L 12

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
```

```
_$

Ps
--
YZ

Z$

Z$

Z$


Z$


Z$





Z$

Z$

Z$

Z$

Z$

Z$


Z$
```

FILEID**SYSGETJPI

B 14

```
  SSSSSSSS  YY      YY    SSSSSSSS    GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT          JJ  PPPPPPPP      IIIIII
  SSSSSSSS  YY      YY    SSSSSSSS    GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT          JJ  PPPPPPPP      IIIIII
SS          YY      YY  SS          GG          EE              TT              JJ  PP      PP      II
SS          YY      YY  SS          GG          EE              TT              JJ  PP      PP      II
SS            YY  YY    SS          GG          EE              TT              JJ  PP      PP      II
SS            YY  YY    SS          GG          EE              TT              JJ  PP      PP      II
  SSSSSS        YY        SSSSSS    GG          EEEEEEEE        TT              JJ  PPPPPPPP          II
  SSSSSS        YY        SSSSSS    GG          EEEEEEEE        TT              JJ  PPPPPPPP          II
        SS      YY              SS  GG  GGGGGG  EE              TT      JJ      JJ  PP                II
        SS      YY              SS  GG  GGGGGG  EE              TT      JJ      JJ  PP                II
        SS      YY              SS  GG      GG  EE              TT      JJ      JJ  PP                II          ....
        SS      YY              SS  GG      GG  EE              TT      JJ      JJ  PP                II          ....
SSSSSSSS        YY      SSSSSSSS      GGGGGG    EEEEEEEEEE      TT        JJJJJJ    PP              IIIIII        ....
SSSSSSSS        YY      SSSSSSSS      GGGGGG    EEEEEEEEEE      TT        JJJJJJ    PP              IIIIII        ....

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

Table of contents

```
0000          .TITLE  SYSGETJPI - GET JOB PROCESS INFORMATION SYSTEM SERVICE
0000          .IDENT  'V04-000'
0000  ;
0000  ;****************************************************************************
0000  ;*                                                                          *
0000  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000  ;*  ALL RIGHTS RESERVED.                                                    *
0000  ;*                                                                          *
0000  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000  ;*  TRANSFERRED.                                                            *
0000  ;*                                                                          *
0000  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000  ;*  CORPORATION.                                                            *
0000  ;*                                                                          *
0000  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000  ;*                                                                          *
0000  ;*                                                                          *
0000  ;****************************************************************************
0000
0000  ;++
0000  ; FACILITY: VMS Executive, System services.
0000  ;
0000  ; ABSTRACT:
0000  ;
0000  ;       Return accounting, quota, and informational data about the current
0000  ;       process, or any other process.
0000  ;
0000  ; ENVIRONMENT: Kernel Mode
0000  ;
0000  ; AUTHOR: Henry M. Levy , CREATION DATE: 20-October-1977
0000  ;
0000  ; MODIFIED BY:
0000  ;
0000  ;       V03-024 MSH0071         Michael S. Harvey       26-Jul-1984
0000  ;               Don't clobber user address space when issuing JPI items
0000  ;               that get serviced via kernel ASTs.
0000  ;
0000  ;       V03-023 MSH0062         Michael S. Harvey       6-Jul-1984
0000  ;               Don't skip the NULL process when wildcarding through
0000  ;               the process set.
0000  ;
0000  ;       V03-022 MSH0040         Michael S. Harvey       1-May-1984
0000  ;               Look for image name in designated area whenever an AME
0000  ;               is running in the process.
0000  ;
0000  ;       V03-021 HWS0055         Harold Schultz          11-Apr-1984
0000  ;               Add JPI$_MASTER_PID item so that the PID of the
0000  ;               master process in a job can be accessed.
0000  ;               (since the MPID in the JIB is in internal format, it
0000  ;                is translated to extended format before it is returned)
```

```
0000  ;                  Change JPI$_PROC_INDEX special processing entry point to
0000  ;                  be local.
0000  ;
0000  ;        V03-020 CWH3020         CW Hobbs                20-Mar-1984
0000  ;                  Add JPI$_PROC_INDEX item so that applications which used to
0000  ;                  look at the low word of the PID can adapt.
0000  ;
0000  ;        V03-019 MSH0010         Michael S. Harvey       14-Feb-1984
0000  ;                  Restructure internal item information tables to
0000  ;                  accomodate counted strings up to 255 bytes.
0000  ;
0000  ;        V03-018 KFH0009         Ken Henderson            7 Sep 1983
0000  ;                  Fix KFH0008 more thoroughly
0000  ;                  Also add SPC_MODE routine.
0000  ;
0000  ;        V03-017 KFH0008         Ken Henderson           29 Aug 1983
0000  ;                  Add documentation on how
0000  ;                  to add itemcodes.
0000  ;
0000  ;        V03-016 KFH0008         Ken Henderson           18 Aug 1983
0000  ;                  Don't return ACCVIO if can't get data
0000  ;
0000  ;        V03-015 WMC0001         Wayne Cardoza           28-Jul-1983
0000  ;                  Allow chained item lists.
0000  ;
0000  ;        V03-014 KFH0007         Ken Henderson           18 Jul 1983
0000  ;                  Fixed IMAGNAME bug in source.
0000  ;
0000  ;        V03-013 KFH0006         Ken Henderson           12 Jul 1983
0000  ;                  Fixed STRDSC bug in source.
0000  ;
0000  ;        V03-012 KFH0005         Ken Henderson           16 Jun 1983
0000  ;                  Fixed various bugs in source.
0000  ;
0000  ;        V03-011 KFH0004         Ken Henderson           27 May 1983
0000  ;                  Added HEXSTR datatype to macro.
0000  ;                  Fixed wildcarding bug, DELPEN bug,
0000  ;                  and sKAST buffer length bug. Cleaned
0000  ;                  up use of stack scratch space.
0000  ;
0000  ;        V03-010 KFH0003         Ken Henderson           24 Mar 1983
0000  ;                  Fetch other processes' PHD items directly
0000  ;                  if the header is resident, and allow
0000  ;                  NULL and SWAPPER processes to be visible.
0000  ;
0000  ;        V03-009 KFH0002         Ken Henderson            1 Mar 1983
0000  ;                  Mods to support bitfield item-codes.
0000  ;
0000  ;        V03-008 CWH1002         CW Hobbs                25-Feb-1983
0000  ;                  Modify to use extended pids.  Use SCH$C_SWPPIX to locate the
0000  ;                  pcb of the swapper.
0000  ;
0000  ;        V03-007 KFH0001         Ken Henderson           10 Feb 1983
0000  ;                  Condense the table macros into JPI_ITEM_CODE,
0000  ;                  and add call the JPI_GENERATE_TABLE to invoke
0000  ;                  JPI_ITEM_CODE for each item-code.
0000  ;
```

```
0000   ;        V03-006 LJK0187         Lawrence J. Kenah       22-Oct-1982
0000   ;                Correct erroneous reference to ASTCNT. Make routine that checks
0000   ;                accessibility of image name a global routine.
0000   ;
0000   ;        V03-005 LJK0157         Lawrence J. Kenah       7-Apr-1982
0000   ;                Add support for JPI$_IMAGECOUNT for LIB$SPAWN's benefit
0000   ;
0000   ;        V03-004 LJK0155         Lawrence J. Kenah       1-Apr-1982
0000   ;                Handle quota deductions in consistent fashion.
0000   ;
0000   ;        V03-003 MSH0001         Maryann Hinden          23-Mar-1982
0000   ;                Fix broken BSBW's.
0000   ;
0000   ;        V03-002 DWT0032         David Thiel             22-Mar-1982
0000   ;                Correct length of probe for returning asynchronous result
0000   ;                length.
0000   ;
0000   ;        V03-001 LJK0146         Lawrence J. Kenah       16-Mar-1982
0000   ;                Correct bugs along code path for not enough nonpaged pool.
0000   ;                Use action routine to convert AUTHPRI item. Do not return
0000   ;                address items for another process. Add additional check
0000   ;                that item list has not changed before second scan.
0000   ;
0000   ;--
```

```
0000    ;                        GUIDE TO GETJPI/GETSYI/GETDVI
0000    ;                        -----------------------------
0000    ;
0000    ;Overview
0000    ;--------
0000    ;
0000    ;These three system services are table-driven. The macro definition files
0000    ;that help define their tables are shared with DCL and the RTL. This results
0000    ;in new item-codes becoming useable with DCL's F$GETXXI lexical functions and
0000    ;the RTL's LIB$GETXXI routines automatically. Additionally, new SYSBOOT
0000    ;parameters become item-codes to the GETSYIs.
0000    ;
0000    ;The macro definition files are called JPITABLE.MAR, SYITABLE.MAR, and
0000    ;DVITABLE.MAR, and live in MASD$:<VMSLIB.SRC>. During a systembuild, they
0000    ;are inserted into the library SYS$LIBRARY:SYSBLDMLB.MLB. DCL and the RTL
0000    ;and SYS use this library to define their GETXXI tables. The system
0000    ;parameter file <SYS.SRC>SYSPARAM.MAR has also been conditionalized to be
0000    ;used to define GETSYI item-codes and is also inserted into SYSBLDMLB.MLB.
0000    ;
0000    ;
0000    ;NOTE:   SYSBLDMLB.MLB is a general macro library for holding macro
0000    ;        definitions that are shared between facilities, but will not
0000    ;        ship to the customer.
0000    ;
0000    ;
0000    ;When adding an item-code, at least two files need to be edited. One of the
0000    ;macro files listed above, as well as an SDL file that defines the 16-bit
0000    ;number which is the user-visible item-code. Also, if a SYSBOOT parameter is
0000    ;added, an SDL file needs to be updated to define the new GETSYI item-code.
0000    ;
0000    ;The GETDVI service actually uses only one table, but the GETSYI and GETJPI
0000    ;services use several. The JPITABLE file defines all the tables for GETJPI
0000    ;and the SYITABLE file defines all the tables for GETSYI. The different
0000    ;tables group the pieces of data according to method of retrieval.
0000    ;
0000    ;In some cases, the piece of data to be returned by the service requires
0000    ;special processing to fetch, calculate, or format it before returning it.
0000    ;In these cases, the code of the system service needs to be enhanced.
0000    ;If the data returned is a new format for DCL, the lexical function
0000    ;module of DCL may need to be enhanced. This is also true for the RTL code.
```

```
0000   ;The Macros
0000   ;----------
0000   ;
0000   ;A two-level scheme exists for defining the item tables used by the three
0000   ;services and the other facilities. A commonly defined macro (called
0000   ;JPI_GENERATE_TABLE, SYI_GENERATE_TABLE, or DVI_GENERATE_TABLE) contains
0000   ;multiple calls to a lower-level macro (called JPI_ITEM_CODE, SYI_ITEM_CODE,
0000   ;or DVI_ITEM_CODE) which actually defines each element in the table.
0000   ;While the _GENERATE_TABLE macros are commonly defined, the _ITEM_CODE macros
0000   ;are individually defined according to the needs of facility. (For instance,
0000   ;the LEXICON module must store the name of the item as an ASCIC string - in
0000   ;order to match it with the string supplied in the F$GETXXI function call;
0000   ;the other facilities need not store the item name in text.)
0000   ;
0000   ;When an item-code must be added, an additional call to the _ITEM_CODE macro
0000   ;must be added to the appropriate _GENERATE_TABLE macro. In the case of GETJPI
0000   ;and GETDVI, the _GENERATE_TABLE macro is defined in the JPITABLE and DVITABLE
0000   ;modules. The SYI_GENERATE_TABLE macro is defined by the SYSPARAM module
0000   ;- all the calls to the PARAMETER and PQL macros are 'collected' into the
0000   ;SYI_GENERATE_TABLE macro. When used in that mode (when GETSYISW is defined),
0000   ;the SYI_ITEMTABLES macro also becomes part of the SYI_GENERATE_TABLE macro.
0000   ;SYI_ITEMTABLES is defined in the SYITABLE module and contains all the calls
0000   ;to the SYI_ITEM_CODE macro that are Not related to SYSBOOT parameters.
0000   ;When GETSYISW is defined in SYSPARAM, the PARAMETER macro does not allocate
0000   ;or store memory, but rather passes some of the arguments to it on through via
0000   ;a call to SYI_ITEM_CODE. That is how all the calls to PARAMETER become calls
0000   ;to SYI_ITEM_CODE.
0000   ;
0000   ;The following is the situation that exists when the symbol GETSYISW is defined.
0000   ;The non-SYSBOOT items are defined by the macro SYI_ITEMTABLES in SYITABLE.MAR.
0000   ;The SYSBOOT items are defined by each invokation of the PARAMETER macro in
0000   ;SYSPARAM.MAR. Note that each invokation of the PQL macro in SYSPARAM.MAR
0000   ;invokes the PARAMETER macro twice. When GETSYISW is defined, the PARAMETER
0000   ;macro merely passes its arguments through to a call to the SYI_ITEM_CODE
0000   ;macro. The SYI_ITEM_CODE macro is locally defined as needed by the facility.
0000   ;
0000   ;+---------------------------------------------------------------------+
0000   ;|                     SYI_GENERATE_TABLE                              |
0000   ;|   SYI_ITEMTABLES          |                                         |
0000   ;|                           | PARAMETER   |           PQL             |
0000   ;|                           |             | PARAMETER   | PARAMETER   |
0000   ;|             |             |             |             |             |
0000   ;|SYI_ITEM_CODE|SYI_ITEM_CODE|SYI_ITEM_CODE|SYI_ITEM_CODE|SYI_ITEM_CODE|
0000   ;+---------------------------------------------------------------------+
0000   ;\                          / \                                        /
0000   ; \                        /   \                                     /
0000   ;     FROM SYITABLE.MAR                  FROM SYSPARAM.MAR
0000   ;    (NON-SYSBOOT ITEMS)                  (SYSBOOT ITEMS)
```

```
0000            .SBTTL  DECLARATIONS
0000    ;
0000    ; INCLUDE FILES:
0000    ;
0000
0000            $ACBDEF                                 ; AST control block parameters
0000            $DYNDEF                                 ; dynamic memory block types
0000            $IFDDEF                                 ; image file descriptor block
0000            $IPLDEF                                 ; interrupt priority levels
0000            $JIBDEF                                 ; define job information block
0000            $JPIDEF                                 ; define GETJPI item identifiers
0000            $PCBDEF                                 ; define process control block
0000            $PHDDEF                                 ; define process header
0000            $PSLDEF                                 ; processor state longword
0000            $STATEDEF                               ; scheduler state definitions
0000            $PRDEF                                  ; define processor registers
0000            $PRIDEF                                 ; define priority increment classes
0000            $RSNDEF                                 ; define resource wait codes
0000            $SSDEF                                  ; define status codes
0000
0000    ;
0000    ; MACROS:
0000    ;
0000
0000    ;
0000    ; Macros to define entries in the six item information tables.
0000    ; There is a table for each data structure from which the user may
0000    ; request information, and one table for information returned as an
0000    ; address.  Tables are indexed by low byte of item identifier.
0000    ; Refer to "OWN STORAGE:" for pictures of the table entries.
0000    ;
0000
0000            .MACRO  JPI_ITEM_CODE   BASE,-          ; for service to use
0000                                    NAME,-          ; of the item-code
0000                                    SOURCE,-        ; of the data
0000                                    DTYPE,-         ; of returned value
0000                                    BITPOS,-        ; of 'bitval' field
0000                                    BITSIZ,-        ; of 'bitval' field
0000                                    OUTLEN,-        ; of returned value
0000                                    STRUCT          ; containing the field
0000
0000            .IF NOT_DEFINED JPI$_'NAME
0000            .WARN ; JPI$_'NAME' IS NOT DEFINED IN STARDEFFL.SDL
0000            .ENDC
0000
0000            STEP = 4
0000            .IIF IDENTICAL <BASE><PCB>,     STEP    = 5
0000            .IIF IDENTICAL <BASE><PHD>,     STEP    = 5
0000            .IIF IDENTICAL <BASE><ADR>,     STEP    = 5
0000            .IIF IDENTICAL <BASE><CTL>,     STEP    = 7
0000            .IIF IDENTICAL <BASE><PCBFLD>,  STEP    = 7
0000            .IIF IDENTICAL <BASE><PHDFLD>,  STEP    = 7
0000
0000            .=BASE'TBL+<<JPI$_'NAME&^XFF>*STEP>
0000
0000            .IIF IDENTICAL <BASE><PCB>,     .WORD   'STRUCT'$'SOURCE
0000            .IIF IDENTICAL <BASE><PHD>,     .WORD   'STRUCT'$'SOURCE
```

```
0000            .IIF IDENTICAL <BASE><PCBFLD>,  .WORD   PCB$'SOURCE
0000            .IIF IDENTICAL <BASE><PHDFLD>,  .WORD   PHD$'SOURCE
0000            .IIF IDENTICAL <BASE><ADR>,     .LONG   SOURCE
0000            .IIF IDENTICAL <BASE><CTL>,     .LONG   SOURCE
0000
0000            .IF IDENTICAL <BASE><PCBFLD>
0000            .WORD   <'BITSIZ'-1>@11!PCB$V_'BITPOS'
0000            .ENDC
0000
0000            .IF IDENTICAL <BASE><PHDFLD>
0000            .WORD   <'BITSIZ'-1>@11!PHD$V_'BITPOS'
0000            .ENDC
0000
0000            .IF DIFFERENT <BASE><ADR>
0000
0000            $XX$ = VALUE
0000            .IIF IDENTICAL <DTYPE><HEXNUM>, $XX$ = VALUE
0000            .IIF IDENTICAL <DTYPE><DECNUM>, $XX$ = VALUE
0000            .IIF IDENTICAL <DTYPE><HEXSTR>, $XX$ = BSTRING
0000            .IIF IDENTICAL <DTYPE><PRVMSK>, $XX$ = BSTRING
0000            .IIF IDENTICAL <DTYPE><PRTMSK>, $XX$ = BSTRING
0000            .IIF IDENTICAL <DTYPE><PADSTR>, $XX$ = BSTRING
0000            .IIF IDENTICAL <DTYPE><CNTSTR>, $XX$ = CSTRING
0000            .IIF IDENTICAL <DTYPE><STRDSC>, $XX$ = DSTRING
0000            .IIF IDENTICAL <DTYPE><BITVEC>, $XX$ = VALUE
0000            .IIF IDENTICAL <DTYPE><BITVAL>, $XX$ = BOOLE
0000            .IIF IDENTICAL <DTYPE><STDUIC>, $XX$ = VALUE
0000            .IIF IDENTICAL <DTYPE><STDTIM>, $XX$ = BSTRING
0000            .IIF IDENTICAL <DTYPE><ACPTYP>, $XX$ = BSTRING
0000
0000            .BYTE   $XX$
0000            .BYTE   OUTLEN
0000
0000            .ENDC   ; IF DIFFERENT <BASE><ADR>
0000
0000            .BYTE   JPI$C_'STRUCT'TYPE
0000
0000            .ENDM   JPI_ITEM_CODE
0000
0000
0000    ;
0000    ; This macro defines the entries to the table of special items.
0000    ; The items in this table must be handled by action routines
0000    ; before being returned.  Each entry has a word item identifier
0000    ; followed by the address of an action routine.
0000    ;
0000
0000            .MACRO  SPECIAL_ITEM NAME,ROUTINE
0000            .WORD   JPI$_'NAME
0000            .ADDRESS ROUTINE
0000            .ENDM   SPECIAL_ITEM
0000
0000    ;
0000    ; This macro defines flag bits.
0000    ;
0000
0000            .MACRO  JPIBITS NAME,SIZE
```

```
                 0000          JPI_V_'NAME' = JPI_BIT
                 0000          JPI_S_'NAME' = SIZE
                 0000          JPI_BIT = JPI_BIT + SIZE
                 0000          .ENDM   JPIBITS
                 0000
                 0000    ;
                 0000    ; EQUATED SYMBOLS:
                 0000    ;
                 0000
        00000004 0000          EFN = 4                              ; event flag number argument
        00000008 0000          PIDADR = 8                           ; address of PID
        0000000C 0000          PRCNAM = 12                          ; address of name descriptor
        00000010 0000          ITMLST = 16                          ; address of item identifiers
        00000014 0000          IOSB = 20                            ; I/O status block address
        00000018 0000          ASTADR = 24                          ; ast routine address
        0000001C 0000          ASTPRM = 28                          ; ast parameter
                 0000
                 0000    ;
                 0000    ; Space is left on stack for routines which may
                 0000    ; manipulate values before returning them.
                 0000    ;
                 0000
        FFFFFFD8 0000          LOCAL_SPACE = -40
        FFFFFFD8 0000          SCRATCH = LOCAL_SPACE+0
        FFFFFFE8 0000          BITDEF = LOCAL_SPACE+16
        FFFFFFEC 0000          PHDTEMP = LOCAL_SPACE+20
        FFFFFFF4 0000          WSLIST = LOCAL_SPACE+28
        FFFFFFF8 0000          DIRCNT = LOCAL_SPACE+32
        FFFFFFFC 0000          FLAGS = LOCAL_SPACE+36
                 0000
                 0000    ;
                 0000    ; Bit definitions for flags longword on stack
                 0000    ;
                 0000
        00000000 0000          JPI_BIT = 0
                 0000          JPIBITS WILD,1                       ; we're doing a wildcard operation
                 0000          JPIBITS NULLSWAP,1                   ; the target is NULL or SWAPPER
                 0000          JPIBITS STRDSC,1                     ; the datatype is a string descriptor
                 0000
                 0000    ;
                 0000    ; Max structure number definitions
                 0000    ;
                 0000
        00000001 0000          MAX_ADR_ITEM = <JPI$_LASTADR&^XFF>-1 ; maximum ADRTBL item number
        00000010 0000          MAX_CTL_ITEM = <JPI$_LASTCTL&^XFF>-1 ; maximum CTLTBL item number
        00000025 0000          MAX_PCB_ITEM = <JPI$_LASTPCB&^XFF>-1 ; maximum PCBTBL item number
        0000001B 0000          MAX_PHD_ITEM = <JPI$_LASTPHD&^XFF>-1 ; maximum PHDTBL item number
        FFFFFFFF 0000          MAX_PCBFLD_ITEM = <JPI$_LASTPCBFLD&^XFF>-1 ; max PCBFLDTBL item number
        FFFFFFFF 0000          MAX_PHDFLD_ITEM = <JPI$_LASTPHDFLD&^XFF>-1 ; max PHDFLDTBL item number
                 0000
                 0000    ;
                 0000    ; Data type codes (all numeric types have same code)
                 0000    ;
                 0000
        00000000 0000          VALUE = 0                            ; numeric value
        00000001 0000          BSTRING = 1                          ; blank filled string
        00000002 0000          CSTRING = 2                          ; counted ascii string
```

```
00000003  0000              BOOLE = 3                                       ; bit value
00000004  0000              DSTRING = 4                                     ; string descriptor
          0000
          0000      ;
          0000      ; AST control block extensions
          0000      ;
          0000              $DEFINI ACB
          0000
0000001C  0000              .=ACB$L_KAST+4                                          ;
          001C
          001C      $DEF    ACB_L_DADDR     .BLKL   1                               ; data buffer address
          0020      $DEF    ACB_L_EFN       .BLKL   1                               ; event flag number
          0024      $DEF    ACB_L_IOSB      .BLKL   1                               ; completion AST routine addr
          0028      $DEF    ACB_L_OPID      .BLKL   1                               ; original requester's PID
          002C      $DEF    ACB_L_IMGCNT    .BLKL   1                               ; PHD$L_IMGCNT of requester
          0030      $DEF    ACB_L_COUNT     .BLKL   1                               ; item descriptor count
          0034      $DEF    ACB_L_ILIST                                             ; item descriptor list
          0034
0000000C  0034              ACB_C_IDESC = 12                                        ; item descriptor size
          0034
          0034              $DEFEND ACB
          0000
          0000
          0000      ;
          0000      ; OWN STORAGE:
          0000      ;
          0000
00000000  0000              .PSECT  YF$$SYSGETJPI
          0000
          0000      ;
          0000      ; This array contains the maximum item number for each of the
          0000      ; six item data structures, indexed by structure number.
          0000      ;
          0000
          0000      MAXCOUNT:
      01  0000              .BYTE   MAX_ADR_ITEM
      10  0001              .BYTE   MAX_CTL_ITEM
      25  0002              .BYTE   MAX_PCB_ITEM
      1B  0003              .BYTE   MAX_PHD_ITEM
      FF  0004              .BYTE   MAX_PCBFLD_ITEM
      FF  0005              .BYTE   MAX_PHDFLD_ITEM
```

```
          0006  ;
          0006  ; Define the six item data structures.  Each data structure is indexed
          0006  ; by item identifier.
          0006  ;
          0006
          0006  ADRTBL:
          0006          ;-------------------------;
          0006          ; .LONG  ADDRESS          ;
          0006          ; .BYTE  JPI$C_ADRTYPE    ;
          0006          ;-------------------------;
          0006
00000010  0006          .BLKB   5*<MAX_ADR_ITEM+1>               ; define adr table
          0010
          0010  CTLTBL:
          0010          ;-------------------------;
          0010          ; .LONG  ADDRESS          ;
          0010          ; .BYTE  DTYPE            ;
          0010          ; .BYTE  LENGTH           ;
          0010          ; .BYTE  JPI$C_CTLTYPE    ;
          0010          ;-------------------------;
          0010
00000087  0010          .BLKB   7*<MAX_CTL_ITEM+1>               ; define ctl table
          0087
          0087  PCBTBL:
          0087          ;-------------------------;
          0087          ; .WORD  XXX$OFFSET       ;
          0087          ; .BYTE  DTYPE            ;
          0087          ; .BYTE  LENGTH           ;
          0087          ; .BYTE  JPI$C_PCBTYPE    ;
          0087          ;-------------------------;
          0087
00000145  0087          .BLKB   5*<MAX_PCB_ITEM+1>               ; define pcb table
          0145
          0145  PHDTBL:
          0145          ;-------------------------;
          0145          ; .WORD  XXX$OFFSET       ;
          0145          ; .BYTE  DTYPE            ;
          0145          ; .BYTE  OUTLEN           ;
          0145          ; .BYTE  JPI$C_PHDTYPE    ;
          0145          ;-------------------------;
          0145
000001D1  0145          .BLKB   5*<MAX_PHD_ITEM+1>               ; define phd table
          01D1
          01D1  PCBFLDTBL:
          01D1          ;------------------------------;
          01D1          ; .WORD  XXX$OFFSET            ;
          01D1          ; .WORD <BITSIZ-1>@11!BITPOS   ;
          01D1          ; .BYTE  DTYPE                 ;
          01D1          ; .BYTE  OUTLEN                ;
          01D1          ; .BYTE  JPI$C_PCBFLDTYPE      ;
          01D1          ;------------------------------;
          01D1
000001D1  01D1          .BLKB   7*<MAX_PCBFLD_ITEM+1>            ; define pcbfld table
          01D1
          01D1  PHDFLDTBL:
          01D1          ;------------------------------;
          01D1          ; .WORD  XXX$OFFSET            ;
          01D1                ; .WORD <BITSIZ-1>@11!BITPOS  ;
          01D1                ; .BYTE  DTYPE                ;
          01D1                ; .BYTE  OUTLEN               ;
          01D1                ; .BYTE  JPI$C_PHDFLDTYPE     ;
          01D1                ;-----------------------------;
          01D1
 000001D1 01D1                .BLKB   7*<MAX_PHDFLD_ITEM+1>            ; define phdfld table
          01D1
          01D1                .SAVE                                    ; save current location
          01D1
          01D1        ;
          01D1        ;*************************************************************
          01D1        ;
          01D1        ; GENERATE THE SIX TABLES USING THE COMMONLY DEFINED MACRO
          01D1        ;
          01D1        ;*************************************************************
          01D1        ;
          01D1                JPI_GENERATE_TABLE
          01D1
          01D1
 000001D1 01D1                .RESTORE                                 ; restore location
          01D1
          01D1        ;
          01D1        ; Table to define items which must be handled
          01D1        ; by action routines.
          01D1        ;
          01D1
          01D1        SPECIAL:
          01D1                SPECIAL_ITEM    PRI,SPC_PRI              ; handle priority ...
          01D7                SPECIAL_ITEM    PRIB,SPC_PRI             ; ... evaluations
          01DD                SPECIAL_ITEM    AUTHPRI,SPC_PRI          ; all of them
          01E3                                                         ; compute working set
          01E3                SPECIAL_ITEM    WSAUTH,SPC_WORKSET       ; ...parameters
          01E9                SPECIAL_ITEM    WSQUOTA,SPC_WORKSET      ;
          01EF                SPECIAL_ITEM    WSEXTENT,SPC_WORKSET     ;
          01F5                SPECIAL_ITEM    WSAUTHEXT,SPC_WORKSET    ;
          01FB                SPECIAL_ITEM    DFWSCNT,SPC_WORKSET      ;
          0201                SPECIAL_ITEM    IMAGNAME,SPC_IMAGNAME    ; find image name
          0207                SPECIAL_ITEM    MODE,SPC_MODE            ; return the process' mode
          020D                SPECIAL_ITEM    PROC_INDEX,SPC_PROC_INDEX ; create a process index
          0213                SPECIAL_ITEM    MASTER_PID,SPC_MASTER_PID ; return PID of master proc.
          0219
 0000000C 0219        SPECIAL_LEN = <.-SPECIAL>/6                      ; compute number of entries
```

```
0219          .SBTTL  SYSGETJPI - GETJPI main program
0219
0219  ;++
0219  ;
0219  ; FUNCTIONAL DESCRIPTION:
0219  ;
0219  ;       This service allows a process to receive information about itself, or
0219  ;       any process which it has the UIC privilege to examine.
0219  ;
0219  ; CALLING SEQUENCE:
0219  ;
0219  ;       CALLS/CALLG
0219  ;
0219  ; INPUTS:
0219  ;
0219  ;       EFN(AP) = number of the event flag to set when all of the requested
0219  ;                 data is valid.
0219  ;       PIDADR(AP) = address of a longword containing the process ID of the
0219  ;                    process for which the information is being requested
0219  ;       PRCNAM(AP) = address of a string descriptor for the process name
0219  ;                    of the process for which the information is requested
0219  ;       ITMLST(AP) = address of a list of item descriptors of the form:
0219  ;
0219  ;               +------------------------------+
0219  ;               !  ITEM CODE    !  BUF. LENGTH !
0219  ;               +------------------------------+
0219  ;               !       BUFFER ADDRESS         !
0219  ;               +------------------------------+
0219  ;               !  ADDRESS TO RETURN LENGTH    !
0219  ;               +------------------------------+
0219  ;
0219  ;       IOSB(AP) = address of a quadword I/O status block to receive final
0219  ;               status
0219  ;       ASTADR(AP) = address of an AST routine to be called when all of the
0219  ;               requested data has been supplied.
0219  ;       ASTPRM(AP) = 32 bit ast parameter
0219  ;
0219  ; IMPLICIT INPUTS:
0219  ;
0219  ;       none
0219  ;
0219  ; OUTPUTS:
0219  ;
0219  ;       none
0219  ;
0219  ; IMPLICIT OUTPUTS:
0219  ;
0219  ;       none
0219  ;
0219  ; ROUTINE VALUE:
0219  ;
0219  ;       SS$_NORMAL -> normal completion
0219  ;       SS$_ACCVIO -> ITMLST can not be read by the calling access mode,
0219  ;                     or the return buffer or return length word can not
0219  ;                     be written by the calling access mode
0219  ;       SS$_BADPARAM -> an invalid item identifier was supplied
0219  ;       SS$_IVLOGNAM -> zero or greater than maximum length process name string
```

```
                    0219  ;         SS$_NONEXPR -> nonexistant/deleted process or invalid process ID
                    0219  ;                        specified
                    0219  ;         SS$_NOPRIV -> calling process does not have privilege to get information
                    0219  ;                        about the specified process.
                    0219  ;
                    0219  ; SIDE EFFECTS:
                    0219  ;
                    0219  ;         none
                    0219  ;--
                    0219
                00000000          .PSECT  YEXEPAGED                    ; only entry mask in this program section
                    0000
               0FFC 0000          .ENTRY  EXE$GETJPI,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
        0214'   31  0002          BRW     EXE_GETJPI                   ; transfer to real procedure
                    0005
                00000219          .PSECT  YF$$SYSGETJPI
                    0219
                    0219  EXE_GETJPI:
    5E   D8 AE  DE  0219          MOVAL   LOCAL_SPACE(SP),SP           ; allocate local space on stack
         FC AD  D4  021D          CLRL    FLAGS(FP)                    ; reset the flags longword
          0769  30  0220          BSBW    NAMPID                       ; get PID/PCB address of desired process
         72 50  E9  0223          BLBC    R0,15$                       ; exit if invalid process specified
                    0226  ;
                    0226  ; Check for, and clear possible IOSB
                    0226  ;
    51   14 AC  D0  0226          MOVL    IOSB(AP),R1                  ; get IOSB address
            0B  13  022A          BEQL    3$                           ; branch if none
                    022C          IFWRT   #8,(R1),29$                  ; check access to it
          00E6  31  0232          BRW     30$                          ; ACCVIO
            61  7C  0235  29$:    CLRQ    (R1)                         ; clear IOSB
                    0237  ;
                    0237  ; Check for, and clear event flag
                    0237  ;
    53   04 AC  9A  0237  3$:     MOVZBL  EFN(AP),R3                   ; get event flag number
  00000000'EF   16  023B          JSB     SCH$CLREF                    ; clear this event flag
         54 50  E9  0241  4$:     BLBC    R0,15$                       ; and return on errors.
                    0244  ;
                    0244  ; Validate AST, if present. Note R4 still has our PCB address, and R9
                    0244  ; has the PCB address of the process we want information from.
                    0244  ;
         18 AC  D5  0244          TSTL    ASTADR(AP)
            08  13  0247          BEQL    5$                           ; no AST to check.
         38 A4  B5  0249          TSTW    PCB$W_ASTCNT(R4)             ; is quota exceeded?
            03  14  024C          BGTR    5$                           ; nope
          00CF  31  024E          BRW     35$                          ; quota exceeded - return error
                    0251  ;
                    0251  ; R10 is used to count the items that are in the other process's address
                    0251  ; space. The accumulated size of the user buffers is kept track of
                    0251  ; on top of the stack. DIRCNT(FP) counts the number of PHD cells that
                    0251  ; were successfully fetched from another processes' header.
                    0251  ;
            5A  D4  0251  5$:     CLRL    R10                          ; no items yet
            00  DD  0253          PUSHL   #0                           ; no accumulated size either
         F8 AD  D4  0255          CLRL    DIRCNT(FP)                   ; no fetched items yet
                    0258  ;
                    0258  ; Loop through the item descriptor blocks, validating the requested item
                    0258  ; identifiers and moving accessible items.  A zero item identifier terminates
```

```
                        0258   658 ; the list.
                        0258   659 ;
                        0258   660
      55   10 AC    D0  0258   661          MOVL    ITMLST(AP),R5                 ; get item descriptor list address
                        025C   662 6$:      IFRD    #4,(R5),10$                   ; check first longword readable
         00B6       31  0262   663          BRW     30$
                        0265   664
      55      65    D0  0265   665 7$:      MOVL    (R5),R5                       ; get pointer to next chained item list
              F2    11  0268   666          BRB     6$                            ; process it
                        026A   667
                        026A   668 10$:
      56      85    3C  026A   669          MOVZWL  (R5)+,R6                      ; get buffer size
      51      85    3C  026D   670          MOVZWL  (R5)+,R1                      ; get item identifier
              63    13  0270   671          BEQL    49$                           ; done if zero, take normal exit
 FFFF 8F      51    B1  0272   672          CMPW    R1,#JPI$_CHAIN                ; is it a chained item list
              EC    13  0277   673          BEQL    7$
                        0279   674          IFRD    #12,(R5),11$                  ; check rest of this descriptor ...
         0099       31  027F   675          BRW     30$                           ; ... plus first longword of next one
      57      85    7D  0282   676 11$:     MOVQ    (R5)+,R7                      ; get buffer address and return address
              51    DD  0285   677          PUSHL   R1                            ; save R1 across accessibility check
      50      57    D0  0287   678          MOVL    R7,R0                         ; buffer address to R0
      51      56    D0  028A   679          MOVL    R6,R1                         ; and size to R1
              53    D4  028D   680          CLRL    R3                            ; PROBE will use PSL<PRVMOD>
     00000000'EF    16  028F   681          JSB     EXE$PROBEW                    ; check write accessibility of buffer
              51  8ED0  0295   682          POPL    R1                            ; restore R1 for use by CHECKITEM
           3C 50    E9  0298   683 15$:     BLBC    R0,31$                        ; return error if inaccessible
              55    DD  029B   684          PUSHL   R5                            ; save R5 from action routines
            02E4    30  029D   685          BSBW    CHECKITEM                     ; validate identifier and get item info.
           37 50    E9  02A0   686          BLBC    R0,41$                        ; invalid item if error
 00000000'EF  59    D1  02A3   687          CMPL    R9,SCH$GL_CURPCB              ; is this for current process?
              17    12  02AA   688          BNEQ    16$                           ; branch if not
 5E FC AD     01    E1  02AC   689          BBC     #JPI_V_NULLSWAP,FLAGS(FP),20$
                        02B1   690                                                ; branch if it's current 'full' process
                        02B1   691
                        02B1   692          CASE    R2,<-                         ; the current process is null or swap
                        02B1   693                  45$,-    ; ADR
                        02B1   694                  45$,-    ; CTL
                        02B1   695                  20$,-    ; PCB
                        02B1   696                  20$,-    ; PHD
                        02B1   697                  20$,-    ; PCBFLD
                        02B1   698                  20$,-    ; PHDFLD
                        02B1   699                  45$-     ; JIB
                        02B1   700                  >B,#1
                        02C3   701
                        02C3   702 16$:     CASE    R2,<-                         ; it isn't the current process
                        02C3   703                  45$,-    ; ADR
                        02C3   704                  18$,-    ; CTL
                        02C3   705                  20$,-    ; PCB
                        02C3   706                  17$,-    ; PHD
                        02C3   707                  20$,-    ; PCBFLD
                        02C3   708                  17$,-    ; PHDFLD
                        02C3   709                  21$-     ; JIB
                        02C3   710                  >B,#1
                        02D5   711
              5E    11  02D5   712 49$:     BRB     50$                           ; HELPER BRANCHES
         0064       31  02D7   713 31$:     BRW     GRET
         0048       31  02DA   714 41$:     BRW     40$
```

```
                              02DD
         1B 24 A9    12   E1  02DD  17$:    BBC     #PCB$V_PHDRES,PCB$L_STS(R9),18$ ; is the header resident?
                              02E2          ; R0 returned from MOVEPHD can be the following:
                              02E2          ; SS$_ACCVIO - routine MOVEIT couldn't stuff RETLEN
                              02E2          ; SS$_NONEXPR - got into MOVEPHD and DELPEN was set
                              02E2          ; SS$_NORMAL - everything fine - got the data
                              02E2          ; 0 - got into MOVEPHD and PHD had gone away - get with sKAST
               03EB  30       02E2          BSBW    MOVEPHD                 ; must be, go get the data
              10 50  E8       02E5          BLBS    R0,102$                 ; got it! count it.
            0C   50  D1       02E8          CMPL    R0,#SS$_ACCVIO          ; was the retlen bad?
                 51  13       02EB          BEQL    GRET                    ; EQL means it was
         38 FC AD    01   E0  02ED          BBS     #JPI_V_NULLSWAP,FLAGS(FP),45$ ; couldn't get Null/Swap? oh oh.
                 50  D5       02F2          TSTL    R0                      ; now check for PHD no longer RES
                 0C  13       02F4          BEQL    19$                     ; EQL means PHD no longer RES
                 46  11       02F6          BRB     GRET                    ; whatever the error, go return it
                              02F8
              F8 AD  D6       02F8  102$:   INCL    DIRCNT(FP)              ; successful at getting it directly
                 05  11       02FB          BRB     19$
                              02FD
         28 FC AD    01   E0  02FD  18$:    BBS     #JPI_V_NULLSWAP,FLAGS(FP),45$
                              0302                                          ; Null and Swapper don't have CTL reg.
                 5A  D6       0302  19$:    INCL    R10                     ; count up one more for sKAST later.
            04 AE    56   C0  0304          ADDL2   R6,4(SP)                ; Add in size of user's buffer
                 08  11       0308          BRB     25$
         1B FC AD    01   E0  030A  21$:    BBS     #JPI_V_NULLSWAP,FLAGS(FP),45$
                              030F                                          ; Null and Swapper don't have a JIB
               0379  30       030F  20$:    BSBW    MOVEIT                  ; move item to user
              55 8ED0         0312  25$:    POPL    R5                      ; restore R5
              26 50  E9       0315          BLBC    R0,GRET                 ; return length not writeable
               FF4F  31       0318          BRW     10$                     ; back for next descriptor
                              031B
            50   0C  3C       031B  30$:    MOVZWL  #SS$_ACCVIO,R0          ; access violation
                 1E  11       031E          BRB     GRET
                              0320
            50   1C  3C       0320  35$:    MOVZWL  #SS$_EXQUOTA,R0         ; AST quota exceeded
                 19  11       0323          BRB     GRET
                              0325
            50   14  3C       0325  40$:    MOVZWL  #SS$_BADPARAM,R0        ; illegal item or request
                 14  11       0328          BRB     GRET
                              032A
        54   D8 AD   DE       032A  45$:    MOVAL   SCRATCH(FP),R4          ; make a 16-byte zeroed buffer
                 64  7C       032E          CLRQ    (R4)
              08 A4  7C       0330          CLRQ    8(R4)
                 DA  11       0333          BRB     20$                     ; ... through common subroutine
                              0335
            50   01  3C       0335  50$:    MOVZWL  #SS$_NORMAL,R0          ; normal return
        5A   F8 AD   D1       0338          CMPL    DIRCNT(FP),R10          ; any items we couldn't get?
                 6E  12       033C          BNEQ    RESCAN                  ; if so, go obtain them.
                              033E  ;
                              033E  ; Set the event flag, post the completion status, and declare a completion AST
                              033E  ;
                 50  DD       033E  GRET:   PUSHL   R0                      ; save completion status
    54   00000000'EF  D0      0340          MOVL    SCH$GL_CURPCB,R4        ; get PCB address
           51   60 A4  D0     0347          MOVL    PCB$L_PID(R4),R1        ; get process's PID
                 52  D4       034B          CLRL    R2                      ; set null priority increment
           53   04 AC  D0     034D          MOVL    EFN(AP),R3              ; get event flag number to set
        00000000'EF  16       0351          JSB     SCH$POSTEF              ; set the event flag
```

```
          51    14 AC     D0  0357  10$:     MOVL     IOSB(AP),R1                    ; get address of IOSB
                   09     13  035B           BEQL     20$                            ; branch if none
                              035D           IFNOWRT  #8,(R1),20$                    ; check if writable
             61    6E     D0  0363           MOVL     (SP),(R1)                      ; store completion status
          55    18 AC     D0  0366  20$:     MOVL     ASTADR(AP),R5                  ; get address of AST routine
                   15     13  036A           BEQL     30$                            ; branch if none specified
                   54     DC  036C           MOVPSL   R4                             ; get PSL
 54    54    02    16     EF  036E           EXTZV    #PSL$V_PRVMOD,#PSL$S_PRVMOD,R4,R4 ; extract previous mode
                              0373           $DCLAST_S (R5),ASTPRM(AP),R4            ; queue the completion AST
                50 8ED0       0381  30$:     POPL     R0                             ; restore completion status
                   04         0384           RET                                     ; and return.
```

```
                          0385          .SBTTL  RESCAN - Rescan item list creating list of items in process
                          0385     ;++
                          0385     ;
                          0385     ; FUNCTIONAL DESCRIPTION:
                          0385     ;
                          0385     ;        Routine to obtain information that is contained in another process's
                          0385     ;        virtual address space.  This is accomplished by first creating a list
                          0385     ;        of items that are to be obtained from the other process. An AST is
                          0385     ;        then queued to the process to execute a routine in this service that
                          0385     ;        copies the desired items to a buffer in non-paged pool.  The routine
                          0385     ;        then queues another AST back to the requesting process to execute
                          0385     ;        another routine in this service to copy the items from the system
                          0385     ;        buffer to the requester's buffers.
                          0385     ;
                          0385     ; CALLING SEQUENCE:
                          0385     ;
                          0385     ;        Branch
                          0385     ;
                          0385     ; INPUTS:
                          0385     ;
                          0385     ;        R10 = number of items that are in other process's address space
                          0385     ;        R11 = PID of other process
                          0385     ;        (SP) = accumulated size of user buffers. A buffer of this size
                          0385     ;               will be allocated from nonpaged pool to hold data from
                          0385     ;               the target process.
                          0385     ;
                          0385     ; OUTPUTS:
                          0385     ;
                          0385     ;        none
                          0385     ;
                          0385     ; IMPLICIT OUTPUTS:
                          0385     ;
                          0385     ;        An extended AST control block is allocated and filled-in with the
                          0385     ;        usual AST parameters with the extension containing a list of
                          0385     ;        item descriptors.  A data buffer is also allocated to contain the
                          0385     ;        item data.
                          0385     ;
                          0385     ; ROUTINE VALUE:
                          0385     ;
                          0385     ;        none
                          0385     ;
                          0385     ; SIDE EFFECTS:
                          0385     ;
                          0385     ;        lots
                          0385     ;--
                          0385
                          0385          .ENABLE LOCAL_BLOCK
                          0385
0A 24 A4    0A      E1    0385     5$:  BBC     #PCB$V_SSRWAIT,PCB$L_STS(R4),7$ ; do not wait if set
                          038A          ENBINT                          ; allow interrupts again
   50    0124 8F    3C    038D          MOVZWL  #SS$_INSFMEM,R0         ; indicate no pool left
              AA    11    0392     6$:  BRB     GRET                    ; and join common exit path
                          0394
                          0394     ; There is not enough nonpaged pool. The process must be placed into resource
                          0394     ; wait until pool becomes available.
                          0394
              50    DC    0394     7$:  MOVPSL  R0                      ; get current PSL
```

```
     6E   50   001F0000 8F   CB  0396          BICL3   #PSL$M_IPL,R0,(SP)            ; wait at IPL 0 to allow ASTs
                    50  03   3C  039E          MOVZWL  #RSN$_NPDYNMEM,R0             ; ... for some pool to be given back
             00000000'GF     16  03A1          JSB     G^SCH$RWAIT                   ; quota check will be repeated
                    51  56   D0  03A7          MOVL    R6,R1                         ;  when process executes again
                        14   11  03AA          BRB     8$                            ; pool available. repeat quota check
                                 03AC
                                 03AC  RESCAN:
                                 03AC  ;
                                 03AC  ; Allocate an extended AST block
                                 03AC  ;
          54   00000000'EF   D0  03AC          MOVL    SCH$GL_CURPCB,R4              ; get current PCB address
             51     5A  0C   C5  03B3          MULL3   #ACB_C_IDESC,R10,R1           ; compute size of item descriptors
                    51  34   C0  03B7          ADDL2   #ACB_L_ILIST,R1               ; plus header
                    51  6E   C0  03BA          ADDL2   (SP),R1                       ; plus buffer size
                    56  51   D0  03BD          MOVL    R1,R6                         ; save request size for later storage
             00000000'GF     16  03C0  8$:     JSB     G^EXE$BUFFRQUOTA              ; check buffer quota
                     C9 50   E9  03C6          BLBC    R0,6$                         ; quit if not enough quota
                                 03C9  9$:     DSBINT  120$                          ; elevate IPL to SYNCH
             00000000'GF     16  03D3          JSB     G^EXE$ALONONPAGED             ; now allocate the chunk of pool
-                    A9 50   E9  03D9          BLBC    R0,5$                         ; get out if no pool available
                                 03DC  ;
                                 03DC  ; Fill-in the standard AST parameters and header information
                                 03DC  ;
                                 03DC          ENBINT                                ; allow scheduling again
          50    0080 C4      D0  03DF          MOVL    PCB$L_JIB(R4),R0              ; get JIB address
             20 A0      56   C2  03E4          SUBL    R6,JIB$L_BYTCNT(R0)           ; adjust buffer quota
             0C A2      5B   D0  03E8          MOVL    R11,ACB$L_PID(R2)             ; PID of target process
                    5B  52   D0  03EC          MOVL    R2,R11                        ; save address of AST block
             08 AB      56   B0  03EF          MOVW    R6,ACB$W_SIZE(R11)            ; save block size for deallocation
             0A AB      02   90  03F3          MOVB    #DYN$C_ACB,ACB$B_TYPE(R11) ; set block type
                        51   DC  03F7          MOVPSL  R1                            ; get PSL
     51     51     02   16   EF  03F9          EXTZV   #PSL$V_PRVMOD,#PSL$S_PRVMOD,R1,R1 ; get requester's mode
  0B AB     51     80   8F   89  03FE          BISB3   #<1@ACB$V_KAST>,R1,ACB$B_RMOD(R11) ; and put into block
      18 AB     082C'CF      9E  0404          MOVAB   W^MOVEFU,ACB$L_KAST(R11)        ; special kernel address
        10 AB     18 AC      D0  040A          MOVL    ASTADR(AP),ACB$L_AST(R11) ; return ast address
                        13   13  040F          BEQL    10$                           ; skip quota check if none
                    50  1C   3C  0411          MOVZWL  #SS$_EXQUOTA,R0               ; assume exceeded
                     38 A4   D5  0414          TSTL    PCB$W_ASTCNT(R4)              ; any left
                        03   12  0417          BNEQ    91$                           ; hop, skip to error return
                      0089   31  0419          BRW     35$
                     38 A4   B7  041C  91$:    DECW    PCB$W_ASTCNT(R4)              ; subtract from quota
       00 0B AB         06   E2  041F          BBSS    #ACB$V_QUOTA,ACB$B_RMOD(R11),10$ ; and record that fact in ACB
          14 AB     1C AC    D0  0424  10$:    MOVL    ASTPRM(AP),ACB$L_ASTPRM(R11) ; and parameter
          20 AB     04 AC    9A  0429          MOVZBL  EFN(AP),ACB_L_EFN(R11)    ; efn to set on return
          24 AB     14 AC    D0  042E          MOVL    IOSB(AP),ACB_L_IOSB(R11) ; address of possible iosb
          28 AB     60 A4    D0  0433          MOVL    PCB$L_PID(R4),ACB_L_OPID(R11) ; our PID
       53    00000000'EF     D0  0438          MOVL    CTL$GL_PHD,R3                 ; get address of process header
        2C AB    00F4 C3     D0  043F          MOVL    PHD$L_IMGCNT(R3),ACB_L_IMGCNT(R11) ; sequence number of this image
            30 AB       5A   D0  0445          MOVL    R10,ACB_L_COUNT(R11)          ; item count
               1C AB         D4  0449          CLRL    ACB_L_DADDR(R11)              ; no data buffer yet
                                 044C                                                ; allocated but location not recorded
                    044C     ;
                    044C     ; Loop through the list, copying the item descriptors for items in the
                    044C     ; process's address space to the extended AST block.
                    044C     ;
                    044C     ; The item descriptor list will look like:
                    044C     ;
                    044C     ;         +--------------------------------+
                    044C     ;         ! item code      ! buffer length !
                    044C     ;         +--------------------------------+
                    044C     ;         !    user buffer address         !
                    044C     ;         +--------------------------------+
                    044C     ;         ! address to return length       !
                    044C     ;         +--------------------------------+
                    044C     ;
                    044C     ; When the item list comes back from the kernel ASTs, the item code field
                    044C     ; is overwritten with the actual length of the source data for each item.
                    044C     ;
     58    34 AB    DE 044C                MOVAL   ACB_L_ILIST(R11),R8       ; get address of item descriptor list
     57    10 AC    D0 0450                MOVL    ITMLST(AP),R7             ; get address of item specifier list
                    0454     12$:          IFNORD  #4,(R7),30$               ; check first longword still readable
                    045A
        56    87    3C 045A     15$:       MOVZWL  (R7)+,R6                  ; get user buffer size
        51    87    3C 045D                MOVZWL  (R7)+,R1                  ; get item identifier
              46    13 0460                BEQL    40$                       ; if zero, we're done with list.
                    0462                   IFNORD  #12,(R7),30$              ; check still readable
 FFFF 8F      51    B1 0468                CMPW    R1,#JPI$_CHAIN            ; is it a chained item list
              2B    13 046D                BEQL    19$
        7E    87    7D 046F                MOVQ    (R7)+,-(SP)               ; get user buffer and length addresses
          010F      30 0472                BSBW    CHECKITEM                 ; get structure type into R2
         27   50    E9 0475                BLBC    R0,20$                    ; make sure argument list has not changed
        50    53    D0 0478                MOVL    R3,R0                     ; save item length
        53    8E    7D 047B                MOVQ    (SP)+,R3                  ; get user buffer and length addresses
     55    AA 8F    9A 047E                MOVZBL  #<<1@JPI$C_PCBTYPE>!-     ; create mask of types in system space
                    0482                              <1@JPI$C_PCBFLDTYPE>!- ;  plus address type, which is
                    0482                              <1@JPI$C_JIBTYPE>!-    ;  returned as zero if not for caller
                    0482                              <1@JPI$C_ADRTYPE>>,R5  ;
  D4 55       52    E0 0482                BBS     R2,R5,15$                 ; branch if we already got it
              5A    D7 0486                DECL    R10                       ; decrement item counter
              15    19 0488                BLSS    20$                       ; error if count goes negative
        6E    56    C2 048A                SUBL2   R6,(SP)                   ; subtract user buffer size from input
              10    19 048D                BLSS    20$                       ; error if result goes negative
        88    56    B0 048F                MOVW    R6,(R8)+                  ; copy user buffer size
                    0492
        88    51    B0 0492     18$:       MOVW    R1,(R8)+                  ; copy item identifier
        88    53    7D 0495                MOVQ    R3,(R8)+                  ; copy user buffer and length address
              C0    11 0498                BRB     15$                       ; and loop through till done.
                    049A
        57    87    D0 049A     19$:       MOVL    (R7)+,R7                  ; pointer to next item list
              B5    11 049D                BRB     12$                       ; go process it
                    049F
              14    DD 049F     20$:       PUSHL   S^#SS$_BADPARAM           ; set bad parameters failure
              02    11 04A1                BRB     35$                       ;
                    04A3
              0C    DD 04A3     30$:       PUSHL   S^#SS$_ACCVIO             ; set access violation failure
          008E      31 04A5     35$:       BRW     100$                      ;
                    04A8
              5A    D5 04A8     40$:       TSTL    R10                       ; count should be zero
              F3   12  04AA   950          BNEQ    20$                           ; error if it is not.
              8E   D5  04AC   951          TSTL    (SP)+                         ; so should size be zero
              EF   12  04AE   952          BNEQ    20$                           ; error if it is not.
       1C AB  58   D0  04B0   953          MOVL    R8,ACB_L_DADDR(R11)           ; fill in buffer address
                       04B4   954  ;
                       04B4   955  ; The AST is queued to the destination process unless it has delete or
                       04B4   956  ; suspend pending set, or is currently suspended.
                       04B4   957  ;
                       04B4   958  50$:    SETIPL  120$                          ; raise IPL to synch, lock code
          55  5B   D0  04BB   959          MOVL    R11,R5                        ; set address of AST block
       54  0C A5   3C  04BE   960          MOVZWL  ACB$L_PID(R5),R4              ; PIX of destination process
54  00000000'FF44  D0  04C2   961          MOVL    @SCH$GL_PCBVEC[R4],R4         ; get PCB address
   0C A5    60 A4  D1  04CA   962          CMPL    PCB$L_PID(R4),ACB$L_PID(R5) ; see if PIDs the same
              59   12  04CF   963          BNEQ    80$                           ; and exit if not
   54 24 A4   01   E0  04D1   964          BBS     #PCB$V_DELPEN,PCB$L_STS(R4),80$ ; or if delete pending
   56 24 A4   0B   E0  04D6   965          BBS     #PCB$V_SUSPEN,PCB$L_STS(R4),90$ ; or if suspend pending
      2C A4   09   B1  04DB   966          CMPW    #SCH$C_SUSP,PCB$W_STATE(R4)
              50   13  04DF   967          BEQL    90$                           ; process suspended, error exit
      2C A4   0A   B1  04E1   968          CMPW    #SCH$C_SUSPO,PCB$W_STATE(R4)
              4A   13  04E5   969          BEQL    90$                           ; or suspended out of memory
      2C A4   02   B1  04E7   970          CMPW    #SCH$C_MWAIT,PCB$W_STATE(R4)
              44   13  04EB   971          BEQL    90$                           ; or an indeterminately long wait state
          52  04   D0  04ED   972          MOVL    #PRI$_TICOM,R2                ; give a big priority increment
   AE FC AD   01   E0  04F0   973          BBS     #JPI_V_NULLSWAP,FLAGS(FP),30$ ; don't ever queue it to these!
     00000000'EF   16  04F5   974          JSB     SCH$QAST                      ; queue AST to other process
                       04FB   975  ;
                       04FB   976  ; If process is in compute state and at a lower priority than the requesting
                       04FB   977  ; process, boost its current priority to the requesting process's current
                       04FB   978  ; priority.  (Required because event reporting won't normally boost a COM
                       04FB   979  ; state process's priority).
                       04FB   980  ;
      2C A4   0C   B1  04FB   981          CMPW    #SCH$C_COM,PCB$W_STATE(R4) ; process in compute state?
              06   13  04FF   982          BEQL    60$                           ;
      2C A4   0D   B1  0501   983          CMPW    #SCH$C_COMO,PCB$W_STATE(R4) ; or compute out of memory
              1C   12  0505   984          BNEQ    70$                           ;
53   00000000'EF   D0  0507   985  60$:    MOVL    SCH$GL_CURPCB,R3              ; get requestor's PCB address
     50    0B A3   90  050E   986          MOVB    PCB$B_PRI(R3),R0              ; get requestor's current priority
      0B A4   50   91  0512   987          CMPB    R0,PCB$B_PRI(R4)              ; other process have a higher priority?
              0B   1E  0516   988          BGEQU   70$                           ; if GEQU yes - don't boost priority
          50  10   91  0518   989          CMPB    #16,R0                        ; will boost be into realtime priority?
              06   1A  051B   990          BGTRU   70$                           ; if GTRU yes - don't boost priority
     00000000'EF   16  051D   991          JSB     SCH$CHSEP                     ; boost other process's priority
          50  01   3C  0523   992  70$:    MOVZWL  #SS$_NORMAL,R0                ; so far, so good.
                       0526   993          SETIPL  #0                            ;
                   04  0529   994          RET                                   ;
                       052A   995  ;
                       052A   996  ; Error recovery when the process we want to send the AST to has vanished,
                       052A   997  ; has delete pending, or is suspended; we must release both blocks
                       052A   998  ;
    7E    08E8 8F  3C  052A   999  80$:    MOVZWL  #SS$_NONEXPR,-(SP)            ; non-existent process
              05   11  052F  1000          BRB     100$
                       0531  1001
    7E    03A4 8F  3C  0531  1002  90$:    MOVZWL  #SS$_SUSPENDED,-(SP)          ; process is suspended
                       0536  1003                                                ;
     50    08 AB   3C  0536  1004  100$:   MOVZWL  ACB$W_SIZE(R11),R0            ; need to restore BYTCNT quota
54   00000000'GF   D0  053A  1005          MOVL    G^SCH$GL_CURPCB,R4            ; to caller of $GETJPI
    51    0080 C4  D0  0541  1006          MOVL    PCB$L_JIB(R4),R1              ; get JIB address
```

```
           20 A1    50   C0  0546   1007             ADDL2   R0,JIB$L_BYTCNT(R1)         ; and give back quota
        03 0B AB    06   E1  054A   1008             BBC     #ACB$V_QUOTA,ACB$B_RMOD(R11),105$ ; also ASTCNT if that
              38 A4      B6  054F   1009             INCW    PCB$W_ASTCNT(R4)            ; was subtracted before
           50       5B   D0  0552   1010 105$:       MOVL    R11,R0                      ; get address of AST block
       00000000'GF       16  0555   1011             JSB     G^EXE$DEANONPAGED           ; deallocate the block
                    50 8ED0  055B   1012             POPL    R0                          ; restore status
                             055E   1013             SETIPL  #0                          ; restore IPL to allow page faults
       08E8 8F      50   B1  0561   1014             CMPW    R0,#SS$_NONEXPR             ; is error nonexistent process?
                    04   13  0566   1015             BEQL    130$                        ; branch if yes
                  FDD3   31  0568   1016 110$:       BRW     GRET                        ;
                             056B   1017
                             056B   1018
                             056B   1019 ;
                             056B   1020 ; The preceding code must raise IPL to synchronize access to process database,
                             056B   1021 ; but since it is paged it must be locked in memory. The usage of the SETIPL
                             056B   1022 ; macro above, both raises IPL and faults the code into memory.
                             056B   1023 ;
                             056B   1024 120$:
                         08  056B   1025             .BYTE   IPL$_SYNCH                  ; end of locked code region
                             056C   1026             ASSUME  <.-5$> LE 512               ; only 512 bytes can be locked
                             056C   1027             ASSUME  <.-50$> LE 512              ; only 512 bytes can be locked
                             056C   1028
                             056C   1029 ;  If process has disappeared (has already been deleted or is in a delete
                             056C   1030 ;  pending state) in the interval between selection and queuing the AST,
                             056C   1031 ;  and the initial call indicated wild card mode, then go back to the
                             056C   1032 ;  beginning of the service. Note that wild card mode is indicated by a
                             056C   1033 ;  negative number (usually -1) in the upper wcrd of the PID argument in
                             056C   1034 ;  the caller's argument list.
                             056C   1035
       51     08 AC      D0  056C   1036 130$:       MOVL    PIDADR(AP),R1               ; Get PIDADR from argument list
                 F6      13  0570   1037             BEQL    110$                        ; If not there, can't be wild card mode
                             0572   1038             IFNORD  #2,2(R1),110$               ; Don't repeat if cannot read parameter
              02 A1      B5  0579   1039             TSTW    2(R1)                       ; Look at wild card indicator
                 EA      18  057C   1040             BGEQ    110$                        ; Must be negative for wild card mode
            5E   5D      D0  057E   1041             MOVL    FP,SP                       ; Restore SP to its value on entry
              FA7E'      31  0581   1042             BRW     EXE$GETJPI + 2              ;  and go back to the beginning.
                             0584   1043
                             0584   1044             .DISABLE        LOCAL_BLOCK
                             0584   1045
```

```
0584            .SBTTL  CHECKITEM - Validate item identifier
0584
0584    ;++
0584    ;
0584    ; FUNCTIONAL DESCRIPTION:
0584    ;
0584    ;       Routine to validate item identifier and return information
0584    ;       about the item.
0584    ;
0584    ; CALLING SEQUENCE:
0584    ;
0584    ;       JSB/BSB
0584    ;
0584    ; INPUTS:
0584    ;
0584    ;       R1 = item identifier
0584    ;       R9 = Target PCB address
0584    ;
0584    ; IMPLICIT INPUTS:
0584    ;
0584    ;       none
0584    ;
0584    ; OUTPUTS:
0584    ;
0584    ;       R1 = item identifier
0584    ;       R2 = structure number
0584    ;       R3 = item length
0584    ;       R4 = item address (actual address for PCB data, assumes current process
0584    ;               for other data) < if we're getting PHD data directly, it will
0584    ;                                 be the PHD offset, not the address >
0584    ;       R5 = item type code
0584    ;
0584    ; IMPLICIT OUTPUTS:
0584    ;
0584    ;       none
0584    ;
0584    ; ROUTINE VALUE:
0584    ;
0584    ;       R0 low bit clear -> successful return
0584    ;       R0 low bit set -> invalid item identifier
0584    ;
0584    ; SIDE EFFECTS:
0584    ;
0584    ;       none
0584    ;--
```

```
                               0584 CHECKITEM:
                   50    D4    0584             CLRL     R0                     ; assume error
               53  51    9A    0586             MOVZBL   R1,R3                  ; get item number
       52  51  08  08    EF    0589             EXTZV    #8,#8,R1,R2            ; get structure number
                   1F    13    058E             BEQL     79$                    ; error if structure number zero
               06  52    91    0590             CMPB     R2,#JPI$C_MAXSTRUC     ; structure number valid?
                   1A    1A    0593             BGTRU    79$                    ; error if not
        FA64 CF42  53    91    0595             CMPB     R3,MAXCOUNT-1[R2]      ; check max item values (1 origin)
                   12    1A    059B             BGTRU    79$                    ; error if illegal item number
                   54    D4    059D             CLRL     R4                     ; assume zero base
                               059F             CASE     R2,<-                  ; case on structure base
                               059F                              10$,-        ; ADR
                               059F                              50$,-        ; CTL
                               059F                              20$,-        ; PCB
                               059F                              30$,-        ; PHD
                               059F                              100$,-       ; PCBFLD
                               059F                              110$-        ; PHDFLD
                               059F                              >B,#1
                               05AF
                               05AF 79$:
                 00BB    31    05AF             BRW      80$                    ; CASE out of bounds - return
                               05B2
                               05B2 10$:                                        ; item is an address
       54   FA4F CF43    DE    05B2             MOVAL    ADRTBL[R3],R4          ; address is table address
               54  53    C0    05B8             ADDL     R3,R4                  ; base+indexvalue*5
           52   04 A4    9A    05BB             MOVZBL   4(R4),R2               ; get structure type code
               53  04    D0    05BF             MOVL     #4,R3                  ; size of data is four bytes
               55  00    9A    05C2             MOVZBL   #VALUE,R5              ; item is a value
                 00A3    31    05C5             BRW      70$                    ; all done
                               05C8 20$:                                        ; item is from PCB
               54  59    D0    05C8             MOVL     R9,R4                  ; get back PCB address
         55   FAB8 CF    DE    05CB             MOVAL    PCBTBL,R5              ; get address of PCB item table
                   15    11    05D0             BRB      40$                    ; continue
                               05D2 30$:                                        ; item is from process header
      00000000'EF  59    D1    05D2             CMPL     R9,SCH$GL_CURPCB       ; is the target process our own?
                   07    12    05D9             BNEQ     35$                    ; NEQ means it's not, don't touch CTL
       54  00000000'9F   D0    05DB             MOVL     @#CTL$GL_PHD,R4        ; get process header address
         55   FB5F CF    DE    05E2 35$:        MOVAL    PHDTBL,R5              ; get address of PHD item table
                               05E7 40$:
               53  05    C4    05E7             MULL     #5,R3                  ; each element is 5 bytes long
               53  55    C0    05EA             ADDL     R5,R3                  ; compute address in item table
               55  83    3C    05ED             MOVZWL   (R3)+,R5               ; get offset into data structure
                               05F0
                   3E    11    05F0             BRB      60$                    ;
                               05F2 100$:                                       ; item is from PCBFLD
               54  59    D0    05F2             MOVL     R9,R4                  ; get back PCB address
               53  07    C4    05F5             MULL     #7,R3                  ; each element is 7 bytes long
       53   FBD4 CF43    9E    05F8             MOVAB    PCBFLDTBL[R3],R3       ; get address of PCBFLD item
                   19    11    05FE             BRB      120$                   ; continue
                               0600 110$:                                       ; item is from PHDFLD
      00000000'EF  59    D1    0600             CMPL     R9,SCH$GL_CURPCB       ; is the target process our own?
                   07    12    0607             BNEQ     115$                   ; NEQ means not, don't touch CTL
       54  00000000'9F   D0    0609             MOVL     @#CTL$GL_PHD,R4        ; get process header address
               53  07    C4    0610 115$:       MULL     #7,R3                  ; each element is 7 bytes long
       53   FBB9 CF43    9E    0613             MOVAB    PHDFLDTBL[R3],R3       ; get address of PHDFLD item
                               0619 120$:
               55  83    3C    0619             MOVZWL   (R3)+,R5               ; get offset into data structure
           E8 AD    83    3C  061C              MOVZWL  (R3)+,BITDEF(FP)          ; save the BITSIZ and BITPOS
                              0620
                    0E    11  0620              BRB     60$                       ;
                              0622   50$:                                         ; item is in control region
              53    07    C4  0622              MULL    #7,R3                     ; compute index into item table
                    54    D4  0625              CLRL    R4                        ; assume zero base value
      53    F9E4 CF43     9E  0627              MOVAB   CTLTBL[R3],R3             ; get address of item information
              55    83    D0  062D              MOVL    (R3)+,R5                  ; get item address
         52    02 A3      9A  0630   60$:       MOVZBL  2(R3),R2                  ; fetch actual structure type
              52    07    91  0634              CMPB    #JPI$C_JIBTYPE,R2         ; is it the JIB?
                    05    12  0637              BNEQ    65$                       ; br if not
       54     0080 C9     D0  0639              MOVL    PCB$L_JIB(R9),R4          ; else get address of JIB
              54    55    C0  063E   65$:       ADDL    R5,R4                     ; form complete address
              55    83    9A  0641              MOVZBL  (R3)+,R5                  ; get item type code
              55    04    D1  0644              CMPL    #DSTRING,R5               ; is it a string descriptor?
                    06    12  0647              BNEQ    67$                       ; NEQ means nope
           FC AD    04    C8  0649              BISL2   #<1@JPI_V_STRDSC>,FLAGS(FP) ; it's special, flag it
                    04    11  064D              BRB     69$
           FC AD    04    CA  064F   67$:       BICL2   #<1@JPI_V_STRDSC>,FLAGS(FP) ; not special, clear flag
              53    63    9A  0653   69$:       MOVZBL  (R3),R3                   ; get item length
              52    05    91  0656              CMPB    #JPI$C_PCBFLDTYPE,R2      ; is it a bit field?
                    0E    13  0659              BEQL    90$                       ; EQL means it is
              52    06    91  065B              CMPB    #JPI$C_PHDFLDTYPE,R2      ; is it s bit field?
                    0B    12  065E              BNEQ    70$                       ; NEQ means it's not a FLD at all
     00000000'EF    59    D1  0660              CMPL    R9,SCH$GL_CURPCB          ; is the target process our own?
                    02    12  0667              BNEQ    70$                       ; NEQ means it's not
                    03    10  0669   90$:       BSBB    EXTFLD
                    50    D6  066B   70$:       INCL    R0                        ; set successful return
                          05  066D   80$:       RSB                               ; return to caller
```

```
                              066E                   .SBTTL  EXTFLD - Extract a bitfield from a datum
                              066E
                              066E   ;++
                              066E   ;
                              066E   ; FUNCTIONAL DESCRIPTION:
                              066E   ;
                              066E   ;       Routine to fetch bitfield data from within a data cell.
                              066E   ;
                              066E   ; CALLING SEQUENCE:
                              066E   ;
                              066E   ;       JSB/BSB
                              066E   ;
                              066E   ; INPUTS:
                              066E   ;
                              066E   ;       BITDEF(FP) = BITPOS/BITSIZ fields from item table
                              066E   ;       R4 = address of cell containing data
                              066E   ;
                              066E   ; IMPLICIT INPUTS:
                              066E   ;
                              066E   ;       none
                              066E   ;
                              066E   ; OUTPUTS:
                              066E   ;
                              066E   ;       R4 = new address on stack where bit is saved
                              066E   ;
                              066E   ; IMPLICIT OUTPUTS:
                              066E   ;
                              066E   ;       none
                              066E   ;
                              066E   ; ROUTINE VALUE:
                              066E   ;
                              066E   ;       none
                              066E   ;
                              066E   ; SIDE EFFECTS:
                              066E   ;
                              066E   ;       none
                              066E   ;--
                              066E
                              066E   EXTFLD:
                     0C   BB  066E           PUSHR   #^M<R2,R3>                      ; get some room
 52   E8 AD   05     0B   EF  0670           EXTZV   #11,#5,BITDEF(FP),R2            ; get BITSIZ-1
                     52   D6  0676           INCL    R2                              ; make it BITSIZ
 53   E8 AD   0B     00   EF  0678           EXTZV   #0,#11,BITDEF(FP),R3            ; get BITPOS
 E8 AD   64   52     53   EF  067E           EXTZV   R3,R2,(R4),BITDEF(FP)           ; get the bitfield
           54   E8   AD   DE  0684           MOVAL   BITDEF(FP),R4                   ; point to it
                     0C   BA  0688           POPR    #^M<R2,R3>                      ; restore the registers
                     05       068A           RSB
```

```
068B          .SBTTL  MOVEIT - Move data to user's buffer
068B
068B    ;++
068B    ;
068B    ; FUNCTIONAL DESCRIPTION:
068B    ;
068B    ;         Move the requested data to user buffer. Zero fill to end of buffer.
068B    ;         Return actual data length to user.  Assumes user's buffer has
068B    ;         been probed.
068B    ;
068B    ; CALLING SEQUENCE:
068B    ;
068B    ;         JSB/BSB
068B    ;
068B    ; INPUTS:
068B    ;
068B    ;         R1 = item identifier
068B    ;         R2 = data structure number
068B    ;         R3 = item length
068B    ;         R4 = item address
068B    ;         R5 = item type code
068B    ;         R6 = user buffer length
068B    ;         R7 = user buffer address
068B    ;         R8 = address to return length
068B    ;         R11 = PID of process to get data from
068B    ;
068B    ; IMPLICIT INPUTS:
068B    ;
068B    ;         none
068B    ;
068B    ; OUTPUTS:
068B    ;
068B    ;         none
068B    ;
068B    ; IMPLICIT OUTPUTS:
068B    ;
068B    ;         none
068B    ;
068B    ; ROUTINE VALUE:
068B    ;
068B    ;         R0 low bit set -> success
068B    ;         R0 low bit clear -> access violation on write of length
068B    ;
068B    ; SIDE EFFECTS:
068B    ;
068B    ;         Registers R1-R4 destroyed
068B    ;--
```

```
                                  068B  1276 MOVEIT:
                                  068B  1277
                                  068B  1278 ;
                                  068B  1279 ; Call routine to check for special conditions
                                  068B  1280 ;
                                  068B  1281
                      00A0   30   068B  1282         BSBW    CHECK_SPC
                                  068E  1283
                                  068E  1284 ;
                                  068E  1285 ; Check for counted string, and find actual length if so.
                                  068E  1286 ;
                                  068E  1287
                  55   02    D1   068E  1288         CMPL    #CSTRING,R5                  ; is this special string?
                       03    12   0691  1289         BNEQ    10$                          ; branch if not
                  53   84    9A   0693  1290         MOVZBL  (R4)+,R3                     ; get length and skip length byte
                                  0696  1291 ;
                                  0696  1292 ; Check that process still exists.  This assures that data address is good.
                                  0696  1293 ;
                  50   5B    3C   0696  1294 10$:    MOVZWL  R11,R0                       ; get process ID index
      50   00000000'FF40     D0   0699  1295         MOVL    @SCH$GL_PCBVEC[R0],R0        ; get PCB address
              5B    60 A0    D1   06A1  1296         CMPL    PCB$L_PID(R0),R11            ; same PID?
                       02    13   06A5  1297         BEQL    15$                          ; branch if yes
                       53    D4   06A7  1298         CLRL    R3                           ; else, zero data size
                                  06A9  1299 ;
                                  06A9  1300 ; Move the data
                                  06A9  1301 ;
                       28    BB   06A9  1302 15$:    PUSHR   #^M<R3,R5>                   ; save needed registers from movc
   67   56   00   64   53    2C   06AB  1303         MOVC5   R3,(R4),#0,R6,(R7)           ; move data to user's buffer, zero fill
                       28    BA   06B1  1304         POPR    #^M<R3,R5>                   ; restore registers
                       58    D5   06B3  1305         TSTL    R8                           ; did caller want return length?
                       11    13   06B5  1306         BEQL    30$                          ; branch if not
                                  06B7  1307         IFNOWRT #2,(R8),40$                  ; exit if word not writable
                  56   53    D1   06BD  1308         CMPL    R3,R6                        ; see how much was moved
                       03    15   06C0  1309         BLEQ    20$                          ; use valid data length if it fit
                  53   56    D0   06C2  1310         MOVL    R6,R3                        ; else give him "too short" buffer size
                  68   53    B0   06C5  1311 20$:    MOVW    R3,(R8)                      ; return length to user
                  50   01    3C   06C8  1312 30$:    MOVZWL  #SS$_NORMAL,R0               ; set success code
                       05         06CB  1313         RSB
                                  06CC  1314
                  50   0C    3C   06CC  1315 40$:    MOVZWL  #SS$_ACCVIO,R0               ; couldn't stuff RETLEN cell
                       05         06CF  1316         RSB                                  ; return
```

```
                           06D0            .SBTTL  MOVEPHD - Fetch data from other process' PHD
                           06D0
                           06D0       ;++
                           06D0       ;
                           06D0       ; FUNCTIONAL DESCRIPTION:
                           06D0       ;
                           06D0       ;        Disable interrupts and fetch the data from the other process' header.
                           06D0       ;        Always get WSLIST in case it'll be needed. Make R4 point to the
                           06D0       ;        saved data, and call EXTFLD if it's a PHDFLDTYPE item.
                           06D0       ;
                           06D0       ; CALLING SEQUENCE:
                           06D0       ;
                           06D0       ;        JSB/BSB
                           06D0       ;
                           06D0       ; INPUTS:
                           06D0       ;
                           06D0       ;        R1 = item identifier
                           06D0       ;        R2 = data structure number
                           06D0       ;        R3 = item length
                           06D0       ;        R4 = offset into other process' PHD
                           06D0       ;        R5 = item type code
                           06D0       ;        R6 = user buffer length
                           06D0       ;        R7 = user buffer address
                           06D0       ;        R8 = address to return length
                           06D0       ;        R11 = PID of process to get data from
                           06D0       ;
                           06D0       ; IMPLICIT INPUTS:
                           06D0       ;
                           06D0       ;        none
                           06D0       ;
                           06D0       ; OUTPUTS:
                           06D0       ;
                           06D0       ;        none
                           06D0       ;
                           06D0       ; IMPLICIT OUTPUTS:
                           06D0       ;
                           06D0       ;        none
                           06D0       ;
                           06D0       ; ROUTINE VALUE:
                           06D0       ;
                           06D0       ;        SS$_ACCVIO - routine MOVEIT couldn't stuff RETLEN
                           06D0       ;        SS$_NONEXPR - got into MOVEPHD and DELPEN was set
                           06D0       ;        SS$_NORMAL - everything fine - got the data
                           06D0       ;        0 - got into MOVEPHD and PHD had gone away - get with sKAST
                           06D0       ;
                           06D0       ; SIDE EFFECTS:
                           06D0       ;
                           06D0       ;        Registers R1-R4 destroyed
                           06D0       ;--
                           06D0
                           06D0      MOVEPHD:
             7E    55   7D 06D0              MOVQ    R5,-(SP)                  ; save R5 and R6
              000006D3     06D3      LOCK_BEGIN = .
                           06D3              DSBINT  LOCK_IPL                  ; raise IPL to Synch and lock code
             50    5B   3C 06DD              MOVZWL  R11,R0                    ; get process ID index
   50   00000000'FF40   D0 06E0              MOVL    @SCH$GL_PCBVEC[R0],R0     ; get PCB address
          5B    60 A0   D1 06E8              CMPL    PCB$L_PID(R0),R11         ; same PID?
```

```
             37  12 06EC              BNEQ    90$                     ; NEQ means not the same
    32 24 A0 12  E1 06EE              BBC     #PCB$V_PHDRES,PCB$L_STS(R0),90$ ; if the PHD isn't there, exit
    26 24 A0 01  E0 06F3              BBS     #PCB$V_DELPEN,PCB$L_STS(R0),85$ ; if process will go away, exit
       55    6C A0  D0 06F8           MOVL    PCB$L_PHD(R0),R5        ; get the header address
    F4 AD    08 A5  3C 06FC           MOVZWL  PHD$W_WSLIST(R5),WSLIST(FP) ; save the WSLIST just in case
     56    55    54  C1 0701          ADDL3   R4,R5,R6                ; PHD offset + PHD address => R6
       EC AD    66  7D 0705           MOVQ    (R6),PHDTEMP(FP)        ; save the data from the PHD
                       0709           ENBINT                          ; allow interrupts again
       54    EC AD  DE 070C           MOVAL   PHDTEMP(FP),R4          ; point to the saved data
             55  8E  7D 0710          MOVQ    (SP)+,R5                ; restore R5 and R6
             52  06  91 0713          CMPB    #JPI$C_PHDFLDTYPE,R2    ; is it a bit field?
                 03  12 0716          BNEQ    30$                     ; NEQ means it is not
               FF53  30 0718          BSBW    EXTFLD                  ; extract out the bitfield
               FF6D  31 071B    30$:  BRW     MOVEIT                  ; now "fetch" the data the normal way
                        071E
       50   08E8 8F  3C 071E    85$:  MOVZWL  #SS$_NONEXPR,R0         ; process going away
                 02  11 0723          BRB     95$
                        0725
                 50  D4 0725    90$:  CLRL    R0                      ; PHD not resident anymore
                        0727    95$:  ENBINT
             55    8E  7D 072A        MOVQ    (SP)+,R5                ; clean off the stack, restore R5,R6
                       05 072D        RSB
                          072E
```

```
                              072E        .SBTTL  SPECIAL - Handle special conditions
                              072E
                              072E  ;++
                              072E  ;
                              072E  ; FUNCTIONAL DESCRIPTION:
                              072E  ;
                              072E  ;       These routines handle data items which must be transformed
                              072E  ;       before they are returned to the user.  Generally, some
                              072E  ;       transformation is applied to the data item and the newly
                              072E  ;       computed item is stored in LOCAL_SPACE on the stack.
                              072E  ;       The handling routine then changes R4 to point to LOCAL_SPACE
                              072E  ;       so that MOVEIT will move the item from local storage.
                              072E  ;
                              072E  ; CALLING SEQUENCE:
                              072E  ;
                              072E  ;       JSB/BSB
                              072E  ;
                              072E  ; INPUTS:
                              072E  ;
                              072E  ;       R1 = item identifier
                              072E  ;       R3 = item length
                              072E  ;       R4 = item address
                              072E  ;       R6 = user's buffer length
                              072E  ;       R9 = PCB address of target process
                              072E  ;
                              072E  ; IMPLICIT INPUTS:
                              072E  ;
                              072E  ;       none
                              072E  ;
                              072E  ; OUTPUTS:
                              072E  ;
                              072E  ;       none
                              072E  ;
                              072E  ; IMPLICIT OUTPUTS:
                              072E  ;
                              072E  ;       none
                              072E  ;
                              072E  ; ROUTINE VALUE:
                              072E  ;
                              072E  ;       none
                              072E  ;
                              072E  ; SIDE EFFECTS:
                              072E  ;
                              072E  ;       none
                              072E  ;--
                              072E
                              072E  CHECK_SPC:
                              072E
                              072E  ;
                              072E  ; Registers R5 and R6 are saved at this level and may be used by
                              072E  ; the action routines without being saved.  Action routines are JSB'ed
                              072E  ; to with R5 containing the address of LOCAL_SPACE on the stack.
                              072E  ;
                              072E
           7E   55       7D   072E          MOVQ    R5,-(SP)                ; save registers
           55   0C       D0   0731          MOVL    #SPECIAL_LEN,R5         ; get number of table entries
     56  FA99   CF       DE   0734          MOVAL   SPECIAL,R6              ; get address of table
```

```
                                0739  1456 10$:
                   86    51  B1 0739  1457          CMPW    R1,(R6)+                  ; does entry match item?
                         08  13 073C  1458          BEQL    20$                       ; yes, go handle it
                   56    04  C0 073E  1459          ADDL    #4,R6                     ; skip handler address
                      F5 55  F5 0741  1460          SOBGTR  R5,10$                    ; scan rest of table
                         06  11 0744  1461          BRB     30$                       ; nothing to do, exit
                                0746  1462 20$:
              55      D8 AD  DE 0746  1463          MOVAL   LOCAL_SPACE(FP),R5        ; load local address for action routine
                         96  16 074A  1464          JSB     @(R6)+                    ; call action routine
                                074C  1465 30$:
                   55    8E  7D 074C  1466          MOVQ    (SP)+,R5                  ; restore registers
                             05 074F  1467          RSB
                                0750  1468 ;
                                0750  1469 ; Data handling routines
                                0750  1470 ;
                                0750  1471
                                0750  1472 ;
                                0750  1473 ; Internal priority must be subtracted from 31 before being returned.
                                0750  1474 ;
                                0750  1475
                                0750  1476 SPC_PRI:
         65      1F      64  83 0750  1477          SUBB3   (R4),#31,(R5)             ; compute external priority
                   54    55  D0 0754  1478          MOVL    R5,R4                     ; change address for move routine
                             05 0757  1479          RSB
                                0758  1480
                                0758  1481 ;
                                0758  1482 ; Check the PCB$L_STS bits to get the mode
                                0758  1483 ;
                                0758  1484
                                0758  1485 SPC_MODE:
              56      24 A9  D0 0758  1486          MOVL    PCB$L_STS(R9),R6          ; the bits are in the STS
                   65    01  D0 075C  1487          MOVL    #JPI$K_NETWORK,(R5)       ; assume network
                11 56    15  E0 075F  1488          BBS     #PCB$V_NETWRK,R6,10$      ; if set, all done
                   65    02  D0 0763  1489          MOVL    #JPI$K_BATCH,(R5)         ; now try batch mode
                0A 56    0E  E0 0766  1490          BBS     #PCB$V_BATCH,R6,10$
                   65    03  D0 076A  1491          MOVL    #JPI$K_INTERACTIVE,(R5)   ; now try interactive mode
                03 56    19  E0 076D  1492          BBS     #PCB$V_INTER,R6,10$
                   65    00  D0 0771  1493          MOVL    #JPI$K_OTHER,(R5)         ; it must be 'other' mode
                   54    55  D0 0774  1494 10$:     MOVL    R5,R4                     ; point at the 'data'
                             05 0777  1495          RSB
                                0778  1496
                                0778  1497 ;
                                0778  1498 ; Working set pointers are indices into working set list
                                0778  1499 ; and must be subtracted from first list element.
                                0778  1500 ;
                                0778  1501
                                0778  1502 SPC_WORKSET:
      00000000'EF        59  D1 0778  1503          CMPL    R9,SCH$GL_CURPCB          ; is the target process our own?
                         07  13 077F  1504          BEQL    15$                       ; EQL means it is
       65     64      F4 AD  A3 0781  1505          SUBW3   WSLIST(FP),(R4),(R5)      ; use the saved WSLIST
                         0C  11 0786  1506          BRB     17$                       ; don't touch CTL here
      56     00000000'9F     D0 0788  1507 15$:     MOVL    @#CTL$GL_PHD,R6           ; get process header address
       65     64      08 A6  A3 078F  1508          SUBW3   PHD$W_WSLIST(R6),(R4),(R5) ; compute argument size
                   54    55  D0 0794  1509 17$:     MOVL    R5,R4                     ; change item address
                         64  B6 0797  1510          INCW    (R4)                      ; must add one to index
                             05 0799  1511          RSB
                                079A  1512
```

```
                          079A   ;
                          079A   ; Convert the pcb vector index into a process index.  This is so that applications
                          079A   ; which used to use the low word of the PID as an index can adapt.  This item,
                          079A   ; JPI$_PROC_INDEX, is very similar to the current pix:
                          079A   ;
                          079A   ;         PROC_INDEX is a number between 1 and the sysgen parameter MAXPROCESSCNT.
                          079A   ;         This means that it is a small number, and that it is practical to
                          079A   ;         statically allocate bitvectors or other vectors for the maximum
                          079A   ;         number of processes expected.
                          079A   ;
                          079A   ;         At any instant, no more than one process will have a given PROC_INDEX.
                          079A   ;         In particular, no other process will have the same PROC_INDEX as you.
                          079A   ;         This guarantees no collisions.  If the application wanted to know
                          079A   ;         about reuse of the PROC_INDEX, it could store the EPID in the
                          079A   ;         vector and do its own check.
                          079A   ;
                          079A   ; No program should assume that PROC_INDEX is anything more than this.  Note that
                          079A   ; the sole purpose of the following arithmetic is to make sure that PROC_INDEX is
                          079A   ; NOT the pcb vector index!
                          079A   ;
                          079A   SPC_PROC_INDEX:
           56       64  3C 079A          MOVZWL  (R4),R6                  ; Get the PIX into a register
   65 00000000'EF       3C 079D          MOVZWL  SGN$GW_MAXPRCCT,(R5)     ; Move the MAXPROCESSCNT into the scratch
           65       56  C2 07A4          SUBL2   R6,(R5)                  ; Convert 0 to N-1 to range N to 1
           54       55  D0 07A7          MOVL    R5,R4                    ; Point at the new value for move routine
                        05 07AA          RSB
                           07AB
                           07AB
                           07AB  ;
                           07AB  ; Convert the MPID from the JIB to extended format.
                           07AB  ;
                           07AB  ; Inputs:
                           07AB  ;
                           07AB  ;         R4 = Addr of MPID in internal format
                           07AB  ;         R5 = Addr of scratch buffer
                           07AB  ;
                           07AB  ; Outputs:
                           07AB  ;
                           07AB  ;         R4 = Addr. MPID in extended format
                           07AB  ;
                           07AB
                           07AB  SPC_MASTER_PID:
           50       64  D0 07AB          MOVL    (R4),R0                  ; get MPID
      00000000'GF       16 07AE          JSB     G^EXE$IPID_TO_EPID       ; convert it to extended format
           65       50  D0 07B4          MOVL    R0,(R5)                  ; store converted PID in scratch buffer
           54       55  D0 07B7          MOVL    R5,R4                    ; point to the converted PID
                        05 07BA          RSB
```

```
                          07BB  ;
                          07BB  ; The current image file name is in the Image File Descriptor Block.  It
                          07BB  ; is also in user writable memory, so all addresses must be probed.
                          07BB  ;
                          07BB  ; Inputs:
                          07BB  ;         R4 = CTL$GL_IMGHDRBF, address of image header buffer
                          07BB  ;
                          07BB  ; Outputs:
                          07BB  ;         R3 = size of image file name
                          07BB  ;         R4 = address of image file name
                          07BB  ;
                          07BB
                          07BB EXE$CHKIMAGNAME::
         54     64   D0   07BB            MOVL    (R4),R4                  ; get address of image header buffer
                19   13   07BE            BEQL    11$                      ; if EQL, no image active
      55     04 A4   D0   07C0            MOVL    4(R4),R5                 ; get address of image file descriptor
                          07C4            IFNORD  #8,IFD$Q_CURPROG(R5),11$,#PSL$C_USER ; check access to desc
      53     14 A5   7D   07CB            MOVQ    IFD$Q_CURPROG(R5),R3     ; get image name descriptor
         53     53   3C   07CF            MOVZWL  R3,R3                    ; assure size of string is in range
                          07D2            IFNORD  R3,(R4),11$,#PSL$C_USER  ; check access to string
                     05   07D8            RSB                              ;
                          07D9 11$:
                53   7C   07D9            CLRQ    R3                       ; zero string descriptor
                     05   07DB            RSB                              ;
                          07DC
                          07DC  ;
                          07DC  ; The current image file name is in the Image File Descriptor Block.
                          07DC  ; Probe it for maximum protection.
                          07DC  ;
                          07DC  ; If a compatibility mode exception handler has been declared for the
                          07DC  ; process, assume that an AME is running. Further assume that the second
                          07DC  ; compatibility mode context page has been patterned enough after the
                          07DC  ; image file descriptor block such that an alternate image name can be
                          07DC  ; found there. In this case, return that image name. If the name is null,
                          07DC  ; fall back to the name in the Image File Descriptor Block. Note that the
                          07DC  ; second compatibility mode context page is user writeable, so it must
                          07DC  ; be probed.
                          07DC  ;
                          07DC  ; Inputs:
                          07DC  ;         R4 = CTL$GL_IMGHDRBF, address of image header buffer
                          07DC  ;         8(SP) = user's buffer length
                          07DC  ;
                          07DC  ; Outputs:
                          07DC  ;         R3 = size of image file name
                          07DC  ;         R4 = address of image file name
                          07DC  ;
                          07DC
                          07DC SPC_IMAGNAME:
      56     08 AE   D0   07DC            MOVL    8(SP),R6                 ; get the user's buffer length
         54     64   D0   07E0            MOVL    (R4),R4                  ; get address of image header buffer
                43   13   07E3            BEQL    10$                      ; if EQL, no image active
         00000000'GF   D5   07E5          TSTL    G^CTL$GL_CMHANDLR        ; is there an AME running?
                1A   13   07EB            BEQL    5$                       ; if EQL no, use image in IFD
  55     00000200'GF   DE   07ED          MOVAL   G^CTL$AL_CMCNTX+^X200,R5 ; point to second c-mode context page
                          07F4            IFNORD  #8,IFD$Q_CURPROG(R5),10$,#PSL$C_USER ; check access to desc
      53     14 A5   D0   07FB            MOVL    IFD$Q_CURPROG(R5),R3     ; get length of image name string
                06   13   07FF            BEQL    5$                       ; if EQL, string is null, get from IFD
```

```
54    18 A5   D0  0801             MOVL    IFD$Q_CURPROG+4(R5),R4         ; get address of string
         0F   11  0805             BRB     6$                             ; join common code
                  0807
55    04 A4   D0  0807  5$:        MOVL    4(R4),R5                       ; get address of image file descriptor
                  080B             IFNORD  #8,IFD$Q_CURPROG(R5),10$,#PSL$C_USER ; check access to desc
53    14 A5   7D  0812             MOVQ    IFD$Q_CURPROG(R5),R3           ; get image name descriptor
   53    53   3C  0816  6$:        MOVZWL  R3,R3                          ; assure size of string is in range
   56    53   D1  0819             CMPL    R3,R6                          ; is it bigger than the user's buffer?
         03   15  081C             BLEQ    7$
   53    56   D0  081E             MOVL    R6,R3                          ; yes, make user's size the real size
                  0821  7$:        IFNORD  R3,(R4),10$,#PSL$C_USER        ; check access to string
              05  0827             RSB                                    ;
                  0828  10$:
         53   7C  0828             CLRQ    R3                             ; zero string descriptor
              05  082A             RSB                                    ;
                  082B
                  082B  LOCK_IPL:
              08  082B             .BYTE   IPL$_SYNCH
                  082C  LOCK_END:
                  082C             ASSUME  <LOCK_END-LOCK_BEGIN> LE 512
```

```
                        082C              .SBTTL  MOVEFU - Move data from user to system buffer
                        082C      ;++
                        082C      ;
                        082C      ; FUNCTIONAL DESCRIPTION:
                        082C      ;
                        082C      ;       This routine is entered as the result of a special kernel AST
                        082C      ;       generated by a process requesting information through $GETJPI
                        082C      ;       on another process.  MOVEFU is passed control information and
                        082C      ;       the item list in the AST packet.  Also chained into the AST
                        082C      ;       packet is another packet for returning the data.  This packet
                        082C      ;       is returned by issuing a special kernel AST to the process
                        082C      ;       requesting the information, to the label MOVETU in GETJPI.
                        082C      ;
                        082C      ; CALLING SEQUENCE:
                        082C      ;
                        082C      ;       JSB (as the result of a special kernel AST)
                        082C      ;
                        082C      ; INPUTS:
                        082C      ;
                        082C      ;       R0:R3 - scratch
                        082C      ;       R4 - PCB ADDRESS
                        082C      ;       R5 - AST control block address
                        082C      ;       Control block (see below)
                        082C      ;
                        082C      ; OUTPUTS:
                        082C      ;
                        082C      ;       None
                        082C      ;
                        082C      ; ROUTINE VALUE:
                        082C      ;
                        082C      ;       None
                        082C      ;
                        082C      ; SIDE EFFECTS:
                        082C      ;
                        082C      ;       If the process requesting information still exists, a special
                        082C      ;       kernel AST is issued to address MOVETU to process the filled
                        082C      ;       information packets.
                        082C      ;
                        082C      ;--
                        082C              .enable lsb
                        082C      MOVEFU:
        2FF0 8F    BB   082C              PUSHR   #^M<R4,R5,R6,R7,R8,R9,R10,R11,FP>
        5D    5E   D0   0830              MOVL    SP,FP                   ; set address of local storage
     5E    D8 AE   DE   0833              MOVAL   LOCAL_SPACE(SP),SP      ; allocate local storage
  0C A5    28 A5   D0   0837              MOVL    ACB_L_OPID(R5),ACB$L_PID(R5) ; turn the block around
18 A5    08C1'CF   9E   083C              MOVAB   W^MOVETU,ACB$L_KAST(R5) ; new AST routine
  0B A5    80 8F   88   0842              BISB2   #<1@ACB$V_KAST>,ACB$B_RMOD(R5)  ; set special kernel bit again
     5A    30 A5   D0   0847              MOVL    ACB_L_COUNT(R5),R10     ; get item count
     5B    1C A5   D0   084B              MOVL    ACB_L_DADDR(R5),R11     ; get data block address
     57    34 A5   DE   084F              MOVAL   ACB_L_ILIST(R5),R7      ; point to start of item list
        59    54   D0   0853              MOVL    R4,R9                   ; setup for call to CHECKITEM
                        0856      ;
                        0856      ; Loop through item descriptor list, moving data to the system buffer
                        0856      ;
        56    87   3C   0856      10$:    MOVZWL  (R7)+,R6                ; get user buffer size
     EC AD    57   D0   0859              MOVL    R7,PHDTEMP(FP)          ; save address of item identifier
        51    87   3C   085D              MOVZWL  (R7)+,R1                ; item identifier
```

```
                      FD21   30  0860   1696            BSBW     CHECKITEM                      ; get address and size of item
                      FEC8   30  0863   1697            BSBW     CHECK_SPC                      ; check for special types
                 55     02   D1  0866   1698            CMPL     #CSTRING,R5                    ; counted string?
                        03   12  0869   1699            BNEQ     20$                            ; branch if not
                 53     84   9A  086B   1700            MOVZBL   (R4)+,R3                       ; get length and skip length byte
                                 086E   1701
              EC BD     53   B0  086E   1702 20$:       MOVW     R3,@PHDTEMP(FP)                ; clobber item code with source size
     6B   56    00 64   53   2C  0872   1703            MOVC5    R3,(R4),#0,R6,(R11)            ; move data to system buffer
                 5B     53   D0  0878   1704            MOVL     R3,R11                         ; update system buffer pointer
                 57     08   C0  087B   1705            ADDL     #8,R7                          ; update item descriptor pointer
                                 087E   1706                                                    ; by skipping user buffer addresses
                     D5 5A   F5  087E   1707            SOBGTR   R10,10$                        ; decrement item count and loop
                                 0881   1708 ;
                                 0881   1709 ; We have moved all the data to the system buffer. Restore registers, and
                                 0881   1710 ; check to see if the requesting process is still active, before we queue the
                                 0881   1711 ; return kernel AST.
                                 0881   1712 ;
              5E   28 AE     DE  0881   1713            MOVAL    -LOCAL_SPACE(SP),SP           ; remove local storage from stack
                   2FF0 8F   BA  0885   1714            POPR     #^M<R4,R5,R6,R7,R8,R9,R10,R11,FP>
                                 0889   1715 30$:       SETIPL   50$                            ; raise IPL to synch, lock code.
              51   28 A5     3C  0890   1716            MOVZWL   ACB_L_OPID(R5),R1              ; old PID
        51   00000000'FF41   D0  0894   1717            MOVL     @SCH$GL_PCBVEC[R1],R1          ; PCB address associated.
           0C A5   60 A1     D1  089C   1718            CMPL     PCB$L_PID(R1),ACB$L_PID(R5)          ; same PID in both places?
                        11   12  08A1   1719            BNEQ     40$                            ; error if not.
           0C 24 A1     01   E0  08A3   1720            BBS      #PCB$V_DELPEN,PCB$L_STS(R1),40$ ; error if delete pending
                        52   D4  08A8   1721            CLRL     R2                             ; null priority increment
            00000000'EF      16  08AA   1722            JSB      SCH$QAST                       ; queue the AST
                                 08B0   1723            SETIPL   #IPL$_ASTDEL                   ; drop back to AST delivery level
                             05  08B3   1724            RSB                                     ;
                                 08B4   1725 ;
                                 08B4   1726 ; If the process did not exist, or was marked for delete, deallocate the
                                 08B4   1727 ; blocks and return.
                                 08B4   1728 ;
                                 08B4   1729 40$:       SETIPL   #IPL$_ASTDEL                   ; drop back to AST delivery level
                 50     55   D0  08B7   1730            MOVL     R5,R0                          ; get AST block address
                                 08BA   1731 DEANONPAGED:                                       ; local point for all calls to EXE$D...
            00000000'GF      17  08BA   1732            JMP      G^EXE$DEANONPAGED              ; deallocate it and exit
                                 08C0   1733 ;
                                 08C0   1734 ; The preceding code must raise IPL to synchronize access to process database,
                                 08C0   1735 ; but since it is paged it must be locked in memory. The usage of the SETIPL
                                 08C0   1736 ; macro above, both raises IPL and faults the code into memory.
                                 08C0   1737 ;
                                 08C0   1738 50$:
                             08  08C0   1739            .BYTE    IPL$_SYNCH                     ; end of locked code region
                                 08C1   1740            ASSUME   <.-30$> LE 512                 ; only 512 bytes can be locked
                                 08C1   1741            .disable lsb
                                 08C1   1742
```

```
                         08C1              .SBTTL  MOVETU - Move data from system buffer to user
                         08C1      ;++
                         08C1      ;
                         08C1      ; FUNCTIONAL DESCRIPTION:
                         08C1      ;
                         08C1      ;       MOVETU is entered as the result of a special kernel AST queued by
                         08C1      ;       the routine MOVEFU from the process we were requesting information
                         08C1      ;       from on a GETJPI system service.  The data buffer has been filled,
                         08C1      ;       and now we must move that data from the system buffer to the user.
                         08C1      ;
                         08C1      ;       Prior to storing the data, we check to see if the copy of PHD$L_IMGCNT
                         08C1      ;       that was saved in the packet is the same as that in the process header.
                         08C1      ;       If they are not equal, it means the image that issued the GETJPI service
                         08C1      ;       has exited, and a new image is in memory; we should not move the data
                         08C1      ;       to the user.
                         08C1      ;
                         08C1      ; CALLING SEQUENCE:
                         08C1      ;
                         08C1      ;       JSB (as the result of a special kernel AST)
                         08C1      ;
                         08C1      ; INPUTS:
                         08C1      ;
                         08C1      ;       R0:R3 - scratch
                         08C1      ;       R4 - PCB address
                         08C1      ;       R5 - AST control block address
                         08C1      ;       Control block data
                         08C1      ;
                         08C1      ; OUTPUTS:
                         08C1      ;
                         08C1      ;       none
                         08C1      ;
                         08C1      ; ROUTINE VALUE:
                         08C1      ;
                         08C1      ;       none
                         08C1      ;
                         08C1      ; SIDE EFFECTS:
                         08C1      ;
                         08C1      ;       Attempts to move data to user buffers, as requested by original
                         08C1      ;       GETJPI request. May cause setting of event flags, IOSB, and
                         08C1      ;       possibly an AST to the requestor.  Errors in processing result
                         08C1      ;       in an attempt to post the error status in the IOSB, if specified.
                         08C1      ;--
                         08C1      MOVETU:
                         08C1      ;
                         08C1      ; See if PHD$L_IMGCNT has what we think it has in it, and free the blocks
                         08C1      ; and exit if it doesn't; if not equal, a different image is running!
                         08C1      ;
53   00000000'EF    D0   08C1              MOVL    CTL$GL_PHD,R3                   ; get process header address
 2C A5    00F4 C3   D1   08C8              CMPL    PHD$L_IMGCNT(R3),ACB_L_IMGCNT(R5) ; see if the same thing.
             1A     13   08CE              BEQL    10$                             ; go move data if equal
   03 0B A5   06    E1   08D0              BBC     #ACB$V_QUOTA,ACB$B_RMOD(R5),5$ ; has AST quota been charged?
           38 A4    B6   08D5              INCW    PCB$W_ASTCNT(R4)                ; give it back
           50  55   D0   08D8      5$:     MOVL    R5,R0                           ; get address of AST block
    51    0080 C4   D0   08DB              MOVL    PCB$L_JIB(R4),R1                ; get address of JIB
        52   08 A0  3C   08E0              MOVZWL  ACB$W_SIZE(R0),R2               ; convert count to longword
         20 A1  52  C0   08E4              ADDL    R2,JIB$L_BYTCNT(R1)             ; restore buffer quota
                D0  11   08E8              BRB     DEANONPAGED                     ; deallocate AST block and exit
```

```
                       08EA
                       08EA  10$:
        0FF0 8F    BB  08EA        PUSHR   #^M<R4,R5,R6,R7,R8,R9,R10,R11>
   59     0B A5    9A  08EE        MOVZBL  ACB$B_RMOD(R5),R9        ; get requester's access mode
   5A     30 A5    D0  08F2        MOVL    ACB_L_COUNT(R5),R10      ; get item count
   51     1C A5    D0  08F6        MOVL    ACB_L_DADDR(R5),R1       ; get data buffer address
   56     34 A5    DE  08FA        MOVAL   ACB_L_ILIST(R5),R6       ; get starting address of the list
                       08FE  ;
                       08FE  ; Loop through item descriptor list, moving data to user buffer(s)
                       08FE  ;
                       08FE  20$:
          57  86   3C  08FE        MOVZWL  (R6)+,R7                 ; user buffer length
          58  86   3C  0901        MOVZWL  (R6)+,R8                 ; actual data length
          55  86   D0  0904        MOVL    (R6)+,R5                 ; user buffer address
                       0907  ;
                       0907  ; Check that requester still has write access to his buffer
                       0907  ;
          50  55   D0  0907        MOVL    R5,R0                    ; buffer address to R0
              51   DD  090A        PUSHL   R1                       ; Save R1
          51  57   D0  090C        MOVL    R7,R1                    ; and size to R1
          53  59   D0  090F        MOVL    R9,R3                    ; use access mode value from ACB for PROBE
     00000000'EF   16  0912        JSB     EXE$PROBEW               ; check write accessibility of buffer
             51 8ED0   0918        POPL    R1                       ; Restore R1
          22  50   E9  091B        BLBC    R0,50$                   ; get out if buffer inaccessible
                       091E  ;
                       091E  ; Now actually move the data
                       091E  ;
  65      61  57   28  091E        MOVC3   R7,(R1),(R5)             ; move data to user buffer
          50  86   D0  0922        MOVL    (R6)+,R0                 ; get address to store actual length
              11   13  0925        BEQL    40$                      ; branch if no length wanted
                       0927        IFNOWRT #2,(R0),50$,R9           ; requester still have access to buffer?
          57  58   D1  092D        CMPL    R8,R7                    ; actual data length less than user's?
              03   15  0930        BLEQ    30$                      ; branch if yes - use actual length
          58  57   D0  0932        MOVL    R7,R8                    ; use user buffer length
          60  58   B0  0935  30$:  MOVW    R8,(R0)                  ; return buffer length
          C3  5A   F5  0938  40$:  SOBGTR  R10,20$                  ; decrement item count and loop
          50  01   3C  093B        MOVZWL  #SS$_NORMAL,R0           ; set successful completion
              03   11  093E        BRB     60$                      ;
          50  0C   3C  0940  50$:  MOVZWL  #SS$_ACCVIO,R0           ; set access violation failure
                       0943  ;
                       0943  ; Restore original registers, set the event flag, and post completion status
                       0943  ;
        0FF0 8F    BA  0943  60$:  POPR    #^M<R4,R5,R6,R7,R8,R9,R10,R11>
              50   DD  0947        PUSHL   R0                       ; save status
   53     20 A5    D0  0949        MOVL    ACB_L_EFN(R5),R3         ; get event flag number
   51     60 A4    D0  094D        MOVL    PCB$L_PID(R4),R1         ; and PID for process
              52   D4  0951        CLRL    R2                       ; set null priority increment
     00000000'GF   16  0953        JSB     G^SCH$POSTEF             ; set the event flag
             50 8ED0   0959        POPL    R0                       ; restore exit status
   53     24 A5    D0  095C        MOVL    ACB_L_IOSB(R5),R3        ; possible IOSB address?
              0A   13  0960        BEQL    70$                      ; branch if none supplied
                       0962        IFNOWRT #4,(R3),70$,ACB$B_RMOD(R5) ; check if IOSB still accessable
          63  50   D0  0969        MOVL    R0,(R3)                  ; store completion status
                       096C  ; Return the BYTCNT quota to the caller
                       096C  ;
   52     08 A5    3C  096C  70$:  MOVZWL  ACB$W_SIZE(R5),R2        ; convert to longword
   51   0080 C4    D0  0970        MOVL    PCB$L_JIB(R4),R1         ; get JIB address
```

```
         20 A1    52   C0  0975  1858          ADDL    R2,JIB$L_BYTCNT(R1)          ; restore buffer quota
                           0979  1859  ;
                           0979  1860  ; If an AST was specified, queue it to caller and return.
                           0979  1861  ;
            10 A5      D5  0979  1862          TSTL    ACB$L_AST(R5)                ; is an address supplied?
               08      13  097C  1863          BEQL    80$                          ; branch if not.
               52      D4  097E  1864          CLRL    R2                           ; no priority increment
      00000000'EF      17  0980  1865          JMP     SCH$QAST                     ; queue AST to user and exit
                           0986  1866  ;
                           0986  1867  ; No AST specified, deallocate the AST control block and return.
                           0986  1868  ;
            50    55   D0  0986  1869  80$:    MOVL    R5,R0                        ; set the address of the AST block
                FF2E   31  0989  1870          BRW     DEANONPAGED                  ; deallocate the block and exit
```

```
                              098C          .SBTTL  NAMPID - Get specified process ID
                              098C  ;++
                              098C  ;
                              098C  ; FUNCTIONAL DESCRIPTION:
                              098C  ;
                              098C  ;       Routine to convert a process name to a PID and check privileges. If a
                              098C  ;       valid PID or process name is specified, the standard conversion
                              098C  ;       routine EXE$NAMPID is simply called.  If, however, a PID that implies
                              098C  ;       a "wildcard" PID (-1) is specified, then the next active process is
                              098C  ;       chosen as the process ID to pass to EXE$NAMPID.  EXE$NAMPID then
                              098C  ;       checks the requestor's privilege to obtain information about the
                              098C  ;       process and returns the process's PCB address.
                              098C  ;
                              098C  ; INPUTS:
                              098C  ;
                              098C  ;       R4 = current process PCB address
                              098C  ;       PIDADR(AP) = address of specified PID
                              098C  ;       PRCNAM(AP) = address of specified process name descriptor
                              098C  ;
                              098C  ; OUTPUTS:
                              098C  ;
                              098C  ;       R0 = success/failure of operation
                              098C  ;       R4 = current process PCB address
                              098C  ;       R9 = specified process PCB address
                              098C  ;       R11 = specified process PID
                              098C  ;       @PIDADR(AP) = specified process PID or special "wildcard" context PID
                              098C  ;--
                              098C  NAMPID:
                              098C          .ENABL  LSB
         56   08 AC     D0    098C          MOVL    PIDADR(AP),R6           ; get PID address
                 43     13    0990          BEQL    20$                     ; if eql - none
                              0992          IFNOWRT #4,(R6),50$             ; check access to PID
         50      66     D0    0998          MOVL    (R6),R0                 ; get PID
                 38     18    099B          BGEQ    20$                     ; if geq - standard extended PID
                              099D  ;
                              099D  ; "Wildcard" type PID specified
                              099D  ;
         55      50     32    099D          CVTWL   R0,R5                   ; get PIX (Process Index) from PID
   50   00000000'GF     DE    09A0          MOVAL   G^SCH$GL_NULLPCB,R0     ; special case handling for NULL proc
      FC AD      01     C8    09A7          BISL2   #<1@JPI_V_WILD>,FLAGS(FP) ; mark wildcarding in progress
                 55     B6    09AB  10$:    INCW    R5                      ; increment PIX
  00000000'EF    55     B1    09AD          CMPW    R5,SCH$GL_MAXPIX        ; is PIX in valid range?
                 56     1A    09B4          BGTRU   60$                     ; if gtru, no - no more processes
         55      55     3C    09B6          MOVZWL  R5,R5                   ; clean out top half of R5
                 11     13    09B9          BEQL    15$                     ; if eql, this indicates actual NULL proc
  50   00000000'FF45    D0    09BB          MOVL    @SCH$GL_PCBVEC[R5],R0   ; get PCB address
  00000000'8F    50     D1    09C3          CMPL    R0,#SCH$GL_NULLPCB      ; unused process slot?
                 DF     13    09CA          BEQL    10$                     ; if eql, yes - try next one
   DA 24 A0      01     E0    09CC  15$:    BBS     #PCB$V_DELPEN,PCB$L_STS(R0),10$ ; also get next one if this
                              09D1                                          ; one is going away
         66   64 A0     D0    09D1          MOVL    PCB$L_EPID(R0),(R6)     ; store extended PID in argument list
                              09D5  ;
                              09D5  ; Convert process name to PID, if specified, and check privileges
                              09D5  ;
         5C      04     C0    09D5  20$:    ADDL    #4,AP                   ; make PIDADR top argument
     00000000'EF        16    09D8          JSB     25$                     ; get into nonpaged code
                              09DE          .SAVE_PSECT                     ; save current .PSECT context
```

```
                      09DE
                      09DE      ;  The reason for jumping to the nonpaged exec rather than dynamically
                      09DE      ;  locking down pageable pages is that EXE$NAMPID cannot be entered
                      09DE      ;  above IPL 2 and the dynamic locking would cause that to happen. The
                      09DE      ;  reason that EXE$NAMPID must be entered at IPL 2 or lower is that it
                      09DE      ;  touches the caller's argument list (which contains arguments that
                      09DE      ;  could fault) and page faults are not allowed above IPL 2.
                      09DE
                  00000000                .PSECT   AEXENONPAGED                 ; EXE$NAMPID returns at IPL$_SYNCH
          FFFD'   30  0000      25$:      BSBW     EXE$NAMPID                   ; get PCB address and check privileges
                      0003                SETIPL   #0                           ; restore IPL - PCB is no longer locked
                  05  0006                RSB                                   ; go back to paged code
                      0007
                  000009DE                .RESTORE_PSECT                        ; get paged .PSECT context back
       5C    04   C2  09DE                SUBL     #4,AP                        ; restore argument pointer
       5B    51   D0  09E1                MOVL     R1,R11                       ; save PID
 07 FC AD    00   E1  09E4                BBC      #JPI_V_WILD,FLAGS(FP),30$ ; "wildcard" PID specified?
       66    51   B0  09E9                MOVW     R1,(R6)                      ; restore process index context
    02 A6    01   AE  09EC                MNEGW    #1,2(R6)                     ; else, set continuation context
                      09F0      ;
                      09F0      ; Check PID address and return
                      09F0      ;
       0C 50      E9  09F0      30$:      BLBC     R0,40$                       ; branch if error
       59    54   D0  09F3                MOVL     R4,R9                        ; save PCB address
       00'   5B   B1  09F6                CMPW     R11,S^#SCH$C_SWPPIX          ; is PID a normal one?
             04   1A  09F9                BGTRU    40$                          ; if GTRU, yes
    FC AD    02   C8  09FB                BISL2    #<1@JPI_V_NULLSWAP>,FLAGS(FP) ; indicate Null or Swapper process
54   00000000'EF  D0  09FF      40$:      MOVL     SCH$GL_CURPCB,R4             ; restore current PCB address
             05       0A06                RSB
                      0A07
       50    0C   3C  0A07      50$:      MOVZWL   #SS$_ACCVIO,R0               ; set access violation
             F3   11  0A0A                BRB      40$                          ;
   50   09A8 8F   3C  0A0C      60$:      MOVZWL   #SS$_NOMOREPROC,R0           ; set no more processes
             EC   11  0A11                BRB      40$                          ;
                      0A13
                      0A13                .END
```

```
$$T1                    = 00000000              EXE$NAMPID                ********    X   04
$XX$                    = 00000000              EXE$PROBEW                ********    X   02
ACB$B_RMOD              = 0000000B              EXE_GETJPI                00000219 R      02
ACB$B_TYPE              = 0000000A              EXTFLD                    0000066E R      02
ACB$L_AST               = 00000010              FLAGS                   = FFFFFFFC
ACB$L_ASTPRM            = 00000014              GRET                      0000033E R      02
ACB$L_KAST              = 00000018              IFD$Q_CURPROG           = 00000014
ACB$L_PID               = 0000000C              IOSB                    = 00000014
ACB$V_KAST              = 00000007              IPL$_ASTDEL             = 00000002
ACB$V_QUOTA             = 00000006              IPL$_SYNCH              = 00000008
ACB$W_SIZE              = 00000008              ITMLST                  = 00000010
ACB_C_IDESC             = 0000000C              JIB$B_JOBTYPE           = 00000068
ACB_L_COUNT               00000030              JIB$L_BYTCNT            = 00000020
ACB_L_DADDR               0000001C              JIB$L_BYTLM             = 00000024
ACB_L_EFN                 00000020              JIB$L_MPID              = 00000054
ACB_L_ILIST               00000034              JIB$L_PGFLCNT           = 0000003C
ACB_L_IMGCNT              0000002C              JIB$L_PGFLQUOTA         = 00000038
ACB_L_IOSB                00000024              JIB$T_ACCOUNT           = 00000018
ACB_L_OPID                00000028              JIB$T_USERNAME          = 0000000C
ADRTBL                    00000006 R      02    JIB$W_ENQCNT            = 0000004C
ASTADR                  = 00000018              JIB$W_ENQLM             = 0000004E
ASTPRM                  = 0000001C              JIB$W_FILCNT            = 00000030
BITDEF                  = FFFFFFE8              JIB$W_FILLM             = 00000032
BOOLE                   = 00000003              JIB$W_MAXDETACH         = 00000052
BSTRING                 = 00000001              JIB$W_MAXJOBS           = 00000050
CHECKITEM                 00000584 R      02    JIB$W_PRCCNT            = 00000044
CHECK_SPC                 0000072E R      02    JIB$W_PRCLIM            = 00000046
CSTRING                 = 00000002              JIB$W_SHRFLIM           = 0000004A
CTL$AL_CMCNTX             ********    X   02    JIB$W_TQCNT             = 00000034
CTL$AL_FINALEXC           ********    X   02    JIB$W_TQLM              = 00000036
CTL$AQ_EXCVEC             ********    X   02    JPI$C_ADRTYPE           = 00000001
CTL$GB_MSGMASK            ********    X   02    JPI$C_CTLTYPE           = 00000002
CTL$GL_CMHANDLR           ********    X   02    JPI$C_JIBTYPE           = 00000007
CTL$GL_CREPRC_FLAGS       ********    X   02    JPI$C_MAXSTRUC          = 00000006
CTL$GL_IMGHDRBF           ********    X   02    JPI$C_PCBFLDTYPE        = 00000005
CTL$GL_PHD                ********    X   02    JPI$C_PCBTYPE           = 00000003
CTL$GL_SITESPEC           ********    X   02    JPI$C_PHDFLDTYPE        = 00000006
CTL$GL_UAF_FLAGS          ********    X   02    JPI$C_PHDTYPE           = 00000004
CTL$GL_VIRTPEAK           ********    X   02    JPI$K_BATCH             = 00000002
CTL$GL_VOLUMES            ********    X   02    JPI$K_INTERACTIVE       = 00000003
CTL$GL_WSPEAK             ********    X   02    JPI$K_NETWORK           = 00000001
CTL$GQ_LOGIN              ********    X   02    JPI$K_OTHER             = 00000000
CTL$GQ_PROCPRIV           ********    X   02    JPI$_ACCOUNT            = 00000203
CTL$GT_CLINAME            ********    X   02    JPI$_APTCNT             = 0000030A
CTL$GT_TABLENAME          ********    X   02    JPI$_ASTACT             = 00000300
CTLTBL                    00000010 R      02    JPI$_ASTCNT             = 0000030E
DEANONPAGED               000008BA R      02    JPI$_ASTEN              = 00000301
DIRCNT                  = FFFFFFF8              JPI$_ASTLM              = 00000409
DSTRING                 = 00000004              JPI$_AUTHPRI            = 00000418
DYN$C_ACB               = 00000002              JPI$_AUTHPRIV           = 00000412
EFN                     = 00000004              JPI$_BIOCNT             = 0000030F
EXE$ALONONPAGED           ********    X   02    JPI$_BIOLM              = 00000310
EXE$BUFFRQUOTA            ********    X   02    JPI$_BUFIO              = 0000040C
EXE$CHKIMAGNAME           000007BB RG     02    JPI$_BYTCNT             = 00000311
EXE$DEANONPAGED           ********    X   02    JPI$_BYTLM              = 0000031A
EXE$GETJPI                00000000 RG     03    JPI$_CHAIN              = FFFFFFFF
EXE$IPID_TO_EPID          ********    X   02    JPI$_CLINAME            = 0000020A
```

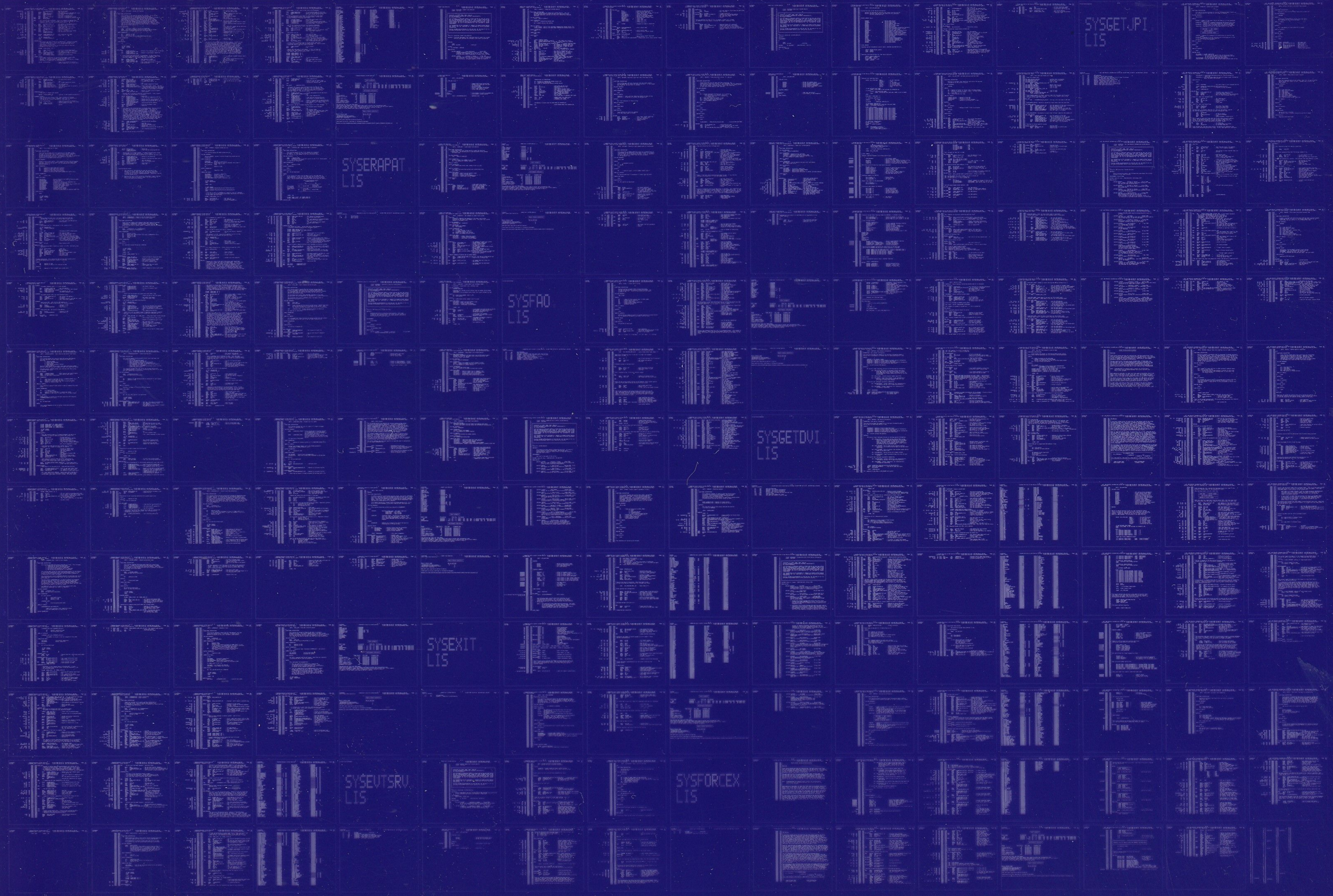
```
JPI$_CPULIM                      = 0000040D          JPI$_SITESPEC                    = 00000208
JPI$_CPUTIM                      = 00000407          JPI$_STATE                       = 00000306
JPI$_CREPRC_FLAGS                = 0000020C          JPI$_STS                         = 00000305
JPI$_CURPRIV                     = 00000400          JPI$_SWPFILLOC                   = 00000321
JPI$_DFPFC                       = 00000406          JPI$_TABLENAME                   = 0000020B
JPI$_DFWSCNT                     = 00000403          JPI$_TERMINAL                    = 0000031D
JPI$_DIOCNT                      = 00000312          JPI$_TMBU                        = 0000030B
JPI$_DIOLM                       = 00000313          JPI$_TQCNT                       = 00000315
JPI$_DIRIO                       = 0000040B          JPI$_TQLM                        = 00000410
JPI$_EFCS                        = 00000317          JPI$_UAF_FLAGS                   = 0000020D
JPI$_EFCU                        = 00000318          JPI$_UIC                         = 00000304
JPI$_EFWM                        = 00000316          JPI$_USERNAME                    = 00000202
JPI$_ENQCNT                      = 0000031F          JPI$_VIRTPEAK                    = 00000200
JPI$_ENQLM                       = 00000320          JPI$_VOLUMES                     = 00000205
JPI$_EXCVEC                      = 00000100          JPI$_WSAUTH                      = 00000401
JPI$_FILCNT                      = 00000314          JPI$_WSAUTHEXT                   = 00000417
JPI$_FILLM                       = 0000040F          JPI$_WSEXTENT                    = 00000416
JPI$_FINALEXC                    = 00000101          JPI$_WSPEAK                      = 00000201
JPI$_FREP0VA                     = 00000404          JPI$_WSQUOTA                     = 00000402
JPI$_FREP1VA                     = 00000405          JPI$_WSSIZE                      = 00000411
JPI$_FREPTECNT                   = 00000415          JPI_BIT                          = 00000003
JPI$_GPGCNT                      = 0000030C          JPI_S_NULLSWAP                   = 00000001
JPI$_GRP                         = 00000308          JPI_S_STRDSC                     = 00000001
JPI$_IMAGECOUNT                  = 0000041A          JPI_S_WILD                       = 00000001
JPI$_IMAGNAME                    = 00000207          JPI_V_NULLSWAP                   = 00000001
JPI$_IMAGPRIV                    = 00000413          JPI_V_STRDSC                     = 00000002
JPI$_JOBPRCCNT                   = 0000031E          JPI_V_WILD                       = 00000000
JPI$_JOBTYPE                     = 00000323          LOCAL_SPACE                      = FFFFFFD8
JPI$_LASTADR                     = 00000102          LOCK_BEGIN                       = 000006D3 R       02
JPI$_LASTCTL                     = 00000211          LOCK_END                           0000082C R       02
JPI$_LASTPCB                     = 00000326          LOCK_IPL                           0000082B R       02
JPI$_LASTPCBFLD                  = 00000500          MAXCOUNT                           00000000 R       02
JPI$_LASTPHD                     = 0000041C          MAX_ADR_ITEM                     = 00000001
JPI$_LASTPHDFLD                  = 00000600          MAX_CTL_ITEM                     = 00000010
JPI$_LOGINTIM                    = 00000206          MAX_PCBFLD_ITEM                  = FFFFFFFF
JPI$_MASTER_PID                  = 00000325          MAX_PCB_ITEM                     = 00000025
JPI$_MAXDETACH                   = 0000020E          MAX_PHDFLD_ITEM                  = FFFFFFFF
JPI$_MAXJOBS                     = 0000020F          MAX_PHD_ITEM                     = 0000001B
JPI$_MEM                         = 00000307          MOVEFU                             0000082C R       02
JPI$_MODE                        = 00000322          MOVEIT                             0000068B R       02
JPI$_MSGMASK                     = 00000209          MOVEPHD                            000006D0 R       02
JPI$_OWNER                       = 00000303          MOVETU                             000008C1 R       02
JPI$_PAGEFLTS                    = 0000040A          NAMPID                             0000098C R       02
JPI$_PAGFILCNT                   = 00000414          PCB$B_ASTACT                     = 0000000C
JPI$_PAGFILLOC                   = 00000419          PCB$B_ASTEN                      = 0000000D
JPI$_PGFLQUOTA                   = 0000040E          PCB$B_PRI                        = 0000000B
JPI$_PHDFLAGS                    = 0000041B          PCB$B_PRIB                       = 0000002F
JPI$_PID                         = 00000319          PCB$L_EFCS                       = 00000050
JPI$_PPGCNT                      = 0000030D          PCB$L_EFCU                       = 00000054
JPI$_PRCCNT                      = 0000031B          PCB$L_EFWM                       = 0000004C
JPI$_PRCLM                       = 00000408          PCB$L_EOWNER                     = 00000068
JPI$_PRCNAM                      = 0000031C          PCB$L_EPID                       = 00000064
JPI$_PRI                         = 00000302          PCB$L_JIB                        = 00000080
JPI$_PRIB                        = 00000309          PCB$L_PHD                        = 0000006C
JPI$_PROCPRIV                    = 00000204          PCB$L_PID                        = 00000060
JPI$_PROC_INDEX                  = 00000324          PCB$L_STS                        = 00000024
JPI$_SHRFILLM                    = 00000210          PCB$L_UIC                        = 000000BC
```

```
PCB$L_WSSWP                 = 00000020            PSL$M_IPL             = 001F0000
PCB$T_LNAME                 = 00000070            PSL$S_PRVMOD          = 00000002
PCB$T_TERMINAL              = 00000044            PSL$V_PRVMOD          = 00000016
PCB$V_BATCH                 = 0000000E            RESCAN                  000003AC R      02
PCB$V_DELPEN                = 00000001            RSN$_NPDYNMEM         = 00000003
PCB$V_INTER                 = 00000019            SCH$CHSEP               ********   X    02
PCB$V_NETWRK                = 00000015            SCH$CLREF               ********   X    02
PCB$V_PHDRES                = 00000012            SCH$C_COM             = 0000000C
PCB$V_SSRWAIT               = 0000000A            SCH$C_COMO            = 0000000D
PCB$V_SUSPEN                = 0000000B            SCH$C_MWAIT           = 00000002
PCB$W_APTCNT                = 00000030            SCH$C_SUSP            = 00000009
PCB$W_ASTCNT                = 00000038            SCH$C_SUSPO           = 0000000A
PCB$W_BIOCNT                = 0000003A            SCH$C_SWPPIX            ********   X    02
PCB$W_BIOLM                 = 0000003C            SCH$GL_CURPCB           ********   X    02
PCB$W_DIOCNT                = 0000003E            SCH$GL_MAXPIX           ********   X    02
PCB$W_DIOLM                 = 00000040            SCH$GL_NULLPCB          ********   X    02
PCB$W_GPGCNT                = 00000034            SCH$GL_PCBVEC           ********   X    02
PCB$W_GRP                   = 000000BE            SCH$POSTEF              ********   X    02
PCB$W_MEM                   = 000000BC            SCH$QAST                ********   X    02
PCB$W_PPGCNT                = 00000036            SCH$RWAIT               ********   X    02
PCB$W_PRCCNT                = 00000042            SCRATCH               = FFFFFFD8
PCB$W_STATE                 = 0000002C            SGN$GW_MAXPRCCT         ********   X    02
PCB$W_TMBU                  = 00000032            SPC_IMAGNAME            000007DC R      02
PCBFLDTBL                     000001D1 R      02  SPC_MASTER_PID          000007AB R      02
PCBTBL                        00000087 R      02  SPC_MODE                00000758 R      02
PHD$B_AUTHPRI               = 0000010C            SPC_PRI                 00000750 R      02
PHD$B_DFPFC                 = 00000034            SPC_PROC_INDEX          0000079A R      02
PHD$L_BIOCNT                = 00000058            SPC_WORKSET             00000778 R      02
PHD$L_CPULIM                = 0000005C            SPECIAL                 000001D1 R      02
PHD$L_CPUTIM                = 00000038            SPECIAL_LEN           = 0000000C
PHD$L_DIOCNT                = 00000054            SS$_ACCVIO            = 0000000C
PHD$L_FREP0VA               = 00000028            SS$_BADPARAM          = 00000014
PHD$L_FREP1VA               = 00000030            SS$_EXQUOTA           = 0000001C
PHD$L_FREPTECNT             = 0000002C            SS$_INSFMEM           = 00000124
PHD$L_IMGCNT                = 000000F4            SS$_NOMOREPROC        = 000009A8
PHD$L_PAGEFLTS              = 0000004C            SS$_NONEXPR           = 000008E8
PHD$L_PAGFIL                = 0000001C            SS$_NORMAL            = 00000001
PHD$Q_AUTHPRIV              = 000000E0            SS$_SUSPENDED         = 000003A4
PHD$Q_IMAGPRIV              = 000000E8            STEP                  = 00000005
PHD$Q_PRIVMSK               = 00000000            SYS$DCLAST              ********  GX    02
PHD$W_ASTLM                 = 00000040            VALUE                 = 00000000
PHD$W_DFWSCNT               = 0000001A            WSLIST                = FFFFFFF4
PHD$W_FLAGS                 = 00000036
PHD$W_WSAUTH                = 0000000A
PHD$W_WSAUTHEXT             = 00000014
PHD$W_WSEXTENT              = 00000016
PHD$W_WSLIST                = 00000008
PHD$W_WSQUOTA               = 00000018
PHD$W_WSSIZE                = 00000050
PHDFLDTBL                     000001D1 R      02
PHDTBL                        00000145 R      02
PHDTEMP                     = FFFFFFEC
PIDADR                      = 00000008
PR$_IPL                     = 00000012
PRCNAM                      = 0000000C
PRI$_TICOM                  = 00000004
PSL$C_USER                  = 00000003
```

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

PSECT name                  Allocation           PSECT No.  Attributes
----------                  ----------           ---------  ----------
.  ABS  .                   00000000  (     0.)  00 (   0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                       00000034  (    52.)  01 (   1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
YF$$SYSGETJPI               00000A13  (  2579.)  02 (   2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
YEXEPAGED                   00000005  (     5.)  03 (   3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
AEXENONPAGED                00000007  (     7.)  04 (   4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE

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

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    29    00:00:00.08    00:00:01.20
Command processing               121    00:00:00.68    00:00:04.14
Pass 1                           518    00:00:27.14    00:01:02.51
Symbol table sort                  0    00:00:02.07    00:00:07.15
Pass 2                           343    00:00:06.17    00:00:20.11
Symbol table output               38    00:00:00.25    00:00:01.44
Psect synopsis output              3    00:00:00.03    00:00:00.03
Cross-reference output             0    00:00:00.00    00:00:00.00
Assembler run totals            1054    00:00:36.43    00:01:36.59

The working set limit was 2100 pages.
135941 bytes (266 pages) of virtual memory were used to buffer the intermediate code.
There were 70 pages of symbol table space allocated to hold 1273 non-local and 119 local symbols.
1964 source lines were read in Pass 1, producing 28 object records in Pass 2.
51 pages of virtual memory were used to define 36 macros.

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

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[SYSLIB]SYSBLDMLB.MLB;1                      1
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                          18
_$255$DUA28:[SYSLIB]STARLET.MLB;2                       11
TOTALS (all libraries)                                  30

1632 GETS were required to define 30 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:SYSGETJPI/OBJ=OBJ$:SYSGETJPI MSRC$:SYSGETJPI/UPDATE=(ENH$:SYSGETJPI)+EXECML$/LIB+SYS$LIBRARY:SYSBLDMLB/LIB
```

0384 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



SYSGETJPI
LIS

SYSERAPAT
LIS

SYSFAO
LIS

SYSGETDVI
LIS

SYSEXIT
LIS

SYSEVTSRV
LIS

SYSFORCEX
LIS

0385 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY