```
DDDDDDDDDDDD        CCCCCCCCCCCC  LLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDD         DDD  CCC              LLL
DDDDDDDDDDDD         CCCCCCCCCCC  LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLLLLLLLLLLLLLL
DDDDDDDDDDDD        CCCCCCCCCCCC  LLLLLLLLLLLLLLL
```

FILEID**LEXICON

```
LL          EEEEEEEEEE  XX      XX    IIIIII      CCCCCCCC    OOOOOO    NN      NN
LL          EEEEEEEEEE  XX      XX    IIIIII      CCCCCCCC    OOOOOO    NN      NN
LL          EE          XX      XX      II      CC          OO      OO  NN      NN
LL          EE          XX      XX      II      CC          OO      OO  NN      NN
LL          EE            XX  XX        II      CC          OO      OO  NNNN    NN
LL          EE            XX  XX        II      CC          OO      OO  NNNN    NN
LL          EEEEEEEE        XX          II      CC          OO      OO  NN  NN  NN
LL          EEEEEEEE        XX          II      CC          OO      OO  NN  NN  NN
LL          EE            XX  XX        II      CC          OO      OO  NN    NNNN
LL          EE            XX  XX        II      CC          OO      OO  NN    NNNN
LL          EE          XX      XX      II      CC          OO      OO  NN      NN      ....
LL          EE          XX      XX      II      CC          OO      OO  NN      NN      ....
LLLLLLLLLL  EEEEEEEEEE  XX      XX    IIIIII      CCCCCCCC    OOOOOO    NN      NN      ....
LLLLLLLLLL  EEEEEEEEEE  XX      XX    IIIIII      CCCCCCCC    OOOOOO    NN      NN      ....

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  LEXICON - LEXICAL FUNCTION PARSE AND EXECUTION ROUTINES
0000            .IDENT  'V04-000'
0000
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    ; LEXICAL FUNCTION PARSE AND EXECUTION SUPPORT ROUTINES
0000    ;
0000    ; D. N. CUTLER 3-FEB-78
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-034 HWS0103         Harold Schultz  15-Aug-1984
0000    ;               When unable to find lexical function, undo search
0000    ;               optimization in function name before outputting
0000    ;               error message (Clear high bit in first character
0000    ;               of function name).
0000    ;
0000    ;       V03-033 PCG0024         Peter George    31-May-1984
0000    ;               Preclear XAB used by F$FILE to avoid RMS accvio.
0000    ;
0000    ;       V03-032 HWS0062         Harold Schultz  27-Apr-1984
0000    ;               In F$PARSE, return requested portion of file spec.
0000    ;               even if parse error. (undo PCG0018). Return null only
0000    ;               if no specific portion of file spec requested.
0000    ;
0000    ;       V03-031 HWS0051         Harold Schultz  05-Apr-1984
0000    ;               When doing an F$PARSE, cleanup any remaining internal
0000    ;               RMS context.
0000    ;
0000    ;       V03-030 HWS0046         Harold Schultz  02-Apr-1984
0000    ;               Translate concealed logical names when checking if a
0000    ;               file is "known" with F$FILE.
0000    ;
0000    ;       V03-029 PCG0023         Peter George    02-Mar-1984
0000    ;               Clean up F$SEARCH if input is "".
```

```
0000    ;
0000    ;       V03-028 HWS0010         Harold Schultz  13-Feb-1984
0000    ;               Use PRC_V_CARRCNTL instead of contents of prompt string
0000    ;               to check for presence/absence of CR/LF
0000    ;               Enable a parameter list of parse types to be passed to
0000    ;               F$PARSE. Add new parse type NO_CONCEAL.
0000    ;               In F$PARSE, check validity of name of filespec field parameter
0000    ;               before performing the parse.
0000    ;
0000    ;       V03-027 PCG0022         Peter George    02-Feb-1984
0000    ;               Add SYNTAX_ONLY argument to F$PARSE.
0000    ;               Have F$ID return a null string on all system service errors.
0000    ;
0000    ;       V03-026 RAS0247         Ron Schaefer    27-Jan-1984
0000    ;               Fix F$PARSE/F$SEARCH to release RMS context correctly
0000    ;               when doing the null parse.
0000    ;
0000    ;       V03-025 PCG0021         Peter George    16-Jan-1984
0000    ;               Fix broken branch.
0000    ;
0000    ;       V03-024 PCG0020         Peter George    12-Jan-1984
0000    ;               Fix broken branch.
0000    ;
0000    ;       V03-023 PCG0019         Peter George    11-Jan-1984
0000    ;               Have F$PARSE do a syntax-only parse.
0000    ;               Add F$EDIT lowercase.
0000    ;
0000    ;       V03-022 PCG0018         Peter George    28-Nov-1983
0000    ;               Return a null string no matter what part of an
0000    ;               non-existent file is requested in F$PARSE.
0000    ;
0000    ;       V03-021 TMK0001         Todd M. Katz            12-Oct-1983
0000    ;               Translate logical names using LNM$DCL_LOGICAL instead of
0000    ;               LNM$DEFAULT_SEARCH as the table name.
0000    ;
0000    ;       V03-020 PCG0017         Peter George    19-Sep-1983
0000    ;               Fix bug in GETJPI table expansion.
0000    ;
0000    ;       V03-019 PCG0016         Peter George    12-Sep-1983
0000    ;               Init NAM block ESS field in F$FILE.
0000    ;               Use $GETJPI definition of MODE.
0000    ;
0000    ;       V03-018 PCG0015         Peter George    17-Aug-1983
0000    ;               Change case of weekday strings.
0000    ;               Change the way that default protection is retrieved.
0000    ;               Specify that a NAM block is need for known file opens.
0000    ;               Have F$USER return new format UIC's.
0000    ;               Increase accuracy of F$GETJPI error reporting.
0000    ;               Add F$IDENTIFIER.
0000    ;
0000    ;       V03-017 PCG0014         Peter George    27-Jul-1983
0000    ;               Support both F$LOGICAL and F$TRNLNM.
0000    ;               Add LENGTH item to F$TRNLNM.
0000    ;
0000    ;       V03-016 PCG0013         Peter George    23-Jun-1983
0000    ;               Finish adding new arguments to F$CVTIME.
0000    ;               Use event flags more intelligently.
```

```
0000  ;                    Use !%I FAO directive in FORMAT_UIC.
0000  ;                    Add support for JNL ACP.
0000  ;                    Skip blanks between lexical function name and open paren.
0000  ;                    Make F$XLOGICAL F$LOGICAL.  Add TABLE item to F$LOGICAL.
0000  ;                    Add "OTHER" process mode type.
0000  ;
0000  ;           V03-015 PCG0012          Peter George    13-Jun-1983
0000  ;                    Fix bug in F$ENV("CAPTIVE").
0000  ;                    Fix broken branch.
0000  ;                    Signal P2 error for NOSUCHNODE in F$GETSYI.
0000  ;
0000  ;           V03-014 PCG0011          Peter George    01-Jun-1983
0000  ;                    Add second and third arguments to F$CVTIME.
0000  ;                    Add F$ENV("OUTPUT_RATE").
0000  ;
0000  ;           V03-013 PCG0010          Peter George    27-May-1983
0000  ;                    Add STDTIM data type.
0000  ;                    Make modifications to F$GETSYI.
0000  ;
0000  ;           V03-012 PCG0009          Peter George    30-Apr-1983
0000  ;                    Add F$XLOGICAL.
0000  ;                    Add F$ENV("CAPTIVE","INTERACTIVE").
0000  ;                    Add second argument to F$VERIFY.
0000  ;                    Add second argument to F$GETSYI.
0000  ;                    Add HEXSTR data type.
0000  ;
0000  ;           V03-011 PCG0008          Peter George    30-Mar-1983
0000  ;                    Add F$ENV("KEY_STATE","VERIFY_PROCEDURE","VERIFY_IMAGE").
0000  ;                    Return standardized protection string everywhere.
0000  ;                    Remove all references to PRC_V_GOTO.
0000  ;
0000  ;           V03-010 PCG0007          Peter George    03-Mar-1983
0000  ;                    Move value conversion and editing routines to CONVERT.
0000  ;                    Set UPD and DEL share options in F$FILE.
0000  ;                    Add F$FILE ("KNOWN").
0000  ;
0000  ;           V03-009 KFH0003          Ken Henderson    2 Mar 1983
0000  ;                    Modified JPI_ITEM_CODE macro to conform to new
0000  ;                    JPITABLE.MAR.
0000  ;
0000  ;           V03-008 KFH0002          Ken Henderson   25 Feb 1983
0000  ;                    Changed GETSYI macro to SYI_ITEM_CODE to inherit
0000  ;                    all 217 SYSBOOT parameters and other new GETSYI
0000  ;                    item-codes. Also changed GETDVI macro to DVI_ITEM_CODE
0000  ;                    to inherit all new GETDVI item-codes. Also added
0000  ;                    the new PSECT DCL$ZDATA to contain all the tables.
0000  ;
0000  ;           V03-007 PCG0006          Peter George    15-Feb-1983
0000  ;                    Cleanup some F$SEARCH code.
0000  ;                    Change F$EDIT COMMENT to UNCOMMENT.
0000  ;                    Change F$ENV FILENAME to PROCEDURE.
0000  ;                    Remove F$ENV MESSAGE.
0000  ;                    Change F$ENV MESSAGE_QUALIFIERS to MESSAGE.
0000  ;
0000  ;           V03-006 KFH0001          Ken Henderson    10 Feb 1983
0000  ;                    Changed GETJPI macro into JPI_ITEM_CODE
0000  ;                    and added call to JPI_GENERATE_TABLE macro.
```

```
0000  ;
0000  ;          V03-005 PCG0005         Peter George    07-Jan-1983
0000  ;                  Add F$ENVIRONMENT.
0000  ;
0000  ;          V03-004 PCG0004         Peter George    15-Dec-1982
0000  ;                  Add AUTHPRI, PAGEFILLOC, SWPFILLOC F$GETJPI fields.
0000  ;                  Do not report F$SEARCH errors.
0000  ;
0000  ;          V03-003 PCG0003         Peter George    14-Nov-1982
0000  ;                  Fold DCL$REMBLANKS and DCL$UPCASE into DCL$EDIT.
0000  ;
0000  ;          V03-002 PCG0002         Peter George    07-Oct-1982
0000  ;                  Add F$TYPE, F$EDIT, and F$ELEMENT.
0000  ;                  Add DCL$UPCASE.  Tighten up bounds checking in
0000  ;                  F$CVSI and F$CVUI.
0000  ;
0000  ;          V03-001 PHL0101         Peter H. Lipman 28-Jun-1982
0000  ;                  Use the new services, $GETDVIW, $GETJPIW, $GETSYIW
0000  ;
0000  ;---
```

```
          0000  ;
          0000  ; MACRO LIBRARY CALLS
          0000  ;
          0000
          0000          PRCDEF                          ;DEFINE PROCESS WORK AREA
          0000          WRKDEF                          ;DEFINE COMMAND WORK AREA
          0000          SYMDEF                          ;DEFINE TYPES OF SYMBOLS
          0000          IDFDEF                          ;DEFINE INDIRECT FRAME BLOCK
          0000          EDITDEF                         ;DEFINE EDIT CODES
          0000          $PPDDEF                         ;DEFINE LOGINOUT/DCL COMMUNICATION AREA
          0000          $CLIMSGDEF                      ;DEFINE ERROR/STATUS VALUES
          0000          $DIBDEF                         ;DEFINE DEVICE INFORMATION FIELDS
          0000          $DEVDEF                         ;DEFINE DEVICE CHARACTERISTICS BITS
          0000          $DVIDEF                         ;DEFINE GETDVI INFORMATION FIELDS
          0000          $JPIDEF                         ;DEFINE JOB/PROCESS INFORMATION ID'S
          0000          $LNMDEF                         ;DEFINE LOGICAL NAME CODES
          0000          $PSLDEF                         ;DEFINE PSL CODES
          0000          $SYIDEF                         ;DEFINE GETSYI INFORMATION ID'S
          0000          $FABDEF                         ;FAB DEFINITIONS
          0000          $NAMDEF                         ;NAM DEFINITIONS
          0000          $XABDEF                         ;XAB DEFINITIONS
          0000          $XABALLDEF                      ;ALLOCATION XAB DEFINITIONS
          0000          $XABDATDEF                      ;DATE/TIME XAB DEFINITIONS
          0000          $XABFHCDEF                      ;FILE HEADER XAB DEFINITIONS
          0000          $XABPRODEF                      ;PROTECTION XAB DEFINITIONS
          0000          $XABSUMDEF                      ;SUMMARY ISAM XAB DEFINITIONS
          0000          $STSDEF                         ;STATUS DEFINITIONS
          0000
          0000  ;
          0000  ; DEFINE THIS SO SYSPARAM WILL BE CONDITIONALIZED CORRECTLY
          0000  ;
00000000  0000          GETSYISW = 0
          0000
          0000  ;
          0000  ; FUNCTION ARGUMENT DEFINITIONS
          0000  ;
          0000
00000010  0000  MAX_ARGS = 16                           ;MAXIMUM NUMBER OF ARGUMENTS ALLOWED
00000000  0000  NPARM=0                                 ;NUMBER OF PARAMETERS
00000004  0000  P1=4                                    ;STACK OFFSET TO PARAMETER 1
0000000C  0000  P2=12                                   ;STACK OFFSET TO PARAMETER 2
00000014  0000  P3=20                                   ;STACK OFFSET TO PARAMETER 3
0000001C  0000  P4=28                                   ;STACK OFFSET TO PARAMETER 4
00000024  0000  P5=36                                   ;STACK OFFSET TO PARAMETER 5
0000002C  0000  P6=44                                   ;STACK OFFSET TO PARAMETER 6
          0000
          0000  ;
          0000  ; DEFINE FORMAT OF LONGWORD DESCRIBING FUNCTION ARGUMENTS
          0000  ;
          0000  ;       BYTE 0:         NUMBER OF ARGUMENTS ALLOWED WITH FUNCTION
          0000  ;       BYTES 1-3:      ARRAY OF 4 NIBBLES, EACH NIBBLE DESCRIBING AN ARGUMENT
          0000  ;
          0000  ;                       BITS 0-1: TYPE OF VALUE ALLOWED
          0000  ;                                 0 = BINARY
          0000  ;                                 1 = STRING
          0000  ;                                 2 = REFERENCE TO SYMBOL NAME
          0000  ;                                 3 = EITHER STRING OR BINARY
```

```
                  0000  ;                         BIT 2:    TRUE IF ARGUMENT OPTIONAL, ELSE REQUIRED
                  0000  ;                         BIT 3:    UNUSED
                  0000  ;
                  0000  ;        IF THE NUMBER OF ARGUMENTS EXCEEDS 4, THEN ALL REMAINING ARGUMENTS
                  0000  ;        ARE TREATED AS "EITHER STRING OR BINARY".
                  0000  ;
                  0000
        00000000  0000  ARG_V_TYPE       = 0                    ; STARTING POSITION OF TYPE FIELD
        00000002  0000  ARG_S_TYPE       = 2                    ; SIZE OF TYPE FIELD
        00000000  0000  ARG_C_BINARY     = 0                    ; BINARY
        00000001  0000  ARG_C_STRING     = 1                    ; STRING
        00000002  0000  ARG_C_SYMREF     = 2                    ; REFERENCE TO SYMBOL NAME
        00000003  0000  ARG_C_ANY        = 3                    ; EITHER STRING OR BINARY
        00000002  0000  ARG_V_OPTIONAL   = 2                    ; TRUE IF ARGUMENT OPTIONAL
        00000003  0000  ARG_V_KEYWORD    = 3                    ; TRUE IF KEYWORD ARGUMENT
        00000008  0000  ARG_C_ENTSIZE    = 8                    ; SIZE OF ENTRY (NAME PLUS ARGS)
                  0000
                  0000  ;
                  0000  ; MACRO TO GENERATE LEXICAL FUNCTION DEFINITION TABLE
                  0000  ;
                  0000
                  0000          .MACRO  FIRST4  NAME
                  0000          .NCHR   $ZZ$,NAME
                  0000          $XX$ = 128
                  0000          .IRPC   $YY$,NAME
                  0000          .BYTE   ^A"'$YY$'"!$XX$
                  0000          $XX$ = 0
                  0000          .ENDR
                  0000          .REPT   4-$ZZ$
                  0000          .ASCII  " "
                  0000          .ENDR
                  0000          .ENDM   FIRST4
                  0000
                  0000          .MACRO  FUNCARG TYPE,PRESENCE=REQUIRED,FORMAT=VALUE
                  0000  ;       RETURNS $XX$ WITH NIBBLE DESCRIBING ARGUMENT
                  0000          $XX$ = -1
                  0000          .IIF IDN %EXTRACT(0,1,TYPE),<N>,         $XX$ = ARG_C_BINARY
                  0000          .IIF IDN %EXTRACT(0,1,TYPE),<S>,         $XX$ = ARG_C_STRING
                  0000          .IIF IDN %EXTRACT(0,1,TYPE),<R>,         $XX$ = ARG_C_SYMREF
                  0000          .IIF IDN %EXTRACT(0,1,TYPE),<A>,         $XX$ = ARG_C_ANY
                  0000          .IF LT  $XX$
                  0000          .ERROR  ;INVALID ARGUMENT TYPE.  (STRING, NUMERIC, REFERENCE OR ANY)
                  0000          .ENDC
                  0000          .IIF IDN <PRESENCE><OPTIONAL>, $XX$ = $XX$ + <1@ARG_V_OPTIONAL>
                  0000          .IIF IDN <FORMAT><KEYWORD>, $XX$ = $XX$ + <1@ARG_V_KEYWORD>
                  0000          .ENDM   FUNCARG
                  0000
                  0000          .MACRO  FUNCT   NAME,ARGTYPES,FLAGS
                  0000          FIRST4  %EXTRACT(0,4,NAME)
                  0000          NARGS = 0
                  0000          .IRP    $ZZ$,<ARGTYPES>
                  0000          NARGS = NARGS + 1
                  0000          .ENDR
                  0000          .IF IDN <FLAGS><VARIABLE>
                  0000          NARGS = MAX_ARGS
                  0000          .ENDC
                  0000          $ARGS$ = 0
```

```
0000            $BITS = 0
0000            .IRP    $YY$,<ARGTYPES>
0000            FUNCARG $YY$
0000            $ARGS$ = $ARGS$ + <$XX$ @ $BITS$>
0000            $BITS = $BITS + 4
0000            .IIF GT $BITS-24, .ERROR ;TOO MANY EXPLICTLY DEFINED ARGUMENTS
0000            .ENDR
0000            .LONG   NARGS + <$ARGS$ @ 8>
0000            .ENDM   FUNCT
```

```
    0000  ;
    0000  ; LEXICAL FUNCTION DEFINITION TABLE
    0000  ;
    0000
00000000            .PSECT  DCL$ZDATA,BYTE,RD,NOWRT
    0000
    0000  LXFUNTABLE:
    0000            FUNCT   CVSINTEGER,       <N,N,S>             ;CONVERT SIGNED INTEGER
    0008            FUNCT   CVUINTEGER,       <N,N,S>             ;CONVERT UNSIGNED INTEGER
    0010            FUNCT   CVTIME,           <<S,OPTIONAL>,-     ;CONVERT TO COMPARABLE TIME STRING
    0010                                      <S,OPTIONAL,KEYWORD>,-
    0010                                      <S,OPTIONAL,KEYWORD>>
    0018            FUNCT   DIRECTORY                             ;DEFAULT DIRECTORY STRING
    0020            FUNCT   EDIT,             <S,-                ;EDIT A STRING
    0020                                      <S,REQUIRED,KEYWORD>>
    0028            FUNCT   ELEMENT,          <N,S,S>             ;FIND THE SPECIFIED ELEMENT
    0030            FUNCT   ENVIRONMENT,      <<S,REQUIRED,KEYWORD>> ;GET DCL CONTEXT INFO
    0038            FUNCT   EXTRACT,          <N,N,S>             ;EXTRACT SUBSTRING FROM STRING
    0040            FUNCT   FAO,              <S,ANY,ANY,-        ;FAO SERVICE
    0040                                      ANY,ANY,ANY>,VARIABLE
    0048            FUNCT   FILE_ATTRIBUTES,<S,-                  ;GET FILE ATTRIBUTES
    0048                                      <S,REQUIRED,KEYWORD>>
    0050            FUNCT   GETDVI,           <S,-                ;GETDVI SYSTEM SERVICE
    0050                                      <S,REQUIRED,KEYWORD>>
    0058            FUNCT   GETJPI,           <S,-                ;GETJPI SYSTEM SERVICE
    0058                                      <S,REQUIRED,KEYWORD>>
    0060            FUNCT   GETSYI,           <<S,REQUIRED,KEYWORD>,- ;GETSYI SYSTEM SERVICE
    0060                                      <S,OPTIONAL,KEYWORD>>   ;
    0068            FUNCT   IDENTIFIER,       <ANY,-              ;TRANSLATE IDENTIFIER
    0068                                      <S,REQUIRED,KEYWORD>>
    0070            FUNCT   INTEGER,          <ANY>               ;CONVERT EXPRESSION TO INTEGER
    0078            FUNCT   LENGTH,           <S>                 ;STRING LENGTH IN BINARY
    0080            FUNCT   LOCATE,           <S,S>               ;FIND OFFSET TO SUBSTRING
    0088            FUNCT   LOGICAL,          <S>                 ;LOGICAL NAME TRANSLATION
    0090            FUNCT   MESSAGE,          <N>                 ;GET MESSAGE STRING
    0098            FUNCT   MODE                                  ;KEYWORD DESCRIBING TYPE OF JOB
    00A0            FUNCT   PARSE,            <-                  ;PARSE A FILESPEC
    00A0                                      <S>,-               ;       PRIMARY FILESPEC
    00A0                                      <S,OPTIONAL>,-      ;       DEFAULT FILESPEC
    00A0                                      <S,OPTIONAL>,-      ;       RELATED (STICKY) FILESPEC
    00A0                                      <S,OPTIONAL,KEYWORD>,-; FILESPEC FIELD DESIRED
    00A0                                      <S,OPTIONAL,KEYWORD>>;  TYPE OF PARSE
    00A8            FUNCT   PID,              <REF>               ;GET NEXT PROCESS ID STRING
    00B0            FUNCT   PRIVILEGE,        <<S,REQUIRED,KEYWORD>>  ;TRUE IF PRIVS ARE SET/CLEAR
    00B8            FUNCT   PROCESS                               ;PROCESS NAME STRING
    00C0            FUNCT   SEARCH,           <S,<N,OPTIONAL>>        ;GET NEXT WILDCARD FILESPEC
    00C8            FUNCT   SETPRV,           <<S,REQUIRED,KEYWORD>>  ;SETPRV SYSTEM SERVICE
    00D0            FUNCT   STRING,           <ANY>               ;CONVERT EXPRESSION TO STRING
    00D8            FUNCT   TIME                                  ;TIME OF DAY STRING
    00E0            FUNCT   TRNLNM,           <<S>,-              ;LOGICAL NAME TRANSLATION
    00E0                                      <S,OPTIONAL>,-      ;       TABLE NAME
    00E0                                      <N,OPTIONAL>,-      ;       TRANSLATION NUMBER
    00E0                                      <S,OPTIONAL,KEYWORD>,- ;ACCESS MODE
    00E0                                      <S,OPTIONAL,KEYWORD>,- ;ATTRIBUTE
    00E0                                      <S,OPTIONAL,KEYWORD>>  ;ITEM
    00E8            FUNCT   TYPE              <REF>               ;TYPE OF SYMBOL
    00F0            FUNCT   USER                                  ;UIC STRING
```

```
            00F8          FUNCT   VERIFY,         <<N,OPTIONAL>,- ;VERIFY MODE IN BINARY, AND
            00F8                                   <N,OPTIONAL>>  ;OPTIONALLY CHANGE VERIFY MODE
            0100
00000100    0100    LXFUNTABLEN=.-LXFUNTABLE                      ;LENGTH OF LEXICAL FUNCTION TABLE
            0100
            0100    ;
            0100    ; DEFINE THE DTYPES THAT LEXICON USES (SUBSET OF ALL POSSIBLE DTYPES)
            0100    ;
00000000    0100          BIN = 0
00000001    0100          STR = 1
00000002    0100          HEX = 2
00000003    0100          PRV = 3
00000004    0100          BIT = 4
00000005    0100          HEXSTR = 5
00000006    0100          STDTIM = 6
            0100
            0100    ;
            0100    ; MACRO TO DEFINE VALID $GETJPI ITEMS WHICH MAY BE USED IN F$GETJPI
            0100    ; THIS MACRO IS CALLED BY THE COMMONLY DEFINED MACRO: JPI_GENERATE_TABLE
            0100    ;
            0100          .MACRO  JPI_ITEM_CODE   BASE,-          ; of EXE$GETJPI internal tables
            0100                                  NAME,-          ; of the item-code
            0100                                  SOURCE,-        ; of the data
            0100                                  DTYPE,-         ; data type and usage
            0100                                  BITPOS,-        ; of the 'bitval' bit
            0100                                  BITSIZ,-        ; of the 'bitval' bit
            0100                                  OUTLEN,-        ; of the data
            0100                                  STRUCT          ; structure containing the data
            0100
            0100          .WORD   JPI$_'NAME
            0100
            0100          $XX$ = BIN
            0100          .IIF IDN <DTYPE><DECNUM>,       $XX$=BIN
            0100          .IIF IDN <DTYPE><PADSTR>,       $XX$=STR
            0100          .IIF IDN <DTYPE><CNTSTR>,       $XX$=STR
            0100          .IIF IDN <DTYPE><HEXNUM>,       $XX$=HEX
            0100          .IIF IDN <DTYPE><PRVMSK>,       $XX$=PRV
            0100          .IIF IDN <DTYPE><STDUIC>,       $XX$=BIN
            0100          .IIF IDN <DTYPE><STDTIM>,       $XX$=STR
            0100          .IIF IDN <DTYPE><STRDSC>,       $XX$=STR
            0100          .IIF IDN <DTYPE><BITVEC>,       $XX$=BIN
            0100          .IIF IDN <DTYPE><BITVAL>,       $XX$=BIT
            0100          .IIF IDN <DTYPE><PRTMSK>,       $XX$=BIN
            0100          .IIF IDN <DTYPE><ACPTYP>,       $XX$=BIN
            0100          .IIF IDN <DTYPE><HEXSTR>,       $XX$=HEXSTR
            0100
            0100          .BYTE   $XX$
            0100          .ASCIC  "NAME"
            0100          .ENDM   JPI_ITEM_CODE
```

```
                  0100  ;
                  0100  ;*************************************************************
                  0100  ;
                  0100  ; DEFINE VALID $ ITEMS WHICH MAY BE USED IN F$GETJPI
                  0100  ;
                  0100  ;*************************************************************
                  0100  ;
                  0100
                  0100  GETJPI_TABLE:
                  0100
                  0100  ; Each element of GETJPI_TABLE is:
                  0100  ;
                  0100          ;---------------------;
                  0100          ; .WORD   JPI$_'NAME  ;
                  0100          ; .BYTE   DTYPE       ;
                  0100          ; .ASCIC  'NAME'      ;
                  0100          ;---------------------;
                  0100
                  0100          JPI_GENERATE_TABLE
                  047C
        00000000  047C          .LONG   0
                  0480
                  0480  ;
                  0480  ; MACRO TO DEFINE VALID $GETSYI ITEMS WHICH MAY BE USED IN F$GETSYI
                  0480  ; THIS MACRO IS CALLED BY THE COMMONLY DEFINED MACRO: SYI_GENERATE_TABLE
                  0480  ;
                  0480          .MACRO  SYI_ITEM_CODE   BASE,-          ; of EXE$GETSYI internal tables
                  0480                                  NAME,-          ; of the item-code
                  0480                                  SOURCE,-        ; of the data
                  0480                                  DTYPE,-         ; data type and usage
                  0480                                  BITPOS,-        ; of FLD data items
                  0480                                  BITSIZ,-        ; of FLD data items
                  0480                                  OUTLEN          ; of the returned data
                  0480
                  0480          .WORD   SYI$_'NAME
                  0480
                  0480          $XX$ = BIN
                  0480  ;       .IIF IDN <DTYPE><DECNUM>,       $XX$=BIN
                  0480  ;       .IIF IDN <DTYPE><PADSTR>,       $XX$=STR
                  0480  ;       .IIF IDN <DTYPE><CNTSTR>,       $XX$=STR
                  0480  ;       .IIF IDN <DTYPE><HEXNUM>,       $XX$=HEX
                  0480  ;       .IIF IDN <DTYPE><PRVMSK>,       $XX$=PRV
                  0480  ;       .IIF IDN <DTYPE><STDUIC>,       $XX$=BIN
                  0480  ;       .IIF IDN <DTYPE><STDTIM>,       $XX$=STR
                  0480  ;       .IIF IDN <DTYPE><STRDSC>,       $XX$=STR
                  0480  ;       .IIF IDN <DTYPE><BITVEC>,       $XX$=BIN
                  0480  ;       .IIF IDN <DTYPE><BITVAL>,       $XX$=BIT
                  0480  ;       .IIF IDN <DTYPE><HEXSTR>,       $XX$=HEXSTR
                  0480
                  0480  ;
                  0480  ; The above mapping is identical to GETJPI, but the GETSYI code can't
                  0480  ; yet handle DTYPEs other than HEX, BIN, STR, BIT, and HEXSTR
                  0480  ; so the following table is used in the meantime until everything is done.
                  0480  ;
                  0480
                  0480          .IIF IDN <DTYPE><DECNUM>,       $XX$=BIN
                  0480          .IIF IDN <DTYPE><PADSTR>,       $XX$=STR
```

```
0480            .IIF IDN <DTYPE><CNTSTR>,        $XX$=STR
0480            .IIF IDN <DTYPE><HEXNUM>,        $XX$=HEX
0480            .IIF IDN <DTYPE><PRVMSK>,        $XX$=BIN
0480            .IIF IDN <DTYPE><STDUIC>,        $XX$=BIN
0480            .IIF IDN <DTYPE><STDTIM>,        $XX$=STDTIM
0480            .IIF IDN <DTYPE><STRDSC>,        $XX$=STR
0480            .IIF IDN <DTYPE><BITVEC>,        $XX$=BIN
0480            .IIF IDN <DTYPE><BITVAL>,        $XX$=BIT
0480            .IIF IDN <DTYPE><HEXSTR>,        $XX$=HEXSTR
0480
0480            .BYTE   $XX$
0480            .ASCIC  "NAME"
0480            .ENDM   SYI_ITEM_CODE
0480
0480    ;
0480    ;*************************************************************
0480    ;
0480    ; DEFINE VALID $GETSYI ITEMS WHICH MAY BE USED IN F$GETSYI
0480    ;
0480    ;*************************************************************
0480    ;
0480
0480    GETSYI_TABLE:
0480
0480    ; Each element of GETSYI_TABLE is:
0480    ;
0480            ;---------------------;
0480            ; .WORD    SYI$_'NAME ;
0480            ; .BYTE    DTYPE      ;
0480            ; .ASCIC   "NAME"     ;
0480            ;---------------------;
0480
0480            SYI_GENERATE_TABLE
```

```
                              0C9C
                              0C9C            .NLIST  CND
                              0C9C            PARAMETER       ADDRESS=EXE$GL_DEFFLAGS,-
                              0C9C                            DEFAULT=1,-
                              0C9C                            MAX=1,-
                              0C9C                            MIN=0,-
                              0C9C                            NAME=BUGREBOOT,-
                              0C9C                            BIT=EXE$V_BUGREBOOT,-
                              0C9C                            TYPE=<DYNAMIC,SYS>,-
                              0C9C                            UNIT=Boolean
                              0C9C
                    00000004  0C9C            OUTLEN = 4
                              0C9C            SYI_ITEM_CODE   FLD,-
                              0C9C                            <BUGREBOOT>,-
                              0C9C                            <EXE$GL_DEFFLAGS>,-
                              0C9C                            BITVAL,-
                              0C9C                            <EXE$V_BUGREBOOT>,-
                              0C9C                            1,-
                              0C9C                            1
                              0C9C
                        2001  0C9C            .WORD   SYI$_BUGREBOOT
                              0C9E
                    00000000  0C9E            $XX$ = BIN
                              0C9E      ;     .IIF IDN <BITVAL><DECNUM>,      $XX$=BIN
                              0C9E      ;     .IIF IDN <BITVAL><PADSTR>,      $XX$=STR
                              0C9E      ;     .IIF IDN <BITVAL><CNTSTR>,      $XX$=STR
                              0C9E      ;     .IIF IDN <BITVAL><HEXNUM>,      $XX$=HEX
                              0C9E      ;     .IIF IDN <BITVAL><PRVMSK>,      $XX$=PRV
                              0C9E      ;     .IIF IDN <BITVAL><STDUIC>,      $XX$=BIN
                              0C9E      ;     .IIF IDN <BITVAL><STDTIM>,      $XX$=STR
                              0C9E      ;     .IIF IDN <BITVAL><STRDSC>,      $XX$=STR
                              0C9E      ;     .IIF IDN <BITVAL><BITVEC>,      $XX$=BIN
                              0C9E      ;     .IIF IDN <BITVAL><BITVAL>,      $XX$=BIT
                              0C9E      ;     .IIF IDN <BITVAL><HEXSTR>,      $XX$=HEXSTR
                              0C9E
                              0C9E      ;
                              0C9E      ; The above mapping is identical to GETJPI, but the GETSYI code can't
                              0C9E      ; yet handle DTYPEs other than HEX, BIN, STR, BIT, and HEXSTR
                              0C9E      ; so the following table is used in the meantime until everything is done.
                              0C9E      ;
                              0C9E
                              0C9E
                          04  0C9E            .BYTE   $XX$
 54 4F 4F 42 45 52 47 55 42 00'  0C9F            .ASCIC  "BUGREBOOT"
                          09  0C9F
                              0CA9
                              0CA9
```

```
                              148D
                    00000000  148D            .LONG   0
                              1491
                              1491   ;
                              1491   ; PRODUCE A TABLE OF ASCIC NAMES DESCRIBING EACH OF THE PROCESS STATES
                              1491   ;
                              1491
                              1491            .MACRO  $DEFINI NAME,P1,P2
                              1491            .ENDM
                              1491            .MACRO  $DEFEND NAME,P1,P2
                              1491            .ENDM
                              1491            .MACRO  $EQU SYMBOL,VALUE
                              1491   S = %LOCATE(< >,SYMBOL)+1
                              1491   L = %LENGTH(SYMBOL)-S
                              1491            .ASCIC  /%EXTRACT(S,L,SYMBOL)/
                              1491            .BLKB   8-<L+1>
                              1491            .ENDM   $EQU
                              1491
                              1491   STATE_TABLE:
     4C 41 47 45 4C 4C 49 00' 1491            .ASCIC  'ILLEGAL'                   ; FOR ZERO CASE
                          07  1491
                              1499            $STATEDEF                           ; TABLE OF PROCESS STATE NAMES
                              1509
                    00000003  1509            .MDELETE        $DEFINI,$DEFEND,$EQU
                              1509
                              1509
                              1509   ;
                              1509   ; MACRO TO DEFINE VALID DATA ITEMS WHICH MAY BE USED IN F$FILE_ATTRIBUTES
                              1509   ;
                              1509
                              1509            .MACRO  FILITEM NAME,BLOCK,FIELD,SIZE,BITNUM,TYPE=BINARY
                              1509            $XX$ = 0
                              1509            .IIF IDN <BLOCK><FAB>,              $XX$=0
                              1509            .IIF IDN <BLOCK><NAM>,              $XX$=1
                              1509            .IIF IDN <BLOCK><XABALL>,           $XX$=2
                              1509            .IIF IDN <BLOCK><XABDAT>,           $XX$=3
                              1509            .IIF IDN <BLOCK><XABFHC>,           $XX$=4
                              1509            .IIF IDN <BLOCK><XABPRO>,           $XX$=5
                              1509            .IIF IDN <BLOCK><XABSUM>,           $XX$=6
                              1509            .BYTE   $XX$                        ; RMS BLOCK ID
                              1509            $XX$ = 0
                              1509            .IIF NB <FIELD>,$XX$ = 'FIELD'
                              1509            .BYTE   $XX$                        ; OFFSET TO FIELD IN RMS BLOCK
                              1509            $XX$ = 0
                              1509            .IIF IDN <SIZE><BIT>,               $XX$=1
                              1509            .IIF IDN <SIZE><BYTE>,              $XX$=8
                              1509            .IIF IDN <SIZE><WORD>,              $XX$=16
                              1509            .IIF IDN <SIZE><LONG>,              $XX$=32
                              1509            .BYTE   $XX$                        ; SIZE OF FIELD
                              1509            $XX$ = 0
                              1509            .IIF NB <BITNUM>,$XX$ = 'BITNUM'
                              1509            .BYTE   $XX$                        ; BIT NUMBER
                              1509            $XX$ = 0
                              1509            .IIF IDN <TYPE><BINARY>,            $XX$=0
                              1509            .IIF IDN <TYPE><DATE>,              $XX$=1
                              1509            .IIF IDN <TYPE><ASCIC>,             $XX$=2
                              1509            .IIF IDN <TYPE><FILEID>,            $XX$=3
```

```
             1509          .IIF IDN <TYPE><ORG>,           $XX$=4
             1509          .IIF IDN <TYPE><RAT>,           $XX$=5
             1509          .IIF IDN <TYPE><RFM>,           $XX$=6
             1509          .IIF IDN <TYPE><UIC>,           $XX$=7
             1509          .IIF IDN <TYPE><PROT>,          $XX$=8
             1509          .IIF IDN <TYPE><EOF>,           $XX$=9
             1509          .IIF IDN <TYPE><BOOLEAN>,       $XX$=10
             1509          .IIF IDN <TYPE><KNOWN>,         $XX$=11       ; F$FILE DEPENDS ON THIS BEI
             1509          .BYTE   $XX$                    ; TYPE OF DATA ITEM
             1509          .ASCIC  "NAME"                  ; ASCIC NAME OF DATA ITEM
             1509          .ENDM   FILITEM
             1509
             1509  ;
             1509  ; DEFINE XAB BLOCK ID'S AND LENGTHS
             1509  ;
             1509
             1509  XAB_INFO:
      20 14  1509          .BYTE   XAB$C_ALL,XAB$C_ALLLEN
      2C 12  150B          .BYTE   XAB$C_DAT,XAB$C_DATLEN
      2C 1D  150D          .BYTE   XAB$C_FHC,XAB$C_FHCLEN
      58 13  150F          .BYTE   XAB$C_PRO,XAB$C_PROLEN
      0C 16  1511          .BYTE   XAB$C_SUM,XAB$C_SUMLEN
             1513
             1513  ;
             1513  ; DEFINE VALID DATA ITEMS WHICH MAY BE USED IN F$FILE_ATTRIBUTES
             1513  ;
             1513
             1513  FILE_TABLE:
             1513          FILITEM ALQ,FAB,FAB$L_ALQ,SIZE=LONG          ;ALLOCATION QUANTITY
             151C          FILITEM BKS,FAB,FAB$B_BKS,SIZE=BYTE          ;BUCKET SIZE
             1525          FILITEM BLS,FAB,FAB$W_BLS,SIZE=WORD          ;BLOCK SIZE
             152E          FILITEM DEQ,FAB,FAB$W_DEQ,SIZE=WORD          ;DEFAULT EXTENSION QUANTITY
             1537          FILITEM FSZ,FAB,FAB$B_FSZ,SIZE=BYTE          ;FIXED CONTROL AREA SIZE
             1540          FILITEM MRN,FAB,FAB$L_MRN,SIZE=LONG          ;MAXIMUM RECORD NUMBER
             1549          FILITEM MRS,FAB,FAB$W_MRS,SIZE=WORD          ;MAXIMUM RECORD SIZE
             1552          FILITEM ORG,FAB,FAB$B_ORG,SIZE=BYTE,TYPE=ORG   ;FILE ORGANIZATION
             155B          FILITEM RAT,FAB,FAB$B_RAT,SIZE=BYTE,TYPE=RAT   ;RECORD ATTRIBUTES
             1564          FILITEM RFM,FAB,FAB$B_RFM,SIZE=BYTE,TYPE=RFM   ;RECORD FORMAT
             156D          FILITEM KNOWN,NAM,FAB$L_CTX,SIZE=LONG,TYPE=KNOWN;KNOWN FILE CHECK
             1578          FILITEM CTG,FAB,FAB$L_FOP,SIZE=BIT,BITNUM=FAB$V_CTG,TYPE=BOOLEAN ;CONTIGUOUS
             1581          FILITEM CBT,FAB,FAB$L_FOP,SIZE=BIT,BITNUM=FAB$V_CBT,TYPE=BOOLEAN ;CONTIGUOUS
             158A          FILITEM RCK,FAB,FAB$L_FOP,SIZE=BIT,BITNUM=FAB$V_RCK,TYPE=BOOLEAN ;READ CHECK
             1593          FILITEM WCK,FAB,FAB$L_FOP,SIZE=BIT,BITNUM=FAB$V_WCK,TYPE=BOOLEAN ;WRITE CHEC
             159C          FILITEM CDT,XABDAT,XAB$Q_CDT,TYPE=DATE   ;CREATION DATE/TIME
             15A5          FILITEM EDT,XABDAT,XAB$Q_EDT,TYPE=DATE   ;EXPIRATION DATE/TIME
             15AE          FILITEM RDT,XABDAT,XAB$Q_RDT,TYPE=DATE   ;REVISION DATE/TIME
             15B7          FILITEM BDT,XABDAT,XAB$Q_BDT,TYPE=DATE   ;BACKUP DATE/TIME
             15C0          FILITEM RVN,XABDAT,XAB$W_RVN,SIZE=WORD   ;REVISION NUMBER
             15C9          FILITEM UIC,XABPRO,XAB$W_MBM,SIZE=LONG,TYPE=UIC  ;OWNER UIC STRING
             15D2          FILITEM GRP,XABPRO,XAB$W_GRP,SIZE=WORD   ;OWNER GROUP NUMBER
             15DB          FILITEM MBM,XABPRO,XAB$W_MBM,SIZE=WORD   ;OWNER MEMBER NUMBER
             15E4          FILITEM PRO,XABPRO,XAB$W_PRO,SIZE=WORD,TYPE=PROT  ;FILE PROTECTION BITMASK
             15ED          FILITEM EOF,XABFHC,XAB$L_EBK,SIZE=LONG,TYPE=EOF  ;END-OF-FILE BLOCK NUMBER
             15F6          FILITEM NOA,XABSUM,XAB$B_NOA,SIZE=BYTE   ;NUMBER OF AREAS
             15FF          FILITEM NOK,XABSUM,XAB$B_NOK,SIZE=BYTE   ;NUMBER OF KEYS
             1608          FILITEM PVN,XABSUM,XAB$W_PVN,SIZE=WORD   ;PROLOGUE VERSION NUMBER
             1611          FILITEM DID,NAM,NAM$W_DID,TYPE=FILEID    ;DIRECTORY ID NUMBER
```

```
                     161A          FILITEM FID,NAM,NAM$W_FID,TYPE=FILEID   ;FILE ID NUMBER
                     1623          FILITEM DVI,NAM,NAM$T_DVI,TYPE=ASCIC    ;DEVICE NAME STRING
00000000 00000000    162C          .QUAD   0
                     1634
                     1634  ;
                     1634  ; MACRO TO DEFINE VALID F$PARSE FILESPEC FIELDS
                     1634  ;
                     1634
                     1634          .MACRO  FILFLD  NAME,LENGTH,ADDRESS
                     1634          .ASCIC  "NAME"
                     1634          .BYTE   LENGTH                  ;OFFSET TO LENGTH BYTE IN NAM
                     1634          .BYTE   ADDRESS                 ;OFFSET TO ADDRESS LONGWORD IN NAM
                     1634          .ENDM
                     1634
                     1634  ;
                     1634  ; DEFINE VALID FILESPEC FIELDS WHICH MAY BE RETURNED BY F$PARSE
                     1634  ;
                     1634
                     1634  PARSE_FIELDS:
                     1634          FILFLD  NODE,NAM$B_NODE,NAM$L_NODE
                     163B          FILFLD  DEVICE,NAM$B_DEV,NAM$L_DEV
                     1644          FILFLD  DIRECTORY,NAM$B_DIR,NAM$L_DIR
                     1650          FILFLD  NAME,NAM$B_NAME,NAM$L_NAME
                     1657          FILFLD  TYPE,NAM$B_TYPE,NAM$L_TYPE
                     165E          FILFLD  VERSION,NAM$B_VER,NAM$L_VER
                  00 1668          .BYTE   0
                     1669
                     1669          .MACRO  PRSTYP  NAME,FIELD
                     1669          .ASCIC  "NAME"
                     1669          .BYTE   NAM$M_'FIELD'
                     1669          .ENDM
                     1669
                     1669  PARSE_TYPES:
                     1669          PRSTYP  SYNTAX_ONLY,SYNCHK
                     1676          PRSTYP  NO_CONCEAL,NOCONCEAL
                  00 1682          .BYTE   0
                     1683
                     1683  ;
                     1683  ; MACRO TO DEFINE VALID $GETDVI ITEMS WHICH MAY BE USED IN F$GETDVI
                     1683  ; THIS MACRO IS CALLED BY THE COMMONLY DEFINED MACRO: DVI_GENERATE_TABLE
                     1683  ;
                     1683          .MACRO  DVI_ITEM_CODE   NAME,-          ; of the item-code
                     1683                                  SPECIAL,-       ; flag
                     1683                                  SOURCE,-        ; of the data
                     1683                                  DTYPE,-         ; data type and usage
                     1683                                  BITPOS,-        ; of the 'bitval' bit
                     1683                                  OUTLEN,-        ; of the returned data
                     1683                                  STRUCT,-        ; structure containing the data
                     1683                                  DEVTYP          ; flag
                     1683
                     1683          ;
                     1683          ; SPLDEVNAM and EXISTS are special cases
                     1683          ;
                     1683          .IF IDN <NAME><SPLDEVNAM>
                     1683          .WORD   DVI$_DEVNAM
                     1683          .IFF
                     1683
```

```
                1683            .IF IDN <NAME><EXISTS>
                1683            .WORD   DVI$_DEVCHAR
                1683            .IFF
                1683            .WORD   DVI$_'NAME
                1683            .ENDC
                1683
                1683            .ENDC
                1683
                1683            $XX$ = 0
                1683            .IIF IDN <DTYPE><DECNUM>,       $XX$=0
                1683            .IIF IDN <DTYPE><PADSTR>,       $XX$=1
                1683            .IIF IDN <DTYPE><CNTSTR>,       $XX$=1
                1683            .IIF IDN <DTYPE><HEXNUM>,       $XX$=4
                1683            .IIF IDN <DTYPE><PRVMSK>,       $XX$=0
                1683            .IIF IDN <DTYPE><STDUIC>,       $XX$=2
                1683            .IIF IDN <DTYPE><STDTIM>,       $XX$=0
                1683            .IIF IDN <DTYPE><STRDSC>,       $XX$=1
                1683            .IIF IDN <DTYPE><BITVEC>,       $XX$=0
                1683            .IIF IDN <DTYPE><BITVAL>,       $XX$=3
                1683            .IIF IDN <DTYPE><PRTMSK>,       $XX$=5
                1683            .IIF IDN <DTYPE><ACPTYP>,       $XX$=6
                1683            .IIF IDN <DTYPE><HEXSTR>,       $XX$=7
                1683
                1683            .BYTE   $XX$
                1683
                1683            .IF IDN <NAME><EXISTS>
                1683            .BYTE   DEV$V_EXISTS
                1683            .IFF
                1683            .BYTE   0
                1683            .ENDC
                1683
                1683            .ASCIC  "NAME"
                1683            .ENDM   DVI_ITEM_CODE
                1683
       FFFFFFFF 1683        DEV$V_EXISTS = -1
```

```
                              1683  ;
                              1683  ;*************************************************************
                              1683  ;
                              1683  ; DEFINE VALID $GETDVI ITEMS WHICH MAY BE USED IN F$GETDVI
                              1683  ;
                              1683  ;*************************************************************
                              1683  ;
                              1683
                              1683  GETDVI_TABLE:
                              1683
                              1683  ; Each element of GETDVI_TABLE is:
                              1683  ;
                              1683  ;       ;-----------------------;
                              1683  ;       ; .WORD    DVI$_'NAME   ;
                              1683  ;       ; .BYTE    DTYPE        ;
                              1683  ;       ; .BYTE    DEV$V_EXISTS ;
                              1683  ;       ; .ASCIC   "NAME"       ;
                              1683  ;       ;-----------------------;
                              1683
                              1683          DVI_GENERATE_TABLE
                              1E3D
                              1E3D          ;
                              1E3D          ; These codes are special to LEXICON, they aren't 'real' item-codes
                              1E3D          ;
                              1E3D          DVI_ITEM_CODE SPLDEVNAM, F, DEVNAM, PADSTR, 0, 4, UCB, ANY
                              1E4B          DVI_ITEM_CODE EXISTS, F, DEVCHAR, BITVAL, 0, 1, UCB, ANY
                              1E56
      00000000 00000000       1E56          .QUAD   0                       ; END = ZERO LENGTH NAME
                              1E5E
                              1E5E  ;
                              1E5E  ; BUILD TABLE OF ACP TYPES
                              1E5E  ;
              00000001        1E5E          $XX$ = 1
                              1E5E          .MACRO  ACP NAME
                              1E5E          ASSUME  DVI$C_ACP_'NAME EQ $XX$
                              1E5E          $XX$ = $XX$ + 1
                              1E5E          .ASCIC  "NAME"
                              1E5E          .BLKB   8 - <%LENGTH(NAME)+1>
                              1E5E          .ENDM   ACP
                              1E5E
                              1E5E  ACP_TABLE:
  4C 41 47 45 4C 4C 49 00'    1E5E          .ASCIC  'ILLEGAL'
                        07    1E5E
                              1E66          ACP     F11V1
                              1E6E          ACP     F11V2
                              1E76          ACP     MTA
                              1E7E          ACP     NET
                              1E86          ACP     REM
                              1E8E          ACP     JNL
  4C 41 47 45 4C 4C 49 00'    1E96          .ASCIC  'ILLEGAL'
                        07    1E96
                              1E9E
                              1E9E  ;
                              1E9E  ; MACRO TO DEFINE VALID F$EDIT ITEM NAMES
                              1E9E  ;
                              1E9E          .MACRO  EDIT    NAME
                              1E9E          .ASCIC  "NAME"
```

```
                1E9E            .BYTE   EDIT_M_'NAME
                1E9E            .ENDM
                1E9E
                1E9E    ;
                1E9E    ; DEFINE VALID F$EDIT ITEM NAMES
                1E9E    ;
                1E9E    EDIT_TABLE:
                1E9E            ASSUME  EDIT_V_UNCOMMENT EQ 0
                1E9E            ASSUME  EDIT_V_COMPRESS EQ 1
                1E9E            ASSUME  EDIT_V_COLLAPSE EQ 2
                1E9E            ASSUME  EDIT_V_TRIM EQ 3
                1E9E            ASSUME  EDIT_V_UPCASE EQ 4
                1E9E            ASSUME  EDIT_V_LOWERCASE EQ 5
                1E9E            ASSUME  EDIT_V_EOL EQ 6
                1E9E            EDIT    UNCOMMENT
                1EA9            EDIT    COMPRESS
                1EB3            EDIT    COLLAPSE
                1EBD            EDIT    TRIM
                1EC3            EDIT    UPCASE
                1ECB            EDIT    LOWERCASE
          00    1ED6            .BYTE   0
                1ED7
                1ED7    ;
                1ED7    ; MACRO TO DEFINE VALID F$ENVIRONMENT ITEM NAMES
                1ED7    ;
    00000000    1ED7            $XX$ = 0
                1ED7            .MACRO  ENVIRON NAME
                1ED7            .ASCIC  "NAME"
                1ED7            .BYTE   $XX$
                1ED7            $XX$ = $XX$ + 1
                1ED7            .ENDM
                1ED7
                1ED7    ;
                1ED7    ; DEFINE VALID F$ENVIRONMENT ITEM NAMES
                1ED7    ;
                1ED7    ENVIRON_TABLE:
                1ED7            ENVIRON CONTROL
                1EE0            ENVIRON NOCONTROL
                1EEB            ENVIRON DEFAULT
                1EF4            ENVIRON DEPTH
                1EFB            ENVIRON ON_SEVERITY
                1F08            ENVIRON PROMPT
                1F10            ENVIRON PROMPT_CONTROL
                1F20            ENVIRON PROTECTION
                1F2C            ENVIRON MESSAGE
                1F35            ENVIRON MAX_DEPTH
                1F40            ENVIRON ON_CONTROL_Y
                1F4E            ENVIRON PROCEDURE
                1F59            ENVIRON KEY_STATE
                1F64            ENVIRON VERIFY_PROCEDURE
                1F76            ENVIRON VERIFY_IMAGE
                1F84            ENVIRON CAPTIVE
                1F8D            ENVIRON INTERACTIVE
                1F9A            ENVIRON OUTPUT_RATE
          00    1FA7            .BYTE   0
                1FA8
                1FA8    ;
```

```
          1FA8  ; MACRO TO DEFINE VALID F$TRNLNM ITEM NAMES
          1FA8  ;
          1FA8          .MACRO  LOGICAL NAME,ITEM
          1FA8          .ASCIC  "NAME"
          1FA8          .WORD   LNM$_'ITEM
          1FA8          $XX$ = 0
          1FA8          .IIF IDN <ITEM><ATTRIBUTES>,$XX$ = LNM$V_'NAME
          1FA8          .BYTE   $XX$
          1FA8          .ENDM
          1FA8
          1FA8  ;
          1FA8  ; DEFINE VALID F$TRNLNM ITEM NAMES
          1FA8  ;
          1FA8  LOGICAL_TABLE:
          1FA8          LOGICAL ACCESS_MODE,ACMODE
          1FB7          LOGICAL CONCEALED,ATTRIBUTES
          1FC4          LOGICAL CONFINE,ATTRIBUTES
          1FCF          LOGICAL CRELOG,ATTRIBUTES
          1FD9          LOGICAL LENGTH,LENGTH
          1FE3          LOGICAL MAX_INDEX,MAX_INDEX
          1FF0          LOGICAL NO_ALIAS,ATTRIBUTES
          1FFC          LOGICAL TABLE,ATTRIBUTES
          2005          LOGICAL TABLE_NAME,TABLE
          2013          LOGICAL TERMINAL,ATTRIBUTES
          201F          LOGICAL VALUE,STRING
      00  2028          .BYTE   0
          2029
          2029  ;
          2029  ; MACRO TO DEFINE VALID F$TRNLNM ATTRIBUTE TYPES
          2029  ;
          2029          .MACRO  LOGATTR NAME
          2029          .ASCIC  "NAME"
          2029          $XX$ = 0
          2029          .IIF IDN <NAME><CASE_BLIND>,$XX$ = LNM$M_'NAME
          2029          .LONG   $XX$
          2029          .ENDM
          2029
          2029  ;
          2029  ; DEFINE VALID F$TRNLNM ATTRIBUTE TYPES
          2029  ;
          2029  LOGATTR_TABLE:
          2029          LOGATTR CASE_BLIND
          2038          LOGATTR CASE_SENSITIVE
      00  204B          .BYTE   0
          204C
          204C  ;
          204C  ; MACRO TO DEFINE VALID ACCESS MODES
          204C  ;
00000000  204C          $XX$ = 0
          204C          .MACRO  ACCESS  NAME
          204C          .ASCIC  "NAME"
          204C          .BYTE   $XX$
          204C          $XX$ = $XX$ + 1
          204C          .ENDM
          204C
          204C  ;
          204C  ; DEFINE VALID ACCESS MODES (MUST BE IN PSL$C_XXX ORDER)
```

```
                    204C    ;
                    204C    ACCESS_TABLE:
                    204C            ACCESS  KERNEL
                    2054            ACCESS  EXECUTIVE
                    205F            ACCESS  SUPERVISOR
                    206B            ACCESS  USER
              00    2071            .BYTE   0
                    2072
                    2072    ;
                    2072    ; BUILD TABLE OF MESSAGE SEVERITIES
                    2072    ;
                    2072            .MACRO  SEVERITY NAME
                    2072            .ASCIC  "NAME"
                    2072            .BLKB   8 - <%LENGTH(NAME)+1>
                    2072            .ENDM   SEVERITY
                    2072
                    2072    SEVERITY_TABLE:
                    2072            ASSUME  STS$K_WARNING EQ 0
                    2072            SEVERITY        WARNING
                    207A            ASSUME  STS$K_ERROR EQ 2
                    207A            SEVERITY        ERROR
                    2082            ASSUME  STS$K_SEVERE EQ 4
                    2082            SEVERITY        SEVERE
                    208A            SEVERITY        NONE            ;PLACE HOLDER FOR NEXT "NONE"
                    2092            SEVERITY        NONE
                    209A
                    209A    ;
                    209A    ; MACRO TO DEFINE VALID F$CVTIME TYPE NAMES
                    209A    ;
        00000000    209A            $XX$ = 0
                    209A            .MACRO  CVTTYPE NAME
                    209A            .ASCIC  "NAME"
                    209A            .BYTE   $XX$
                    209A            $XX$ = $XX$ + 1
                    209A            .ENDM
                    209A
                    209A    ;
                    209A    ; DEFINE VALID F$CVTIME TYPE NAMES
                    209A    ;
                    209A    CVTIME_TYPES:
                    209A            CVTTYPE COMPARISON
                    20A6            CVTTYPE ABSOLUTE
                    20B0            CVTTYPE DELTA
              00    20B7            .BYTE   0
                    20B8
                    20B8    ;
                    20B8    ; MACRO TO DEFINE VALID F$CVTIME ITEM NAMES
                    20B8    ;
                    20B8            .MACRO  CVTIME  NAME,CSTART,CCOUNT,ASTART,ACOUNT,DSTART,DCOUNT
                    20B8            .ASCIC  "NAME"
                    20B8            .BYTE   CSTART
                    20B8            .BYTE   CCOUNT
                    20B8            .BYTE   ASTART
                    20B8            .BYTE   ACOUNT
                    20B8            .BYTE   DSTART
                    20B8            .BYTE   DCOUNT
                    20B8            .ENDM
```

```
                                    20B8
                                    20B8  ;
                                    20B8  ; DEFINE VALID F$CVTIME ITEM NAMES
                                    20B8  ; (0,0 MEANS UNSUPPORTED FOR DELTA TIMES, WEEKDAY FOR OTHERS)
                                    20B8  ;
                                    20B8  CVTIME_ITEMS:
                                    20B8          CVTIME  DATETIME,0,7,0,7,0,5
                                    20C7          CVTIME  DATE,0,3,0,3,0,1
                                    20D2          CVTIME  TIME,3,4,3,4,1,4
                                    20DD          CVTIME  DAY,2,1,0,1,0,1
                                    20E7          CVTIME  MONTH,1,1,1,1,0,0
                                    20F3          CVTIME  YEAR,0,1,2,1,0,0
                                    20FE          CVTIME  HOUR,3,1,3,1,1,1
                                    2109          CVTIME  MINUTE,4,1,4,1,2,1
                                    2116          CVTIME  SECOND,5,1,5,1,3,1
                                    2123          CVTIME  HUNDREDTH,6,1,6,2,4,1
                                    2133          CVTIME  WEEKDAY,0,0,0,0,0,0
                                00  2141          .BYTE   0
                                    2142
                                    2142  MONTH_TABLE:
2D 52 41 4D 2D 42 45 46 2D 4E 41 4A 2142          .ASCII  'JAN-FEB-MAR-APR-MAY-JUN-JUL-AUG-SEP-OCT-NOV-DEC-'
2D 4E 55 4A 2D 59 41 4D 2D 52 50 41 214E
2D 50 45 53 2D 47 55 41 2D 4C 55 4A 215A
2D 43 45 44 2D 56 4F 4E 2D 54 43 4F 2166
                                    2172  MONTH_NO_TABLE:
36 30 35 30 34 30 33 30 32 30 31 30 2172          .ASCII  '010203040506070809101112??'
32 31 31 31 30 31 39 30 38 30 37 30 217E
                              3F 3F 218A
                                    218C
                                    218C  ;
                                    218C  ; BUILD TABLE OF WEEKDAYS
                                    218C  ;
                                    218C          .MACRO  DAY NAME
                                    218C          .ASCIC  "NAME"
                                    218C          .BLKB   10 - <%LENGTH(NAME)+1>
                                    218C          .ENDM   DAY
                                    218C
                                    218C  WEEKDAY_TABLE:
                                    218C          DAY     Monday
                                    2196          DAY     Tuesday
                                    21A0          DAY     Wednesday
                                    21AA          DAY     Thursday
                                    21B4          DAY     Friday
                                    21BE          DAY     Saturday
                                    21C8          DAY     Sunday
                                    21D2
                                    21D2  ;
                                    21D2  ; MACRO TO DEFINE VALID F$IDENTIFIER KEYWORDS
                                    21D2  ;
                          00000000  21D2          $XX$ = 0
                                    21D2          .MACRO  IDENT   NAME
                                    21D2          .ASCIC  "NAME"
                                    21D2          .BYTE   $XX$
                                    21D2          $XX$ = $XX$ + 1
                                    21D2          .ENDM
                                    21D2
                                    21D2  ;
```

```
                                21D2  ; DEFINE VALID F$IDENTIFIER KEYWORDS
                                21D2  ;
                                21D2  IDENT_TABLE:
                                21D2          IDENT   NAME_TO_NUMBER
                                21E2          IDENT   NUMBER_TO_NAME
                            00  21F2          .BYTE   0
                                21F3
                                21F3  ;
                                21F3  ; MACRO TO DEFINE VALID PROCESS MODES
                                21F3  ;
                      00000000  21F3          $XX$ = 0
                                21F3          .MACRO  PRCMODE NAME
                                21F3          .ASCIC  "NAME"
                                21F3          .BYTE   $XX$
                                21F3          $XX$ = $XX$ + 1
                                21F3          .ENDM
                                21F3
                                21F3  ;
                                21F3  ; DEFINE VALID PROCESS MODES (MUST BE IN JPI$K_XXX ORDER)
                                21F3  ;
                                21F3  MODE_TABLE:
                                21F3          PRCMODE OTHER
                                21FA          PRCMODE NETWORK
                                2203          PRCMODE BATCH
                                220A          PRCMODE INTERACTIVE
                            00  2217          .BYTE   0
                                2218
                                2218  BATCH:
             48 43 54 41 42 00' 2218          .ASCIC  'BATCH'                 ;BATCH JOB
                            05  2218
                                221E  INTERACTIVE:
45 56 49 54 43 41 52 45 54 4E 49 00' 221E     .ASCIC  'INTERACTIVE'           ;INTERACTIVE JOB
                            0B  221E
                                222A  NETWORK:
          4B 52 4F 57 54 45 4E 00' 222A       .ASCIC  'NETWORK'               ;NETWORK JOB
                            07  222A
                                2232  OTHER:
             52 45 48 54 4F 00' 2232          .ASCIC  'OTHER'                 ;OTHER NON-INTERACTIVE JOB
                            05  2232
                                2238
                                2238  ;
                                2238  ; LIST MESSAGE FIELDS
                                2238  ;
                54 58 45 54 00' 2238  TEXT:   .ASCIC  'TEXT'
                            04  2238
54 41 43 49 46 49 54 4E 45 44 49 00' 223D IDENT: .ASCIC 'IDENTIFICATION'
                      4E 4F 49  2249
                            0E  223D
                                224C  SEVERITY:
       59 54 49 52 45 56 45 53 00' 224C       .ASCIC  'SEVERITY'
                            08  224C
                                2255  FACILITY:
       59 54 49 4C 49 43 41 46 00' 2255       .ASCIC  'FACILITY'
                            08  2255
                                225E
                                225E
                4C 4C 41 4F 4E  225E  NOALL:  .ASCII  'NOALL'                 ;KEYWORD FOR [NO]ALL PRIVILEGES
```

```
          45 53 4C 41 46 00' 2263   1046 FALSE:  .ASCIC  'FALSE'            ;BOOLEAN FALSE VALUE
                         05  2263
             45 55 52 54 00' 2269   1047 TRUE:   .ASCIC  'TRUE'             ;BOOLEAN TRUE VALUE
                         04  2269
                             226E   1048
       47 4E 49 52 54 53 00' 226E   1049 STRING: .ASCIC  'STRING'           ;STRING SYMBOL
                         06  226E
                             2275   1050 INTEGER:
    52 45 47 45 54 4E 49 00' 2275   1051         .ASCIC  'INTEGER'          ;INTEGER SYMBOL
                         07  2275
                             227D   1052
47 4F 4C 5F 4C 43 44 24 4D 4E 4C 00' 227D 1053 TABDEF: .ASCIC 'LNM$DCL_LOGICAL' ;DEFAULT LOGICAL NAME TABLE
                 4C 41 43 49 2289
                         0F  227D
       3A 30 41 4C 4E 5F 00' 228D   1054 NLA0:   .ASCIC  '_NLA0:'           ;NULL DEVICE
                         06  228D
```

```
    2294                .SBTTL  EVALUATE LEXICAL FUNCTION STRING
    2294
00000000                .PSECT  DCL$ZCODE,BYTE,RD,NOWRT
    0000
    0000        ;---
    0000        ; DCL$FUNCTION - EVALUATE A LEXICAL FUNCTION STRING
    0000        ;
    0000        ; THIS ROUTINE CAN BE CALLED TO EVALUATE A STRING WHICH CONTAINS
    0000        ; A LEXICAL FUNCTION AND ITS ARGUMENTS, AND RETURNS ITS VALUE.
    0000        ;
    0000        ; INPUTS:
    0000        ;
    0000        ;       R1/R2 = DESCRIPTOR OF FUNCTION/SYMBOL STRING
    0000        ;       R3/R4 = DESCRIPTOR OF SCRATCH STORAGE
    0000        ;
    0000        ; OUTPUTS:
    0000        ;
    0000        ;       R0 = STATUS
    0000        ;       R1/R2 = DESCRIPTOR OF VALUE STRING (NULL IF ERROR)
    0000        ;---
    0000                .IF DF  SUPRESS_CODE                    ;NOBODY CALLS THIS AT THE MOMENT
    0000
    0000        DCL$FUNCTION::
    0000                PUSHR   #^M<R3,R4,R5,R6,R7>             ;SAVE REGISTERS
    0000                PUSHL   WRK_L_EXPANDPTR(R10)            ;SAVE EXPANSION BUFFER POINTER
    0000                PUSHL   WRK_L_CHARPTR(R10)              ;SAVE INPUT BUFFER POINTER
    0000                STATUS  BUFOVF                          ;ASSUME BUFFER OVERFLOW
    0000                CMPL    R1,R3                           ;CHECK IF STRING TOO BIG FOR BUFFER
    0000                BGEQ    90$                             ;SKIP IT IF ITS TOO LONG
    0000                MOVL    R4,WRK_L_EXPANDPTR(R10)         ;PUT EXPANSION IN FRONT OF BUFFER
    0000                MOVAB   -2(R4)[R3],R0                   ;CALCULATE ENDING ADDRESS OF BUFFER
    0000                SUBL    R1,R0                           ;SUBTRACT LENGTH OF INPUT STRING
    0000                MOVL    R0,WRK_L_CHARPTR(R10)           ;SET INPUT CHARACTER POINTER TO STRING
    0000                MOVC    R1,(R2),(R0)                    ;STORE INPUT STRING ON STACK
    0000                CLRB    (R3)+                           ;TERMINATE INPUT STRING WITH EOL CHAR
    0000                BSBW    DCL$GETOKEN                     ;GET FUNCTION NAME STRING
    0000                MOVQ    R1,R6                           ;AND PASS TO DCL$LEXIF
    0000                BSBB    DCL$LEXIF                       ;EVALUATE LEXICAL FUNCTION
    0000        90$:    POPL    WRK_L_CHARPTR(R10)              ;RESTORE INPUT BUFFER POINTER
    0000                POPL    WRK_L_EXPANDPTR(R10)            ;RESTORE EXPANSION BUFFER POINTER
    0000                POPR    #^M<R3,R4,R5,R6,R7>             ;RESTORE REGISTERS
    0000                RSB
    0000
    0000                .ENDC
```

```
                              0000          .SBTTL  LEXICAL FUNCTION DISPATCHER
                              0000  ;+
                              0000  ; DCL$LEXIF - LEXICAL FUNCTION DISPATCH ROUTINE
                              0000  ;
                              0000  ; THIS ROUTINE IS CALLED WHEN A STRING SUBSTITUTION IS SPECIFIED AND THE STRING
                              0000  ; IS CURRENTLY UNDEFINED.
                              0000  ;
                              0000  ; INPUTS:
                              0000  ;
                              0000  ;       R6 = LENGTH OF SYMBOL NAME.
                              0000  ;       R7 = ADDRESS OF SYMBOL NAME.
                              0000  ;       R10 = ADDRESS OF COMMAND WORK AREA
                              0000  ;       R11 = ADDRESS OF PROCESS WORK AREA
                              0000  ;
                              0000  ; OUTPUTS:
                              0000  ;
                              0000  ;       THE SYMBOL NAME IS MATCHED WITH THE LEXICAL FUNCTION TABLE. IF NO MATCH
                              0000  ;       OCCURS, THEN A NULL STRING IS RETURNED FOR SUBSTITUTION IN THE ORIGINAL
                              0000  ;       COMMAND LINE. IF A MATCH OCCURS, THEN THE APPROPRIATE LEXICAL FUNCTION
                              0000  ;       IS EVALUATED AND THE RESULTANT STRING IS RETURNED.
                              0000  ;
                              0000  ;       R1/R2 = A QUADWORD DESCRIBING THE FUNCTION RESULT
                              0000  ;               IF R2 NONZERO, THEN R1/R2 IS A STRING DESCRIPTOR
                              0000  ;               IF R2 ZERO, THEN R1 IS A BINARY LONGWORD VALUE.
                              0000  ;
                              0000  ;       INPUT AND EXPANSION POINTERS ARE UPDATED AS APPROPRIATE.
                              0000  ;-
                              0000
                              0000  DCL$LEXIF::                                           ;LEXICAL FUNCTION DISPATCH
            13F8 8F     BB    0000          PUSHR   #^M<R3,R4,R5,R6,R7,R8,R9,AP> ; SAVE REGISTERS
              5C 5E     D0    0004          MOVL    SP,AP                         ;MARK STACK ADDRESS
              56 06     D1    0007          CMPL    #6,R6                         ;SPECIFIED NAME LONGER THAN 6 CHARACTERS?
                 03     1A    000A          BGTRU   10$                           ;IF GTRU NO
              56 06     D0    000C          MOVL    #6,R6                         ;REDUCE LENGTH TO 6 CHARACTERS
       87  2446 8F      B1    000F  10$:    CMPW    #^A/F$/,(R7)+                 ;POSSIBLY LEXICAL FUNCTION NAME?
                 0D     12    0014          BNEQ    60$                           ;IF NEQ NO
              FFE7'     30    0016          BSBW    DCL$SETNBLK                   ;CHECK CHARACTER AFTER FUNCTION NAME
              50 28     91    0019          CMPB    #^A/(/,R0                     ;POSSIBLY LEXICAL FUNCTION?
                 05     12    001C          BNEQ    60$                           ;IF NEQ NO
              56 02     C2    001E          SUBL    #2,R6                         ;REDUCE LENGTH OF FUNCTION NAME
                 24     14    0021          BGTR    18$                           ;IF LEQ NOT LEXICAL FUNCTION NAME
                 50     7C    0023  60$:    CLRQ    R0                            ;FAILURE, NO SUCH FUNCTION, NULL VALUE
              52 5E     D0    0025          MOVL    SP,R2                         ;INDICATE RESULT IS A STRING
               015F     31    0028          BRW     90$                           ;EXIT
                              002B  70$:    STATUS  IVFNAM                        ;SET INVALID FUNCTION NAME STATUS
                 07     11    0032          BRB     85$
                              0034  80$:    STATUS  ABFNAM                        ;SET AMBIGUOUS FUNCTION NAME STATUS
         67  80 8F      8A    003B  85$:    BICB    #^X80,(R7)                    ;UNDO MATCH OPTIMIZATION
       F48A CA  77      3E    003F          MOVAW   -(R7),WRK_L_MARKPTR(R10) ;SET STARTING ADDRESS OF FUNCTION NAME
               160A     31    0044          BRW     DCL$PARSERR
                              0047
                              0047  ;
                              0047  ; SEARCH LEXICAL FUNCTION NAME TABLE FOR A MATCH
                              0047  ;
              FFB6'     30    0047  18$:    BSBW    DCL$MOVCHAR                   ;COPY ( CHARACTER
         67  80 8F      88    004A          BISB    #^X80,(R7)                    ;SET FORCE MATCH FLAG
 0100 8F  67  56        39    004E          MATCHC  R6,(R7),#LXFUNTABLEN,LXFUNTABLE ;SEARCH FOR FUNCTION NAME MATCH
```

```
              00000000'EF      0054
                       D0  12  0059          BNEQ    70$                     ;BRANCH IF NO MATCH FOUND
                 52    56  C0  005B          ADDL    R6,R2                   ;CALCULATE BYTES REMAINING INCLUDING SUBSTRI
            58   52    08  C7  005E          DIVL3   #ARG_C_ENTSIZE,R2,R8    ;CALCULATE COMPLETE TABLE ENTRY NUMBER
   63       0A   67    56  39  0062          MATCHC  R6,(R7),#10,(R3)        ;SCAN FOR DUPLICATE MATCH
                       CB  13  0067          BEQL    80$                     ;IF EQL DUPLICATE MATCH FOUND
                               0069  ;
                               0069  ; DETERMINE THE FUNCTION INDEX (R8) AND RETRIEVE THE NUMBER OF ARGUMENTS
                               0069  ; THAT THE FUNCTION TAKES AS WELL AS THE ARGUMENT TYPES (NUMERIC OR STRING).
                               0069  ;
            58   20    58  C3  0069          SUBL3   R8,#LXFUNTABLEN/ARG_C_ENTSIZE,R8 ;CALCULATE REAL FUNCTION INDEX
                               006D          ASSUME  ARG_C_ENTSIZE EQ 8
   57       00000004'EF48  7E  006D          MOVAQ   LXFUNTABLE+4[R8],R7     ;POINT TO ARGUMENT INFORMATION LONGWORD
                 57    67  D0  0075          MOVL    (R7),R7                 ;SAVE IT IN REGISTER
                               0078  ;
                               0078  ; PARSE EACH FUNCTION ARGUMENT, STORING EITHER THE STRING DESCRIPTOR OR
                               0078  ; THE NUMERIC VALUE IN THE ARGUMENT DESCRIPTORS.  STRINGS ARE SAVED ON
                               0078  ; THE STACK IN ORDER TO MAKE FUNCTIONS RECURSIVE.
                               0078  ;
            5E   FF7C  CE  9E  0078          MOVAB   -<MAX_ARGS*8+4>(SP),SP  ;ALLOCATE ARGUMENT DESCRIPTORS + COUNT
0080 8F     FF 8F  6E  00  2C  007D          MOVC5   #0,(SP),#^XFF,#MAX_ARGS*8,4(SP) ;INIT ALL ARGS TO MISSING (-1)
              04 AE            0085
                       5C  DD  0087          PUSHL   AP                      ;SAVE STARTING STACK ADDRESS IN -4(AP)
           5C    04 AE     9E  0089          MOVAB   4(SP),AP                ;POINT TO START OF ARGUMENT LIST
                 55    5C  D0  008D          MOVL    AP,R5                   ;ADDRESS OF NEXT DESCRIPTOR TO FILL
                       85  D4  0090          CLRL    (R5)+                   ;INITIALIZE # ARGUMENTS SPECIFIED
                               0092
                    FF6B'  30  0092  20$:    BSBW    DCL$SETNBLK             ;SKIP LEADING BLANKS AND TABS
                    FF68'  30  0095          BSBW    DCL$MARK                ;MARK CURRENT PARSE POSITION
                 57    6C  91  0098          CMPB    NPARM(AP),R7            ;ANY MORE ARGUMENTS TO PROCESS?
                       03  19  009B          BLSS    21$                     ;IF GEQ YES
                     00B5  31  009D          BRW     50$                     ;END OF ARGUMENTS
                 56    03  D0  00A0  21$:    MOVL    #ARG_C_ANY,R6           ;ASSUME EITHER BINARY OR STRING
                 06    6C  91  00A3          CMPB    NPARM(AP),#6            ;BEYOND EXPLICITLY DEFINED ARGS?
                       0C  18  00A6          BGEQ    23$                     ;IF SO, ASSUME "ANY"
            56   6C    02  78  00A8          ASHL    #2,NPARM(AP),R6         ;CALCULATE OFFSET TO NIBBLE
                 56    08  C0  00AC          ADDL    #8,R6                   ;BIT POSITION OF NIBBLE IN R7
   56       57   08    56  EF  00AF          EXTZV   R6,#8,R7,R6             ;EXTRACT NIBBLE DESCRIBING ARGUMENT
                               00B4                                          ;AND NIBBLE DESCRIBING NEXT ARGUMENT
                               00B4                                          ;AS WELL, FOR END-OF-LIST CHECKS
                               00B4  ;
                               00B4  ; CHECK IF ARGUMENT MISSING
                               00B4  ;
                 2C    50  91  00B4  23$:    CMPB    R0,#^A','               ;CHECK IF ARGUMENT NULL
                       44  13  00B7          BEQL    28$                     ;BRANCH IF NULL
                 29    50  91  00B9          CMPB    R0,#^A')'               ;CHECK IF ARGUMENT NULL
                       3F  13  00BC          BEQL    28$                     ;BRANCH IF NULL
                               00BE  ;
                               00BE  ; PARSE ARGUMENT EXPRESSION
                               00BE  ;
   02       56   02    00  ED  00BE          CMPZV   #ARG_V_TYPE,#ARG_S_TYPE,R6,#ARG_C_SYMREF ;REFERENCE TO SYMBOL?
                       42  13  00C3          BEQL    30$                     ;BRANCH IF SO
                    FF38'  30  00C5          BSBW    DCL$EXPRESS             ;GET VALUE OF EXPRESSION
                    26 50  E9  00C8          BLBC    R0,22$                  ;BRANCH IF ERROR DETECTED
                               00CB  ;
                               00CB  ; VERIFY THAT THE EXPRESSION ARGUMENT WAS THE EXPECTED TYPE (STRING OR BINARY)
                               00CB  ; IF IT WASN'T, THEN CONVERT THE EXPRESSION TO THE PROPER MODE, IF POSSIBLE.
```

```
                               00CB  ;
 03   56   02   00    ED  00CB        CMPZV   #ARG_V_TYPE,#ARG_S_TYPE,R6,#ARG_C_ANY ;ANY TYPE ALLOWED?
                3D    13  00D0        BEQL    40$                       ;BRANCH IF SO
                52    D5  00D2        TSTL    R2                        ;CHECK IF STRING OR BINARY LONGWORD
                          00D4        ASSUME  ARG_C_BINARY EQ 0
                          00D4        ASSUME  ARG_C_STRING EQ 1
           0D   56    E8  00D4        BLBS    R6,27$                    ;BRANCH IF EXPECTED STRING ARGUMENT
                36    13  00D7        BEQL    40$                       ;OK IF EXPECTED NUMERIC AND GOT IT
       00000000'EF    16  00D9        JSB     DCL$CVT_BINARY            ;CONVERT STRING TO BINARY
           0F   50    E9  00DF        BLBC    R0,22$                    ;BRANCH IF ERROR DETECTED
                2B    11  00E2        BRB     40$
                29    12  00E4  27$:  BNEQ    40$                       ;OK IF EXPECTED STRING AND GOT IT
           50   51    D0  00E6        MOVL    R1,R0                     ;SET BINARY VALUE TO CONVERT
       00000000'EF    16  00E9        JSB     DCL$CBTA_DEC              ;CONVERT BINARY VALUE TO DECIMAL STRING
                1E    11  00EF        BRB     40$                       ;USE CONVERTED STRING AS ARGUMENT
                          00F1  ;
                          00F1  ; AN ERROR HAS OCCURRED.  IF WE ARE IN A COMMENT, JUST STORE
                          00F1  ; AN ARGUMENT OF NULL OR ZERO AND CONTINUE.
                          00F1  ;
   03 F0 AA     0C    E0  00F1  22$:  BBS     #WRK_V_COMMENT,WRK_W_FLAGS(R10),24$ ;SKIP IF IN COMMENT
              1558    31  00F6        BRW     DCL$PARSERR               ;OTHERWISE, REPORT THE ERROR
                51    7C  00F9  24$:  CLRQ    R1                        ;IF IN COMMENT, CONTINUE WITH A ZERO
                12    11  00FB        BRB     40$
                          00FD  ;
                          00FD  ; PROCESS AN EMPTY ARGUMENT.  IF ALLOWED, -1 WILL APPEAR IN THE QUADWORD
                          00FD  ; TO INDICATE MISSING ARGUMENT.
                          00FD  ;
      31 56     02    E0  00FD  28$:  BBS     #ARG_V_OPTIONAL,R6,43$    ;BRANCH IF OPTIONAL VALUES ALLOWED
             FEFC'    30  0101        BSBW    DCL$MOVCHAR               ;MOVE DELIMITER TO EXPANSION BUFFER
              0088    31  0104  29$:  BRW     95$                       ;AND REPORT MISSING ARGUMENT
                          0107  ;
                          0107  ; PARSE ARGUMENT TO BE STORED BY REFERENCE RATHER THAN BY VALUE
                          0107  ;
             FEF6'    30  0107  30$:  BSBW    DCL$GETOKEN               ;GET ARGUMENT SYMBOL OR LITERAL
           22   62    91  010A        CMPB    (R2),#^A'"'               ;QUOTED STRING?
                F5    13  010D        BEQL    29$                       ;IF SO, ERROR - NOT BY REFERENCE
                          010F  ;
                          010F  ; STORE STRING ARGUMENT ON STACK TO PRESERVE IT OVER RECURSION
                          010F  ;
                52    D5  010F  40$:  TSTL    R2                        ;STRING ARGUMENT?
                1C    13  0111        BEQL    42$                       ;IF NOT, JUST STORE LONGWORD
      03 56     03    E1  0113        BBC     #ARG_V_KEYWORD,R6,41$     ;KEYWORD ARGUMENT?
             FEE6'    30  0117        BSBW    DCL$SQUEEZE               ;EDIT THE KEYWORD LIST
           5E   51    C2  011A  41$:  SUBL    R1,SP                     ;ALLOCATE SPACE FOR STRING ON STACK
           85   51    D0  011D        MOVL    R1,(R5)+                  ;SAVE STRING LENGTH
           85   5E    D0  0120        MOVL    SP,(R5)+                  ;SAVE (COPIED) STRING ADDRESS
                55    DD  0123        PUSHL   R5                        ;SAVE REGISTERS
 04 AE     62   51    28  0125        MOVC    R1,(R2),4(SP)             ;SAVE ARGUMENT STRING ON STACK
             55 8ED0      012A        POPL    R5                        ;RESTORE REGISTERS
                06    11  012D        BRB     45$
                          012F
           65   51    7D  012F  42$:  MOVQ    R1,(R5)                   ;STORE VALUE IN QUADWORD
           55   08    C0  0132  43$:  ADDL    #8,R5                     ;SKIP TO NEXT ARGUMENT DESCRIPTOR
                6C    D6  0135  45$:  INCL    NPARM(AP)                 ;INCREMENT # ARGUMENTS FOUND
             FEC6'    30  0137        BSBW    DCL$SETNBLK               ;SKIP TRAILING SPACES
           57   6C    91  013A        CMPB    NPARM(AP),R7              ;LAST PARAMETER TO PROCESS?
                16    13  013D        BEQL    50$                       ;IF EQL YES
```

```
                    FEBE'  30  013F        BSBW    DCL$MOVCHAR                     ;GET NEXT INPUT CHARACTER
               50     2C   91  0142        CMPB    #^A/,/,R0                       ;TRAILING COMMA?
                      03   12  0145        BNEQ    48$                             ;BRANCH IF NOT
                    FF48   31  0147        BRW     20$                             ;LOOP THRU ALL ARGUMENTS
               06     57   91  014A 48$:   CMPB    R7,#6                           ;VARIABLE # ARGUMENTS?
                      09   1A  014D        BGTRU   55$                             ;BRANCH IF SO
            05 56     06   E0  014F        BBS     #4+ARG_V_OPTIONAL,R6,55$  ;OK IF NEXT ARGUMENT IS OPTIONAL
                      3A   11  0153        BRB     95$                             ;ERROR IF REQUIRED ARGUMENT NOT PRESENT
                               0155
                    FEA8'  30  0155 50$:   BSBW    DCL$MOVCHAR                     ;GET NEXT INPUT CHARACTER
               50     29   91  0158 55$:   CMPB    #^A/)/,R0                       ;TRAILING RIGHT PARENTHESIS?
                      3B   12  015B        BNEQ    100$                            ;IF NEQ NO
                    FEA0'  30  015D        BSBW    DCL$MARK                        ;MARK CURRENT PARSE POSITION
                               0160 ;
                               0160 ; CONSTRUCT A DESCRIPTOR OF SCRATCH BUFFER SPACE AND CALL THE APPROPRIATE
                               0160 ; LEXICAL FUNCTION ROUTINE TO OBTAIN THE RESULT STRING.
                               0160 ;
               53    F486 CA   D0  0160        MOVL    WRK_L_EXPANDPTR(R10),R3 ;BUILD DESCRIPTOR OF SCRATCH AREA
      52   F48E CA   F486 CA   C3  0165        SUBL3   WRK_L_EXPANDPTR(R10),WRK_L_CHARPTR(R10),R2 ;SET UNUSED LENGTH
                      4C   10  016D        BSBB    DISPATCH                        ;DISPATCH FUNCTION
                               016F ;
                               016F ; R1/R2 NOW DESCRIBE THE FUNCTION RESULT:
                               016F ;         IF R2 NONZERO, R1/R2 ARE A STRING DESCRIPTOR
                               016F ;         IF R2 ZERO, R1 IS A BINARY LONGWORD VALUE
                               016F ;
                               016F ; RESULTANT STRING MIGHT NOW RESIDE ON THE STACK (E.G. AN EXTRACT
                               016F ; OF AN ARGUMENT STORED ON THE STACK).  WE MUST COPY IT TO THE EXPANSION
                               016F ; BUFFER BEFORE RETURNING IT TO THE CALLER.
                               016F ;
                      52   D5  016F        TSTL    R2                              ;CHARACTER STRING?
                      11   13  0171        BEQL    58$                             ;BRANCH IF NOT
               56     51   D0  0173        MOVL    R1,R6                           ;SAVE LENGTH OF STRING
       F486 DA  62    51   28  0176        MOVC    R1,(R2),@WRK_L_EXPANDPTR(R10) ;COPY IT TO UNUSED EXPANSION AREA
               51     56   D0  017C        MOVL    R6,R1                           ;RETURN LENGTH OF RESULT STRING
               52    F486 CA   D0  017F        MOVL    WRK_L_EXPANDPTR(R10),R2 ;AND ADDRESS OF RESULT STRING
               50     01   D0  0184 58$:   MOVL    #1,R0                           ;SUCCESS
               5E     7C   D0  0187        MOVL    -(AP),SP                        ;REMOVE ALL TEMPORARIES FROM STACK
               13F8 8F     BA  018A 90$:   POPR    #^M<R3,R4,R5,R6,R7,R8,R9,AP> ; RESTORE REGISTERS
                      05       018E        RSB
                               018F
                               018F ;
                               018F ; INVALID SYMBOL DELIMITER
                               018F ;
          50   00038268 8F     D0  018F 95$:   MOVL    #CLI$_ARGREQ,R0                 ;SET ARGUMENT REQUIRED STATUS
                      07   11  0196        BRB     105$                            ;
                               0198 100$:  STATUS  SYMDEL                          ;SET INVALID SYMBOL DELIMITER STATUS
       F486 CA   F48E CA   D0  019F 105$:  MOVL    WRK_L_CHARPTR(R10),WRK_L_EXPANDPTR(R10) ;SET ADDRESS OF NEXT CHARACT
  F48A CA   F486 CA   01   C3  01A6        SUBL3   #1,@RK_L_EXPANDPTR(R10),@RK_L_MARKPTR(R10) ;SET ADDRESS OF INVALID D
                 F48E DA   95  01AE        TSTB    @WRK_L_CHARPTR(R10)             ;AT END OF LINE?
                      04   12  01B2        BNEQ    110$                            ;IF NEQ NO
                 F48E CA   D7  01B4        DECL    WRK_L_CHARPTR(R10)              ;BACK UP TO END OF LINE
                    1496   31  01B8 110$:  BRW     DCL$PARSERR                     ;
```

```
01BB   ;
01BB   ; DISPATCH TO PROPER LEXICAL FUNCTION ROUTINE
01BB   ;
01BB   ; ON INPUT TO ROUTINE:
01BB   ;
01BB   ;       R2/R3 = DESCRIPTOR OF SCRATCH AREA
01BB   ;       R10 = ADDRESS OF PARSING WORK AREA (WRK)
01BB   ;       R11 = ADDRESS OF PROCESS WORK AREA (PRC)
01BB   ;
01BB   ;       P1(AP) = DESCRIPTOR OF FIRST ARGUMENT VALUE
01BB   ;       P2(AP) = DESCRIPTOR OF SECOND ARGUMENT VALUE
01BB   ;       P3(AP) = DESCRIPTOR OF THIRD ARGUMENT VALUE
01BB   ;       ETC. (UP TO MAX_ARGS)
01BB   ;
01BB   ; ON OUTPUT FROM ROUTINE:
01BB   ;
01BB   ;       R1/R2 = A QUADWORD DESCRIBING THE FUNCTION RESULT
01BB   ;               IF R2 NONZERO, THEN R1/R2 IS A STRING DESCRIPTOR
01BB   ;               IF R2 ZERO, THEN R1 IS A BINARY LONGWORD VALUE.
01BB   ;
01BB   ;       R3-R9 DESTROYED.
01BB   ;
01BB
01BB   DISPATCH:
01BB           CASE    R8,<-
01BB                   F$CVSINTEGER,-          ;CONVERT SIGNED INTEGER
01BB                   F$CVUINTEGER,-          ;CONVERT UNSIGNED INTEGER
01BB                   F$CVTIME,-              ;CONVERT TO COMPARABLE DATE/TIME
01BB                   F$DIRECTORY,-           ;DEFAULT DIRECTORY
01BB                   F$EDIT,-                ;EDIT A STRING
01BB                   F$ELEMENT,-             ;FIND A STRING ELEMENT
01BB                   F$ENVIRONMENT,-         ;GET DCL INFORMATION
01BB                   F$EXTRACT,-             ;EXTRACT SUBSTRING FROM STRING
01BB                   F$FAO,-                 ;FORMATTED ASCII OUTPUT
01BB                   F$FILE,-                ;GET FILE ATTRIBUTES
01BB                   F$GETDVI,-              ;GETDVI SYSTEM SERVICE
01BB                   F$GETJPI,-              ;GETJPI SYSTEM SERVICE
01BB                   F$GETSYI,-              ;GETSYI SYSTEM SERVICE
01BB                   F$IDENTIFIER,-          ;CONVERT IDENTIFIER
01BB                   F$INTEGER,-             ;CONVERT TO INTEGER
01BB                   F$LENGTH,-              ;STRING LENGTH
01BB                   F$LOCATE,-              ;LOCATE SUBSTRING
01BB                   F$LOGICAL,-             ;LOGICAL NAME TRANSLATION
01BB                   F$MESSAGE,-             ;GET MESSAGE
01BB                   F$MODE,-                ;JOB MODE
01BB                   F$PARSE,-               ;PARSE FILE SPECIFICATION
01BB                   F$PID,-                 ;GET NEXT PROCESS ID
01BB                   F$PRIVILEGE,-           ;TRUE IF PRIVS SET/CLEAR
01BB                   F$PROCESS,-             ;PROCESS NAME
01BB                   F$SEARCH,-              ;SEARCH FOR NEXT WILDCARD FILESPEC
01BB                   F$SETPRV,-              ;SETPRV SYSTEM SERVICE
01BB                   F$STRING,-              ;CONVERT TO ASCII STRING
01BB                   F$TIME,-                ;CURRENT TIME AND DATE
01BB                   F$TRNLNM,-              ;LOGICAL NAME TRANSLATION
01BB                   F$TYPE,-                ;TYPE OF SYMBOL
01BB                   F$USER,-                ;USER IDENTIFICATION
01BB                   F$VERIFY,-              ;VERIFY MODE
```

```
            01BB  1380          >                       ;
     51  D4 01FF  1381     CLRL  R1                     ;RETURN NULL STRING
         05 0201  1382     RSB                          ;
```

```
                                  0202        .SBTTL  CONVERT SIGNED AND UNSIGNED INTEGER FIELD
                                  0202  ;+
                                  0202  ; F$CVSINTEGER - CONVERT SIGNED INTEGER
                                  0202  ; F$CVUINTEGER - CONVERT UNSIGNED INTEGER
                                  0202  ;
                                  0202  ; THIS LEXICAL FUNCTION RETURNS THE BINARY DATA EXTRACTED FROM AN ARBITRARY
                                  0202  ; FIELD IN A STRING.
                                  0202  ;
                                  0202  ;       F$CVSINTEGER(POSITION, SIZE, STRING)
                                  0202  ;       F$CVUINTEGER(POSITION, SIZE, STRING)
                                  0202  ;
                                  0202  ; INPUTS:
                                  0202  ;
                                  0202  ;       P1(AP) = STARTING BIT NUMBER
                                  0202  ;       P2(AP) = NUMBER OF BITS
                                  0202  ;       P3(AP) = DESCRIPTOR OF TEXT STRING.
                                  0202  ;
                                  0202  ; OUTPUTS:
                                  0202  ;
                                  0202  ;       R1 = CONVERTED BINARY LONGWORD
                                  0202  ;       R2 = 0 TO INDICATE BINARY DATA TYPE
                                  0202  ;-
                                  0202
                                  0202        .ENABL  LSB
                                  0202  F$CVSINTEGER:                                     ;CONVERT SIGNED INTEGER
                   55    01  D0   0202        MOVL    #1,R5                             ;SET SIGNED CONVERSION INDICATOR
                         02  11   0205        BRB     10$                               ;
                                  0207  F$CVUINTEGER:                                     ;CONVERT UNSIGNED INTEGER
                         55  D4   0207        CLRL    R5                                ;SET UNSIGNED CONVERSION INDICATOR
             0C AC       20  D1   0209  10$:  CMPL    #32,P2(AP)                        ;SIZE VALUE GREATER THAN 1 LONGWORD?
                         30  1F   020D        BLSSU   90$                               ;IF GEQU NO
                         51  D4   020F        CLRL    R1                                ;ASSUME FIELD IS NOT DEFINED
       53    04 AC    0C AC  C1   0211        ADDL3   P2(AP),P1(AP),R3                  ;CALCULATE ENDING BIT POSITION
          52    14 AC    08  C5   0217        MULL3   #8,P3(AP),R2                      ;CALCULATE SIZE OF STRING IN BITS
             52       04 AC  D1   021C        CMPL    P1(AP),R2                         ;FIELD POSITION BEYOND END OF STRING?
                         1D  1E   0220        BGEQU   90$                               ;IF GEQU YES
                   52    53  C2   0222        SUBL    R3,R2                             ;FIELD POSITION TOTALLY WITHIN STRING?
                         18  19   0225        BLSS    90$                               ;IF GEQ YES
                      0A 55  E8   0227        BLBS    R5,50$                            ;IF LBS SIGNED CONVERSION
 51    18 BC    0C AC    04 AC  EF   022A        EXTZV   P1(AP),P2(AP),@P3+4(AP),R1 ;EXTRACT FIELD
                         08  11   0232        BRB     60$                               ;
 51    18 BC    0C AC    04 AC  EE   0234  50$:  EXTV    P1(AP),P2(AP),@P3+4(AP),R1 ;EXTRACT SIGNED FIELD
                         52  D4   023C  60$:  CLRL    R2                                ;MARK RESULT IS BINARY DATA
                         05       023E        RSB                                       ;
                                  023F
       50    00038228 8F  D0   023F  90$:  MOVL    #CLI$_INVRANGE,R0                 ;SET INVALID RANGE MESSAGE
                    13CA  31   0246        BRW     ERROR                             ;SIGNAL ERROR
                                  0249        .DSABL  LSB
```

```
                              0249        .SBTTL  CONVERT TO COMPARABLE DATE/TIME STRING
                              0249  ;+
                              0249  ; F$CVTIME - CONVERT TO COMPARABLE DATE/TIME STRING
                              0249  ;
                              0249  ; THIS FUNCTION CONVERTS A STANDARD VMS DATE/TIME STRING INTO A
                              0249  ; STRING OF THE FORM "YYYY-MM-DD HH:MM:SS.NN", WHICH CAN BE COMPARED
                              0249  ; TO ANOTHER STRING OF THE SAME FORM USING THE STRING COMPARISION
                              0249  ; OPERATORS.  IT CAN ALSO BE USED TO GENERATE A STANDARDIZED DATE/TIME
                              0249  ; STRING OR TO EXTRACT ITEMS FROM A STARDIZED DATE/TIME STRING.
                              0249  ;
                              0249  ;       F$CVTIME(TIME, ITEM)
                              0249  ;
                              0249  ; INPUTS:
                              0249  ;
                              0249  ;       P1(AP) = DESCRIPTOR OF DATE/TIME STRING
                              0249  ;       P2(AP) = DESCRIPTOR OF TIME TYPE
                              0249  ;       P3(AP) = DESCRIPTOR OF ITEM KEYWORD
                              0249  ;
                              0249  ; OUTPUTS:
                              0249  ;
                              0249  ;       R1/R2 = DESCRIPTOR OF REQUESTED DATE/TIME STRING
                              0249  ;-
                              0249  TIME_DELIMS:
               20 2E 2D 3A    0249        .ASCII  /:-. /
                              024D
                              024D  F$CVTIME:
                              024D  ;
                              024D  ; Build the result buffer descriptor.
                              024D  ;
        7E    53    18    C1  024D        ADDL3   #24,R3,-(SP)                   ;BUILD TIME BUFFER DESCRIPTOR
              59    53    D0  0251        MOVL    R3,R9                          ;SAVE ADDR OF SECOND DESCR
              7E    52    D0  0254        MOVL    R2,-(SP)                       ;
              6E    2F    D1  0257        CMPL    #24+23,(SP)                    ;TIME BUFFER LARGE ENOUGH?
                    06    1B  025A        BLEQU   10$                            ;IF LEQU YES
              5E    08    C0  025C        ADDL    #2*4,SP                        ;RESTORE THE STACK
                  13AA    31  025F        BRW     BUFOVF                         ;REPORT BUFFER OVERFLOW
                              0262
                              0262  ;
                              0262  ; Get the type information.
                              0262  ;
                    58    D4  0262  10$:  CLRL    R8                             ;ASSUME NO TYPE SPECIFIED
                 0C AC    D5  0264        TSTL    P2(AP)                         ;WAS A TYPE SPECIFIED?
                    30    19  0267        BLSS    20$                            ;NO, THEN USE DEFAULT
      54     0000209A'EF  9E  0269        MOVAB   CVTIME_TYPES,R4                ;GET ADDRESS OF ITEM TABLE
              55    64    9A  0270  15$:  MOVZBL  (R4),R5                        ;GET LENGTH OF ITEM
                    12    13  0273        BEQL    16$                            ;BRANCH IF END OF TABLE
55   00   10 BC  0C AC    2D  0275        CMPC5   P2(AP),@P2+4(AP),#0,R5,1(R4)   ;DOES ITEM MATCH
                 01 A4        027C
                    14    13  027E        BEQL    17$                            ;BRANCH IF SO
        54     02 A445    9E  0280        MOVAB   2(R4)[R5],R4                   ;SKIP TO NEXT TABLE ENTRY
                    E9    11  0285        BRB     15$                            ;
              5E    08    C0  0287  16$:  ADDL    #2*4,SP                        ;RESTORE THE STACK
                              028A        STATUS  IVKEYW                         ;SET STATUS
                  138A    31  0291        BRW     P2_ERROR                       ;REPORT KEYWORD ERROR
        58     01 A445    9A  0294  17$:  MOVZBL  1(R4)[R5],R8                   ;SET THE TYPE
                              0299
                              0299  ;
```

```
                          0299  ; Call LIB$CVT_TIME or LIB$CVT_DTIME to get the time string.
                          0299  ;
        55     5E     D0  0299  20$:   MOVL     SP,R5                          ;GET ADDRESS OF BUFFER DESCR
               7E     7C  029C         CLRQ     -(SP)                          ;ALLOC TIME QUADWORD
               53     D4  029E         CLRL     R3                             ;SET ADDRESS OF INPUT DESCR
            04 AC     D5  02A0         TSTL     P1(AP)                         ;WAS P1 SPECIFIED?
               0D     19  02A3         BLSS     23$                            ;NO, THEN SKIP
        51  04 AC     7D  02A5         MOVQ     P1(AP),R1                      ;GET INPUT DESCRIPTOR
             FD54'    30  02A9         BSBW     DCL$TRIM                       ;UPCASE AND TRIM IT
     04 AC     51     7D  02AC         MOVQ     R1,P1(AP)                      ;RETURN THE DESCRIPTOR
               03     11  02B0         BRB      25$                            ;
            04 AC     7C  02B2  23$:   CLRQ     P1(AP)                         ;USE A NULL INPUT DESCRIPTOR
        53  04 AC     7E  02B5  25$:   MOVAQ    P1(AP),R3                      ;GET INPUT DESCRIPTOR ADDR
               55     DD  02B9         PUSHL    R5                             ;PUSH ADDRESS OF DESCRIPTOR TWICE
               55     DD  02BB         PUSHL    R5                             ;
            F8 A5     7F  02BD         PUSHAQ   -8(R5)                         ;PUSH TIME BUFFER ADDR
               53     DD  02C0         PUSHL    R3                             ;PUSH INPUT DESCRIPTOR
        02     58     D1  02C2         CMPL     R8,#2                          ;IS DELTA TIME REQUESTED?
               17     13  02C5         BEQL     27$                            ;YES, THEN BRANCH
                          02C7
  00000000'EF  04     FB  02C7         CALLS    #4,LIB$CVT_TIME                ;PARSE THE TIME SPECIFICATION
            24 50     E8  02CE         BLBS     R0,30$                         ;BRANCH IF SUCCESS
        5E     10     C0  02D1         ADDL     #16,SP                         ;RESTORE THE STACK
                          02D4         STATUS   IVATIME                        ;SET ERROR STATUS
           133A       31  02DB         BRW      P1_ERROR                       ;REPORT TIME SYNTAX ERROR
                          02DE
  00000000'EF  04     FB  02DE  27$:   CALLS    #4,LIB$CVT_DTIME               ;PARSE THE TIME SPECIFICATION
            0D 50     E8  02E5         BLBS     R0,30$                         ;BRANCH IF SUCCESS
        5E     10     C0  02E8         ADDL     #16,SP                         ;RESTORE THE STACK
                          02EB         STATUS   IVDTIME                        ;SET ERROR STATUS
           1323       31  02F2         BRW      P1_ERROR                       ;REPORT TIME SYNTAX ERROR
                          02F5                                                 ;
                          02F5  ;
                          02F5  ; Get the item information.
                          02F5  ;
     54    000020B8'EF    9E  02F5  30$:   MOVAB    CVTIME_ITEMS,R4                ;GET ADDRESS OF ITEM TABLE
               55 64      9A  02FC         MOVZBL   (R4),R5                        ;GET LENGTH OF ITEM
        54  01 A445       9E  02FF         MOVAB    1(R4)[R5],R4                   ;GET ADDRESS OF DATA
        7E     6448       3C  0304         MOVZWL   (R4)[R8],-(SP)                 ;GET DEFAULT DATA
            14 AC         D5  0308         TSTL     P3(AP)                         ;WAS AN ITEM SPECIFIED?
               3E         19  030B         BLSS     40$                            ;NO, THEN USE DEFAULT
     54    000020B8'EF    9E  030D         MOVAB    CVTIME_ITEMS,R4                ;GET ADDRESS OF ITEM TABLE
               55 64      9A  0314  35$:   MOVZBL   (R4),R5                        ;GET LENGTH OF ITEM
               12         13  0317         BEQL     36$                            ;BRANCH IF END OF TABLE
 55   00   18 BC  14 AC   2D  0319         CMPC5    P3(AP),@P3+4(AP),#0,R5,1(R4)   ;DOES ITEM MATCH
            01 A4             0320
               14         13  0322         BEQL     37$                            ;BRANCH IF SO
        54  07 A445       9E  0324         MOVAB    7(R4)[R5],R4                   ;SKIP TO NEXT TABLE ENTRY
               E9         11  0329         BRB      35$                            ;
        5E     14         C0  032B  36$:   ADDL     #20,SP                         ;RESTORE THE STACK
                              032E         STATUS   IVKEYW                         ;SET STATUS
           12EC           31  0335         BRW      P3_ERROR                       ;REPORT KEYWORD ERROR
        54  01 A445       9E  0338  37$:   MOVAB    1(R4)[R5],R4                   ;GET ADDRESS OF DATA
        6E     6448       3C  033D         MOVZWL   (R4)[R8],(SP)                  ;SET THE POSITION AND LENGTH
               08         12  0341         BNEQ     40$                            ;IF NONZERO, THEN CONTINUE
        02     58         D1  0343         CMPL     R8,#2                          ;IS DELTA TIME REQUESTED?
               E3         13  0346         BEQL     36$                            ;YES, THEN ERROR
```

```
               0081   31  0348         BRW     90$                            ;NO, THEN RETURN WEEKDAY
                          034B
                          034B  ;
                          034B  ; Return requested information.
                          034B  ;
           56    0C AE   7D  034B  40$:    MOVQ    12(SP),R6                      ;GET TIME STRING DESCRIPTOR
                    58   D5  034F          TSTL    R8                             ;WAS ''COMPARISON'' SPECIFIED?
                    42   12  0351          BNEQ    60$                            ;NO, THEN BRANCH
                          0353
                          0353  ;
                          0353  ; Convert to ASCII comparable time.
                          0353  ;
           56       17   D1  0353          CMPL    #23,R6                         ;IS DAY ONE DIGIT OR TWO?
                    05   13  0356          BEQL    50$                            ;BRANCH IF TWO
           77       30   90  0358          MOVB    #^A'0',-(R7)                   ;INSERT A ZERO
                    56   D6  035B          INCL    R6                             ;INCREASE TIME LENGTH
     54    00002142'EF   9E  035D  50$:    MOVAB   MONTH_TABLE,R4                 ;POINT TO MONTH TABLE
                    50   D4  0364          CLRL    R0                             ;START WITH MONTH #0
           84    03 A7   D1  0366  51$:    CMPL    3(R7),(R4)+                    ;DOES MONTH MATCH?
                    04   13  036A          BEQL    52$                            ;BRANCH IF FOUND
        F6 50       0C   F2  036C          AOBLSS  #12,R0,51$                     ;LOOP UNTIL END OF TABLE
           53       59   D0  0370  52$:    MOVL    R9,R3                          ;POINT TO SECOND DATE/TIME BUFFER
           83    07 A7   D0  0373          MOVL    7(R7),(R3)+                    ;MOVE YEAR NUMBER
           83       2D   90  0377          MOVB    #^A'-',(R3)+                   ;MOVE DASH
     83    00002172'EF40 B0  037A          MOVW    MONTH_NO_TABLE[R0],(R3)+       ;MOVE MONTH NUMBER
           83       2D   90  0382          MOVB    #^A'-',(R3)+                   ;MOVE DASH
           83       67   B0  0385          MOVW    (R7),(R3)+                     ;MOVE DAY NUMBER
           83    0B A7   7D  0388          MOVQ    11(R7),(R3)+                   ;MOVE 12 CHARACTERS OF TIME STRING
           83    13 A7   D0  038C          MOVL    19(R7),(R3)+                   ;
           57       59   D0  0390          MOVL    R9,R7                          ;RESET TIME POINTER
                    56   D7  0393          DECL    R6                             ;DECREASE TIME LENGTH
                          0395
                          0395  ;
                          0395  ; Extract requested field.
                          0395  ;
     55    6E    08    08   EF  0395  60$:    EXTZV   #8,#8,(SP),R5                  ;GET NUMBER OF DELIMS
           54       6E   9A  039A          MOVZBL  (SP),R4                        ;GET STARTING DELIM #
                    0D   13  039D          BEQL    67$                            ;BRANCH IF ZERO
                    56   D7  039F  65$:    DECL    R6                             ;DECR STRING SIZE
   FEA2 CF    04    87   3A  03A1          LOCC    (R7)+,#4,TIME_DELIMS           ;IS CHAR A DELIM?
                    F6   13  03A7          BEQL    65$                            ;NO, THEN SKIP IT
                 F3 54   F5  03A9          SOBGTR  R4,65$                         ;COPY UNTIL NO MORE DELIMS
           52       59   D0  03AC  67$:    MOVL    R9,R2                          ;INIT RESULT DESCR
                    53   D4  03AF          CLRL    R3                             ;
           89       87   90  03B1  70$:    MOVB    (R7)+,(R9)+                    ;COPY A CHARACTER
                    53   D6  03B4          INCL    R3                             ;INCR THE RESULT LENGTH
                    56   D7  03B6          DECL    R6                             ;DECR STRING SIZE
                    0B   13  03B8          BEQL    75$                            ;DONE IF ZERO
   FE89 CF    04    67   3A  03BA          LOCC    (R7),#4,TIME_DELIMS            ;IS NEXT CHAR A DELIM?
                    EF   13  03C0          BEQL    70$                            ;NO, THEN MOVE NEXT CHAR
                 EC 55   F5  03C2          SOBGTR  R5,70$                         ;COPY UNTIL NO MORE DELIMS
           51       53   D0  03C5  75$:    MOVL    R3,R1                          ;GET RESULT LENGTH
           5E       14   C0  03C8          ADDL    #20,SP                         ;RESTORE THE STACK
                         05  03CB          RSB                                    ;RETURN
                          03CC
                          03CC  ;
                          03CC  ; Return day of week.
```

```
                                    03CC  1602 ;
   0C AE    04 AE   23C34600 8F  7B 03CC  1603 90$:    EDIV    #600000000,4(SP),12(SP),R0      ; GET MINUTES SINCE BASE TIME VALUE
                             50     03D6
                         10 AE   D4 03D7  1604         CLRL    16(SP)                          ;
   0C AE    0C AE   000005A0 8F  7B 03DA  1605         EDIV    #60*24,12(SP),12(SP),R0         ; GET DAYS SINCE BASE TIME
                             50     03E4
                   0C AE     02  C0 03E5  1606         ADDL    #2,12(SP)                       ; ADJUST FOR BASE DAY BEING WED.
                         10 AE   D4 03E9  1607         CLRL    16(SP)                          ;
      50    0C AE   0C AE    07  7B 03EC  1608         EDIV    #7,12(SP),12(SP),R0             ; GET DAY OF WEEK
              52   0000218C'EF   9E 03F3  1609         MOVAB   WEEKDAY_TABLE,R2                ; GET WEEKDAY TABLE ADDR
                         50  0A  C4 03FA  1610         MULL    #10,R0                          ; GET WEEKDAY TABLE OFFSET
                     52    6240  9E 03FD  1611         MOVAB   (R2)[R0],R2                     ; GET ASCIC WEEKDAY
                         51  82  9A 0401  1612         MOVZBL  (R2)+,R1                        ; GET WEEKDAY DESCR
                         5E  14  C0 0404  1613         ADDL    #20,SP                          ; RESTORE THE STACK
                                 05 0407  1614         RSB                                     ; RETURN
```

```
                           0408           .SBTTL  RETURN DEFAULT DIRECTORY NAME STRING
                           0408   ;+
                           0408   ; F$DIRECTORY - RETURN DEFAULT DIRECTORY NAME STRING
                           0408   ;
                           0408   ; THIS LEXICAL FUNCTION RETURNS THE ADDRESS AND LENGTH OF THE CURRENT DEFAULT
                           0408   ; DIRECTORY STRING.
                           0408   ;
                           0408   ;       F$DIRECTORY()
                           0408   ;
                           0408   ; INPUTS:
                           0408   ;
                           0408   ;       NONE.
                           0408   ;
                           0408   ; OUTPUTS:
                           0408   ;
                           0408   ;       R1 = LENGTH OF DEFAULT DIRECTORY STRING.
                           0408   ;       R2 = ADDRESS OF DEFAULT DIRECTORY STRING.
                           0408   ;-
                           0408
                           0408   F$DIRECTORY:                              ;RETURN DEFAULT DIRECTORY STRING
52   00000000'9F      9E   0408           MOVAB   @#PIO$GT_DDSTRING,R2      ;GET ADDRESS OF DEFAULT DIRECTORY STRING
          51   82     9A   040F           MOVZBL  (R2)+,R1                  ;GET LENGTH OF DEFAULT DIRECTORY STRING
                      05   0412           RSB                               ;
```

```
                              0413          .SBTTL  EDIT A STRING
                              0413  ;+
                              0413  ; F$EDIT - EDIT A STRING
                              0413  ;
                              0413  ; THIS LEXICAL FUNCTION PERFORMS THE EDITS A STRING, AS DIRECTED BY AN
                              0413  ; ITEM LIST.
                              0413  ;
                              0413          F$EDIT (STRING, ITEM_LIST)
                              0413  ;
                              0413  ; INPUTS:
                              0413  ;
                              0413          P1(AP) = DESCRIPTOR OF TEXT STRING
                              0413          P2(AP) = ITEM LIST
                              0413  ;
                              0413  ; OUTPUTS:
                              0413  ;
                              0413          R1 = LENGTH OF EDITED STRING.
                              0413          R2 = ADDRESS OF EDITED STRING.
                              0413  ;-
                              0413
                              0413  F$EDIT:
        54    0C AC   7D      0413          MOVQ    P2(AP),R4               ;GET DESCRIPTOR OF COMMA SEPARATED LIST
                 58   D4      0417          CLRL    R8                      ;CLEAR THE FLAGS
           52    54   7D      0419  10$:    MOVQ    R4,R2                   ;ASSUME NO COMMA IN LIST
  58   00000040  8F   C8      041C          BISL    #EDIT_M_EOL,R8          ;ASSUME END OF LIST
       65    54  2C   3A      0423          LOCC    #^A',',R4,(R5)          ;LOCATE END OF NEXT KEYWORD
                 12   13      0427          BEQL    20$                     ;BRANCH IF NOT FOUND, USE ENTIRE STRING
  58   00000040  8F   CA      0429          BICL    #EDIT_M_EOL,R8          ;NOT END OF LIST
       52    54  50   C3      0430          SUBL3   R0,R4,R2                ;ISOLATE THE KEYWORD
             54  70   9E      0434          MOVAB   -(R0),R4                ;CHOP THE KEYWORD OFF THE LIST
        55    01 A1   9E      0437          MOVAB   1(R1),R5                ;
                              043B
  56   00001E9E'EF    9E      043B  20$:    MOVAB   EDIT_TABLE,R6           ;GET ADDRESS OF TABLE
           57    66   9A      0442  30$:    MOVZBL  (R6),R7                 ;GET LENGTH OF ITEM NAME
                 36   13      0445          BEQL    90$                     ;BRANCH IF END OF TABLE
           7E    52   7D      0447          MOVQ    R2,-(SP)                ;SAVE REGISTERS
           7E    54   7D      044A          MOVQ    R4,-(SP)                ;
       00    63  52   2D      044D          CMPC5   R2,(R3),#0,-            ;DOES ITEM NAME MATCH?
        01 A6    57           0451                  R7,1(R6)                ;
                 0D   13      0454          BEQL    50$                     ;BRANCH IF ITEM FOUND
           54    8E   7D      0456          MOVQ    (SP)+,R4                ;RESTORE REGISTERS
           52    8E   7D      0459          MOVQ    (SP)+,R2                ;
       56    02 A647  9E      045C          MOVAB   2(R6)[R7],R6            ;SKIP TO NEXT ENTRY IN TABLE
                 DF   11      0461          BRB     30$                     ;LOOP UNTIL FOUND
           54    8E   7D      0463  50$:    MOVQ    (SP)+,R4                ;RESTORE REGISTERS
           52    8E   7D      0466          MOVQ    (SP)+,R2                ;
       58    01 A647  88      0469          BISB    1(R6)[R7],R8            ;SET OPTION BIT
       A7    58  06   E5      046E          BBCC    #EDIT_V_EOL,R8,10$      ;PROCESS NEXT KEYWORD IN LIST
                              0472
        51    04 AC   7D      0472  60$:    MOVQ    P1(AP),R1               ;GET DESCRIPTOR INPUT STRING
           50    58   D0      0476          MOVL    R8,R0                   ;GET EDITING FLAGS
              FB84'   30      0479          BSBW    DCL$EDIT                ;EDIT THE STRING
                      05      047C          RSB
                              047D
                              047D  90$:    STATUS  IVKEYW                  ;UNDEFINED ITEM NAME
               1197   31      0484          BRW     P2_ERROR
```

```
                          0487        .SBTTL  FIND A STRING ELEMENT
                          0487  ;+
                          0487  ; F$ELEMENT - FIND A STRING ELEMENT
                          0487  ;
                          0487  ; THIS LEXICAL FUNCTION EXTRACTS A SPECIFIED STRING ELEMENT FROM A STRING.
                          0487  ;
                          0487  ;       F$ELEMENT (NUMBER, DELIMITER, STRING)
                          0487  ;
                          0487  ; INPUTS:
                          0487  ;
                          0487  ;       P1(AP) = ELEMENT NUMBER (BASED AT ZERO)
                          0487  ;       P2(AP) = DESCRIPTOR OF SINGLE CHARACTER DELIMITER
                          0487  ;       P3(AP) = DESCRIPTOR OF TEXT STRING.
                          0487  ;
                          0487  ; OUTPUTS:
                          0487  ;
                          0487  ;       R1 = LENGTH OF EXTRACTED ELEMENT.
                          0487  ;       R2 = ADDRESS OF EXTRACTED ELEMENT.
                          0487  ;-
                          0487
                          0487  F$ELEMENT:
       53    04 AC   D0   0487          MOVL    P1(AP),R3               ;GET ELEMENT NUMBER
                53   D5   048B          TSTL    R3                      ;POSITIVE?
                30   19   048D          BLSS    85$                     ;NO, THEN ERROR
       54    0C AC   7D   048F          MOVQ    P2(AP),R4               ;GET DELIMITER STRING
       50    14 AC   7D   0493          MOVQ    P3(AP),R0               ;GET TEXT STRING
          54    01   D1   0497          CMPL    #1,R4                   ;IS IT EXACTLY ONE CHAR?
                2D   12   049A          BNEQ    90$                     ;NO, THEN ERROR
                50   D6   049C          INCL    R0                      ;SET INITIAL LENGTH
                51   D7   049E          DECL    R1                      ;SET INITIAL POINTER
                          04A0
                51   D6   04A0  10$:    INCL    R1                      ;POINT PAST DELIMITER
                50   D7   04A2          DECL    R0                      ;
          52    51   D0   04A4          MOVL    R1,R2                   ;REMEMBER START OF ELEMENT
    61    50    65   3A   04A7          LOCC    (R5),R0,(R1)            ;LOCATE DELIMITER IN STRING
                07   13   04AB          BEQL    80$                     ;IF NOT FOUND, THEN RETURN DELIMITER
          F0    53   F4   04AD          SOBGEQ  R3,10$                  ;LOOP UNTIL WE FIND THE RIGHT ELEMENT
          51    52   C2   04B0  70$:    SUBL    R2,R1                   ;CALCULATE LENGTH OF ELEMENT
                     05   04B3          RSB
                          04B4
                53   D5   04B4  80$:    TSTL    R3                      ;IF LAST ELEMENT
                F8   13   04B6          BEQL    70$                     ;THEN RETURN IT
          51    01   D0   04B8          MOVL    #1,R1                   ;RETURN DELIMITER
          52    55   D0   04BB          MOVL    R5,R2                   ;
                     05   04BE          RSB
                          04BF
  50    00038228 8F  D0   04BF  85$:    MOVL    #CLI$_INVRANGE,R0       ;ASSUME RANGE WILL BE INVALID
              114A   31   04C6          BRW     ERROR                   ;SIGNAL ERROR
                          04C9
  50    000388FA 8F  D0   04C9  90$:    MOVL    #CLI$_STRTOOLNG,R0      ;DELIMITER IS MORE THAN ONE CHAR
              114B   31   04D0          BRW     P2_ERROR                ;SIGNAL ERROR
```

```
                                   04D3          .SBTTL  GET DCL INFORMATION
                                   04D3  ;+
                                   04D3  ; F$ENVIRONMENT - GET DCL INFORMATION
                                   04D3  ;
                                   04D3  ; THIS LEXICAL FUNCTION RETRIEVES SPECIFIC INFORMATION ABOUT THE DCL PROCESS
                                   04D3  ; CONTEXT.
                                   04D3  ;
                                   04D3  ;       F$ENVIRONMENT (ITEM)
                                   04D3  ;
                                   04D3  ; INPUTS:
                                   04D3  ;
                                   04D3  ;       P1(AP) = DESCRIPTOR OF ITEM
                                   04D3  ;
                                   04D3  ; OUTPUTS:
                                   04D3  ;
                                   04D3  ;       R1 = LENGTH OF REQUESTED INFORMATION
                                   04D3  ;       R2 = ADDRESS OF REQUESTED INFORMATION
                                   04D3  ;-
                                   04D3
                                   04D3  F$ENVIRONMENT:
     56   00001ED7'EF      9E      04D3          MOVAB   ENVIRON_TABLE,R6              ;GET ADDRESS OF TABLE
                 7E   52   7D      04DA          MOVQ    R2,-(SP)                      ;SAVE THE SCRATCH DESCR
                                   04DD
                 57   66   9A      04DD  10$:    MOVZBL  (R6),R7                       ;GET LENGTH OF ITEM NAME
                      12   13      04E0          BEQL    20$                           ;BRANCH IF END OF TABLE
  00      08 BC   04 AC    2D      04E2          CMPC5   P1(AP),@P1+4(AP),#0,-         ;DOES ITEM NAME MATCH?
             01 A6   57            04E8                  R7,1(R6)                      ;
                      14   13      04EB          BEQL    90$                           ;BRANCH IF ITEM FOUND
          56   02 A647     9E      04ED          MOVAB   2(R6)[R7],R6                  ;SKIP TO NEXT ENTRY IN TABLE
                      E9   11      04F2          BRB     10$                           ;LOOP UNTIL FOUND
                                   04F4
                 52   8E   7D      04F4  20$:    MOVQ    (SP)+,R2                      ;RESTORE THE SCRATCH DESCR
                                   04F7          STATUS  IVKEYW                        ;UNDEFINED ITEM NAME
                    1117   31      04FE          BRW     P1_ERROR
                                   0501
                 52   8E   7D      0501  90$:    MOVQ    (SP)+,R2                      ;RESTORE THE SCRATCH DESCR
                                   0504          CASE    1(R6)[R7],TYPE=B,<-           ;CASE ON ITEM
                                   0504                  100$,-                        ;CONTROL
                                   0504                  110$,-                        ;NOCONTROL
                                   0504                  200$,-                        ;DEFAULT
                                   0504                  300$,-                        ;DEPTH
                                   0504                  400$,-                        ;ON_SEVERITY
                                   0504                  500$,-                        ;PROMPT
                                   0504                  600$,-                        ;PROMPT_CONTROL
                                   0504                  700$,-                        ;PROTECTION
                                   0504                  800$,-                        ;MESSAGE
                                   0504                  900$,-                        ;MAX_DEPTH
                                   0504                  1000$,-                       ;ON_CONTROL_Y
                                   0504                  1100$,-                       ;PROCEDURE
                                   0504                  1200$,-                       ;KEYPAD_STATE
                                   0504                  1300$,-                       ;VERIFY_PROCEDURE
                                   0504                  1400$,-                       ;VERIFY_IMAGE
                                   0504                  1500$,-                       ;CAPTIVE
                                   0504                  1600$,-                       ;INTERACTIVE
                                   0504                  1700$>                        ;OUTPUT_RATE
                                   052E
                                   052E  ;
```

```
                        052E  ; RETURN [NO]CONTROL INFORMATION
                        052E  ;
         50   01  D0    052E  100$:  MOVL    #1,R0                         ;TEST IF BIT IS SET
              02  11    0531         BRB     120$                          ;SKIP
              50  D4    0533  110$:  CLRL    R0                            ;TEST IF BIT IS CLEAR
         52   03  D1    0535  120$:  CMPL    #3,R2                         ;IS RETURN BUFFER BIG ENOUGH?
              2D  1A    0538         BGTRU   190$                          ;BRANCH IF NOT
         56   53  D0    053A         MOVL    R3,R6                         ;COPY ADDRESS OF SCRATCH SPACE
         01   14  ED    053D         CMPZV   #PRC_V_CTRLT,#1,-             ;CHECK CTRL/T
   50  00B4  CB         0540                 PRC_L_OUTOFBAND(R11),R0       ;
              04  12    0544         BNEQ    130$                          ;SKIP IF NOT EQUAL
    83   54  8F  90     0546         MOVB    #^A/T/,(R3)+                  ;INSERT T IN STRING
         01   19  ED    054A  130$:  CMPZV   #PRC_V_CTRLY,#1,-             ;CHECK CTRL/Y
   50  00B4  CB         054D                 PRC_L_OUTOFBAND(R11),R0       ;
              0C  12    0551         BNEQ    150$                          ;SKIP IF NOT EQUAL
         56   53  D1    0553         CMPL    R3,R6                         ;HAVE WE INSERTED ANYTHING YET?
              03  13    0556         BEQL    140$                          ;NO, THEN SKIP
         83   2C  90    0558         MOVB    #^A/,/,(R3)+                  ;INSERT A COMMA
    83   59  8F  90     055B  140$:  MOVB    #^A/Y/,(R3)+                  ;INSERT Y IN STRING
   51    53   56  C3    055F  150$:  SUBL3   R6,R3,R1                      ;GET LENGTH OF RESULT
         52   56  D0    0563         MOVL    R6,R2                         ;GET ADDRESS OF RESULT
              05        0566         RSB                                   ;
                        0567
                        0567  190$:  STATUS  BUFOVF                        ;SIGNAL BUFFER OVERFLOW ERROR
           10A2  31     056E         BRW     ERROR
                        0571
                        0571  ;
                        0571  ; RETURN DEFAULT
                        0571  ;
         7E   52  7D    0571  200$:  MOVQ    R2,-(SP)                      ;PUSH SCRATCH AREA DESCRIPTOR
         58   5E  D0    0574         MOVL    SP,R8                         ;POINT R8 AT IT
          FA86'  30     0577         BSBW    DCL$SHOWDEFINT                ;GET DEFAULT DIRECTORY SPEC
         5E   08  C0    057A         ADDL    #8,SP                         ;RESTORE THE STACK
          FA80'  30     057D         BSBW    DCL$TRIM                      ;REMOVE LEADING BLANKS
              05        0580         RSB                                   ;
                        0581
                        0581  ;
                        0581  ; RETURN DEPTH
                        0581  ;
    51   5C  AB  D0     0581  300$:  MOVL    PRC_L_INDEPTH(R11),R1         ;GET COMMAND PROCEDURE DEPTH
              52  D4    0585         CLRL    R2                            ;RETURN IN INTEGER FORM
              05        0587         RSB                                   ;
                        0588
                        0588  ;
                        0588  ; RETURN ON_SEVERITY
                        0588  ;
    50   6A  AB  9A     0588  400$:  MOVZBL  PRC_W_ONLEVEL(R11),R0         ;GET SEVERITY NUMBER
         50   01  8A    058C         BICB    #1,R0                         ;CLEAR THE CONTROL/Y BIT
  52  00002072'EF40 DE  058F         MOVAL   SEVERITY_TABLE[R0],R2         ;GET ADDRESS OF ASCIC SEVERITY
         51   82  9A    0597         MOVZBL  (R2)+,R1                      ;SET DESCRIPTOR OF SEVERITY
              05        059A         RSB                                   ;
                        059B
                        059B  ;
                        059B  ; RETURN PROMPT
                        059B  ;
   51  00F0  CB  9A     059B  500$:  MOVZBL  PRC_B_PROMPTLEN(R11),R1       ;GET LENGTH OF PROMPT STRING
         51   03  C2    05A0         SUBL    #3,R1                         ;REMOVE LENGTH OF CR/LF/_
```

```
        52   00F4 CB   9E 05A3          MOVAB   PRC_G_PROMPT(R11),R2    ;GET ADDRESS OF PROMPT STRING
                       05 05A8          RSB                             ;
                          05A9
                          05A9  ;
                          05A9  ; RETURN PROMPT_CONTROL
                          05A9  ;
           51      01  D0 05A9  600$:   MOVL    #1,R1                   ;ASSUME CR/LF SPECIFIED
     02 68 AB      00  E0 05AC          BBS     #PRC_V_CARRCNTL,PRC_W_FLAGS(R11),610$  ;SKIP IF SPECIFIED
                   51  D4 05B1          CLRL    R1                      ;RETURN FALSE OTHERWISE
                 11CC  31 05B3  610$:   BRW     FORMAT_BOOLEAN          ;RETURN THE VALUE
                          05B6
                          05B6  ;
                          05B6  ; RETURN PROTECTION
                          05B6  ;
                   7E  D4 05B6  700$:   CLRL    -(SP)                   ;WHERE TO RETURN PROTECTION
           7E      5E  D0 05B8          MOVL    SP,-(SP)                ;NOTE WHERE PROTECTION IS TO BE PUT
                   7E  D4 05BB          CLRL    -(SP)                   ;DON'T WANT TO SET PROTECTION
  00000000'9F      02  FB 05BD          CALLS   #2,@#SYS$SETDFPROT      ;GET DEFAULT PROTECTION
           54      8E  D0 05C4          MOVL    (SP)+,R4                ;COPY PROTECTION TO USEFUL REG
                 110B  31 05C7          BRW     FORMAT_PROT             ;CONVERT INTO ASCII FORM
                          05CA
                          05CA  ;
                          05CA  ; RETURN MESSAGE QUALIFIERS
                          05CA  ;
           58      2F  9A 05CA  800$:   MOVZBL  #^A"/",R8               ;STORE FIELD DELIMITER
           56      53  D0 05CD          MOVL    R3,R6                   ;COPY ADDRESS OF SCRATCH SPACE
           83      58  90 05D0          MOVB    R8,(R3)+                ;INSERT A LEADING SLASH IN STRING
                          05D3
           52      2E  D1 05D3          CMPL    #46,R2                  ;IS SCRATCH BUFFER BIG ENOUGH?
                   8F  1A 05D6          BGTRU   190$                    ;NO, THEN ERROR
     59   00000000'GF  9A 05D8          MOVZBL  G^CTL$GB_MSGMASK,R9     ;GET MESSAGE MASK
                          05DF
     50   00002255'EF  9E 05DF          MOVAB   FACILITY,R0             ;GET ADDRESS OF ASCIC STRING
           51      80  9A 05E6          MOVZBL  (R0)+,R1                ;GET LENGTH/ADDRESS OF STRING
        05 59      03  E0 05E9          BBS     #3,R9,810$              ;IS FACILITY SPECIFIED?
     83   4F4E     8F  B0 05ED          MOVW    #^A/NO/,(R3)+           ;INSERT NO IN STRING
     63     60     51  28 05F2  810$:   MOVC3   R1,(R0),(R3)            ;MOVE STRING INTO DESCRIPTOR
           83      58  90 05F6          MOVB    R8,(R3)+                ;INSERT DELIMITER
                          05F9
     50   0000224C'EF  9E 05F9          MOVAB   SEVERITY,R0             ;GET ADDRESS OF ASCIC STRING
           51      80  9A 0600          MOVZBL  (R0)+,R1                ;GET LENGTH/ADDRESS OF STRING
        05 59      02  E0 0603          BBS     #2,R9,820$              ;IS SEVERITY SPECIFIED?
     83   4F4E     8F  B0 0607          MOVW    #^A/NO/,(R3)+           ;INSERT NO IN STRING
     63     60     51  28 060C  820$:   MOVC3   R1,(R0),(R3)            ;MOVE STRING INTO DESCRIPTOR
           83      58  90 0610          MOVB    R8,(R3)+                ;INSERT DELIMITER
                          0613
     50   0000223D'EF  9E 0613          MOVAB   IDENT,R0                ;GET ADDRESS OF ASCIC STRING
           51      80  9A 061A          MOVZBL  (R0)+,R1                ;GET LENGTH/ADDRESS OF STRING
        05 59      01  E0 061D          BBS     #1,R9,830$              ;IS SEVERITY SPECIFIED?
     83   4F4E     8F  B0 0621          MOVW    #^A/NO/,(R3)+           ;INSERT NO IN STRING
     63     60     51  28 0626  830$:   MOVC3   R1,(R0),(R3)            ;MOVE STRING INTO DESCRIPTOR
           83      58  90 062A          MOVB    R8,(R3)+                ;INSERT DELIMITER
                          062D
     50   00002238'EF  9E 062D          MOVAB   TEXT,R0                 ;GET ADDRESS OF ASCIC STRING
           51      80  9A 0634          MOVZBL  (R0)+,R1                ;GET LENGTH/ADDRESS OF STRING
        05 59      00  E0 0637          BBS     #0,R9,840$              ;IS TEXT SPECIFIED?
     83   4F4E     8F  B0 063B          MOVW    #^A/NO/,(R3)+           ;INSERT NO IN STRING
```

```
        63    60    51    28  0640  840$:   MOVC3   R1,(R0),(R3)            ;MOVE STRING INTO DESCRIPTOR
                              0644
              51    53    56    C3  0644            SUBL3   R6,R3,R1                ;GET LENGTH OF RESULT
                    52    56    D0  0648            MOVL    R6,R2                   ;GET ADDRESS OF RESULT
                          05  064B            RSB                             ;
                              064C
                              064C  ;
                              064C  ; RETURN MAX_DEPTH
                              064C  ;
  51     00000000'8F    D0  064C  900$:   MOVL    #MAX_DEPTH,R1           ;GET MAXIMUM DEPTH OF COMMAND PROCEDURES
                    52    D4  0653            CLRL    R2                      ;RETURN AS INTEGER
                          05  0655            RSB                             ;
                              0656
                              0656  ;
                              0656  ; RETURN ON_CONTROL_Y
                              0656  ;
              51    01    D0  0656  1000$:  MOVL    #1,R1                   ;ASSUME HANDLER SPECIFIED
              00B8  CB    D5  0659            TSTL    PRC_L_ONCTLY(R11)       ;IS IT?
                    02    12  065D            BNEQ    1010$                   ;YES, THEN SKIP
                    51    D4  065F            CLRL    R1                      ;NO, THEN RETURN FALSE
                  111E    31  0661  1010$:  BRW     FORMAT_BOOLEAN          ;RETURN THE VALUE
                              0664
                              0664  ;
                              0664  ; RETURN PROCEDURE
                              0664  ;
        50    00BC  CB    D0  0664  1100$:  MOVL    PRC_L_IDFLNK(R11),R0    ;GET ADDRESS OF IDF BLOCK
          52    68  A0    D0  0669            MOVL    IDF_L_FILENAME(R0),R2   ;GET ADDRESS OF ASCIC FILENAME
                    04    13  066D            BEQL    1110$                   ;SKIP IF NONE
              51    82    9A  066F            MOVZBL  (R2)+,R1                ;GET LENGTH/ADDRESS OF FILENAME
                          05  0672            RSB                             ;
                              0673
                    51    D4  0673  1110$:  CLRL    R1                      ;RETURN NO FILENAME PRESENT
              52    5E    D0  0675            MOVL    SP,R2                   ;INDICATE NULL STRING
                          05  0678            RSB                             ;
                              0679
                              0679  ;
                              0679  ; RETURN KEYPAD_STATE
                              0679  ;
          52    48  AB    D0  0679  1200$:  MOVL    PRC_L_CURRKEY(R11),R2   ;GET ADDRESS OF ASCIC STATE
              51    82    9A  067D            MOVZBL  (R2)+,R1                ;GET LENGTH/ADDRESS OF STATE
                          05  0680            RSB                             ;
                              0681
                              0681  ;
                              0681  ; RETURN VERIFY_PROCEDURE
                              0681  ;
                    51    D4  0681  1300$:  CLRL    R1                      ;ASSUME NOVERIFY
                    07    E1  0683            BBC     #PRC_V_VERIFY,-         ;BRANCH IF NOVERIFY
              02 68 AB        0685                    PRC_Q_FLAGS(R11),1320$  ;
                    51    D6  0688  1310$:  INCL    R1                      ;SET TRUE
                  10F5    31  068A  1320$:  BRW     FORMAT_BOOLEAN          ;RETURN TRUE/FALSE
                              068D
                              068D  ;
                              068D  ; RETURN VERIFY_IMAGE
                              068D  ;
                    51    D4  068D  1400$:  CLRL    R1                      ;ASSUME NOVERIFY
                    07    E1  068F            BBC     #PRC_V_VERIMAGE,-       ;BRANCH IF NOVERIFY
           F5 00AF  CB        0691                    PRC_B_FLAGS2(R11),1320$ ;
```

```
                F1   11  0695  1977          BRB     1310$                    ;SET TRUE
                         0697  1978
                         0697  1979 ;
                         0697  1980 ; RETURN CAPTIVE
                         0697  1981 ;
 50   00000000'GF    9E  0697  1982 1500$:   MOVAB   G^CTL$AG_CLIDATA,R0      ;GET ADDRESS OF PPD
                51   D4  069E  1983          CLRL    R1                       ;ASSUME NOT CAPTIVE
                03   E1  06A0  1984          BBC     #PPD$V_CAPTIVE,-         ;BRANCH IF NOT CAPTIVE ACCOUNT
          E5 02 A0       06A2  1985                  PPD$W_FLAGS(R0),1320$    ;
                E1   11  06A5  1986          BRB     1310$                    ;SET TRUE
                         06A7  1987
                         06A7  1988 ;
                         06A7  1989 ; RETURN INTERACTIVE
                         06A7  1990 ;
                51   D4  06A7  1991 1600$:   CLRL    R1                       ;ASSUME NOT INTERACTIVE
                06   E0  06A9  1992          BBS     #PRC_V_MODE,-            ;BRANCH IF NOT INTERACTIVE
          DC 68 AB       06AB  1993                  PRC_Q_FLAGS(R11),1320$   ;
                D8   11  06AE  1994          BRB     1310$                    ;SET TRUE
                         06B0  1995
                         06B0  1996 ;
                         06B0  1997 ; RETURN OUTPUT_RATE
                         06B0  1998 ;
          7E    52   7D  06B0  1999 1700$:   MOVQ    R2,-(SP)                 ;SET UP RETURN DESCR
                06   E1  06B3  2000          BBC     #PRC_V_MODE,-            ;BRANCH IF INTERACTIVE
          1B 68 AB       06B5  2001                  PRC_Q_FLAGS(R11),1710$   ;
          50    5E   D0  06B8  2002          MOVL    SP,R0                    ;GET ADDR OF OUTPUT DESCR
                         06BB  2003          $ASCTIM_S TIMBUF=(R0),-          ;CONVERT BINARY DELTA TIME TO ASCII
                         06BB  2004                  TIMLEN=(R0),-            ;
                         06BB  2005                  TIMADR=PRC_Q_FLUSHTIME(R11)
             04 50   E9  06CC  2006          BLBC    R0,1710$                 ;BR IF ERROR IN TIME
          51    8E   7D  06CF  2007          MOVQ    (SP)+,R1                 ;GET THE RESULT
                05       06D2  2008          RSB                              ;RETURN DELTA TIME
          51    8E   7D  06D3  2009 1710$:   MOVQ    (SP)+,R1                 ;RETURN NULL
                51   D4  06D6  2010          CLRL    R1                       ;
                05       06D8  2011          RSB                              ;
```

```
                             06D9         .SBTTL  EXTRACT SUBSTRING FROM STRING
                             06D9 ;+
                             06D9 ; F$EXTRACT - EXTRACT SUBSTRING FROM STRING
                             06D9 ;
                             06D9 ; THIS LEXICAL FUNCTION EXTRACTS A SUBSTRING FROM AN ARGUMENT STRING AND RETURNS
                             06D9 ; ITS DESCRIPTOR.
                             06D9 ;
                             06D9 ;       F$EXTRACT(POSITION, SIZE, STRING)
                             06D9 ;
                             06D9 ; INPUTS:
                             06D9 ;
                             06D9 ;       P1(AP) = STARTING CHARACTER NUMBER (BASED AT ZERO)
                             06D9 ;       P2(AP) = NUMBER OF CHARACTERS
                             06D9 ;       P3(AP) = DESCRIPTOR OF TEXT STRING.
                             06D9 ;
                             06D9 ; OUTPUTS:
                             06D9 ;
                             06D9 ;       R1 = LENGTH OF EXTRACTED SUBSTRING.
                             06D9 ;       R2 = ADDRESS OF EXTRACTED SUBSTRING.
                             06D9 ;-
                             06D9
                             06D9 F$EXTRACT:
                  51    D4   06D9         CLRL    R1                          ;ASSUME NULL SUBSTRING
      14 AC    04 AC    D1   06DB         CMPL    P1(AP),P3(AP)               ;STARTING POSITION BEYOND END OF STRING
                  19    1E   06E0         BGEQU   10$                         ;IF GTRU YES
            51 0C AC    D0   06E2         MOVL    P2(AP),R1                   ;GET NUMBER OF CHARACTERS TO EXTRACT
   52 18 AC    04 AC    C1   06E6         ADDL3   P1(AP),P3+4(AP),R2          ;CALCULATE STARTING ADDRESS OF SUBSTRING
      14 AC    04 AC    C2   06EC         SUBL    P1(AP),P3(AP)               ;REDUCE APPARENT STRING LENGTH BY POSITION
         14 AC    51    C2   06F1         SUBL    R1,P3(AP)                   ;REDUCE STRING BY LENGTH OF SUBSTRING
                  04    1E   06F5         BGEQU   10$                         ;IF GEQU STRING LENGTH SUFFICIENT
            51 14 AC    C0   06F7         ADDL    P3(AP),R1                   ;CALCULATE ACTUAL LENGTH OF SUBSTRING
                        05   06FB 10$:    RSB
```

```
                         06FC        .SBTTL  FAO SYSTEM SERVICE
                         06FC  ;+
                         06FC  ; F$FAO - FORMATTED ASCII OUTPUT
                         06FC  ;
                         06FC  ; THIS LEXICAL FUNCTION INVOKES THE $FAO SYSTEM SERVICE WITH
                         06FC  ; EITHER STRINGS OR LONGWORDS AS ARGUMENTS, DEPENDING ON THE
                         06FC  ; TYPE OF THE ARGUMENT.
                         06FC  ;
                         06FC  ; F$FAO(CTRSTR,ARG1,ARG2,....)
                         06FC  ;
                         06FC  ; INPUTS:
                         06FC  ;
                         06FC  ;       NPARM(AP) = NUMBER OF ARGUMENTS SPECIFIED
                         06FC  ;       P1(AP) = FIRST ARGUMENT
                         06FC  ;       P2(AP) = SECOND ARGUMENT
                         06FC  ;       ETC.
                         06FC  ;
                         06FC  ; OUTPUTS:
                         06FC  ;
                         06FC  ;       R1/R2 = RESULTANT STRING FROM $FAO
                         06FC  ;-
                         06FC
                         06FC  F$FAO:
          7E   52   7D   06FC          MOVQ    R2,-(SP)          ;ADDRESS OF RESULT BUFFER DESCRIPTOR
          52   5E   D0   06FF          MOVL    SP,R2             ;ADDRESS OF BUFFER DESCRIPTOR
     50   6C   01   C3   0702          SUBL3   #1,NPARM(AP),R0   ;GET NUMBER OF FAO ARGUMENTS
     51   0C AC40   7E   0706          MOVAQ   P2(AP)[R0],R1     ;START AT LAST ARGUMENT + 1
          FC   A1   D5   070B  10$:    TSTL    -4(R1)            ;BINARY LONGWORD OR STRING DESCRIPTOR?
               07   12   070E          BNEQ    15$               ;BRANCH IF STRING
          51   08   C2   0710          SUBL    #8,R1             ;SKIP TO NEXT ARGUMENT
               61   DD   0713          PUSHL   (R1)              ;PUSH LONGWORD BINARY VALUE
               02   11   0715          BRB     20$
               71   7F   0717  15$:    PUSHAQ  -(R1)             ;PUSH ADDRESS OF DESCRIPTOR
          EF   50   F5   0719  20$:    SOBGTR  R0,10$            ;LOOP THRU ALL FAO ARGUMENTS
               5E   DD   071C          PUSHL   SP                ;ADDRESS OF PARAMETER LIST
               52   DD   071E          PUSHL   R2                ;ADDRESS OF BUFFER DESCRIPTOR
               52   DD   0720          PUSHL   R2                ;ADDRESS OF WORD TO RECEIVE LENGTH
          04   AC   7F   0722          PUSHAQ  P1(AP)            ;ADDRESS OF STRING DESCRIPTOR
 00000000'GF   04   FB   0725          CALLS   #4,G^SYS$FAOL     ;INVOKE FAO SERVICE
          06   50   E9   072C          BLBC    R0,90$            ;BRANCH IF ERROR DETECTED
          5E   52   D0   072F          MOVL    R2,SP             ;POP FAO ARGUMENTS OFF STACK
               06   BA   0732          POPR    #^M<R1,R2>        ;RETRIEVE DESCRIPTOR OF RESULT STRING
                    05   0734  30$:    RSB
                         0735
     5E   52   08   C1   0735  90$:    ADDL3   #8,R2,SP          ;CLEAN STACK
             0EDC   31   0739          BRW     P1_ERROR          ;REPORT ERROR AND DISPLAY P1
```

```
                                  073C          .SBTTL  GET FILE ATTRIBUTES
                                  073C  ;+
                                  073C  ; F$FILE_ATTRIBUTES - GET FILE INFORMATION
                                  073C  ;
                                  073C  ; THIS LEXICAL FUNCTION RETURNS A SPECIFIED ATTRIBUTE FOR A FILE
                                  073C  ; BY USING THE RMS OPEN SYSTEM SERVICE TO OBTAIN THE INFORMATION.
                                  073C  ;
                                  073C  ;       F$FILE_ATTRIBUTES(FILESPEC,ITEM)
                                  073C  ;
                                  073C  ; INPUTS:
                                  073C  ;
                                  073C  ;       P1(AP) = DESCRIPTOR OF FILE SPECIFICATION
                                  073C  ;       P2(AP) = DESCRIPTOR OF RMS ITEM NAME
                                  073C  ;
                                  073C  ; OUTPUTS:
                                  073C  ;
                                  073C  ;       R1/R2 = DESCRIPTOR OF RESULTANT DATA
                                  073C  ;-
                                  073C
                                  073C  F$FILE:
               7E     52     7D   073C          MOVQ    R2,-(SP)                        ;PUSH DESCRIPTOR OF SCRATCH SPACE
               59     5E     D0   073F          MOVL    SP,R9                           ;SAVE ADDRESS OF DESCRIPTOR
        57  00001513'EF      9E   0742          MOVAB   FILE_TABLE,R7                   ;GET ADDRESS OF TABLE
               58   05 A7    9A   0749  10$:    MOVZBL  5(R7),R8                        ;GET LENGTH OF ITEM NAME
                      12     13   074D          BEQL    15$                             ;BRANCH IF END OF TABLE
      00   10 BC   0C AC     2D   074F          CMPC5   P2(AP),@P2+4(AP),#0,-           ;DOES ITEM NAME MATCH?
               06 A7  58          0755                  R8,6(R7)                        ;
                      11     13   0758          BEQL    20$                             ;BRANCH IF ITEM FOUND
            57   06 A748     9E   075A          MOVAB   6(R7)[R8],R7                    ;SKIP TO NEXT ENTRY IN TABLE
                      E8     11   075F          BRB     10$                             ;LOOP UNTIL FOUND
                                  0761  15$:    STATUS  IVKEYW                          ;UNDEFINED ITEM NAME
                    0EB3     31   0768          BRW     P2_ERROR                        ;
               5E   B0 AE    9E   076B  20$:    MOVAB   -FAB$C_BLN(SP),SP               ;ALLOCATE FAB ON STACK
               56     5E     D0   076F          MOVL    SP,R6                           ;GET ADDRESS OF FAB
 66   0050 8F   00    6E  00 2C   0772          MOVC5   #0,(SP),#0,#FAB$C_BLN,(R6)      ;ZERO STORAGE
                                  077A          ASSUME  FAB$B_BLN EQ FAB$B_BID+1
               5003 8F       B0   077A          MOVW    #FAB$C_BID+<FAB$C_BLN@8>,-      ;INITIALIZE FAB
                      66          077E                  FAB$B_BID(R6)                   ;
               50     67     9A   077F          MOVZBL  (R7),R0                         ;GET BLOCK INDEX
               01     50     D1   0782          CMPL    R0,#1                           ;FAB OR NAM BLOCK?
                      47     1F   0785          BLSSU   50$                             ;BRANCH IF FAB ONLY IS NEEDED
                      25     13   0787          BEQL    40$                             ;BRANCH IF NAM BLOCK IS ALSO NEEDED
      52   00001505'EF40     3C   0789          MOVZWL  XAB_INFO-4[R0],R2               ;GET XAB BLOCK ID & LENGTH
          51   52   F8 8F    78   0791          ASHL    #-8,R2,R1                       ;ISOLATE BLOCK LENGTH
               5E     51     C2   0796          SUBL    R1,SP                           ;ALLOCATE SPACE FOR XAB ON STACK
                      52     DD   0799          PUSHL   R2                              ;SAVE R2
 04 AE   51   00   6E  00    2C   079B          MOVC5   #0,(SP),#0,R1,4(SP)             ;ZERO STORAGE
                   52 8ED0        07A2          POPL    R2                              ;RESTORE R2
                                  07A5          ASSUME  XAB$B_BLN EQ XAB$B_COD+1
               6E     52     B0   07A5          MOVW    R2,XAB$B_COD(SP)                ;STORE XAB BLOCK ID/LENGTH
            24 A6     5E     D0   07A8          MOVL    SP,FAB$L_XAB(R6)                ;POINT TO XAB BLOCK
                      20     11   07AC          BRB     50$                             ;
               5E   FEA1 CE  9E   07AE  40$:    MOVAB   -NAM$C_BLN-NAM$C_MAXRSS(SP),SP  ;ALLOCATE SPACE FOR NAM BLOCK
 6E   0060 8F   00    6E  00 2C   07B3          MOVC5   #0,(SP),#0,#NAM$C_BLN,(SP)      ;ZERO STORAGE
                                  07BB          ASSUME  NAM$B_BLN EQ NAM$B_BID+1
               6002 8F       B0   07BB          MOVW    #NAM$C_BID+<NAM$C_BLN@8>,-      ;INITIALIZE NAM
                      6E          07BF                  NAM$B_BID(SP)                   ;
```

```
         0A AE    FF 8F   90  07C0          MOVB    #NAM$C_MAXRSS,NAM$B_ESS(SP)          ;SET SIZE OF EXPANDED BUFFER
         0C AE    60 AE   9E  07C5          MOVAB   NAM$C_BLN(SP),NAM$L_ESA(SP)          ;SET ADDRESS OF RESULT BUFFER
            28 A6     5E  D0  07CA          MOVL    SP,FAB$L_NAM(R6)                     ;POINT TO NAM BLOCK
            04 A7     0B  91  07CE  50$:    CMPB    #11,4(R7)                            ;KNOWN FILE CHECK?
                      09  12  07D2          BNEQ    55$                                  ;NO, THEN SKIP
                              07D4          SETBIT  NAM$V_NOCONCEAL,-                    ;TRANSLATE CONCEALED LOG. NAMES
                              07D4                  NAM$B_NOP(SP)
                              07D8          SETBIT  FAB$V_KFO,FAB$L_FOP(R6)              ;SET KNOWN FILE OPEN
         34 A6    04 AC   90  07DD  55$:    MOVB    P1(AP),FAB$B_FNS(R6)                 ;SET PRIMARY FILESPEC
         2C A6    08 AC   D0  07E2          MOVL    P1+4(AP),FAB$L_FNA(R6)               ;
            16 A6     02  90  07E7          MOVB    #FAB$M_GET,FAB$B_FAC(R6)             ;READ-ONLY ACCESS
         17 A6    4F 8F   90  07EB          MOVB    #FAB$M_GET!FAB$M_PUT!FAB$M_UPD!-     ;ALLOW OTHERS
                              07F0                  FAB$M_DEL!FAB$M_UPI,FAB$B_SHR(R6)    ;
                              07F0          SETBIT  FAB$V_PPF,FAB$L_FOP(R6)              ;USE PROCESS I/O SEGMENT
                              07F5          $OPEN   FAB=(R6)                             ;OPEN THE FILE TO GET REQUESTED INFO
               26 50      E9  07FE          BLBC    R0,85$                               ;IF ERROR, RETURN NULL STRING
                              0801          $CLOSE  FAB=(R6)                             ;CLOSE THE FILE AGAIN
            55    01 A7   9A  080A          MOVZBL  1(R7),R5                             ;GET OFFSET TO FIELD
                      12  13  080E          BEQL    60$                                  ;BRANCH IF NONE
               55     5E  C0  0810          ADDL    SP,R5                                ;POINT TO FIELD
            54    02 A7   9A  0813          MOVZBL  2(R7),R4                             ;GET LENGTH OF FIELD
                      09  13  0817          BEQL    60$                                  ;BRANCH IF NONE
            50    03 A7   9A  0819          MOVZBL  3(R7),R0                             ;GET STARTING BIT NUMBER
   51    65    54     50  EF  081D          EXTZV   R0,R4,(R5),R1                        ;FETCH CONTENTS OF FIELD
                  37'AF   9F  0822  60$:    PUSHAB  B^90$                                ;RETURN AFTER CASE
                      15  11  0825          BRB     GET_FILE_ITEM                        ;GET DATA ITEM
                              0827
            51    0C A6   D0  0827  85$:    MOVL    FAB$L_STV(R6),R1                     ;GET SECONDARY STATUS FROM $OPEN
                      03  13  082B          BEQL    88$                                  ;BRANCH IF NONE
               50     51  D0  082D          MOVL    R1,R0                                ;REPORT SECONDARY ACP STATUS, NOT RM
            5E    08 A9   9E  0830  88$:    MOVAB   8(R9),SP                             ;CLEANUP STACK
                    0DE1  31  0834          BRW     P1_ERROR                             ;REPORT ERROR
                              0837
            5E    08 A9   9E  0837  90$:    MOVAB   8(R9),SP                             ;DEALLOCATE STORAGE ON STACK
                          05  083B          RSB                                          ;
                              083C
                              083C  GET_FILE_ITEM:
                              083C          CASE    4(R7),TYPE=B,<-             ;CASE ON TYPE OF FIELD
                              083C                  100$,-                      ;BINARY
                              083C                  200$,-                      ;DATE/TIME
                              083C                  300$,-                      ;ASCIC STRING
                              083C                  400$,-                      ;FILE ID STRING
                              083C                  500$,-                      ;FILE ORGANIZATION
                              083C                  600$,-                      ;RECORD ATTRIBUTES
                              083C                  700$,-                      ;RECORD FORMAT
                              083C                  800$,-                      ;OWNER UIC
                              083C                  900$,-                      ;PROTECTION
                              083C                  1000$,-                     ;EOF - # BLOCKS IN USE
                              083C                  1100$,-                     ;BOOLEAN
                              083C                  1200$>                      ;KNOWN FILE
                              0859
                              0859  ;
                              0859  ; RETURN BINARY NUMBER
                              0859  ;
                      52  D4  0859  100$:   CLRL    R2                          ;INDICATE BINARY RESULT
                          05  085B          RSB
                              085C  ;
```

```
                               085C  ; RETURN DATE/TIME STRING
                               085C  ;
                               085C  200$:   $ASCTIM_S (R9),(R9),(R5)          ;CONVERT TO ASCII DATE/TIME STRING
                51   69   7D   086B          MOVQ    (R9),R1                   ;GET DESCRIPTOR OF STRING
                          05   086E          RSB
                               086F  ;
                               086F  ; RETURN ASCIC STRING
                               086F  ;
                51   85   9A   086F  300$:   MOVZBL  (R5)+,R1                  ;GET LENGTH OF STRING
                69   51   D1   0872          CMPL    R1,(R9)                   ;ENOUGH ROOM FOR STRING?
                     0F   1A   0875          BGTRU   310$                      ;BRANCH IF NOT
                     51   DD   0877          PUSHL   R1                        ;SAVE LENGTH OF STRING
        04 B9   65   51   28   0879          MOVC    R1,(R5),@4(R9)            ;STORE INTO SCRATCH BUFFER
                     51 8ED0   087E          POPL    R1                        ;RESTORE LENGTH OF STRING
           52   04 A9     D0   0881          MOVL    4(R9),R2                  ;GET ADDRESS OF STRING
                          05   0885          RSB
                               0886  310$:   STATUS  BUFOVF                    ;SIGNAL BUFFER OVERFLOW
                   0D8E   31   088D          BRW     P2_ERROR
                               0890  ;
                               0890  ; RETURN FILE ID NUMBER
                               0890  ;
                54   55   D0   0890  400$:   MOVL    R5,R4                     ;PASS ADDRESS OF FILE ID
                   0EBA   31   0893          BRW     FORMAT_FILEID             ;FORMAT INTO ASCII STRING
                               0896  ;
                               0896  ; RETURN FILE ORGANIZATION
                               0896  ;
                               0896          ASSUME  FAB$C_SEQ EQ 0            ; ^X00
                               0896          ASSUME  FAB$C_REL EQ 16           ; ^X10
                               0896          ASSUME  FAB$C_IDX EQ 32           ; ^X20
 58 44 49 20 4C 45 52 20 51 45 53   0896  505$:   .ASCII  'SEQ REL IDX'
                               08A1
          51   51   FC 8F  78   08A1  500$:   ASHL    #-4,R1,R1                 ;GET UPPER NIBBLE AS INDEX
             52   EC AF41  DE   08A6          MOVAL   505$[R1],R2               ;SET ADDRESS OF STRING
                51   03   D0   08AB          MOVL    #3,R1                     ;SET LENGTH OF STRING
                          05   08AE          RSB
                               08AF  ;
                               08AF  ; RETURN RECORD ATTRIBUTES (RAT)
                               08AF  ;
                               08AF          ASSUME  FAB$M_FTN EQ 1
                               08AF          ASSUME  FAB$M_CR EQ 2
                               08AF          ASSUME  FAB$M_PRN EQ 4
             4E 54 46 00'      08AF  605$:   .ASCIC  'FTN'
                      03       08AF
                52 43 00'      08B3          .ASCIC  'CR'
                      02       08B3
                 000008B7      08B6          .BLKB   1                         ;PAD TO LONGWORD BOUNDARY
             4E 52 50 00'      08B7          .ASCIC  'PRN'
                      03       08B7
                      00'      08BB          .ASCIC  ''                        ;USE NULL IF FFS FAILS
                      00       08BB
                               08BC
      51   65   03   00   EA   08BC  600$:   FFS     #0,#3,(R5),R1             ;FIND FIRST BIT SET
           52   EA AF41   DE   08C1          MOVAL   605$[R1],R2               ;SET ADDRESS OF STRING
                51   82   9A   08C6          MOVZBL  (R2)+,R1                  ;SET LENGTH OF STRING
                          05   08C9          RSB
                               08CA  ;
                               08CA  ; RETURN RECORD FORMAT (RFM)
```

```
                                  08CA      ;
                                  08CA                ASSUME  FAB$C_UDF EQ 0
                                  08CA                ASSUME  FAB$C_FIX EQ 1
                                  08CA                ASSUME  FAB$C_VAR EQ 2
                                  08CA                ASSUME  FAB$C_VFC EQ 3
                                  08CA                ASSUME  FAB$C_STM EQ 4
                                  08CA                ASSUME  FAB$C_STMLF EQ 5
                                  08CA                ASSUME  FAB$C_STMCR EQ 6
20 52 41 56 20 58 49 46 20 46 44 55 08CA  705$:      .ASCII  'UDF FIX VAR VFC STM STMLF   STMCR   '
4C 4D 54 53 20 4D 54 53 20 43 46 56 08D6
20 20 20 52 43 4D 54 53 20 20 20 46 08E2
            05 00 05 03 03 03 03 03 08EE  707$:      .BYTE   3,3,3,3,3,5,0,5
                        51    06 D1 08F6  700$:      CMPL    #6,R1                        ;IS RFM ON 4 BYTE OFFSET?
                              03 12 08F9            BNEQ    710$                          ;YES, SKIP
                        51    07 D0 08FB            MOVL    #7,R1                         ;NO, SET OFFSET
                   52    C8 AF41 DE 08FE  710$:      MOVAL   705$[R1],R2                  ;SET ADDRESS OF STRING
                   51    E7 AF41 9A 0903            MOVZBL  707$[R1],R1                   ;SET LENGTH OF STRING
                                 05 0908            RSB
                                    0909      ;
                                    0909      ; RETURN OWNER UIC
                                    0909      ;
                        54    51 D0 0909  800$:      MOVL    R1,R4                         ;PASS UIC LONGWORD
                            0D60 31 090C            BRW     FORMAT_UIC                    ;CONVERT INTO ASCII FORM
                                    090F
                                    090F      ;
                                    090F      ; RETURN FILE PROTECTION
                                    090F      ;
                        54    51 D0 090F  900$:      MOVL    R1,R4                         ;PASS PROTECTION WORD
                            0DC0 31 0912            BRW     FORMAT_PROT                   ;CONVERT INTO ASCII FORM
                                    0915
                                    0915      ;
                                    0915      ; RETURN # BLOCKS IN USE
                                    0915      ;
                           04 A5 B5 0915  1000$:     TSTW    XAB$W_FFB-XAB$L_EBK(R5)  ;FIRST FREE BYTE = 0?
                              02 12 0918            BNEQ    1010$                         ;IF NOT, EBK = BLOCKS IN USE
                              51 D7 091A            DECL    R1                            ;ELSE DON'T COUNT LAST BLOCK
                              52 D4 091C  1010$:     CLRL    R2                            ;INDICATE RESULT IS BINARY
                                 05 091E            RSB
                                    091F      ;
                                    091F      ; RETURN BOOLEAN TRUE/FALSE
                                    091F      ;
                            0E60 31 091F  1100$:     BRW     FORMAT_BOOLEAN                ;RETURN "TRUE" OR "FALSE"
                                    0922
                                    0922      ;
                                    0922      ; RETURN BOOLEAN TRUE/FALSE
                                    0922      ;
                              51 D4 0922  1200$:     CLRL    R1                            ;ASSUME NOT KNOWN
                           18 A6 D5 0924            TSTL    FAB$L_CTX(R6)                 ;IS IT?
                              02 13 0927            BEQL    1210$                         ;NO, THEN SKIP
                              51 D6 0929            INCL    R1                            ;YES, THEN RETURN TRUE
                            0E54 31 092B  1210$:     BRW     FORMAT_BOOLEAN                ;RETURN "TRUE" OR "FALSE"
```

```
                          092E          .SBTTL  GETDVI SYSTEM SERVICE
                          092E  ;+
                          092E  ; F$GETDVI - GET DEVICE/VOLUME INFORMATION
                          092E  ;
                          092E  ; THIS LEXICAL FUNCTION INVOKES THE $GETDVI SYSTEM SERVICE TO
                          092E  ; OBTAIN ANY PIECE OF INFORMATION FROM A GIVEN DEVICE.
                          092E  ;
                          092E  ;       F$GETDVI(DEVNAM,ITEM)
                          092E  ;
                          092E  ; INPUTS:
                          092E  ;
                          092E  ;       R2/R3 = DECRIPTOR OF SCRATCH BUFFER
                          092E  ;
                          092E  ;       P1(AP) = DESCRIPTOR OF DEVICE NAME
                          092E  ;
                          092E  ;       P2(AP) = DESCRIPTOR OF $DVI ITEM NAME
                          092E  ;
                          092E  ; OUTPUTS:
                          092E  ;
                          092E  ;       R1/R2 = DESCRIPTOR OF RESULTANT DATA
                          092E  ;-
                          092E
                          092E  F$GETDVI:
                7E    7C  092E          CLRQ    -(SP)                   ;RESERVE IOSB, WILL BE 8(R5)
           7E   52    D0  0930          MOVL    R2,-(SP)                ;SAVE LENGTH OF SCRATCH BUFFER
                7E    D4  0933          CLRL    -(SP)                   ;ALLOCATE A LONGWORD FOR DEVCHAR
           55   5E    D0  0935          MOVL    SP,R5                   ;MAKE R5 POINT TO IT
                7E    D4  0938          CLRL    -(SP)                   ;INDICATE BOTTOM OF ITEM LIST
           F4   AE    9F  093A          PUSHAB  -12(SP)                 ;ADDRESS OF WORD TO RECEIVE DEVCHAR LENGTH
                55    DD  093D          PUSHL   R5                      ;ADDRESS OF BUFFER TO RECEIVE DEVCHAR
     00030004   8F    DD  093F          PUSHL   #<DVI$_DEVCHAR+1>@16+4  ;SECONDARY DEVICE DEVCHAR ITEM CODE
           F4   AE    9F  0945          PUSHAB  -12(SP)                 ;ADDRESS OF WORD TO RECEIVE ITEM LENGTH
           7E   52    7D  0948          MOVQ    R2,-(SP)                ;SAVE DESCRIPTOR OF SCRATCH ITEM BUFFER
                          094B
                          094B  ;
                          094B  ; At this point, the stack looks like below:
                          094B  ;
                          094B                  :-------------------------:
                          094B  ; entry SP=>    :                         :
                          094B                  :-------------------------:
                          094B                  :                         :
                          094B                  :--- IOSB ----------------:
                          094B                  :                         :
                          094B                  :-------------------------:
                          094B  ; scratch len:  :    R2                   :
                          094B                  :-------------------------:
                          094B  ; R5 =>         :    0 (holds DEVCHAR)    :
                          094B                  :-------------------------:
                          094B  ; itmlst end:   :    0                    :
                          094B                  :-------------------------:
                          094B                  : addr of X (retlen)      :
                          094B                  :-------------------------:
                          094B                  :    R5                   :
                          094B                  :-------------------------:
                          094B  ; X:            : DVI$_DEVCHAR+1 ; len=4  :
                          094B                  :-------------------------:
                          094B                  : addr of Y (retlen)      :
```

```
                          094B       ;-----------------------;
                          094B       ;          R3           ;
                          094B       ;-----------------------;
                          094B ; Y:  ; DVI$_requested ; R2   ;
                          094B       ;-----------------------;
                          094B ; SP =>                        ;
                          094B       ;                        ;
                          094B       ;-----------------------;
                          094B
     56   00001683'EF  9E 094B           MOVAB   GETDVI_TABLE,R6          ;GET ADDRESS OF TABLE
              57   56  D0 0952 10$:      MOVL    R6,R7                    ;GET ADDRESS OF ENTRY
              56   04  C0 0955           ADDL    #4,R6                    ;GET ADDRESS OF ASCIC ITEM NAME
              58   66  9A 0958           MOVZBL  (R6),R8                  ;GET LENGTH OF ITEM NAME
                   70  13 095B           BEQL    80$                      ;BRANCH IF END OF TABLE
  00   10 BC    0C AC  2D 095D           CMPC5   P2(AP),@P2+4(AP),#0,-    ;DOES ITEM NAME MATCH?
         01 A6     58     0963                   R8,1(R6)                 ;
                   08  13 0966           BEQL    20$                      ;BRANCH IF ITEM FOUND
              56   58  C0 0968           ADDL    R8,R6                    ;SKIP TO NEXT ENTRY IN TABLE
              56   01  C0 096B           ADDL    #1,R6                    ;BE SURE TO COUNT THE COUNT
                   E2  11 096E           BRB     10$                      ;LOOP UNTIL FOUND
         02 AE     67  B0 0970 20$:      MOVW    (R7),2(SP)               ;SET ITEM CODE INTO LIST
              50   5E  D0 0974 30$:      MOVL    SP,R0                    ;GET ADDRESS OF ITEM LIST
                          0977           $GETDVIW_S DEVNAM=P1(AP),-       ;GET INFORMATION ON SPECIFIED DEVICE
                          0977                   ITMLST=(R0),-            ;
                          0977                   IOSB=8(R5),-             ;
                          0977                   EFN=#EXE$C_SYSEFN        ;
              45   50  E9 0993           BLBC    R0,90$                   ;BRANCH IF ERROR DETECTED
         50    08 A5   3C 0996           MOVZWL  8(R5),R0                 ;GET STATUS FROM IOSB
              3E   50  E9 099A           BLBC    R0,90$                   ;BRANCH IF ERROR DETECTED
              54   50  D0 099D           MOVL    R0,R4                    ;SAVE STATUS
         1D 65     06  E1 09A0           BBC     #DEV$V_SPL,(R5),50$      ;SPOOLED DEVICE?
                   56  D4 09A4           CLRL    R6                       ;SET SPOOLED FLAG
         0E AE     02  B0 09A6           MOVW    #DVI$_DEVCHAR,14(SP)     ;GET DEVCHAR OF PRIMARY DEVICE
         02 AE     20  B1 09AA           CMPW    #DVI$_DEVNAM,2(SP)       ;ARE WE GETTING DEVICE NAME?
                   08  12 09AE           BNEQ    40$                      ;NO, THEN NO SPECIAL PROCESSING
         04 A7     09  91 09B0           CMPB    #9,4(R7)                 ;ARE WE GETTING SPOOLED DEVICE NAME?
                   02  12 09B4           BNEQ    40$                      ;NO, THEN NO SPECIAL PROCESSING
                   BC  11 09B6           BRB     30$                      ;YES, THEN DON'T INCR ITEM CODE
         6E    04 A5   B0 09B8 40$:      MOVW    4(R5),(SP)               ;RESET RESULT LENGTH
              02 AE    B6 09BC           INCW    2(SP)                    ;GET SECONDARY DEVICE DATA
                   B3  11 09BF           BRB     30$                      ;RETRY
              02 AE    B4 09C1 50$:      CLRW    2(SP)                    ;CLEAR ITEM CODE, LEAVING RETLEN
              51   8E  7D 09C4           MOVQ    (SP)+,R1                 ;GET DESCRIPTOR OF STRING/LONGWORD
     5E       55   10  C1 09C7           ADDL3   #16,R5,SP                ;RESTORE STACK
                   33  11 09CB           BRB     GETDVI_VALUE             ;GET VALUE
                          09CD ;
                          09CD ; NO SUCH ITEM NAME
                          09CD ;
                          09CD 80$:      STATUS  IVKEYW                   ;UNDEFINED ITEM NAME
     5E       55   10  C1 09D4           ADDL3   #16,R5,SP                ;CLEAN STACK
                 0C43  31 09D8           BRW     P2_ERROR                 ;REPORT THE ERROR
                          09DB ;
                          09DB ; ERROR DETECTED BY GETDVI
                          09DB ;
     5E       55   10  C1 09DB 90$:      ADDL3   #16,R5,SP                ;CLEAN STACK
     50   00000000'8F  D1 09DF           CMPL    #SS$_IVDEVNAM,R0         ;YES, IS NAME INVALID?
                   09  13 09E6           BEQL    93$                      ;YES, THEN CHECK EXISTS BIT
     50   00000000'8F  D1 09E8           CMPL    #SS$_NOSUCHDEV,R0        ;NO, THEN DOES DEVICE EXIST?
```

```
           0C  12 09EF              BNEQ    95$                     ;NO, THEN DON'T CHECK BIT
  03 A7 FF 8F  91 09F1      93$:    CMPB    #DEV$V_EXISTS,3(R7)     ;DO WE CARE IF IT EXISTS
           05  12 09F6              BNEQ    95$                     ;NO, THEN REPORT ERROR
           51  D4 09F8              CLRL    R1                      ;YES, THEN RETURN FALSE
         0D85  31 09FA              BRW     FORMAT_BOOLEAN          ;RETURN THE VALUE
         0C18  31 09FD      95$:    BRW     P1_ERROR                ;REPORT THE ERROR
                  0A00      ;
                  0A00      ; NORMAL PROCESSING FOR LONGWORDS, STRINGS AND HEX VALUES
                  0A00      ;
                  0A00      GETDVI_VALUE:
                  0A00              CASE    2(R7),TYPE=B,<-         ;CASE ON TYPE OF FIELD
                  0A00                      100$,-                  ;BINARY
                  0A00                      200$,-                  ;ASCIC STRING
                  0A00                      300$,-                  ;OWNER UIC
                  0A00                      400$,-                  ;BOOLEAN
                  0A00                      500$,-                  ;HEX STRING
                  0A00                      600$,-                  ;PROTECTION MASK
                  0A00                      700$,-                  ;ACP TYPE
                  0A00                      800$>                   ;HEX STRING
                  0A15
                  0A15      ;
                  0A15      ; RETURN BINARY VALUE
                  0A15      ;
        51  62 D0 0A15      100$:   MOVL    (R2),R1                 ;GET LONGWORD VALUE
            52 D4 0A18              CLRL    R2                      ;INDICATE RESULT IS BINARY
            05    0A1A              RSB
                  0A1B
                  0A1B      ;
                  0A1B      ; RETURN ASCII STRING
                  0A1B      ;
     02 A7  20 B1 0A1B      200$:   CMPW    #DVI$_DEVNAM,2(R7)      ;ARE WE GETTING DEVICE NAME?
            0C 12 0A1F              BNEQ    210$                    ;NO, THEN NO SPECIAL PROCESSING
     04 A7  09 91 0A21              CMPB    #9,4(R7)                ;ARE WE GETTING SPOOLED DEVICE NAME?
            06 12 0A25              BNEQ    210$                    ;NO, THEN NO SPECIAL PROCESSING
            56 D5 0A27              TSTL    R6                      ;IS DEVICE SPOOLED?
            02 13 0A29              BEQL    210$                    ;YES, THEN USE FETCHED NAME
            51 D4 0A2B              CLRL    R1                      ;NO, THEN RETURN NULL STRING
            05    0A2D      210$:   RSB
                  0A2E
                  0A2E      ;
                  0A2E      ; RETURN OWNER UIC
                  0A2E      ;
        54  62 D0 0A2E      300$:   MOVL    (R2),R4                 ;PASS UIC LONGWORD
          0C3B 31 0A31              BRW     FORMAT_UIC              ;CONVERT INTO ASCII FORM
                  0A34
                  0A34      ;
                  0A34      ; RETURN BOOLEAN TRUE/FALSE
                  0A34      ;
        51  62 9A 0A34      400$:   MOVZBL  (R2),R1                 ; FETCH THE RETURNED VALUE
  03 A7 FF 8F  91 0A37              CMPB    #DEV$V_EXISTS,3(R7)     ; ARE WE CHECKING FOR DEVICE EXISTS?
            03 12 0A3C              BNEQ    410$                    ; NEQ MEANS WE AREN'T
        51  01 D0 0A3E              MOVL    #1,R1                   ; IT ALWAYS EXISTS (AT THIS POINT)
          0D3E 31 0A41      410$:   BRW     FORMAT_BOOLEAN          ; RETURN "TRUE" OR "FALSE"
                  0A44
                  0A44
                  0A44      ;
                  0A44      ; RETURN HEX STRING
```

```
                                0A44      ;
        50    62    D0          0A44      500$:   MOVL    (R2),R0                 ;PASS HEX LONGWORD AS ARGUMENT
              E4    13          0A47              BEQL    210$                    ;IF ZERO, RETURN NULL STRING
     00000000'EF    17          0A49              JMP     DCL$CBTA_HEX            ;CONVERT HEX TO STRING AND EXIT
                                0A4F
                                0A4F      ;
                                0A4F      ; RETURN PROTECTION STRING
                                0A4F      ;
        54    62    D0          0A4F      600$:   MOVL    (R2),R4                 ;PASS PROTECTION MASK
            0C80    31          0A52              BRW     FORMAT_PROT             ;CONVERT INTO ASCII FORM
                                0A55
                                0A55      ;
                                0A55      ; RETURN ACP TYPE
                                0A55      ;
        50    62    D0          0A55      700$:   MOVL    (R2),R0                 ;GET ACP TYPE NUMBER
52  00001E5E'EF40   7E          0A58              MOVAQ   ACP_TABLE[R0],R2        ;GET ADDRESS OF ASCIC ACP TYPE
        51    82    9A          0A60              MOVZBL  (R2)+,R1                ;SET DESCRIPTOR OF ACP TYPE
              05                0A63              RSB
                                0A64
                                0A64      ;
                                0A64      ; RETURN HEX STRING
                                0A64      ;
              51    D5          0A64      800$:   TSTL    R1                      ;IS LENGTH ZERO?
              06    13          0A66              BEQL    810$                    ;YES, RETURN NULL STRING
     00000000'EF    17          0A68              JMP     DCL$CBTA_HEXSTR         ;CONVERT HEX TO STRING AND EXIT
                    05          0A6E      810$:   RSB                             ;
```

```
                              0A6F          .SBTTL  GETJPI SYSTEM SERVICE
                              0A6F     ;+
                              0A6F     ; F$GETJPI - GET JOB/PROCESS INFORMATION
                              0A6F     ;
                              0A6F     ; THIS LEXICAL FUNCTION INVOKES THE $GETJPI SYSTEM SERVICE TO
                              0A6F     ; OBTAIN ANY PIECE OF INFORMATION FROM A GIVEN PROCESS.
                              0A6F     ;
                              0A6F     ;       F$GETJPI(PID,ITEM)
                              0A6F     ;
                              0A6F     ; INPUTS:
                              0A6F     ;
                              0A6F     ;       P1(AP) = DESCRIPTOR OF PID STRING OF DESIRED PROCESS,
                              0A6F     ;                       NULL OR 0 FOR CURRENT PROCESS
                              0A6F     ;       P2(AP) = DESCRIPTOR OF $JPI ITEM NAME
                              0A6F     ;
                              0A6F     ; OUTPUTS:
                              0A6F     ;
                              0A6F     ;       R1/R2 = DESCRIPTOR OF RESULTANT DATA
                              0A6F     ;-
                              0A6F
                              0A6F     F$GETJPI:
              F58E'     30    0A6F          BSBW    DCL$MARK                ;MARK POSITION IN EXPANSION BUFFER
                 7E     7C    0A72          CLRQ    -(SP)                   ;RESERVE IOSB
                 7E     7C    0A74          CLRQ    -(SP)                   ;CLEAR LENGTH AND ITEM ID OF LAST ITEM
                              0A76                                          ;AND ALLOCATE SPACE FOR PID LONGWORD
              F4 AE     9F    0A76          PUSHAB  -12(SP)                 ;ADDRESS OF WORD TO RECEIVE LENGTH
           7E    52     7D    0A79          MOVQ    R2,-(SP)                ;SAVE DESCRIPTOR OF SCRATCH BUFFER
   56    00000100'EF    9E    0A7C          MOVAB   GETJPI_TABLE,R6         ;GET ADDRESS OF TABLE
           57    86     D0    0A83     10$: MOVL    (R6)+,R7                ;GET JPI CODE(0:15),FLAGS(16:23),
                              0A86                                          ;   ITEM STRING LENGTH(24:31)
                 03     12    0A86          BNEQ    15$                     ;SKIP IF NOT AT END OF TABLE
               0092     31    0A88          BRW     80$                     ;ERROR IF END OF TABLE
        58    FF A6     9A    0A8B     15$: MOVZBL  -1(R6),R8               ;GET LENGTH OF ITEM NAME
00   10 BC    0C AC     2D    0A8F          CMPC5   P2(AP),@P2+4(AP),#0,-   ;DOES ITEM NAME MATCH?
           66    58           0A95                  R8,(R6)                 ;
                 05     13    0A97          BEQL    20$                     ;BRANCH IF ITEM FOUND
           56    58     C0    0A99          ADDL    R8,R6                   ;SKIP TO NEXT ENTRY IN TABLE
                 E5     11    0A9C          BRB     10$                     ;LOOP UNTIL FOUND
        02 AE    57     B0    0A9E     20$: MOVW    R7,2(SP)                ;SET ITEM CODE INTO LIST
        52    04 AC     7D    0AA2          MOVQ    P1(AP),R2               ;GET DESCRIPTOR OF PID STRING
           51    00     9A    0AA6          MOVZBL  #PRC_K_HEX,R1           ;SET RADIX FOR CONVERSION
              F554'     30    0AA9          BSBW    DCL$CNVASCBIN           ;CONVERT HEX STRING TO BINARY IN R1
                 62     12    0AAC          BNEQ    70$                     ;BRANCH IF CONVERSION ERROR
        10 AE    51     D0    0AAE          MOVL    R1,16(SP)               ;STORE PID LONGWORD
           50    5E     D0    0AB2          MOVL    SP,R0                   ;GET ADDRESS OF ITEM LIST
                              0AB5          $GETJPIW_S PIDADR=16(R0),-      ;GET INFORMATION ON SPECIFIED PROCESS
                              0AB5                  ITMLST=(R0),-           ;
                              0AB5                  IOSB=20(R0),-           ;
                              0AB5                  EFN=#EXE$C_SYSEFN
              53 50     E9    0ACE          BLBC    R0,90$                  ;BRANCH IF ERROR DETECTED
        50    14 AE     3C    0AD1          MOVZWL  20(SP),R0               ;GET STATUS FROM IOSB
              4C 50     E9    0AD5          BLBC    R0,90$                  ;BRANCH IF ERROR DETECTED
              02 AE     B4    0AD8          CLRW    2(SP)                   ;CLEAR ITEM CODE, LEAVING RETLEN
                 06     BA    0ADB          POPR    #^M<R1,R2>              ;GET DESCRIPTOR OF STRING/LONGWORD
           5E    14     C0    0ADD          ADDL    #5*4,SP                 ;AND CLEAN STACK
   0322 8F       57     B1    0AE0          CMPW    R7,#JPI$_MODE           ;ASKING FOR MODE?
                 03     12    0AE5          BNEQ    30$                     ;IF NOT, THEN SKIP
```

```
              00E0  31  0AE7         BRW     STS_MODE              ;IF SO, CHECK IF USER WANTED "MODE"
         0206 8F  57  B1  0AEA  30$:  CMPW    R7,#JPI$_LOGINTIM     ;ASKING FOR LOGIN TIME?
                03  12  0AEF         BNEQ    40$                   ;IF NOT, THEN SKIP
              00A6  31  0AF1         BRW     GETJPI_LOGINTIM       ;IF SO, FORMAT AS DATE/TIME
         0306 8F  57  B1  0AF4  40$:  CMPW    R7,#JPI$_STATE        ;ASKING FOR STATE?
                03  12  0AF9         BNEQ    50$                   ;IF NOT, THEN SKIP
              00B6  31  0AFB         BRW     GETJPI_STATE          ;IF SO, RETURN STATE KEYWORD
         0304 8F  57  B1  0AFE  50$:  CMPW    R7,#JPI$_UIC          ;ASKING FOR UIC?
                03  12  0B03         BNEQ    60$                   ;IF NOT, THEN SKIP
              00BB  31  0B05         BRW     GETJPI_UIC            ;IF SO, FORMAT UIC SPECIALLY
   57    57  08  10  EF  0B08  60$:  EXTZV   #16,#8,R7,R7          ;EXTRACT ITEM TYPE FIELD
              002B  31  0B0D         BRW     GETJPI_VALUE          ;GET VALUE
                        0B10  ;
                        0B10  ; ERROR CONVERTING PID STRING TO A BINARY LONGWORD
                        0B10  ;
                        0B10  70$:  STATUS  EXPSYN                ;EXPRESSION SYNTAX ERROR
           5E  1C  C0  0B17         ADDL    #7*4,SP               ;CLEAN STACK
              0AFB  31  0B1A  75$:  BRW     P1_ERROR              ;REPORT ERROR
                        0B1D  ;
                        0B1D  ; NO SUCH ITEM NAME
                        0B1D  ;
                        0B1D  80$:  STATUS  IVKEYW                ;UNDEFINED ITEM NAME
                        0B24  ;
                        0B24  ; ERROR DETECTED BY GETJPI
                        0B24  ;
           5E  1C  C0  0B24  90$:  ADDL    #7*4,SP               ;CLEAN STACK
        0000'8F  50  B1  0B27         CMPW    R0,#SS$_SUSPENDED     ;SUSPENDED ERROR?
                0A  13  0B2C         BEQL    95$                   ;RETURN NULL/ZERO IF SUSPENDED
        0000'8F  50  B1  0B2E         CMPW    R0,#SS$_NONEXPR       ;NONEXISTANT PROCESS?
                E5  13  0B33         BEQL    75$                   ;REPORT P1 ERROR
              0AE6  31  0B35         BRW     P2_ERROR              ;REPORT ERROR
                51  D4  0B38  95$:  CLRL    R1                    ;RETURN NULL STRING
                    05  0B3A         RSB
                        0B3B  ;
                        0B3B  ; NORMAL PROCESSING FOR LONGWORDS, STRINGS AND HEX VALUES
                        0B3B  ;
                        0B3B  GETJPI_VALUE:
                        0B3B         CASE    R7,TYPE=B,<-          ;CASE ON ITEM TYPE
                        0B3B                 10$,-                 ;BINARY
                        0B3B                 20$,-                 ;STRING
                        0B3B                 30$,-                 ;HEX LONGWORD STRING
                        0B3B                 40$,-                 ;PRIVILEGE LIST
                        0B3B                 60$,-                 ;BITVAL
                        0B3B                 70$>                  ;HEX STRING
                        0B4B  ;
                        0B4B  ; RETURN BINARY VALUE
                        0B4B  ;
           51  62  D0  0B4B  10$:  MOVL    (R2),R1               ;GET LONGWORD VALUE
                52  D4  0B4E         CLRL    R2                    ;INDICATE RESULT IS BINARY
                    05  0B50  20$:  RSB
                        0B51
                        0B51  ;
                        0B51  ; RETURN HEX STRING
                        0B51  ;
                51  D4  0B51  30$:  CLRL    R1                    ;ASSUME RETURNING NULL STRING
           50  62  D0  0B53         MOVL    (R2),R0               ;PASS LONGWORD PID AS ARGUMENT
                F8  13  0B56         BEQL    20$                   ;IF ZERO, RETURN NULL STRING
```

```
         00000000'EF   17  0B58              JMP     DCL$CBTA_HEX                  ;CONVERT PID TO STRING AND EXIT
                           0B5E
                           0B5E      ;
                           0B5E      ; RETURN PRIVILEGE LIST
                           0B5E      ;
             7E   62   7D  0B5E      40$:    MOVQ    (R2),-(SP)                    ;PUSH PRIVILEGE MASK ON STACK
             56   5E   D0  0B61              MOVL    SP,R6                         ;GET ADDRESS OF MASK
                  51   D4  0B64              CLRL    R1                            ;START WITH PRIV #0
          03 66   51   E1  0B66      45$:    BBC     R1,(R6),50$                   ;SKIP IF PRIV IS NOT PRESENT
                0959   30  0B6A              BSBW    ADD_PRIV_STATE                ;APPEND EACH KEYWORD TO STRING
   F1 51   00000040 8F F2  0B6D      50$:    AOBLSS  #64,R1,45$                    ;LOOP FOR EACH PRIV IN MASK
               F488'   30  0B75              BSBW    DCL$MARKEDTOKEN               ;GET DESCRIPTOR OF STRING
             F48A CA   D0  0B78              MOVL    WRK_L_MARKPTR(R10),-          ;RESTORE POSITION IN EXPANSION BUFFER
             F486 CA       0B7C                      WRK_L_EXPANDPTR(R10)
                  51   D5  0B7F              TSTL    R1                            ;WERE NO PRIVS FOUND?
                  02   13  0B81              BEQL    55$                           ;YES, THEN NO TRAILING COMMA
                  51   D7  0B83              DECL    R1                            ;REMOVE TRAILING COMMA
             5E   08   C0  0B85      55$:    ADDL    #8,SP                         ;RESTORE STACK
                       05  0B88              RSB
                           0B89
                           0B89      ;
                           0B89      ; RETURN BOOLEAN
                           0B89      ;
             51   62   9A  0B89      60$:    MOVZBL  (R2),R1                       ;FETCH THE VALUE
                0BF3   31  0B8C              BRW     FORMAT_BOOLEAN                ;RETURN "TRUE" OR "FALSE"
                           0B8F
                           0B8F      ;
                           0B8F      ; RETURN HEX STRING
                           0B8F      ;
                  51   D5  0B8F      70$:    TSTL    R1                            ;IS LENGTH ZERO?
                  06   13  0B91              BEQL    75$                           ;YES, RETURN NULL STRING
         00000000'EF   17  0B93              JMP     DCL$CBTA_HEXSTR               ;CONVERT HEX TO STRING AND EXIT
                       05  0B99      75$:    RSB                                   ;
                           0B9A
                           0B9A      ;
                           0B9A      ; SPECIAL PROCESSING FOR "LOGINTIM"
                           0B9A      ;
                           0B9A      GETJPI_LOGINTIM:
                  52   DD  0B9A              PUSHL   R2                            ;ADDRESS OF RESULT BUFFER
                  17   DD  0B9C              PUSHL   #23                           ;RESULT BUFFER LENGTH
             50   5E   D0  0B9E              MOVL    SP,R0
                           0BA1              $ASCTIM_S TIMBUF=(R0),-               ;CONVERT QUADWORD TIME TO ASCII
                           0BA1                      TIMADR=(R2)
             51   8E   7D  0BB0              MOVQ    (SP)+,R1                      ;PICK UP RESULTANT STRING DESCRIPTOR
                       05  0BB3              RSB
                           0BB4
                           0BB4      ;
                           0BB4      ; SPECIAL PROCESSING FOR "STATE"
                           0BB4      ;
                           0BB4      GETJPI_STATE:
             50   62   D0  0BB4              MOVL    (R2),R0                       ;GET STATE NUMBER
   52   00001491'EF40  7E  0BB7              MOVAQ   STATE_TABLE[R0],R2            ;GET ADDRESS OF ASCIC STATE NAME
             51   82   9A  0BBF              MOVZBL  (R2)+,R1                      ;SET DESCRIPTOR OF STATE NAME
                       05  0BC2              RSB
                           0BC3
                           0BC3      ;
                           0BC3      ; SPECIAL PROCESSING FOR "UIC"
```

```
                    0BC3  2679 ;
                    0BC3  2680 GETJPI_UIC:
       54    62  D0 0BC3  2681         MOVL    (R2),R4                 ;GET UIC LONGWORD
           0AA6  30 0BC6  2682         BSBW    FORMAT_UIC              ;FORMAT UIC INTO ASCII STRING
                 05 0BC9  2683         RSB
                    0BCA  2684
                    0BCA  2685 ;
                    0BCA  2686 ; WE JUST GOT JPI$_STS - CHECK IF USER WANTS "MODE"
                    0BCA  2687 ;
                    0BCA  2688 STS_MODE:
       54    62  D0 0BCA  2689         MOVL    (R2),R4                 ;GET MODE LONGWORD
             53  D4 0BCD  2690         CLRL    R3                      ;SET LOOP INDEX
  52  000021F3'EF 9E 0BCF  2691        MOVAB   MODE_TABLE,R2           ;GET ADDRESS OF MODE TABLE
       51    82  9A 0BD6  2692 25$:    MOVZBL  (R2)+,R1                ;GET LENGTH OF MODE
             0E  13 0BD9  2693         BEQL    40$                     ;BRANCH IF END OF TABLE
       53    54  D1 0BDB  2694         CMPL    R4,R3                   ;IS THIS THE ONE WE WANT?
             09  13 0BDE  2695         BEQL    40$                     ;YES, THEN RETURN THE STRING
   52  01 A142  9E 0BE0  2696         MOVAB   1(R1)[R2],R2            ;SKIP TO NEXT TABLE ENTRY
             53  D6 0BE5  2697         INCL    R3                      ;INCR LOOP INDEX
             ED  11 0BE7  2698         BRB     25$                     ;
             05     0BE9  2699 40$:    RSB                             ;
```

```
                              0BEA          .SBTTL  GETSYI SYSTEM SERVICE
                              0BEA      ;+
                              0BEA      ; F$GETSYI - GET SYSTEM INFORMATION
                              0BEA      ;
                              0BEA      ; THIS LEXICAL FUNCTION INVOKES THE $GETSYI SYSTEM SERVICE TO
                              0BEA      ; OBTAIN ANY PIECE OF INFORMATION FOR THE CURRENT SYSTEM.
                              0BEA      ;
                              0BEA      ;       F$GETSYI(ITEM,NODE)
                              0BEA      ;
                              0BEA      ; INPUTS:
                              0BEA      ;
                              0BEA      ;       P1(AP) = DESCRIPTOR OF $SYI ITEM NAME
                              0BEA      ;       P2(AP) = DESCRIPTOR OF NODE NAME OF DESIRED SYSTEM,
                              0BEA      ;                       NULL OR 0 FOR CURRENT SYSTEM
                              0BEA      ;
                              0BEA      ; OUTPUTS:
                              0BEA      ;
                              0BEA      ;       R1/R2 = DESCRIPTOR OF RESULTANT DATA
                              0BEA      ;-
                              0BEA
                              0BEA      F$GETSYI:
                   7E    7C   0BEA              CLRQ    -(SP)                   ;RESERVE IOSB
                   7E    7C   0BEC              CLRQ    -(SP)                   ;RESERVE CSID AND CLEAR LENGTH AND ITEM ID 0
                F4 AE    9F   0BEE              PUSHAB  -12(SP)                 ;ADDRESS OF WORD TO RECEIVE LENGTH
               7E  52    7D   0BF1              MOVQ    R2,-(SP)                ;SAVE DESCRIPTOR OF SCRATCH BUFFER
   56  00000480'EF       9E   0BF4              MOVAB   GETSYI_TABLE,R6         ;GET ADDRESS OF TABLE
               57  86    D0   0BFB      10$:    MOVL    (R6)+,R7                ;GET SYI CODE(0:15),TYPE(16:23),
                              0BFE                                              ;   ITEM STRING LENGTH(24:31)
                   03    12   0BFE              BNEQ    15$                     ;BRANCH IF NOT END OF TABLE
                 00AA    31   0C00              BRW     80$                     ;BRANCH IF END OF TABLE
            58  FF A6    9A   0C03      15$:    MOVZBL  -1(R6),R8               ;GET LENGTH OF ITEM NAME
   00    08 BC  04 AC    2D   0C07              CMPC5   P1(AP),@P1+4(AP),#0,-   ;DOES ITEM NAME MATCH?
               66  58         0C0D                      R8,(R6)                 ;
                   05    13   0C0F              BEQL    20$                     ;BRANCH IF ITEM FOUND
               56  58    C0   0C11              ADDL    R8,R6                   ;SKIP TO NEXT ENTRY IN TABLE
                   E5    11   0C14              BRB     10$                     ;LOOP UNTIL FOUND
            02 AE  57    B0   0C16      20$:    MOVW    R7,2(SP)                ;SET ITEM CODE INTO LIST
                0C AC    D5   0C1A              TSTL    P2(AP)                  ;P2 SPECIFIED?
                   05    19   0C1D              BLSS    25$                     ;BRANCH IF NOT
      10 AE     0C AC    7E   0C1F              MOVAQ   P2(AP),16(SP)           ;GET ADDRESS OF NODE DESCRIPTOR
               50  5E    D0   0C24      25$:    MOVL    SP,R0                   ;GET ADDRESS OF ITEM LIST
                              0C27              $GETSYIW_S ITMLST=(R0),-        ;GET INFORMATION
                              0C27                      NODENAME=@16(R0),-      ;
                              0C27                      IOSB=20(R0),-           ;
                              0C27                      EFN=#EXE$C_SYSEFN
               71  50    E9   0C40              BLBC    R0,90$                  ;BRANCH IF ERROR DETECTED
            50  14 AE    3C   0C43              MOVZWL  20(SP),R0               ;GET STATUS FROM IOSB
               6A  50    E9   0C47              BLBC    R0,90$                  ;BRANCH IF ERROR DETECTED
               02  AE    B4   0C4A              CLRW    2(SP)                   ;CLEAR ITEM CODE, LEAVING RETLEN
                   06    BA   0C4D              POPR    #^M<R1,R2>              ;GET DESCRIPTOR OF STRING/LONGWORD
               5E  14    C0   0C4F              ADDL    #5*4,SP                 ;AND CLEAN STACK
                              0C52
   57    57    08  10    EF   0C52              EXTZV   #16,#8,R7,R7            ;GET ITEM TYPE
                              0C57
                              0C57              CASE    R7,TYPE=B,<-            ;CASE ON ITEM TYPE
                              0C57                      91$,-                   ;BINARY
                              0C57                      40$,-                   ;STRING
```

```
                          0C57                            30$,-                       ;HEX LONGWORD STRING
                          0C57                            91$,-                       ;PRIVILEGE LIST
                          0C57                            96$,-                       ;BITVAL
                          0C57                            30$,-                       ;HEX STRING
                          0C57                            100$>                       ;STDTIM
                          0C69
        51      62    9A  0C69  96$:     MOVZBL  (R2),R1                     ;FETCH THE VALUE
           0B13       31  0C6C           BRW     FORMAT_BOOLEAN              ;RETURN "TRUE" OR "FALSE"
        51      62    D0  0C6F  91$:     MOVL    (R2),R1                     ;GET LONGWORD VALUE
                52    D4  0C72           CLRL    R2                          ;INDICATE RESULT IS BINARY
                0A    11  0C74           BRB     40$                         ;
                51    D5  0C76  30$:     TSTL    R1                          ;IS LENGTH ZERO?
                06    13  0C78           BEQL    40$                         ;YES, RETURN NULL STRING
       00000000'EF    17  0C7A           JMP     DCL$CBTA_HEXSTR             ;CONVERT HEX TO STRING AND EXIT
                      05  0C80  40$:     RSB                                 ;RETURN
                          0C81
        F486    CA    C3  0C81  100$:    SUBL3   WRK_L_EXPANDPTR(R10),-      ;CALCULATE SIZE OF BUFFER
   50   F48E    CA        0C85                   WRK_L_CHARPTR(R10),R0       ;
        17      50    D1  0C89           CMPL    R0,#23                      ;IS BUFFER LARGE ENOUGH?
                1C    1F  0C8C           BLSSU   110$                        ;NO, THEN ERROR
        F486    CA    DD  0C8E           PUSHL   WRK_L_EXPANDPTR(R10)        ;CREATE DESCRIPTOR OF SCRATCH BUFFER
                50    DD  0C92           PUSHL   R0                          ;
        50      5E    D0  0C94           MOVL    SP,R0                       ;GET ADDRESS OF BUFFER DESCRIPTOR
                          0C97           $ASCTIM_S  TIMLEN=(R0),-            ;CONVERT QUADWORD TIME TO ASCII
                          0C97                   TIMBUF=(R0),-               ;
                          0C97                   TIMADR=(R2)                 ;
        51      8E    7D  0CA6           MOVQ    (SP)+,R1                    ;
                      05  0CA9           RSB                                 ;
                          0CAA
           095F       31  0CAA  110$:    BRW     BUFOVF                      ;SIGNAL BUFFER OVERFLOW ERROR
                          0CAD
                          0CAD  ;
                          0CAD  ; NO SUCH ITEM NAME
                          0CAD  ;
                          0CAD  80$:     STATUS  IVKEYW                      ;UNDEFINED ITEM NAME
                          0CB4
                          0CB4  ;
                          0CB4  ; ERROR DETECTED BY GETSYI
                          0CB4  ;
        5E      1C    C0  0CB4  90$:     ADDL    #7*4,SP                     ;CLEAN STACK
 50    00000000'8F    D1  0CB7           CMPL    #SS$_NOSUCHNODE,R0          ;CHECK FOR NON-EXISTENT NODE
                0C    12  0CBE           BNEQ    92$                         ;NEQ MEANS IT WASN'T THAT ERROR
   57   10CF    8F    B1  0CC0           CMPW    #SYI$_CLUSTER_MEMBER,R7     ;ARE WE CHECKING FOR EXISTENCE?
                08    12  0CC5           BNEQ    93$                         ;NEQ MEANS WE ARE NOT
                51    D4  0CC7           CLRL    R1                          ;SET 'FALSE'
           0AB6       31  0CC9           BRW     FORMAT_BOOLEAN              ;RETURN "FALSE" TO THE USER
           0949       31  0CCC  92$:     BRW     P1_ERROR                    ;REPORT ERROR
           094C       31  0CCF  93$:     BRW     P2_ERROR                    ;REPORT ERROR
```

```
                              0CD2          .SBTTL  CONVERT IDENTIFIER
                              0CD2  ;+
                              0CD2  ; F$IDENTIFIER - CONVERT IDENTIFIER
                              0CD2  ;
                              0CD2  ; THIS FUNCTION CONVERTS IDENTIFIERS BETWEEN STRINGS AND NUMBERS.
                              0CD2  ;
                              0CD2  ;       F$IDENTIFIER (IDENTIFIER, DIRECTION)
                              0CD2  ;
                              0CD2  ; INPUTS:
                              0CD2  ;
                              0CD2  ;       P1(AP) = IDENTIFIER NAME OR NUMBER
                              0CD2  ;       P2(AP) = "NUMBER" OR "NAME"
                              0CD2  ;
                              0CD2  ; OUTPUTS:
                              0CD2  ;
                              0CD2  ;       R1/R2 = DESCRIPTOR OF RESULTANT IDENTIFIER
                              0CD2  ;-
                              0CD2
                              0CD2  F$IDENTIFIER:
     56    000021D2'EF   9E   0CD2          MOVAB   IDENT_TABLE,R6               ;GET ADDRESS OF TABLE
              57    86   9A   0CD9  10$:    MOVZBL  (R6)+,R7                     ;GET LENGTH OF ITEM NAME
                    5F   13   0CDC          BEQL    95$                          ;ERROR IF ZERO
  00    10 BC    0C AC   2D   0CDE          CMPC5   P2(AP),@P2+4(AP),#0,-        ;DOES ITEM NAME MATCH?
              66    57        0CE4                  R7,(R6)                      ;
                    07   13   0CE6          BEQL    20$                          ;BRANCH IF ITEM FOUND
              56    57   C0   0CE8          ADDL    R7,R6                        ;SKIP TO NEXT ENTRY IN TABLE
                    56   D6   0CEB          INCL    R6                           ;
                    EA   11   0CED          BRB     10$                          ;LOOP UNTIL FOUND
                  6647   95   0CEF  20$:    TSTB    (R6)[R7]                     ;"NAME TO NUMBER"?
                    2A   13   0CF2          BEQL    50$                          ;YES, THEN BRANCH
                              0CF4
               F486 CA   DD   0CF4          PUSHL   WRK_L_EXPANDPTR(R10)         ;CREATE DESCRIPTOR OF SCRATCH BUFFER
               F486 CA   C3   0CF8          SUBL3   WRK_L_EXPANDPTR(R10),-       ;
          7E   F48E CA        0CFC                  WRK_L_CHARPTR(R10),-(SP);    ;
          50        5E   D0   0D00          MOVL    SP,R0                        ;
                              0D03          $IDTOASC_S  ID=P1(AP),-              ;CONVERT NUMBER TO NAME
                              0D03                  NAMLEN=(R0),-                ;
                              0D03                  NAMBUF=(R0)                  ;
              51    8E   7D   0D17          MOVQ    (SP)+,R1                     ;GET RESULT DESCRIPTOR
              1D    50   E9   0D1A          BLBC    R0,90$                       ;BRANCH IF ERROR
                    05        0D1D          RSB                                  ;RETURN
                              0D1E
                    7E   D4   0D1E  50$:    CLRL    -(SP)                        ;ALLOCATE A RETURN BUFFER
              50    5E   D0   0D20          MOVL    SP,R0                        ;
                              0D23          $ASCTOID_S  NAME=P1(AP),-            ;CONVERT NAME TO NUMBER
                              0D23                  ID=(R0)
              51    8E   D0   0D31          MOVL    (SP)+,R1                     ;GET RESULT
                    52   D4   0D34          CLRL    R2                           ;
              01    50   E9   0D36          BLBC    R0,90$                       ;BRANCH IF ERROR
                    05        0D39          RSB                                  ;RETURN
                              0D3A
                              0D3A
                    51   D4   0D3A  90$:    CLRL    R1                           ;RETURN THE NULL STRING
                    05        0D3C          RSB
                              0D3D  95$:    STATUS  IVKEYW                       ;REPORT KEYWORD ERROR
                  08D7   31   0D44          BRW     P2_ERROR                     ;
```

```
                          0D47        .SBTTL  CONVERT TO INTEGER
                          0D47  ;+
                          0D47  ; F$INTEGER - CONVERT TO INTEGER
                          0D47  ;
                          0D47  ; THIS FUNCTION CONVERTS ANY VALUE TO AN INTEGER
                          0D47  ;
                          0D47  ;       F$INTEGER(EXPRESSION)
                          0D47  ;
                          0D47  ; INPUTS:
                          0D47  ;
                          0D47  ;       P1(AP) = EXPRESSION RESULT
                          0D47  ;
                          0D47  ; OUTPUTS:
                          0D47  ;
                          0D47  ;       R1 = INTEGER
                          0D47  ;       R2 = 0 (TO INDICATE RESULT IS BINARY RATHER THAN ASCII)
                          0D47  ;-
                          0D47
                          0D47  F$INTEGER:
       51    04 AC    7D  0D47          MOVQ    P1(AP),R1               ;GET EXPRESSION VALUE
      00000000'EF     16  0D4B          JSB     DCL$CVT_BINARY          ;CONVERT STRING TO BINARY
             01 50    E9  0D51          BLBC    R0,90$                  ;BRANCH IF ERROR
                      05  0D54          RSB
             08C0     31  0D55  90$:    BRW     P1_ERROR                ;REPORT ERROR AND DISPLAY P1
```

```
                    0D58          .SBTTL  RETURN LENGTH OF STRING
                    0D58  ;+
                    0D58  ; F$LENGTH - RETURN LENGTH OF STRING
                    0D58  ;
                    0D58  ; THIS LEXICAL FUNCTION RETURNS THE LENGTH OF ITS ARGUMENT AS A DECIMAL STRING.
                    0D58  ;
                    0D58  ;       F$LENGTH(STRING)
                    0D58  ;
                    0D58  ; INPUTS:
                    0D58  ;
                    0D58  ;       P1(AP) = DESCRIPTOR OF STRING.
                    0D58  ;
                    0D58  ; OUTPUTS:
                    0D58  ;
                    0D58  ;       R1 = STRING LENGTH
                    0D58  ;       R2 = 0 TO INDICATE RESULT IS BINARY
                    0D58  ;-
                    0D58
                    0D58  F$LENGTH:                                     ;RETURN LENGTH OF STRING
51    04 AC    D0   0D58          MOVL    P1(AP),R1                     ;GET VALUE TO BE CONVERTED
         52    D4   0D5C          CLRL    R2                            ;INDICATE BINARY RESULT
               05   0D5E          RSB
```

```
                                   0D5F              .SBTTL  LOCATE SUBSTRING IN STRING
                                   0D5F      ;+
                                   0D5F      ; F$LOCATE - LOCATE SUBSTRING IN STRING
                                   0D5F      ;
                                   0D5F      ; THIS LEXICAL FUNCTION RETURNS THE STARTING POSITION OF A SUBSTRING IN ANOTHER
                                   0D5F      ; STRING. IF THE SUBSTRING IS NOT CONTAINED IN THE SPECIFIED STRING, THEN THE
                                   0D5F      ; LENGTH OF THE SPECIFIED STRING IS RETURNED.
                                   0D5F      ;
                                   0D5F      ;       F$LOCATE(SUBSTRING, STRING)
                                   0D5F      ;
                                   0D5F      ; INPUTS:
                                   0D5F      ;
                                   0D5F      ;       P1(AP) = DESCRIPTOR OF SUBSTRING.
                                   0D5F      ;       P2(AP) = DESCRIPTOR OF STRING TO BE SEARCHED.
                                   0D5F      ;
                                   0D5F      ; OUTPUTS:
                                   0D5F      ;
                                   0D5F      ;       R1 = NUMBER INDICATING POSITION WITHIN STRING
                                   0D5F      ;       R2 = 0 TO INDICATE RESULT IS BINARY
                                   0D5F      ;-
                                   0D5F
                                   0D5F      F$LOCATE:                                              ;LOCATE SUBSTRING IN STRING
10 BC   0C AC   08 BC   04 AC   39 0D5F              MATCHC  P1(AP),@P1+4(AP),P2(AP),@P2+4(AP) ;SEARCH FOR SUBSTRING MATCH
                           04   12 0D68              BNEQ    10$                            ;IF NEQ NO MATCH FOUND
                   53   04 AC   C2 0D6A              SUBL    P1(AP),R3                      ;CALCULATE STARTING ADDRESS OF SUBSTRING
           51      53   10 AC   C3 0D6E      10$:    SUBL3   P2+4(AP),R3,R1                 ;CALCULATE STARTING POSITION OF SUBSTRING
                           52   D4 0D73              CLRL    R2                             ;MARK RESULT IS BINARY
                                05 0D75              RSB
```

```
                          0D76          .SBTTL  LOGICAL NAME TRANSLATION
                          0D76     ;+
                          0D76     ; F$LOGICAL - LOGICAL NAME TRANSLATION
                          0D76     ;
                          0D76     ; THIS LEXICAL FUNCTION OBTAINS THE TRANSLATION FOR A SPECIFED LOGICAL NAME AND
                          0D76     ; RETURN THE EQUIVALENCE NAME.
                          0D76     ;
                          0D76     ;       F$LOGICAL(LOGICAL_NAME)
                          0D76     ;
                          0D76     ; INPUTS:
                          0D76     ;
                          0D76     ;       P1(AP) = DESCRIPTOR OF LOGICAL NAME STRING.
                          0D76     ;
                          0D76     ; OUTPUTS:
                          0D76     ;
                          0D76     ;       R1 = LENGTH OF EQUIVALENCE STRING.
                          0D76     ;       R2 = ADDRESS OF EQUIVALENCE STRING.
                          0D76     ;-
                          0D76
                          0D76     F$LOGICAL:                                   ;LOGICAL NAME TRANSLATION
    50    04 AC    9E     0D76          MOVAB   P1(AP),R0                    ;GET ADDRESS OF LOGICAL NAME DESCRIPTOR
       7E    52    7D     0D7A          MOVQ    R2,-(SP)                     ;SAVE EQUIVALENCE NAME DESCRIPTOR
       7E    52    7D     0D7D          MOVQ    R2,-(SP)                     ;SET EQUIVALENCE NAME DESCRIPTOR
       51    5E    D0     0D80          MOVL    SP,R1                        ;SET ADDRESS OF EQUIVALENCE NAME DESCRIPTOR
                          0D83          $TRNLOG_S (R0),(R1),(R1)             ;TRANSLATE LOGICAL NAME
       51    8E    7D     0D96          MOVQ    (SP)+,R1                     ;RETRIEVE EQUIVALENCE NAME PARAMETERS
          54 50    E9     0D99          BLBC    R0,90$                       ;IF LBC TRANSLATION FAILURE
    50  0000'8F    B1     0D9C          CMPW    #SS$_NOTRAN,R0               ;DID TRANSLATION ACTUALLY OCCUR?
             05    13     0DA1          BEQL    10$                          ;IF EQL, THEN NO
       5E    08    C0     0DA3          ADDL    #8,SP                        ;RESTORE STACK
             34    11     0DA6          BRB     20$                          ;EXAMINE RESULT
                          0DA8     ;
                          0DA8     ; TRY AGAIN WITH UPCASED LOGICAL NAME STRING.
                          0DA8     ;
    51    04 AC    7D     0DA8     10$: MOVQ    P1(AP),R1                    ;CREATE LOGICAL NAME DESCRIPTOR
          F251'    30     0DAC          BSBW    DCL$UPCASE                   ;UPCASE THE LOGICAL NAME STRING
       7E    51    7D     0DAF          MOVQ    R1,-(SP)                     ;PUT THE LOGICAL NAME DESC ON THE STACK
       50    5E    D0     0DB2          MOVL    SP,R0                        ;SET ADDRESS OFF LOGICAL NAME DESCRIPTOR
    51    08 AE    9E     0DB5          MOVAB   8(SP),R1                     ;SET ADDRESS OF EQUIVALENCE NAME DESCRIPTOR
                          0DB9          $TRNLOG_S (R0),(R1),(R1)             ;TRANSLATE LOGICAL NAME
       5E    08    C0     0DCC          ADDL    #8,SP                        ;REMOVE THE LOGICAL NAME DESCRIPTOR
       51    8E    7D     0DCF          MOVQ    (SP)+,R1                     ;RETRIEVE EQUIVALENCE NAME PARAMETERS
          1B 50    E9     0DD2          BLBC    R0,90$                       ;IF LBC TRANSLATION FAILURE
    50  0000'8F    B1     0DD5          CMPW    #SS$_NOTRAN,R0               ;DID TRANSLATION ACTUALLY OCCUR?
             11    13     0DDA          BEQL    30$                          ;IF EQL, THEN NO
                          0DDC     ;
                          0DDC     ; STRIP ESCAPE SEQUENCE OFF EQUIVALENCE NAME STRING.
                          0DDC     ;
             51    D5     0DDC     20$: TSTL    R1                           ;ZERO LENGTH VALUE?
             0F    13     0DDE          BEQL    40$                          ;IF EQL YES
       62    1B    91     0DE0          CMPB    #27,(R2)                     ;FIRST CHARACTER ESCAPE?
             0A    12     0DE3          BNEQ    40$                          ;IF NEQ NO
       52    04    C0     0DE5          ADDL    #4,R2                        ;POINT PAST ESCAPE SEQUENCE
       51    04    C2     0DE8          SUBL    #4,R1                        ;REDUCE LENGTH OF EQUIVALENCE STRING
             02    18     0DEB          BGEQ    40$                          ;IF GEQ OKAY
             51    D4     0DED     30$: CLRL    R1                           ;SET TO RETURN NULL STRING
                   05     0DEF     40$: RSB                                  ;
```

```
                  0DF0  2998
                  0DF0  2999 ;
                  0DF0  3000 ; LOGICAL NAME TRANSLATION FAILURE
                  0DF0  3001 ;
                  0DF0  3002
      0825   31   0DF0  3003 90$:    BRW     P1_ERROR                ;REPORT ERROR AND DISPLAY P1
```

```
0DF3          .SBTTL  LOGICAL NAME TRANSLATION
0DF3    ;+
0DF3    ; F$TRNLNM - LOGICAL NAME TRANSLATION
0DF3    ;
0DF3    ; THIS LEXICAL FUNCTION OBTAINS THE TRANSLATION FOR A SPECIFIED LOGICAL NAME AND
0DF3    ; RETURN THE EQUIVALENCE NAME OR THE REQUESTED ATTRIBUTE.
0DF3    ;
0DF3    ;       F$TRNLNM (NAME, TABLE, INDEX, MODE, ATTR, ITEM)
0DF3    ;
0DF3    ; INPUTS:
0DF3    ;
0DF3    ;       P1(AP) = DESCRIPTOR OF LOGICAL NAME STRING.
0DF3    ;       P2(AP) = DESCRIPTOR OF NAME OF LOGICAL NAME TABLE TO SEARCH.
0DF3    ;       P3(AP) = INDEX OF THE DESIRED TRANSLATION.
0DF3    ;       P4(AP) = DESCRIPTOR OF ACCESS MODE OF TRANSLATION.
0DF3    ;       P5(AP) = DESCRIPTOR OF TYPE OF TRANSLATION.
0DF3    ;       P6(AP) = DESCRIPTOR OF REQUESTED ITEM.
0DF3    ;
0DF3    ; OUTPUTS:
0DF3    ;
0DF3    ;       R1 = LENGTH OF EQUIVALENCE STRING.
0DF3    ;       R2 = ADDRESS OF EQUIVALENCE STRING.
0DF3    ;-
0DF3
0DF3    F$TRNLNM:                                       ;LOGICAL NAME TRANSLATION
0DF3
0DF3    ;++
0DF3    ; Item list created on stack:
0DF3    ;
0DF3    ;                 -------------------------
0DF3    ;       48(R9)    : Translation attributes :
0DF3    ;                 -------------------------
0DF3    ;       44(R9)    :  Attribute bit number  :
0DF3    ;                 -------------------------
0DF3    ;       40(R9)    :      Access mode       :
0DF3    ;                 -------------------------
0DF3    ;                 : Table name descriptor  :
0DF3    ;                 :                        :
0DF3    ;       32(R9)    :                        :
0DF3    ;                 -------------------------
0DF3    ;       28(R9)    :   Translation index    :
0DF3    ;                 -------------------------
0DF3    ;       24(R9)    :    End of item list    :
0DF3    ;                 -------------------------
0DF3    ;       20(R9)    :        12(R9)          :
0DF3    ;                 -------------------------
0DF3    ;       16(R9)    :  Scratch buffer descr  :
0DF3    ;                 ------------
0DF3    ;       12(R9)    :lnm$_string!            :
0DF3    ;                 -------------------------
0DF3    ;        8(R9)    :           0            :
0DF3    ;                 -------------------------
0DF3    ;        4(R9)    :        28(R9)          :
0DF3    ;                 -------------------------
0DF3    ;         (R9)    : lnm$_index!      4     :
0DF3    ;                 -------------------------
0DF3    ;--
```

```
                                 0DF3
                                 0DF3      ;
                                 0DF3      ; Allocate and initialize the item list.
                                 0DF3      ;
              5E    34     C2    0DF3              SUBL      #52,SP                          ;ALLOCATE ITEM LIST ON STACK
              59    5E     D0    0DF6              MOVL      SP,R9                           ;SAVE THE ADDR OF THE LIST
              08    52     D1    0DF9              CMPL      R2,#8                           ;CHECK MIN SIZE OF RESULT BUFFER
                    6F     1F    0DFC              BLSSU     17$                             ;BRANCH IF TOO SMALL
      69    00010004 8F    D0    0DFE              MOVL      #LNM$_INDEX@16+4,(R9)           ;INIT THE ITEM LIST
         04 A9    1C A9    DE    0E05              MOVAL     28(R9),4(R9)                    ;
              08 A9        D4    0E0A              CLRL      8(R9)                           ;
           0C A9    52     7D    0E0D              MOVQ      R2,12(R9)                       ;
           0E A9    02     B0    0E11              MOVW      #LNM$_STRING,14(R9)             ;
         14 A9    0C A9    DE    0E15              MOVAL     12(R9),20(R9)                   ;
              18 A9        7C    0E1A              CLRQ      24(R9)                          ;
      51    0000227D'EF    9E    0E1D              MOVAB     TABDEF,R1                       ;
              50    81     9A    0E24              MOVZBL    (R1)+,R0                        ;
           20 A9    50     7D    0E27              MOVQ      R0,32(R9)                       ;
           28 A9    03     D0    0E2B              MOVL      #PSL$C_USER,40(R9)              ;
              2C A9        D4    0E2F              CLRL      44(R9)                          ;
   30 A9    02000000 8F    D0    0E32              MOVL      #LNM$M_CASE_BLIND,48(R9)        ;
                                 0E3A
                                 0E3A      ;
                                 0E3A      ; Set the translation index.
                                 0E3A      ;
              14 AC        D5    0E3A              TSTL      P3(AP)                          ;WAS INDEX SPECIFIED?
                    05     19    0E3D              BLSS      10$                             ;NO, THEN USE DEFAULT
         1C A9    14 AC    D0    0E3F              MOVL      P3(AP),28(R9)                   ;SAVE SPECIFIED INDEX
                                 0E44
                                 0E44      ;
                                 0E44      ; Set the item code.
                                 0E44      ;
              2C AC        D5    0E44      10$:    TSTL      P6(AP)                          ;WAS AN ITEM SPECIFIED?
                    36     19    0E47              BLSS      20$                             ;NO, THEN USE DEFAULT
      54    00001FA8'EF    9E    0E49              MOVAB     LOGICAL_TABLE,R4                ;GET ADDRESS OF ITEM TABLE
              55    64     9A    0E50      15$:    MOVZBL    (R4),R5                         ;GET LENGTH OF ITEM NAME
                    15     13    0E53              BEQL      16$                             ;BRANCH IF END OF TABLE
55    00    30 BC    2C AC     2D    0E55              CMPC5     P6(AP),@P6+4(AP),#0,R5,1(R4)    ;DOES ITEM NAME MATCH
                 01 A4           0E5C
                    10     13    0E5E              BEQL      18$                             ;BRANCH IF ITEM FOUND
      54    04 A445        9E    0E60              MOVAB     4(R4)[R5],R4                    ;SKIP TO NEXT TABLE ENTRY
                    E9     11    0E65              BRB       15$                             ;
                  0077     31    0E67      14$:    BRW       94$                             ;
                  0080     31    0E6A      16$:    BRW       96$                             ;
                  008F     31    0E6D      17$:    BRW       97$                             ;
      50    01 A445        9E    0E70      18$:    MOVAB     1(R4)[R5],R0                    ;GET THE ADDR OF THE ITEM CODE
           0E A9    60     B0    0E75              MOVW      (R0),14(R9)                     ;SET THE ITEM CODE
      2C A9    03 A445     9A    0E79              MOVZBL    3(R4)[R5],44(R9)                ;SAVE THE ATTR LONGWORD BIT NUM
                                 0E7F
                                 0E7F      ;
                                 0E7F      ; Set the access mode.
                                 0E7F      ;
              1C AC        D5    0E7F      20$:    TSTL      P4(AP)                          ;WAS AN ACCESS MODE SPECIFIED?
                    24     19    0E82              BLSS      30$                             ;NO, THEN USE DEFAULT
      54    0000204C'EF    9E    0E84              MOVAB     ACCESS_TABLE,R4                 ;GET ADDRESS OF ACCESS TABLE
              55    64     9A    0E8B      25$:    MOVZBL    (R4),R5                         ;GET LENGTH OF ACCESS MODE
                    D7     13    0E8E              BEQL      14$                             ;BRANCH IF END OF TABLE
```

```
 55   00   20 BC   1C AC   2D  0E90  3118           CMPC5   P4(AP),@P4+4(AP),#0,R5,1(R4)     ;DOES ACCESS MODE MATCH
                   01 A4       0E97
                      07   13  0E99  3119           BEQL    27$                              ;BRANCH IF ACCESS MODE FOUND
           54    02 A445   9E  0E9B  3120           MOVAB   2(R4)[R5],R4                     ;SKIP TO NEXT TABLE ENTRY
                      E9   11  0EA0  3121           BRB     25$                              ;
        28 A9    01 A445   9A  0EA2  3122 27$:      MOVZBL  1(R4)[R5],40(R9)                 ;SET THE ACCESS MODE
                               0EA8  3123
                               0EA8  3124 ;
                               0EA8  3125 ; Set the table name.
                               0EA8  3126 ;
                   0C AC   D5  0EA8  3127 30$:      TSTL    P2(AP)                           ;WAS TABLE SPECIFIED?
                      05   19  0EAB  3128           BLSS    40$                              ;NO, THEN USE DEFAULT
           20 A9   0C AC   7D  0EAD  3129           MOVQ    P2(AP),32(R9)                    ;SET SPECIFIED TABLE NAME
                               0EB2  3130
                               0EB2  3131 ;
                               0EB2  3132 ; Set the tranlation attribute.
                               0EB2  3133 ;
                   24 AC   D5  0EB2  3134 40$:      TSTL    P5(AP)                           ;WAS AN ATTR SPECIFIED?
                      27   19  0EB5  3135           BLSS    48$                              ;NO, THEN USE DEFAULT
        54    00002029'EF  9E  0EB7  3136           MOVAB   LOGATTR_TABLE,R4                 ;GET ADDRESS OF ATTR TABLE
              55      64   9A  0EBE  3137 45$:      MOVZBL  (R4),R5                          ;GET LENGTH OF ATTR NAME
                      24   13  0EC1  3138           BEQL    95$                              ;BRANCH IF END OF TABLE
 55   00   28 BC   24 AC   2D  0EC3  3139           CMPC5   P5(AP),@P5+4(AP),#0,R5,1(R4)     ;DOES ATTR NAME MATCH
                   01 A4       0ECA
                      07   13  0ECC  3140           BEQL    47$                              ;BRANCH IF ATTR FOUND
           54    05 A445   9E  0ECE  3141           MOVAB   5(R4)[R5],R4                     ;SKIP TO NEXT TABLE ENTRY
                      E9   11  0ED3  3142           BRB     45$                              ;
           50    01 A445   9E  0ED5  3143 47$:      MOVAB   1(R4)[R5],R0                     ;GET ADDR OF ATTIBUTE CODE
           30 A9      60   D0  0EDA  3144           MOVL    (R0),48(R9)                      ;SET THE ATTRIBUTE
                    0036   31  0EDE  3145 48$:      BRW     50$                              ;
                               0EE1  3146
                               0EE1  3147 ;
                               0EE1  3148 ; Signal all errors.
                               0EE1  3149 ;
           52      1C AC   9E  0EE1  3150 94$:      MOVAB   P4(AP),R2
                      0A   11  0EE5  3151           BRB     961$
           52      24 AC   9E  0EE7  3152 95$:      MOVAB   P5(AP),R2
                      04   11  0EEB  3153           BRB     961$
           52      2C AC   9E  0EED  3154 96$:      MOVAB   P6(AP),R2
                               0EF1  3155 961$:     STATUS  IVKEYW
           5E      34 A9   9E  0EF8  3156           MOVAB   52(R9),SP
                    073B   31  0EFC  3157           BRW     PN_ERROR
                               0EFF  3158
           5E      34 A9   9E  0EFF  3159 97$:      MOVAB   52(R9),SP
                    0706   31  0F03  3160           BRW     BUFOVF
                               0F06  3161
           50    0000'8F   B1  0F06  3162 90$:      CMPW    #SS$_BUFFEROVF,R0
                      F2   13  0F0B  3163           BEQL    97$
           5E      34 A9   9E  0F0D  3164           MOVAB   52(R9),SP
                    06FF   31  0F11  3165           BRW     ERROR
                               0F14  3166
                    0088   31  0F14  3167 51$:      BRW     80$
                               0F17  3168
                               0F17  3169 ;
                               0F17  3170 ; Perform the translation.
                               0F17  3171 ;
                               0F17  3172 50$:      $TRNLNM_S  ATTR=48(R9),-                 ;TRANSLATE THE LOGICAL NAME
```

```
                         0F17                      TABNAM=32(R9),-                 ;
                         0F17                      LOGNAM=P1(AP),-                 ;
                         0F17                      ACMODE=40(R9),-                 ;
                         0F17                      ITMLST=(R9)                     ;
     50  0000'8F    B1   0F2C           CMPW       #SS$_NOLOGNAM,R0                ;DID TRANSLATION ACTUALLY OCCUR?
              6C    13   0F31           BEQL       80$                             ;IF NEQ, THEN RETURN NULL STRING
           D0 50    E9   0F33           BLBC       R0,90$                          ;IF LBC, THEN ERROR
                         0F36
                         0F36    ;
                         0F36    ; Return requested information
                         0F36    ;
     02    0E A9    B1   0F36    60$:   CMPW       14(R9),#LNM$_STRING             ;WAS TRANSLATION REQUESTED?
              4B    13   0F3A           BEQL       70$                             ;YES, THEN GO RETURN IT
     04    0E A9    B1   0F3C           CMPW       14(R9),#LNM$_TABLE              ;WAS TABLE NAME REQUESTED?
              45    13   0F40           BEQL       70$                             ;YES, THEN GO RETURN IT
     07    0E A9    B1   0F42           CMPW       14(R9),#LNM$_MAX_INDEX          ;WAS INDEX REQUESTED?
              37    13   0F46           BEQL       69$                             ;YES, THEN GO RETURN IT
     05    0E A9    B1   0F48           CMPW       14(R9),#LNM$_LENGTH             ;WAS LENGTH REQUESTED?
              31    13   0F4C           BEQL       69$                             ;YES, THEN GO RETURN IT
     06    0E A9    B1   0F4E           CMPW       14(R9),#LNM$_ACMODE             ;WAS ACCESS MODE REQUESTED?
              11    13   0F52           BEQL       66$                             ;YES, THEN GO RETURN IT
                         0F54
                         0F54    ;
                         0F54    ; Return requested attribute.
                         0F54    ;
              51    D4   0F54           CLRL       R1                              ;ASSUME FALSE
  02 10 B9   2C A9  E1   0F56           BBC        44(R9),@16(R9),65$              ;BRANCH IF FALSE
              51    D6   0F5C           INCL       R1                              ;SET TRUE
     5E    34 A9    9E   0F5E    65$:   MOVAB      52(R9),SP                       ;RESTORE THE STACK
            081D    31   0F62           BRW        FORMAT_BOOLEAN                  ;RETURN BOOLEAN STRING
                         0F65
                         0F65    ;
                         0F65    ; Return access mode.
                         0F65    ;
  52   0000204C'EF  9E   0F65    66$:   MOVAB      ACCESS_TABLE,R2                 ;GET ADDRESS OF ACCESS TABLE
     50    10 B9    9A   0F6C           MOVZBL     @16(R9),R0                      ;GET ACCESS MODE CODE
              05    11   0F70           BRB        68$                             ;
     52   01 A241   9E   0F72    67$:   MOVAB      1(R2)[R1],R2                    ;SKIP TO NEXT TABLE ENTRY
     51       82    9A   0F77    68$:   MOVZBL     (R2)+,R1                        ;GET LENGTH OF ACCESS MODE
           F5 50    F4   0F7A           SOBGEQ     R0,67$                          ;
              25    11   0F7D           BRB        85$                             ;RETURN THE SPECIFIED VALUE
                         0F7F
                         0F7F    ;
                         0F7F    ; Return max index or length.
                         0F7F    ;
     51    10 B9    D0   0F7F    69$:   MOVL       @16(R9),R1                      ;GET THE INDEX
              52    D4   0F83           CLRL       R2                              ;
              1D    11   0F85           BRB        85$                             ;
                         0F87
                         0F87    ;
                         0F87    ; Strip escape sequence off and return equivalence name string.
                         0F87    ;
           0E A9    B4   0F87    70$:   CLRW       14(R9)                          ;REMOVE THE ITEM CODE
     51    0C A9    7D   0F8A           MOVQ       12(R9),R1                       ;GET RESULT DESCRIPTOR
              51    D5   0F8E           TSTL       R1                              ;ZERO LENGTH VALUE?
              12    13   0F90           BEQL       85$                             ;IF EQL YES
     62       1B    91   0F92           CMPB       #27,(R2)                        ;FIRST CHARACTER ESCAPE?
```

```
         0D  12  0F95  3230          BNEQ    85$                     ;IF NEQ NO
     52  04  C0  0F97  3231          ADDL    #4,R2                   ;POINT PAST ESCAPE SEQUENCE
     51  04  C2  0F9A  3232          SUBL    #4,R1                   ;REDUCE LENGTH OF EQUIVALENCE STRING
         05  18  0F9D  3233          BGEQ    85$                     ;IF GEQ OKAY
         51  D4  0F9F  3234 80$:     CLRL    R1                      ;SET TO RETURN NULL STRING
     52  5E  D0  0FA1  3235          MOVL    SP,R2                   ;
5E   34  A9  9E  0FA4  3236 85$:     MOVAB   52(R9),SP               ;RESTORE THE STACK
         05      0FA8  3237          RSB                             ;
```

```
                    0FA9         .SBTTL  GET MESSAGE TEXT
                    0FA9  ;+
                    0FA9  ; F$MESSAGE - GET MESSAGE TEXT
                    0FA9  ;
                    0FA9  ; THIS LEXICAL FUNCTION OBTAINS THE MESSAGE TEXT ASSOCIATED WITH A MESSAGE CODE.
                    0FA9  ;
                    0FA9  ;        F$MESSAGE(CODE)
                    0FA9  ;
                    0FA9  ; INPUTS:
                    0FA9  ;
                    0FA9  ;        P1(AP) = MESSAGE CODE
                    0FA9  ;
                    0FA9  ; OUTPUTS:
                    0FA9  ;
                    0FA9  ;        R1 = LENGTH OF MESSAGE TEXT STRING.
                    0FA9  ;        R2 = ADDRESS OF MESSAGE TEXT STRING.
                    0FA9  ;-
                    0FA9
                    0FA9  F$MESSAGE:                                    ;GET MESSAGE TEXT
 7E     52     7D   0FA9         MOVQ    R2,-(SP)                       ;SET MESSAGE BUFFER DESCRIPTOR
 50     5E     D0   0FAC         MOVL    SP,R0                          ;SET ADDRESS OF MESSAGE BUFFER DESCRIPTOR
                    0FAF         $GETMSG_S P1(AP),(R0),(R0),#15         ;GET MESSAGE TEXT
 51     8E     7D   0FC1         MOVQ    (SP)+,R1                       ;RETRIEVE MESSAGE TEXT PARAMETERS
               05   0FC4         RSB
```

```
                          0FC5              .SBTTL  RETURN JOB MODE
                          0FC5      ;+
                          0FC5      ; F$MODE - RETURN JOB MODE
                          0FC5      ;
                          0FC5      ; THIS LEXICAL FUNCTION RETURNS A STRING THAT IS EITHER "INTERACTIVE" OR "BATCH"
                          0FC5      ; DEPENDENT ON THE JOB MODE.
                          0FC5      ;
                          0FC5      ;       F$MODE()
                          0FC5      ;
                          0FC5      ; INPUTS:
                          0FC5      ;
                          0FC5      ;       NONE.
                          0FC5      ;
                          0FC5      ; OUTPUTS:
                          0FC5      ;
                          0FC5      ;       R1 = LENGTH OF JOB MODE DESCRIPTION STRING.
                          0FC5      ;       R2 = ADDRESS OF JOB MODE DESCRIPTION STRING.
                          0FC5      ;-
                          0FC5
                          0FC5      F$MODE:                                         ;RETURN JOB MODE
               7E    7C   0FC5              CLRQ    -(SP)                           ;RESERVE IOSB
               7E    7C   0FC7              CLRQ    -(SP)                           ;MARK END OF LIST, NO RETLEN
            F8 AE    9F   0FC9              PUSHAB  -2*4(SP)                        ;SET BUFFER ADDRESS
      03220004 8F    DD   0FCC              PUSHL   #JPI$_MODE@16+4                 ;REQUEST MODE, SET BUFFER LENGTH
         50    5E    D0   0FD2              MOVL    SP,R0                           ;SET ADDRESS OF TIME LIST
                          0FD5              $GETJPIW_S ITMLST=(R0),-                ;GET MODE LONGWORD
                          0FD5                      IOSB=16(R0),-                   ;
                          0FD5                      EFN=#EXE$C_SYSEFN               ;
            27 50    E9   0FED              BLBC    R0,50$                          ;IF LBC SERVICE FAILURE
      50    10 AE    3C   0FF0              MOVZWL  16(SP),R0                       ;GET STATUS FROM IOSB
            20 50    E9   0FF4              BLBC    R0,50$                          ;IF LBC SERVICE FAILURE
                          0FF7
               53    D4   0FF7              CLRL    R3                              ;SET LOOP INDEX
52    000021F3'EF    9E   0FF9              MOVAB   MODE_TABLE,R2                   ;GET ADDRESS OF MODE TABLE
         51    82    9A   1000      25$:    MOVZBL  (R2)+,R1                        ;GET LENGTH OF MODE
               0E    13   1003              BEQL    40$                             ;BRANCH IF END OF TABLE
         53    6E    D1   1005              CMPL    (SP),R3                         ;IS THIS THE ONE WE WANT?
               09    13   1008              BEQL    40$                             ;YES, THEN RETURN THE STRING
      52  01 A142    9E   100A              MOVAB   1(R1)[R2],R2                    ;SKIP TO NEXT TABLE ENTRY
               53    D6   100F              INCL    R3                              ;INCR LOOP INDEX
               ED    11   1011              BRB     25$                             ;
                          1013
         5E    18    C0   1013      40$:    ADDL    #6*4,SP                         ;DEALLOCATE GETJPI ITEM LIST AND IOSB
                     05   1016              RSB
                          1017
         5E    18    C0   1017      50$:    ADDL    #6*4,SP                         ;CLEAN STACK
             05F6    31   101A              BRW     ERROR                           ;REPORT ERROR IN FUNCTION
```

```
                              101D       .SBTTL  PARSE A FILE SPECIFICATION
                              101D  ;+
                              101D  ; F$PARSE - PARSE A FILE SPECIFICATION
                              101D  ;
                              101D  ; THIS FUNCTION PERFORMS A $PARSE SYSTEM SERVICE RETURNING THE
                              101D  ; ENTIRE (OR OPTIONALLY ONLY PARTS OF THE) EXPANDED FILESPEC.
                              101D  ;
                              101D  ; INPUTS:
                              101D  ;
                              101D  ;        P1(AP) = PRIMARY FILESPEC
                              101D  ;        P2(AP) = DEFAULT FILESPEC
                              101D  ;        P3(AP) = RELATED FILESPEC (FOR TEMPORARY STICKYNESS)
                              101D  ;        P4(AP) = NAME OF A FILESPEC FIELD
                              101D  ;                 IF NOT SPECIFIED, THE ENTIRE FILESPEC IS RETURNED.
                              101D  ;        P5(AP) = TYPE OF PARSE (ITEM LIST)
                              101D  ;
                              101D  ;        P2,P3,P4, AND P5 ARE ALL OPTIONAL (INDICATED BY A -1 IN THE FIRST LONGWORD)
                              101D  ;
                              101D  ; OUTPUTS:
                              101D  ;
                              101D  ;        R1/R2 = DESCRIPTOR OF EXPANDED FILE SPECIFICATION
                              101D  ;-
                              101D
                              101D  F$PARSE:
               7E   52   7D   101D          MOVQ    R2,-(SP)                        ;PUSH DESCRIPTOR OF SCRATCH BUFFER
               59   5E   D0   1020          MOVL    SP,R9                           ;SAVE ADDRESS OF DESCRIPTOR
         5E  FDF1   CE   9E   1023          MOVAB   -NAM$C_BLN-FAB$C_BLN-NAM$C_MAXRSS-NAM$C_BLN(SP),SP ;ALLOCATE STORAGE
6E  020F 8F  00   6E   00   2C   1028          MOVC5   #0,(SP),#0,#FAB$C_BLN+NAM$C_BLN+NAM$C_MAXRSS+NAM$C_BLN,(SP) ;ZERO ST
               57   5E   D0   1030          MOVL    SP,R7                           ;GET ADDRESS OF NAM
          56   60   A7   9E   1033          MOVAB   NAM$C_BLN(R7),R6                ;GET ADDRESS OF FAB
                              1037          ASSUME  FAB$B_BLN EQ FAB$B_BID+1
         66  5003   8F   B0   1037          MOVW    #FAB$C_BID+<FAB$C_BLN@8>,FAB$B_BID(R6) ;INITIALIZE FAB
                              103C          ASSUME  NAM$B_BLN EQ NAM$B_BID+1
         67  6002   8F   B0   103C          MOVW    #NAM$C_BID+<NAM$C_BLN@8>,NAM$B_BID(R7) ;INITIALIZE NAM
                              1041  ;
                              1041  ;        VALIDATE PARSE TYPE PARAMETER LIST, IF ANY
                              1041  ;
               24   AC   D5   1041          TSTL    P5(AP)                          ;PARSE TYPE SPECIFIED?
                    67   19   1044          BLSS    18$                             ;IF NOT, SKIP BIT SETTING
                              1046
               7E   58   D0   1046          MOVL    R8,-(SP)                        ;SAVE R8,R10,R11
               7E   5A   7D   1049          MOVQ    R10,-(SP)                       ;
          54   24   AC   7D   104C          MOVQ    P5(AP),R4                       ;R4/R5 = ITEM LIST DESCRIPTOR
                    58   94   1050          CLRB    R8                              ;ASSUME ONLY 1 ITEM IN LIST
                              1052
               52   54   7D   1052  5$:     MOVQ    R4,R2                           ;SETUP FOR TABLE CHECK IN CASE 1 ITEM
         65    54   2C   3A   1055          LOCC    #^A/,/,R4,(R5)                  ;SEARCH ITEM LIST FOR A COMMA
                    0D   13   1059          BEQL    6$                              ;CHECK TABLE NOW IF ONLY 1 ITEM IN LIST
                              105B
                    58   96   105B          INCB    R8                              ;INDICATE AT LEAST ONE MORE ITEM IN LIST
         52    54   50   C3   105D          SUBL3   R0,R4,R2                        ;CALCULATE LENGTH OF THIS ITEM
               54   70   9E   1061          MOVAB   -(R0),R4                        ;STRIP CURRENT ITEM FROM LIST
          55   01   A1   9E   1064          MOVAB   1(R1),R5                        ;
                              1068
    5A   00001669'EF     9E   1068  6$:     MOVAB   PARSE_TYPES,R10                 ;R10 = ADDR. OF PARSE TYPES TABLE
               5B   6A   9A   106F  7$:     MOVZBL  (R10),R11                       ;R11 = LENGTH OF THIS PARSE TYPE
                    16   13   1072          BEQL    10$                             ;HAVE INVALID KEY WORD IF AT TABLE END
```

```
                            1074
            7E  52     7D   1074          MOVQ    R2,-(SP)                    ;SAVE ITEM LENGTH AND ADDR.
       00   63  52     2D   1077          CMPC5   R2,(R3),#0,-                ;CHECK CURR. ITEM AGAINST TAB. ENTRY
          01 AA 5B          107B                  R11,1(R10)
                1A     13   107E          BEQL    12$                         ;IF MATCH, SET BIT IN NAM$B_NOP FLAG
                            1080
            52  8E     7D   1080          MOVQ    (SP)+,R2                    ;RESTORE ITEM LENG. AND ADDR.
       5A  02 AA4B     9E   1083          MOVAB   2(R10)[R11],R10             ;R10 = ADDR. OF NEXT TABLE ENTRY
                E5     11   1088          BRB     7$                          ;CHECK ITEM AGAINST NEXT TABLE ENTRY
                            108A
            5A  8E     7D   108A  10$:    MOVQ    (SP)+,R10                   ;RESTORE WORK REGISTERS
            58  8E     D0   108D          MOVL    (SP)+,R8                    ;
                            1090          STATUS  IVKEYW                      ;INVALID KEYWORD
              0596     31   1097          BRW     P5_ERROR                    ;GOTO ERROR HANDLER
                            109A
            52  8E     7D   109A  12$:    MOVQ    (SP)+,R2                    ;HAVE MATCH. FIRST RESTORE ITEM REG.
    08 A7  01 AA4B     88   109D          BISB    1(R10)[R11],NAM$B_NOP(R7)         ;SET CORRESPONDING BIT
         AB 58  00     E4   10A3          BBSC    #0,R8,5$                    ;CHECK NEXT ITEM IN LIST(IF ANY LEFT)
                            10A7
            5A  8E     7D   10A7          MOVQ    (SP)+,R10                   ;RESTORE REGISTERS
            58  8E     D0   10AA          MOVL    (SP)+,R8                    ;
                            10AD  ;
                            10AD  ;       VALIDATE NAME OF FILESPEC FIELD PARAMETER.
                            10AD  ;
             1C AC     D5   10AD  18$:    TSTL    P4(AP)                      ;FIELD SPECIFIED?
                2E     19   10B0          BLSS    19$                         ;IF NOT, RETURN ENTIRE EXPANDED STRING
    54     00001634'EF 9E   10B2          MOVAB   PARSE_FIELDS,R4             ;GET ADDRESS OF TABLE
            55  64     9A   10B9  300$:   MOVZBL  (R4),R5                     ;GET LENGTH OF ITEM NAME
                12     13   10BC          BEQL    350$                        ;BRANCH IF END OF TABLE
 00    20 BC  1C AC    2D   10BE          CMPC5   P4(AP),@P4+4(AP),#0,-       ;DOES ITEM NAME MATCH?
          01 A4 55          10C4                  R5,1(R4)                    ;
                11     13   10C7          BEQL    380$                        ;BRANCH IF ITEM FOUND
    54     03 A445     9E   10C9          MOVAB   3(R4)[R5],R4                ;SKIP TO NEXT ENTRY IN TABLE
                E9     11   10CE          BRB     300$                        ;LOOP UNTIL FOUND
                            10D0  350$:   STATUS  IVKEYW                      ;UNDEFINED ITEM NAME
              0550     31   10D7          BRW     P4_ERROR
            54  83     9A   10DA  380$:   MOVZBL  (R3)+,R4                    ;GET OFFSET TO FIELD LENGTH
            55  83     9A   10DD          MOVZBL  (R3)+,R5                    ;GET OFFSET TO FIELD ADDRESS
                            10E0  ;
                            10E0  ;       PARSE FILE SPEC.
                            10E0  ;
        28 A6   57     D0   10E0  19$:    MOVL    R7,FAB$L_NAM(R6)            ;SET POINTER TO NAM BLOCK
                            10E4          SETBIT  FAB$V_PPF,FAB$L_FOP(R6)     ;USE PROCESS I/O SEGMENT (SINCE THERE
                            10E9                                              ;   MAY NOT BE ANY IMAGE I/O SEGMENT)
    34 A6   04 AC      90   10E9          MOVB    P1(AP),FAB$B_FNS(R6)        ;STORE PRIMARY NAME STRING INTO FAB
    2C A6   08 AC      D0   10EE          MOVL    P1+4(AP),FAB$L_FNA(R6)
            0C AC      D5   10F3          TSTL    P2(AP)                      ;P2 SPECIFIED?
                0A     19   10F6          BLSS    20$                         ;BRANCH IF NOT
    35 A6   0C AC      90   10F8          MOVB    P2(AP),FAB$B_DNS(R6)        ;STORE DEFAULT NAME STRING INTO FAB
    30 A6   10 AC      D0   10FD          MOVL    P2+4(AP),FAB$L_DNA(R6)
            14 AC      D5   1102  20$:    TSTL    P3(AP)                      ;P3 SPECIFIED?
                18     19   1105          BLSS    25$                         ;BRANCH IF NOT
    50     014F C6     9E   1107          MOVAB   FAB$C_BLN+NAM$C_MAXRSS(R6),R0    ;ADDRESS OF RELATED NAM
                            110C          ASSUME  NAM$B_BLN EQ NAM$B_BID+1
    60     6002 8F     B0   110C          MOVW    #NAM$C_BID+<NAM$C_BLN@8>,NAM$B_BID(R0) ;INITIALIZE RELATED NAM
    03 A0   14 AC      90   1111          MOVB    P3(AP),NAM$B_RSL(R0)        ;INITIALIZE RELATED FILESPEC
    04 A0   18 AC      D0   1116          MOVL    P3+4(AP),NAM$L_RSA(R0)
```

```
         10 A7    50      D0  111B  3426         MOVL    R0,NAM$L_RLF(R7)                ;POINT TO RELATED NAM BLOCK
   0A A7       FF 8F      90  111F  3427 25$:    MOVB    #NAM$C_MAXRSS,NAM$B_ESS(R7)      ;SET SIZE OF EXPANDED BUFFER
   0C A7       50 A6      9E  1124  3428         MOVAB   FAB$C_BLN(R6),NAM$L_ESA(R7)      ;SET ADDRESS OF RESULT BUFFER
                              1129  3429         $PARSE  FAB=(R6)                        ;SETUP EXPANDED NAME STRING, NAM FIELDS
                              1132  3430                                                 ;(SUCH AS FID,DID,DVI,ESL)
            03 50         E8  1132  3431         BLBS    R0,28$                          ;BRANCH IF OK
            0B A7         94  1135  3432         CLRB    NAM$B_ESL(R7)                   ;RETURN NULL ON ANY PARSE ERROR
         51    0B         9A  1138  3433 28$:    MOVZBL  #NAM$B_ESL,R1                   ;ASSUME USING ENTIRE EXPANDED STRING
         52    0C         9A  113B  3434         MOVZBL  #NAM$L_ESA,R2
            1C AC         D5  113E  3435         TSTL    P4(AP)                          ;FIELD SPECIFIED?
               03         19  1141  3436         BLSS    40$                             ;IF NOT, RETURN ENTIRE EXPANDED STRING
         51    54         7D  1143  3437         MOVQ    R4,R1                           ;GET OFFSET TO FIELD LENGTH, ADDRESS
      51    6741          9A  1146  3438 40$:    MOVZBL  (R7)[R1],R1                     ;GET LENGTH OF EXPANDED STRING
         52    57         C0  114A  3439         ADDL    R7,R2                           ;GET ADDRESS OF EXPANDED STRING ADDRESS
         69    51         D1  114D  3440         CMPL    R1,(R9)                         ;ENOUGH ROOM FOR STRING?
               2C         1A  1150  3441         BGTRU   60$                             ;BRANCH IF NOT
               51         DD  1152  3442         PUSHL   R1                              ;SAVE REGISTERS
 04 B9    00 B2    51     28  1154  3443         MOVC    R1,@(R2),@4(R9)                 ;COPY INTO SCRATCH BUFFER
            53 8ED0           115A  3444         POPL    R3                              ;SAVE LENGTH OF EXPANDED STRING
            30 A7         D5  115D  3445         TSTL    NAM$L_WCC(R7)                   ;DID RMS SAVE ANY INTERNAL CONTEXT?
               10         13  1160  3446         BEQL    50$                             ;BRANCH IF NOT
                              1162  3447         ASSUME  FAB$B_DNS EQ FAB$B_FNS+1
            34 A6         B4  1162  3448         CLRW    FAB$B_FNS(R6)                   ;SET PRIMARY/DEFAULT FILESPECS TO NULL
                              1165  3449         SETBIT  NAM$V_SYNCHK,NAM$B_NOP(R7)      ;SET SYNTAX ONLY CHECK FLAG
                              1169  3450         $PARSE  FAB=(R6)                        ;TERMINATE WILDCARD SEQUENCE TO GET RMS
                              1172  3451                                                 ;TO DEALLOCATE INTERNAL CONTEXT STORAGE
         51    53         D0  1172  3452 50$:    MOVL    R3,R1                           ;GET LENGTH OF RESULTANT STRING
      52    04 A9         D0  1175  3453         MOVL    4(R9),R2                        ;SET ADDRESS OF STRING
      5E    08 A9         9E  1179  3454         MOVAB   8(R9),SP                        ;DEALLOCATE STORAGE
                          05  117D  3455         RSB
                              117E  3456
            048B          31  117E  3457 60$:    BRW     BUFOVF                          ;REPORT BUFFER OVERFLOW
```

```
                        1181              .SBTTL  GET NEXT PROCESS ID
                        1181       ;+
                        1181       ; F$PID - GET NEXT PROCESS ID
                        1181       ;
                        1181       ; THIS FUNCTION PERFORMS A WILDCARD GETJPI FUNCTION TO RETURN THE
                        1181       ; NEXT PROCESS PID IN SEQUENCE, GIVEN THE WILDCARD CONTEXT.
                        1181       ;
                        1181       ;      F$PID(CONTEXT_SYMBOL)
                        1181       ;
                        1181       ; INPUTS:
                        1181       ;
                        1181       ;      P1(AP) = DESCRIPTOR OF SYMBOL NAME
                        1181       ;
                        1181       ; OUTPUTS:
                        1181       ;
                        1181       ;      R1/R2 = DESCRIPTOR OF PID CONVERTED TO AN ASCII STRING
                        1181       ;-
                        1181
                        1181       F$PID:
            7E    7C    1181              CLRQ    -(SP)                      ;RESERVE IOSB
            7E    7C    1183              CLRQ    -(SP)                      ;CLEAR LAST ITEM AND RETLEN ADDRESS
         F8 AE    9F    1185              PUSHAB  -8(SP)                     ;ADDRESS TO RECEIVE PID
   03190004 8F    DD    1188              PUSHL   #JPI$_PID@16+4             ;SET ITEM CODE AND BUFLEN
         7E   01  CE    118E              MNEGL   #1,-(SP)                   ;ASSUME CONTEXT LONGWORD OF -1 (BEGIN)
     51    04 AC  7D    1191              MOVQ    P1(AP),R1                  ;GET DESCRIPTOR OF SYMBOL NAME
         EE68'    30    1195              BSBW    DCL$SEARCH                 ;FIND SYMBOL IN SYMBOL TABLE
         0B 50    E9    1198              BLBC    R0,10$                     ;IF NOT FOUND, SEARCH FROM BEGINNING
            52    D5    119B              TSTL    R2                         ;STRING VALUE?
            07    12    119D              BNEQ    10$                        ;IF SO, START SEARCH AT BEGINNING
            51    D5    119F              TSTL    R1                         ;TOP BIT MUST BE SET FOR VALID PID CTX
            03    18    11A1              BGEQ    10$                        ;IF NOT OK, START SEARCH AT BEGINNING
         6E 51    D0    11A3              MOVL    R1,(SP)                    ;STORE CONTEXT LONGWORD ON STACK
         50 5E    D0    11A6       10$:   MOVL    SP,R0                      ;GET ADDRESS OF CONTEXT/ITEM LIST
                        11A9              $GETJPIW_S PIDADR=(R0),-           ;GET NEXT PID IN SEQUENCE
                        11A9                      ITMLST=4(R0),-             ;
                        11A9                      IOSB=20(R0),-              ;
                        11A9                      EFN=#EXE$C_SYSEFN
         07 50    E9    11C2              BLBC    R0,15$                     ;BRANCH IF ERROR
     50    14 AE  3C    11C5              MOVZWL  20(SP),R0                  ;FETCH STATUS FROM IOSB
         10 50    E8    11C9              BLBS    R0,20$                     ;BRANCH IF NO ERROR
   0000'8F   50   B1    11CC       15$:   CMPW    R0,#SS$_NOPRIV             ;NO PRIVILEGE TO GET INFO?
            D3    13    11D1              BEQL    10$                        ;IF SO, TRY NEXT PROCESS
   0000'8F   50   B1    11D3              CMPW    R0,#SS$_NOMOREPROC         ;NO MORE PROCESSES?
            30    12    11D8              BNEQ    80$                        ;IF NOT, REPORT ERROR
            6E    7C    11DA              CLRQ    (SP)                       ;RETURN NULL PID ON THIS INVOCATION
         51 8ED0        11DC       20$:   POPL    R1                         ;GET NEW CONTEXT LONGWORD
     53    04 AC  7D    11DF              MOVQ    P1(AP),R3                  ;SET DESCRIPTOR OF SYMBOL NAME
     55    38 AB  9E    11E3              MOVAB   PRC_Q_LOCAL(R11),R5        ;SET ADDRESS OF SYMBOL TABLE LISTHEAD
         50   02  D0    11E7              MOVL    #SYM_K_BINARY,R0           ;SET TYPE IS BINARY LONGWORD
                        11EA              DISABLE                            ;DISABLE CONTROL/Y AST'S
         EE0D'    30    11F0              BSBW    DCL$ALLOCSYM               ;REPLACE VALUE OF SYMBOL
                        11F3              ENABLE                             ;RE-ENABLE CONTROL/Y AST'S
         50 8ED0        11F5              POPL    R0                         ;GET PID OF NEXT PROCESS
         5E   14  C0    11F8              ADDL    #5*4,SP                    ;AND CLEAN STACK
         52   5E  D0    11FB              MOVL    SP,R2                      ;MARK VALUE IS A STRING
         51   50  D0    11FE              MOVL    R0,R1                      ;END OF WILDCARD SEARCH?
            06    13    1201              BEQL    30$                        ;IF SO, RETURN WITH A NULL STRING
```

```
        00000000'EF   16  1203   3516         JSB     DCL$CBTA_HEX            ;CONVERT PID TO STRING AND EXIT
                      05  1209   3517 30$:    RSB
                          120A   3518
           5E    1C   C0  120A   3519 80$:    ADDL    #7*4,SP                 ;CLEAN STACK
               0408   31  120D   3520         BRW     P1_ERROR                ;REPORT ERROR AND DISPLAY P1
```

```
                           1210          .SBTTL  CHECK IF SPECIFIED PRIVS ARE SET OR CLEAR
                           1210  ;+
                           1210  ; F$PRIVILEGE - CHECK IF SPECIFIED PRIVILEGES ARE SET OR CLEAR
                           1210  ;
                           1210  ; THIS LEXICAL FUNCTION CAN BE USED TO DETERMINE WHETHER THE CURRENT PROCESS
                           1210  ; HAS A GIVEN SET OF PRIVILEGES ON OR OFF, DEPENDING ON THE KEYWORDS GIVEN.
                           1210  ; THE ARGUMENT IS A COMMA SEPARATED LIST OF PRIVILEGE KEYWORDS ABOUT WHICH
                           1210  ; THE CALLER IS INTERESTED.  IF THE PREFIX "NO" APPEARS ON THE KEYWORD, THEN
                           1210  ; THE FUNCTION CHECKS IF THE PRIVILEGE IS OFF.  OTHERWISE, THE FUNCTION CHECKS
                           1210  ; THAT THE PRIVILEGE IS ON.  THE FINAL VALUE OF THE FUNCTION IS THE "AND" OF
                           1210  ; ALL OF THE PRIVILEGE CHECKS - THAT IS, IT RETURNS TRUE IF ALL THE PRIVILEGES
                           1210  ; ARE SET AS SPECIFIED.
                           1210  ;
                           1210  ;       F$PRIVILEGE(PRVSTATES)
                           1210  ;
                           1210  ; INPUTS:
                           1210  ;
                           1210  ;       P1(AP) = DESCRIPTOR OF A COMMA SEPARATED LIST OF PRIVILEGE KEYWORDS.
                           1210  ;
                           1210  ; OUTPUTS:
                           1210  ;
                           1210  ;       R1/R2 = BOOLEAN TRUE OR FALSE STRING
                           1210  ;-
                           1210
                           1210  F$PRIVILEGE:
         56    53   D0     1210          MOVL    R3,R6                   ;GET ADDRESS OF SCRATCH QUADWORD
                           1213          $SETPRV_S PRVPRV=(R6),-         ;GET CURRENT PERMANENT PRIVILEGES
                           1213                  PRMFLG=#1               ;
      54   04 AC    7D     1222          MOVQ    P1(AP),R4               ;GET DESCRIPTOR OF COMMA SEPARATED LIST
         52    54   7D     1226  10$:    MOVQ    R4,R2                   ;ASSUME NO COMMA IN LIST
      65   54  2C   3A     1229          LOCC    #^A',',R4,(R5)          ;LOCATE END OF NEXT KEYWORD
               04   13     122D          BEQL    20$                     ;BRANCH IF NOT FOUND, USE ENTIRE STRING
      52   54  50   C3     122F          SUBL3   R0,R4,R2                ;ISOLATE NEXT KEYWORD
         54    70   9E     1233  20$:    MOVAB   -(R0),R4                ;CHOP KEYWORD OFF STRING
      55   01 A1    9E     1236          MOVAB   1(R1),R5
         7E    52   7D     123A          MOVQ    R2,-(SP)                ;PUSH DESCRIPTOR OF KEYWORD
               7E   DF     123D          PUSHAL  -(SP)                   ;PUSH ADDRESS OF WORD TO GET BIT NUMBER
         08    AE   9F     123F          PUSHAB  8(SP)                   ;PUSH ADDRESS OF DESCRIPTOR
  00000000'GF  02   FB     1242          CALLS   #2,G^PRV$PRIVBIT        ;LOOKUP KEYWORD AND GET PRIV. BIT #
               0E   BA     1249          POPR    #^M<R1,R2,R3>           ;CLEANUP STACK
         1F    50   E9     124B          BLBC    R0,70$                  ;IS ERROR, SIGNAL IT
    4F4E 8F    63   B1     124E          CMPW    (R3),#^A'NO'            ;IS KEYWORD NEGATED?
               06   13     1253          BEQL    30$                     ;IF SO, CHECK IF PRIVILEGE OFF
      0F  66   51   E1     1255          BBC     R1,(R6),50$             ;BRANCH IF PRIVILEGE NOT ON
               04   11     1259          BRB     40$
      09  66   51   E0     125B  30$:    BBS     R1,(R6),50$             ;BRANCH IF PRIVILEGE NOT OFF
               54   D5     125F  40$:    TSTL    R4                      ;ANY MORE KEYWORDS LEFT?
               C3   14     1261          BGTR    10$                     ;IF NOT, LOOP UNTIL DONE
         51    01   D0     1263          MOVL    #1,R1                   ;RETURN SUCCESS
               02   11     1266          BRB     60$
                           1268
               51   D4     1268  50$:    CLRL    R1                      ;RETURN FAILURE
             0515   31     126A  60$:    BRW     FORMAT_BOOLEAN          ;RETURN "TRUE" OR "FALSE"
                           126D
 50   00000000'8F   D1     126D  70$:    CMPL    #PRV$_NOTUNQ,R0         ;AMBIGUOUS KEYWORD?
               09   12     1274          BNEQ    75$                     ;IF NOT, EXIT WITH INVALID KEYWORD
                           1276          STATUS  ABKEYW                  ;SET AMBIGUOUS KEYWORD ERROR
```

```
      07   11  127D  3579          BRB     80$
               127F  3580 75$:     STATUS  IVKEYW                  ;SET INVALID KEYWORD
    038F   31  1286  3581 80$:     BRW     P1_ERROR                ;REPORT ERROR
```

```
                          1289          .SBTTL  RETURN PROCESS NAME
                          1289     ;+
                          1289     ; F$PROCESS - RETURN PROCESS NAME
                          1289     ;
                          1289     ; THIS LEXICAL FUNCTION OBTAINS THE CURRENT PROCESS NAME AND RETURN A DESCRIPTOR
                          1289     ; FOR IT.
                          1289     ;
                          1289     ;        F$PROCESS()
                          1289     ;
                          1289     ; INPUTS:
                          1289     ;
                          1289     ;        R2 = LENGTH OF SCRATCH BUFFER.
                          1289     ;        R3 = ADDRESS OF SCRATCH BUFFER.
                          1289     ;
                          1289     ; OUTPUTS:
                          1289     ;
                          1289     ;        R1 = LENGTH OF PROCESS NAME.
                          1289     ;        R2 = ADDRESS OF PROCESS NAME.
                          1289     ;-
                          1289
                          1289     F$PROCESS:                                      ;RETURN PROCESS NAME
           52   0F   D1   1289          CMPL    #15,R2                          ;BUFFER BIG ENOUGH TO HOLD PROCESS NAME?
                3F   1A   128C          BGTRU   20$                             ;BRANCH IF NOT
                7E   7C   128E          CLRQ    -(SP)                           ;RESERVE IOSB
                7E   D4   1290          CLRL    -(SP)                           ;CLEAR ITEM ID OF LAST ITEM
           F4   AE   9F   1292          PUSHAB  -12(SP)                         ;PUSH ADDRESS TO STORE LENGTH
           7E   52   7D   1295          MOVQ    R2,-(SP)                        ;PUSH OUTPUT BUFFER DESCRIPTOR
   02 AE  031C  8F   B0   1298          MOVW    #JPI$_PRCNAM,2(SP)              ;SET PROCESS NAME IDENTIFICATION
           50   5E   D0   129E          MOVL    SP,R0                           ;SET ADDRESS OF ITEM LIST
                          12A1          $GETJPIW_S ITMLST=(R0),-                ;GET PROCESS NAME
                          12A1                  IOSB=16(R0),-                   ;
                          12A1                  EFN=#EXE$C_SYSEFN               ;
           04   50   E9   12B9          BLBC    R0,10$                          ;IF LBC SERVICE FAILURE
      50   10   AE   3C   12BC          MOVZWL  16(SP),R0                       ;GET STATUS FROM IOSB
           51   8E   7D   12C0     10$: MOVQ    (SP)+,R1                        ;RETRIEVE PROCESS NAME PARAMETERS
           51   51   3C   12C3          MOVZWL  R1,R1                           ;CLEAR ITEM ID
           5E   10   C0   12C6          ADDL    #4*4,SP                         ;CLEAN OFF STACK
           04   50   E9   12C9          BLBC    R0,30$                          ;BRANCH IF ERROR
                     05   12CC          RSB                                     ;
              033C   31   12CD     20$: BRW     BUFOVF                          ;REPORT BUFFER OVERFLOW
              0340   31   12D0     30$: BRW     ERROR                           ;REPORT ERROR IN R0
```

```
                        12D3         .SBTTL  GET NEXT WILDCARD FILESPEC
                        12D3  ;+
                        12D3  ; F$SEARCH - GET NEXT WILDCARD FILESPEC
                        12D3  ;
                        12D3  ; THIS FUNCTION INVOKES THE $SEARCH SYSTEM SERVICE TO RETURN THE NEXT
                        12D3  ; FILESPEC IN SEQUENCE, GIVEN THE WILDCARD CONTEXT.  IF THE EXPANDED
                        12D3  ; NAME STRING IS NOT WILDCARD, THE EXPANDED NAME STRING IS CONVERTED
                        12D3  ; INTO A RESULTANT NAME STRING.
                        12D3  ;
                        12D3  ;        F$SEARCH(ENM, STREAM_ID)
                        12D3  ;
                        12D3  ; INPUTS:
                        12D3  ;
                        12D3  ;        P1(AP) = DESCRIPTOR OF EXPANDED NAME STRING
                        12D3  ;        P2(AP) = OPTIONAL BINARY STREAM IDENTIFIER, DEFAULTS TO 0
                        12D3  ;
                        12D3  ; OUTPUTS:
                        12D3  ;
                        12D3  ;        R1/R2 = DESCRIPTOR OF RESULTANT FILESPEC
                        12D3  ;-
                        12D3
                        12D3  F$SEARCH:
                        12D3  ;
                        12D3  ; SEARCH THE LIST OF SEARCH CONTEXT BLOCKS FOR THE REQUESTED STREAM.
                        12D3  ; THE CONTEXT IS MATCHED BASED ON A USER SPECIFIED 32-BIT BINARY NUMBER.
                        12D3  ;
                        12D3  ; EACH CONTEXT BLOCK HAS THE FOLLOWING FORMAT:
                        12D3  ;
                        12D3  ;        LONGWORD POINTER TO NEXT BLOCK, 0 MEANS END OF LIST
                        12D3  ;        LENGTH OF THIS BLOCK IN BYTES
                        12D3  ;        FAB BLOCK   (FAB$L_CTX CONTAINS THE USER STREAM IDENTIFIER, P2)
                        12D3  ;        NAM BLOCK
                        12D3  ;        EXPANDED NAME STRING BUFFER
                        12D3  ;        RESULTANT NAME STRING BUFFER
                        12D3  ;        PRIMARY FILESPEC BUFFER
                        12D3  ;
                        12D3  ; ON OUTPUT FROM THIS PARAGRAPH:
                        12D3  ;
                        12D3  ;        R6 = ADDRESS OF CURRENT CONTEXT BLOCK, IF FOUND
                        12D3  ;        R7 = ADDRESS OF PREVIOUS CONTEXT BLOCK IN LINKED LIST
                        12D3  ;
    50   00A0 CB   D0   12D3         MOVL    PRC_L_STACKPT(R11),R0   ;GET ADDRESS OF CURRENT INDIRECT FRAME
      57   64 A0   9E   12D8         MOVAB   IDF_L_SEARCHCTX(R0),R7  ;GET ADDRESS OF LISTHEAD
        56   67    D0   12DC  10$:   MOVL    (R7),R6                 ;GET ADDRESS OF NEXT CONTEXT BLOCK
             2E    13   12DF         BEQL    20$                     ;BRANCH IF CONTEXT NOT FOUND
  20 A6   0C AC    D1   12E1         CMPL    P2(AP),FAB$L_CTX+8(R6)  ;DOES THE STREAM IDENTIFIER MATCH?
             05    13   12E6         BEQL    15$                     ;BRANCH IF CONTEXT FOUND
        57   56    D0   12E8         MOVL    R6,R7                   ;SKIP TO NEXT BLOCK IN LIST
             EF    11   12EB         BRB     10$                     ;AND CONTINUE SEARCHING
                        12ED  ;
                        12ED  ; CONTEXT BLOCK FOUND IN LIST.  IF THE INPUT FILESPEC HAS CHANGED,
                        12ED  ; THEN RESTART THIS STREAM WITH NEW FILESPEC.
                        12ED  ;
      55   08 A6   9E   12ED  15$:   MOVAB   8(R6),R5                ;SET ADDRESS OF FAB
      54   50 A5   9E   12F1         MOVAB   FAB$C_BLN(R5),R4        ;SET ADDRESS OF NAM
      50   34 A5   9A   12F5         MOVZBL  FAB$B_FNS(R5),R0        ;GET SIZE OF PREVIOUS FILESPEC
           04 AC   D5   12F9         TSTL    P1(AP)                  ;NULL STRING?
```

```
                        0E  13  12FC          BEQL    17$                     ;YES, THEN CLEAN UP AND EXIT
    04 AC   00   2C B5  50  2D  12FE          CMPC5   R0,@FAB$L_FNA(R5),#0,P1(AP),@P1+4(AP) ;SAME FILESPEC?
                     08 BC      1305
                        51  12  1307          BNEQ    28$                     ;IF NOT, RESTART SEARCH SEQUENCE
                      0089  31  1309          BRW     30$                     ;ELSE, CONTINUE WHERE WE LEFT OFF
                      00A8  31  130C  17$:    BRW     50$                     ;CLEAN UP AND EXIT
                                130F
                                130F  ;
                                130F  ; THE CONTEXT WAS NOT FOUND AMONG THE EXISTING CONTEXT BLOCKS FOR THIS
                                130F  ; INDIRECT LEVEL.  CREATE A NEW CONTEXT BLOCK BY PARSING THE INPUT
                                130F  ; FILESPEC, P1, AND INSERTING THE CONTEXT BLOCK INTO THE LIST.
                                130F  ;
              51    03B5 8F 3C  130F  20$:    MOVZWL  #8+FAB$C_BLN+NAM$C_BLN+<3*NAM$C_MAXRSS>,R1 ;SET SIZE OF BLOCK
                     ECE9'  30  1314          BSBW    DCL$ALLDYNMEM           ;ALLOCATE CONTEXT BLOCK
                     0A 50  E8  1317          BLBS    R0,25$                  ;BRANCH IF OK
                                131A          STATUS  SYMOVF                  ;SYMBOL TABLE OVERFLOW
                      02F4  31  1321          BRW     P1_ERROR                ;REPORT ERROR
                  56    52  D0  1324  25$:    MOVL    R2,R6                   ;SAVE ADDRESS OF ALLOCATED BLOCK
                  66    67  D0  1327          MOVL    (R7),(R6)               ;INSERT INTO LINKED LIST
                  67    56  D0  132A          MOVL    R6,(R7)
               04 A6    51  D0  132D          MOVL    R1,4(R6)                ;STORE SIZE IN NEWLY ALLOCATED BLOCK
08 A6   00B0 8F   00  6E 00 2C  1331          MOVC5   #0,(SP),#0,#FAB$C_BLN+NAM$C_BLN,8(R6)   ;ZERO REST OF BLOCK
               55    08 A6  9E  133A          MOVAB   8(R6),R5                ;SET ADDRESS OF FAB
               54    50 A5  9E  133E          MOVAB   FAB$C_BLN(R5),R4        ;SET ADDRESS OF NAM
                                1342          ASSUME  FAB$B_BLN EQ FAB$B_BID+1
              65    5003 8F B0  1342          MOVW    #FAB$C_BID+<FAB$C_BLN@8>,FAB$B_BID(R5) ;INITIALIZE FAB
                                1347          ASSUME  NAM$B_BLN EQ NAM$B_BID+1
              64    6002 8F B0  1347          MOVW    #NAM$C_BID+<NAM$C_BLN@8>,NAM$B_BID(R4) ;INITIALIZE NAM
               28 A5    54  D0  134C          MOVL    R4,FAB$L_NAM(R5)        ;SET POINTER TO NAM BLOCK
                                1350          SETBIT  FAB$V_PPF,FAB$L_FOP(R5) ;USE PROCESS I/O SEGMENT (SINCE THERE
                                1355                                          ;     MAY NOT BE ANY IMAGE I/O SEGMENT)
            18 A5    0C AC  D0  1355          MOVL    P2(AP),FAB$L_CTX(R5)    ;SAVE USER SEARCH STREAM IDENTIFIER
            34 A5    04 AC  90  135A  28$:    MOVB    P1(AP),FAB$B_FNS(R5)    ;STORE PRIMARY NAME STRING INTO FAB
         2C A5    025E C4   9E  135F          MOVAB   NAM$C_BLN+<2*NAM$C_MAXRSS>(R4),FAB$L_FNA(R5)
                        30  BB  1365          PUSHR   #^M<R4,R5>              ;SAVE REGISTERS
   00FF 8F   00   08 BC  04 AC 2C 1367        MOVC5   P1(AP),@P1+4(AP),#0,#NAM$C_MAXRSS,@FAB$L_FNA(R5)
                     2C B5      1370
                        30  BA  1372          POPR    #^M<R4,R5>              ;RESTORE REGISTERS
            0A A4    FF 8F  90  1374          MOVB    #NAM$C_MAXRSS,NAM$B_ESS(R4) ;SET DESCRIPTOR OF ESA BUFFER
            0C A4    60 A4  9E  1379          MOVAB   NAM$C_BLN(R4),NAM$L_ESA(R4)
            02 A4    FF 8F  90  137E          MOVB    #NAM$C_MAXRSS,NAM$B_RSS(R4) ;SET DESCRIPTOR OF RESULT BUFFER
                   015F C4  9E  1383          MOVAB   NAM$C_BLN+NAM$C_MAXRSS(R4),- ;FOR $SEARCH (DONE HERE SINCE IT
                     04 A4      1387                  NAM$L_RSA(R4)           ;DOESN'T CHANGE DURING ENTIRE SEQUENCE)
                                1389          $PARSE  FAB=(R5)                ;SETUP EXPANDED NAME STRING, NAM FIELDS
                     22 50  E9  1392          BLBC    R0,50$                  ;(SUCH AS FID,DID,DVI,ESL)
                                1395
                                1395  ;
                                1395  ; SEARCH CONTEXT BLOCK IS NOW ASSOCIATED WITH THIS STREAM (IT WAS EITHER
                                1395  ; FOUND IN THE LIST, OR CREATED).  IF THE INPUT FILESPEC HAS CHANGED,
                                1395  ; THEN START OVER WITH NEW FILESPEC.  OTHERWISE, SEARCH FOR THE NEXT
                                1395  ; FILESPEC IN SEQUENCE.
                                1395  ;
                                1395  30$:    $SEARCH FAB=(R5)                ;SEARCH FOR NEXT FILESPEC
                     16 50  E9  139E          BLBC    R0,50$                  ;BRANCH IF ERROR DETECTED
               51    03 A4  9A  13A1          MOVZBL  NAM$B_RSL(R4),R1        ;GET LENGTH OF RESULTANT STRING
               52    04 A4  D0  13A5          MOVL    NAM$L_RSA(R4),R2        ;GET ADDRESS OF RESULTANT STRING
         08 34 A4    08     E0  13A9          BBS     #NAM$V_WILDCARD,NAM$L_FNB(R4),40$ ;IF WILDCARD, LEAVE CONTEXT
```

```
                 7E    51  7D  13AE          MOVQ    R1,-(SP)                ;SAVE RESULTANT STRING
                       0C  10  13B1          BSBB    DELCTX                  ;DEALLOCATE THE CONTEXT BLOCK
                 51    8E  7D  13B3          MOVQ    (SP)+,R1                ;RESTORE RESULTANT STRING
                           05  13B6 40$:     RSB
                               13B7
                               13B7 ;
                               13B7 ; AN ERROR HAS BEEN DURING EITHER THE $PARSE OR $SEARCH FUNCTIONS.
                               13B7 ; IN ANY CASE, TERMINATE THIS SEARCH SEQUENCE BY REMOVING THE CONTEXT
                               13B7 ; BLOCK FROM THE LIST.  RETURN A NULL STRING.  DO NOT REPORT THE ERROR,
                               13B7 ; BUT SET THE STATUS.
                               13B7 ;
                       06  10  13B7 50$:     BSBB    DELCTX                  ;DEALLOCATE THE CONTEXT BLOCK
                       51  D4  13B9          CLRL    R1                      ;RETURN NULL STRING
                 52    5E  D0  13BB          MOVL    SP,R2                   ;
                           05  13BE          RSB                             ;
                               13BF
                               13BF ;
                               13BF ; DEALLOCATE THE CONTEXT BLOCK
                               13BF ;
                               13BF ;        R5 = ADDRESS OF FAB
                               13BF ;        R6 = POINTER TO CONTEXT BLOCK
                               13BF ;        R7 = POINTER TO PREVIOUS BLOCK IN LIST
                               13BF ;
                               13BF ;        R0-R3 DESTROYED.
                               13BF ;
   34 A5   0000228D'EF     90  13BF DELCTX:  MOVB    NLA0,FAB$B_FNS(R5)      ;SET NULL DEVICE NAME
   2C A5   0000228E'EF     9E  13C7          MOVAB   NLA0+1,FAB$L_FNA(R5)    ;
              50    28 A5  D0  13CF          MOVL    FAB$L_NAM(R5),R0        ;GET NAMBLK PTR
                       04  13  13D3          BEQL    10$                     ;IF THERE IS ONE
                               13D5          SETBIT  NAM$V_SYNCHK,NAM$B_NOP(R0) ;SET SYNTAX ONLY CHECK
                               13D9 10$:     $PARSE  FAB=(R5)                ;TERMINATE SEARCH SEQUENCE
                 67    66  D0  13E2          MOVL    (R6),(R7)               ;REMOVE FROM LINKED LIST
                 50    56  D0  13E5          MOVL    R6,R0                   ;SET ADDRESS OF BLOCK TO DEALLOCATE
              51    04 A6  D0  13E8          MOVL    4(R6),R1                ;SET SIZE OF BLOCK TO DEALLOCATE
                    EC11'  30  13EC          BSBW    DCL$DEADYNMEM           ;DEALLOCATE CONTEXT BLOCK
                           05  13EF          RSB
```

```
                                     13F0          .SBTTL  SET/CLEAR SPECIFIED PRIVILEGES
                                     13F0   ;+
                                     13F0   ; F$SETPRV - SET OR CLEAR THE SPECIFIED LIST OF PRIVILEGES
                                     13F0   ;
                                     13F0   ; THIS FUNCTION SETS OR CLEARS (DEPENDING ON WHETHER A "NO" IS PREFIXED
                                     13F0   ; TO THE PRIVILEGE KEYWORD) THE SPECIFIED LIST OF PRIVILEGES IN THE PROCESS
                                     13F0   ; PERMANENT PRIVILEGE MASK.  THE PREVIOUS STATES OF THE LISTED PRIVILEGES
                                     13F0   ; ARE RETURNED AS THE VALUE OF THE FUNCTION.  NOTE THAT ONLY THE PREVIOUS
                                     13F0   ; STATES OF THE PRIVILEGES LISTED IN THE ARGUMENT ARE RETURNED, TO AVOID
                                     13F0   ; DEALING WITH STRINGS OF EXCESSIVE LENGTH.
                                     13F0   ;
                                     13F0   ;       F$SETPRV(PRVSTATES)
                                     13F0   ;
                                     13F0   ; INPUTS:
                                     13F0   ;
                                     13F0   ;       P1(AP) = DESCRIPTOR OF A COMMA SEPARATED LIST OF PRIVILEGE KEYWORDS.
                                     13F0   ;
                                     13F0   ; OUTPUTS:
                                     13F0   ;
                                     13F0   ;       R1/R2 = DESCRIPTOR OF COMMA SEPARATED LIST OF KEYWORDS INDICATING
                                     13F0   ;               THE PREVIOUS STATES OF THESE PRIVILEGES.
                                     13F0   ;-
                                     13F0
                                     13F0   F$SETPRV:
                    56   7E     7E   13F0          MOVAQ   -(SP),R6                      ;GET ADDRESS OF SCRATCH QUADWORD
                       EC0A'    30   13F3          BSBW    DCL$MARK                      ;MARK POSITION IN EXPANSION BUFFER
                                     13F6          $SETPRV_S PRVPRV=(R6),-               ;GET CURRENT PERMANENT PRIVILEGES
                                     13F6                  PRMFLG=#1
                54   04   AC    7D   1405          MOVQ    P1(AP),R4                     ;GET DESCRIPTOR OF COMMA SEPARATED LIST
                     52   54    7D   1409   10$:   MOVQ    R4,R2                         ;ASSUME NO COMMA IN LIST
                65   54   2C    3A   140C          LOCC    #^A',',R4,(R5)                ;LOCATE END OF NEXT KEYWORD
                          04    13   1410          BEQL    20$                           ;BRANCH IF NOT FOUND, USE ENTIRE STRING
                52   54   50    C3   1412          SUBL3   R0,R4,R2                      ;ISOLATE NEXT KEYWORD
                     54   70    9E   1416   20$:   MOVAB   -(R0),R4                      ;CHOP KEYWORD OFF STRING
                55   01   A1    9E   1419          MOVAB   1(R1),R5
                          0C    BB   141D          PUSHR   #^M<R2,R3>                    ;SAVE REGISTERS
  63   52   00   2260'CF  03    2D   141F          CMPC5   #3,W^NOALL+2,#0,R2,(R3) ;IS KEYWORD = "ALL"?
                          08    13   1427          BEQL    22$                           ;BRANCH IF SO
  63   52   00   225E'CF  05    2D   1429          CMPC5   #5,W^NOALL,#0,R2,(R3)   ;IS KEYWORD = "NOALL"?
                          0C    BA   1431   22$:   POPR    #^M<R2,R3>                    ;RESTORE REGISTERS
                          0F    12   1433          BNEQ    30$                           ;BRANCH IF NOT
                          51    D4   1435          CLRL    R1                            ;START WITH PRIVILEGE #0
                        008C    30   1437   25$:   BSBW    ADD_PRIV_STATE                ;APPEND PRIVILEGE KEYWORD TO STRING
        F5 51   00000040  8F    F2   143A          AOBLSS  #64,R1,25$                    ;FOR EACH PRIVILEGE IN MASK
                          16    11   1442          BRB     40$
                     7E   52    7D   1444   30$:   MOVQ    R2,-(SP)                      ;PUSH DESCRIPTOR OF KEYWORD
                          7E    DF   1447          PUSHAL  -(SP)                         ;ADDRESS OF LONGWORD TO RECEIVE BIT #
                     08   AE    9F   1449          PUSHAB  8(SP)                         ;SET ADDRESS OF KEYWORD
             00000000'GF  02    FB   144C          CALLS   #2,G^PRV$PRIVBIT              ;LOOKUP KEYWORD
                          0E    BA   1453          POPR    #^M<R1,R2,R3>                 ;AND GET PRIVILEGE BIT NUMBER IN R1
                     11   50    E9   1455          BLBC    R0,51$                        ;BRANCH IF ERROR DETECTED
                          6C    10   1458          BSBB    ADD_PRIV_STATE                ;APPEND PRIVILEGE KEYWORD TO STRING
                     7E   52    7D   145A   40$:   MOVQ    R2,-(SP)                      ;PUSH DESCRIPTOR OF KEYWORD
                          56    DD   145D          PUSHL   R6                            ;SET ADDRESS OF PRIVILEGE MASK
                     04   AE    9F   145F          PUSHAB  4(SP)                         ;SET ADDRESS OF KEYWORD
             00000000'GF  04    FB   1462          CALLS   #2+2,G^PRV$SETPRIV            ;ENABLE/DISABLE SPECIFIED PRIVILEGE
                     3B   50    E9   1469   51$:   BLBC    R0,50$                        ;BRANCH IF ERROR DETECTED
```

```
              54  D5  146C  3831          TSTL     R4                          ;ANY MORE KEYWORDS?
              99  14  146E  3832          BGTR     10$                         ;IF SO, LOOP UNTIL DONE
          7E  01  CE  1470  3833          MNEGL    #1,-(SP)                    ;CREATE PRIVILEGE MASK WITH ALL
          7E  01  CE  1473  3834          MNEGL    #1,-(SP)                    ;THE PRIVILEGE BITS SET
          52  5E  D0  1476  3835          MOVL     SP,R2                       ;GET ADDRESS OF IT
                      1479  3836          $SETPRV_S PRVADR=(R2),-              ;DISABLE ALL PRIVILEGES
                      1479  3837                   PRMFLG=#1,-                 ;
                      1479  3838                   ENBFLG=#0                   ;
                      1488  3839          $SETPRV_S PRVADR=(R6),-              ;ENABLE NEW PRIVILEGES
                      1488  3840                   PRMFLG=#1,-                 ;
                      1488  3841                   ENBFLG=#1                   ;
          5E  10  C0  1497  3842          ADDL     #16,SP                      ;CLEANUP STACK
           EB63'  30  149A  3843          BSBW     DCL$MARKEDTOKEN             ;GET DESCRIPTOR OF STRING
 F486 CA   F48A CA  D0  149D  3844        MOVL     WRK_L_MARKPTR(R10),WRK_L_EXPANDPTR(R10) ;RESTORE POSITION IN EXPANSI
              51  D7  14A4  3845          DECL     R1                          ;DO NOT RETURN TRAILING COMMA
              05      14A6  3846          RSB
                      14A7  3847
 50   00000000'8F  D1  14A7  3848 50$:    CMPL     #PRV$_NOTUNQ,R0             ;AMBIGUOUS KEYWORD?
              09  12  14AE  3849          BNEQ     55$                         ;IF NOT, EXIT WITH INVALID KEYWORD
                      14B0  3850          STATUS   ABKEYW                      ;SET AMBIGUOUS KEYWORD ERROR
              07  11  14B7  3851          BRB      80$
                      14B9  3852 55$:     STATUS   IVKEYW                      ;SET INVALID KEYWORD
          5E  08  C0  14C0  3853 80$:     ADDL     #8,SP                       ;CLEANUP STACK
            0152  31  14C3  3854          BRW      P1_ERROR                    ;REPORT ERROR
```

```
                            14C6                .SBTTL  APPEND PRIVILEGE STATE TO STRING
                            14C6        ;---
                            14C6        ; ADD_PRIV_STATE - APPEND PRIVILEGE STATE (IN KEYWORD FORM) TO STRING.
                            14C6        ;
                            14C6        ; THIS ROUTINE APPENDS THE PRIVILEGE KEYWORD ASSOCIATED WITH A GIVEN
                            14C6        ; PRIVILEGE BIT AND MASK TO THE CURRENT STRING BEING ASSEMBLED IN THE
                            14C6        ; EXPANSION BUFFER.  IF THE PRIVILEGE BIT NUMBER DOES NOT CORRESPOND
                            14C6        ; TO A KEYWORD, NOTHING IS APPENDED TO THE STRING AND NO ERROR IS RETURNED.
                            14C6        ;
                            14C6        ; INPUTS:
                            14C6        ;
                            14C6        ;       R1 = PRIVILEGE BIT NUMBER
                            14C6        ;       R6 = ADDRESS OF PRIVILEGE MASK
                            14C6        ;
                            14C6        ; OUTPUTS:
                            14C6        ;
                            14C6        ;       THE KEYWORD (NEGATED IF THE PRIVILEGE IS OFF) IS APPENDED TO THE
                            14C6        ;       STRING.
                            14C6        ;
                            14C6        ;       R0 DESTROYED.
                            14C6        ;---
                            14C6
                            14C6        ADD_PRIV_STATE:
                  51   DD   14C6                PUSHL   R1                      ;SAVE PRIVILEGE BIT NUMBER
             F486 CA   DD   14C8                PUSHL   WRK_L_EXPANDPTR(R10)    ;SAVE CURRENT POSITION IN BUFFER
          0E 66   51   E0   14CC                BBS     R1,(R6),30$             ;BRANCH IF PRIVILEGE WAS PREVIOUSLY SET
          50   4E 8F   9A   14D0                MOVZBL  #^A'N',R0               ;INSERT AN "N" INTO THE BUFFER
               EB29'   30   14D4                BSBW    DCL$PUTCHAR
          50   4F 8F   9A   14D7                MOVZBL  #^A'O',R0               ;INSERT AN "O" INTO THE BUFFER
               EB22'   30   14DB                BSBW    DCL$PUTCHAR
             F486 CA   DD   14DE        30$:    PUSHL   WRK_L_EXPANDPTR(R10)    ;CREATE DESCRIPTOR OF EXPANSION BUFFER
7E     F48E CA    6E   C3   14E2                SUBL3   (SP),WRK_L_CHARPTR(R10),-(SP)  ;SCRATCH AREA LEFT
                  5E   DD   14E8                PUSHL   SP                      ;ADDRESS OF WORD TO GET RETLEN
               04 AE   9F   14EA                PUSHAB  4(SP)                   ;ADDRESS OF BUFFER DESCRIPTOR
                  51   DD   14ED                PUSHL   R1                      ;SET PRIVILEGE BIT NUMBER
   00000000'GF    03   FB   14EF                CALLS   #3,G^PRV$KEYWORD        ;GET KEYWORD ASSOCIATED WITH BIT NUMBER
        51   8E   8E   C1   14F6                ADDL3   (SP)+,(SP)+,R1          ;POINT TO STRING END+1
             12   50   E9   14FA                BLBC    R0,50$                  ;IF NO SUCH PRIVILEGE, APPEND NOTHING
             5E   04   C0   14FD                ADDL    #4,SP                   ;THROW AWAY SAVED EXPANSION POINTER
        F486 CA   51   D0   1500                MOVL    R1,WRK_L_EXPANDPTR(R10) ;INSERT INTO EXPANSION BUFFER
             50   2C   9A   1505                MOVZBL  #^A',',R0               ;INSERT AN "," INTO THE BUFFER
               EAF5'   30   1508                BSBW    DCL$PUTCHAR
             51 8ED0        150B        90$:    POPL    R1                      ;RESTORE PRIVILEGE BIT NUMBER
                  05        150E                RSB
                            150F
        F486 CA 8ED0        150F        50$:    POPL    WRK_L_EXPANDPTR(R10)    ;BACKUP POSITION IN BUFFER
             F5   11        1514                BRB     90$                     ;AND EXIT WITHOUT APPENDING ANYTHING
```

```
                          1516      .SBTTL  CONVERT EXPRESSION TO ASCII STRING
                          1516  ;+
                          1516  ; F$STRING - CONVERT TO ASCII STRING
                          1516  ;
                          1516  ; THIS FUNCTION CONVERTS ANY EXPRESSION INTO AN ASCII STRING.  IF THE
                          1516  ; EXPRESSION IS A BINARY VALUE, IT IS CONVERTED INTO DECIMAL ASCII.
                          1516  ;
                          1516  ;       F$STRING(EXPRESSION)
                          1516  ;
                          1516  ; INPUTS:
                          1516  ;
                          1516  ;       P1(AP) = EXPRESSION VALUE
                          1516  ;
                          1516  ; OUTPUTS:
                          1516  ;
                          1516  ;       R1/R2 = DESCRIPTOR OF VALUE STRING
                          1516  ;-
                          1516
                          1516  F$STRING:
   51    04 AC     7D     1516          MOVQ    P1(AP),R1               ;GET EXPRESSION VALUE
   00000000'EF     16     151A          JSB     DCL$CVT_STRING          ;CONVERT TO ASCII STRING
         01 50     E9     1520          BLBC    R0,90$                  ;BRANCH IF ERROR DETECTED
                   05     1523          RSB
          00F1     31     1524  90$:    BRW     P1_ERROR                ;REPORT ERROR AND DISPLAY P1
```

```
                      1527          .SBTTL  RETURN CURRENT TIME AND DATE
                      1527  ;+
                      1527  ; F$TIME - RETURN CURRENT TIME AND DATE
                      1527  ;
                      1527  ; THIS LEXICAL FUNCTION OBTAINS THE CURRENT TIME AND DATE AND RETURNS A STRING
                      1527  ; DESCRIPTOR FOR IT.
                      1527  ;
                      1527  ;       F$TIME()
                      1527  ;
                      1527  ; INPUTS:
                      1527  ;
                      1527  ;       NONE.
                      1527  ;
                      1527  ; OUTPUTS:
                      1527  ;
                      1527  ;       R1 = LENGTH OF TIME AND DATE STRING.
                      1527  ;       R2 = ADDRESS OF TIME AND DATE STRING.
                      1527  ;-
                      1527
                      1527  F$TIME:                                         ;RETURN TIME AND DATE
     7E   52   7D     1527          MOVQ    R2,-(SP)                        ;BUILD TIME BUFFER DESCRIPTOR
     6E   17   D1     152A          CMPL    #23,(SP)                        ;TIME BUFFER LARGE ENOUGH?
          19   1A     152D          BGTRU   50$                             ;IF GTRU NO
     50   5E   D0     152F          MOVL    SP,R0                           ;SET ADDRESS OF TIME BUFFER DESCRIPTOR
                      1532          $ASCTIM_S (R0),(R0)                     ;GET CURRENT TIME
     51   8E   7D     1541          MOVQ    (SP)+,R1                        ;RETRIEVE TIME BUFFER PARAMETERS
       04 50   E9     1544          BLBC    R0,60$                          ;IF LBC CONVERSION FAILURE
               05     1547          RSB                                     ;
                      1548
        00C1   31     1548  50$:    BRW     BUFOVF                          ;REPORT BUFFER OVERFLOW
        00C5   31     154B  60$:    BRW     ERROR                           ;REPORT SYSTEM SERVICE ERROR
```

```
                              154E          .SBTTL  GET SYMBOL TYPE
                              154E  ;+
                              154E  ; F$TYPE - GET SYMBOL TYPE
                              154E  ;
                              154E  ; THIS FUNCTION RETURNS NULL, "INTEGER", OR "STRING" CHARACTER STRINGS
                              154E  ; DEPENDING ON WHETHER THE SPECIFIED SYMBOL IS UNDEFINED, EQUATED TO A
                              154E  ; NUMERIC STRING OR AN INTEGER, OR EQUATED TO A NON-NUMERIC STRING.
                              154E  ;
                              154E  ;       F$TYPE (SYMBOL)
                              154E  ;
                              154E  ; INPUTS:
                              154E  ;
                              154E  ;       P1(AP) = DESCRIPTOR OF SYMBOL NAME
                              154E  ;
                              154E  ; OUTPUTS:
                              154E  ;
                              154E  ;       R1/R2 = DESCRIPTOR OF ASCII STRING DESCRIBING THE SYMBOL TYPE
                              154E  ;-
                              154E
                              154E  F$TYPE:
              7E    52   7D   154E          MOVQ    R2,-(SP)                ;SAVE BUFFER DESCRIPTOR
        51    04    AC   7D   1551          MOVQ    P1(AP),R1               ;GET DESCRIPTOR OF SYMBOL NAME
                 EAA8'   30   1555          BSBW    DCL$SEARCH              ;FIND SYMBOL IN SYMBOL TABLE
              36    50   E9   1558          BLBC    R0,20$                  ;IF NOT FOUND, RETURN NULL
                    52   D5   155B          TSTL    R2                      ;STRING DESCRIPTOR?
                    28   13   155D          BEQL    10$                     ;NO, THEN BRANCH
                    51   D5   155F          TSTL    R1                      ;NULL STRING?
                    18   13   1561          BEQL    5$                      ;YES, THEN BRANCH
              6E    51   D1   1563          CMPL    R1,(SP)                 ;BUFFER LARGE ENOUGH?
                    2D   1A   1566          BGTRU   90$                     ;IF GTRU NO
              6E    51   D0   1568          MOVL    R1,(SP)                 ;COPY SYMBOL LENGTH
     04 BE    62    51   28   156B          MOVC3   R1,(R2),@4(SP)          ;COPY THE SYMBOL VALUE
              52    6E   7D   1570          MOVQ    (SP),R2                 ;MOVE THE DESCRIPTOR INTO R2/R3
              51    01   D0   1573          MOVL    #PRC_K_DEC,R1           ;SET CONVERSION RADIX
                 EA87'   30   1576          BSBW    DCL$CNVASCBIN           ;CONVERT THE STRING TO DECIMAL
                    0C   13   1579          BEQL    10$                     ;IF SUCCESSFUL, THEN INTEGER
     52    0000226E'EF   9E   157B  5$:     MOVAB   STRING,R2               ;IF NOT, THEN RETURN 'STRING'
              51    82   9A   1582          MOVZBL  (R2)+,R1                ;
                    0A   11   1585          BRB     20$                     ;
     52    00002275'EF   9E   1587  10$:    MOVAB   INTEGER,R2              ;RETURN 'INTEGER'
              51    82   9A   158E          MOVZBL  (R2)+,R1                ;
              5E    08   C0   1591  20$:    ADDL    #8,SP                   ;RESTORE THE STACK
                         05   1594          RSB
                              1595
              5E    08   C0   1595  90$:    ADDL    #8,SP                   ;RESTORE THE STACK
                  0071   31   1598          BRW     BUFOVF                  ;REPORT BUFFER OVERFLOW
```

```
                          159B          .SBTTL  RETURN USER IDENTIFICATION CODE
                          159B     ;+
                          159B     ; F$USER - RETURN USER IDENTIFICATION CODE
                          159B     ;
                          159B     ; THIS LEXICAL FUNCTION FUNCTION CONVERTS THE USER IDENTIFICATION CODE TO AN
                          159B     ; ASCII STRING AND RETURNS A DESCRIPTOR FOR IT.
                          159B     ;
                          159B     ;       F$USER()
                          159B     ;
                          159B     ; INPUTS:
                          159B     ;
                          159B     ;       NONE.
                          159B     ;
                          159B     ; OUTPUTS:
                          159B     ;
                          159B     ;       R1 = LENGTH OF USER IDENTIFICATION CODE.
                          159B     ;       R2 = ADDRESS OF USER IDENTIFICATION CODE.
                          159B     ;-
                          159B
                          159B     F$USER:                                      ;RETURN USER IDENTIFICATION CODE
             7E     7C    159B              CLRQ    -(SP)                        ;RESERVE IOSB
             7E     7C    159D              CLRQ    -(SP)                        ;CLEAR LAST ITEM AND LENGTH ADDRESS
          F8 AE     9F    159F              PUSHAB  -8(SP)                       ;PUSH ADDRESS TO STORE USER IDENTIFICATION
    03040004 8F     DD    15A2              PUSHL   #JPI$_UIC@16+4               ;SET PARAMETER ID AND LENGTH
       50    5E     D0    15A8              MOVL    SP,R0                        ;SET ADDRESS OF TIME LIST
                          15AB              $GETJPIW_S ITMLST=(R0),-             ;GET USER IDENTIFICATION CODE
                          15AB                      IOSB=16(R0),-                ;
                          15AB                      EFN=#EXE$C_SYSEFN
          11 50     E9    15C3              BLBC    R0,20$                       ;IF LBC SERVICE FAILURE
    50    10 AE     3C    15C6              MOVZWL  16(SP),R0                    ;GET STATUS FROM IOSB
          0A 50     E9    15CA              BLBC    R0,20$                       ;IF LBC SERVICE FAILURE
       54    6E     D0    15CD              MOVL    (SP),R4                      ;GET UIC
           009C     30    15D0              BSBW    FORMAT_UIC                   ;FORMAT THE UIC
       5E    18     C0    15D3              ADDL    #6*4,SP                      ;REMOVE ITEM LIST AND IOSB FROM STACK
             05           15D6              RSB                                  ;
                          15D7
       5E    18     C0    15D7     20$:     ADDL    #6*4,SP                      ;REMOVE ITEM LIST AND IOSB FROM STACK
             37     11    15DA              BRB     ERROR                        ;
```

```
                     15DC          .SBTTL  RETURN VERIFY MODE
                     15DC   ;+
                     15DC   ; F$VERIFY - RETURN VERIFY MODE
                     15DC   ;
                     15DC   ; THIS LEXICAL FUNCTION RETURNS A BOOLEAN VARIABLE THAT DESCRIBES THE CURRENT
                     15DC   ; STATE OF THE VERIFY MODE AND, IN ADDITION, WILL SET OR CLEAR VERIFICATION
                     15DC   ; IF A NON-NULL ARGUMENT IS SPECIFIED.
                     15DC   ;
                     15DC   ;       F$VERIFY([NEW_VERIFY_MODE [, NEW_IMAGE_VERIFY_MODE]])
                     15DC   ;
                     15DC   ; INPUTS:
                     15DC   ;
                     15DC   ;       P1(AP) = OPTIONAL NEW VERIFY MODE (1 TO SET, 0 TO CLEAR)
                     15DC   ;               (-1 IF UNSPECIFIED)
                     15DC   ;       P2(AP) = OPTIONAL NEW IMAGE VERIFY MODE (1 TO SET, 0 TO CLEAR)
                     15DC   ;               (-1 IF UNSPECIFIED)
                     15DC   ;
                     15DC   ; OUTPUTS:
                     15DC   ;
                     15DC   ;       R1 = BOOLEAN RESULT
                     15DC   ;       R2 = 0 TO INDICATE RESULT IS BINARY
                     15DC   ;-
                     15DC
                     15DC   F$VERIFY:                                              ;RETURN VERIFY MODE
         58    01  D0  15DC          MOVL    #1,R8                                 ;ASSUME VERIFY IS CURRENTLY ON
   02 68 AB    07  E0  15DF          BBS     #PRC_V_VERIFY,PRC_W_FLAGS(R11),10$ ;BR IF ON
               58  D4  15E4          CLRL    R8                                    ;MARK VERIFY TURNED OFF
         04 AC     D5  15E6   10$:   TSTL    P1(AP)                                ;A NON-NULL ARGUMENT?
               1B  19  15E9          BLSS    90$                                   ;NOPE, IT'S NULL, NO CHANGES
                       15EB          CLRBIT  PRC_V_VERIFY,PRC_W_FLAGS(R11) ;TURN OFF VERIFICATION
      05 04 AC     E9  15F0          BLBC    P1(AP),20$                            ;LEAVE VERIFY OFF IF LBC
                       15F4          SETBIT  PRC_V_VERIFY,PRC_W_FLAGS(R11) ;ELSE TURN ON VERIFICATION
   56    0C AC     D0  15F9   20$:   MOVL    P2(AP),R6                             ;A NON-NULL ARGUMENT?
               04  18  15FD          BGEQ    30$                                   ;BRANCH IF NON-NULL
   56    04 AC     D0  15FF          MOVL    P1(AP),R6                             ;ELSE USE FIRST ARGUMENT
            E9FA'  30  1603   30$:   BSBW    DCL$SETVERIFY_IMAGE                   ;SET IMAGE VERIFICATION
      51       58  D0  1606   90$:   MOVL    R8,R1                                 ;RETURN BOOLEAN RESULT
               52  D4  1609          CLRL    R2                                    ;INDICATE RESULT IS BINARY
               05      160B          RSB                                           ;
```

```
                        160C          .SBTTL  ERROR RETURNS
                        160C
                        160C  ;
                        160C  ; BUFFER OVERFLOW
                        160C  ;
                        160C
                        160C  BUFOVF: STATUS  BUFOVF                          ;SET BUFFER OVERFLOW STATUS
         E9EA'    30    1613  ERROR:  BSBW    DCL$MARK                        ;SET FOR NO SEGMENT DISPLAY
           39     11    1616          BRB     DCL$PARSERR
                        1618
                        1618  ;
                        1618  ; SIGNAL ERROR WITH AN ARGUMENT AS THE ERROR SEGMENT
                        1618  ;
                        1618
                        1618  P1_ERROR:
      52   04 AC  9E    1618          MOVAB   P1(AP),R2                       ;USE P1 AS ERROR SEGMENT
           1C     11    161C          BRB     PN_ERROR
                        161E  P2_ERROR:
      52   0C AC  9E    161E          MOVAB   P2(AP),R2                       ;USE P2 AS ERROR SEGMENT
           16     11    1622          BRB     PN_ERROR
                        1624  P3_ERROR:
      52   14 AC  9E    1624          MOVAB   P3(AP),R2                       ;USE P3 AS ERROR SEGMENT
           10     11    1628          BRB     PN_ERROR
                        162A  P4_ERROR:
      52   1C AC  9E    162A          MOVAB   P4(AP),R2                       ;USE P4 AS ERROR SEGMENT
           0A     11    162E          BRB     PN_ERROR
                        1630  P5_ERROR:
      52   24 AC  9E    1630          MOVAB   P5(AP),R2                       ;USE P5 AS ERROR SEGMENT
           04     11    1634          BRB     PN_ERROR
                        1636  P6_ERROR:
      52   2C AC  9E    1636          MOVAB   P6(AP),R2                       ;USE P6 AS ERROR SEGMENT
                        163A
                        163A  PN_ERROR:
           51     D4    163A          CLRL    R1                              ;PRESET TO RETURN NULL STRING
   0F F0 AA  0C   E0    163C          BBS     #WRK_V_COMMENT,WRK_W_FLAGS(R10),30$ ;RETURN NULL IF IN COMMENT
  F48A CA   04 A2 D0    1641          MOVL    4(R2),@RK_L_MARKPTR(R10) ;SET START OF ERROR SEGMENT
F486 CA  04 A2  62 C1   1647          ADDL3   (R2),4(R2),@RK_L_EXPANDPTR(R10) ;SET END OF ERROR SEGMENT
           01     11    164E          BRB     DCL$PARSERR                     ;REPORT ERROR
           05           1650  30$:    RSB                                     ;RETURN WITH FUNCTION=NULL/ZERO
                        1651
                        1651  ;
                        1651  ; ERROR EXIT FROM LEXICAL FUNCTION STRING EXPANSION
                        1651  ;
                        1651
                        1651  DCL$PARSERR::                                   ;
         E9AC'    30    1651          BSBW    DCL$ERRORMSG                    ;OUTPUT ERROR MESSAGE
           50     DD    1654          PUSHL   R0                              ;SAVE FINAL STATUS
    00000000'EF   16    1656          JSB     DCL$FLUSH                       ;FLUSH COMMAND
           50   8ED0    165C          POPL    R0                              ;RETRIEVE FINAL STATUS
    00000000'EF   16    165F          JSB     DCL$SET_STATUS                  ;SET COMMAND STATUS
    00000000'EF   17    1665          JMP     DCL$RESTART                     ;
```

```
                       166B        .SBTTL  CONVERT UIC TO ASCII STRING
                       166B  ;+
                       166B  ; FORMAT_UIC - CONVERT BINARY UIC TO ASCII STRING
                       166B  ;
                       166B  ; CONVERTS A BINARY UIC LONGWORD INTO AN ASCII STRING
                       166B  ;
                       166B  ; INPUTS:
                       166B  ;
                       166B  ;       R4 = UIC LONGWORD
                       166B  ;
                       166B  ; OUTPUTS:
                       166B  ;
                       166B  ;       R1/R2 = DESCRIPTOR OF ASCII STRING
                       166B  ;
                       166B  ;-
                       166B
        49 25 21 00'   166B  UIC:     .ASCIC  '!%I'
                 03    166B
                       166F
                       166F  FORMAT_UIC:
    51    F9 AF  9E    166F           MOVAB   UIC,R1                     ;GET ADDRESS OF ASCIC FAO STRING
       50    81  9A    1673           MOVZBL  (R1)+,R0                   ;CREATE DESCRIPTOR OF FAO STRING
       7E    50  7D    1676           MOVQ    R0,-(SP)                   ;PUSH ONTO STACK
     F486    CA  DD    1679           PUSHL   WRK_L_EXPANDPTR(R10)       ;CREATE DESCRIPTOR OF SCRATCH BUFFER
     F486    CA  C3    167D           SUBL3   WRK_L_EXPANDPTR(R10),-     ;
 7E  F48E    CA        1681                   WRK_L_CHARPTR(R10),-(SP)   ;
       50    5E  D0    1685           MOVL    SP,R0                      ;
                       1688           $FAO_S  CTRSTR=8(R0),-             ;FORMAT INTO ASCII STRING
                       1688                   OUTBUF=(R0),-              ;ADDRESS OF RESULT BUFFER
                       1688                   OUTLEN=(R0),-              ;WORD TO RECEIVE RESULT LENGTH
                       1688                   P1=R4                      ;UIC NUMBER
           1E    BA    1698           POPR    #^M<R1,R2,R3,R4>           ;GET DESCRIPTOR OF STRING
                       169A                                              ;AND CLEAN STACK
                 05    169A           RSB
```

```
                                    169B         .SBTTL  CONVERT PROTECTION MASK TO ASCII STRING
                                    169B ;+
                                    169B ; FORMAT_PROT - CONVERT BINARY PROTECTION MASK TO ASCII STRING
                                    169B ;
                                    169B ; CONVERTS A BINARY PROTECTION MASK WORD INTO AN ASCII STRING
                                    169B ;
                                    169B ; INPUTS:
                                    169B ;
                                    169B ;        R4 = PROTECTION WORD
                                    169B ;        R10 = BASE ADDRESS OF COMMAND WORK AREA
                                    169B ;
                                    169B ; OUTPUTS:
                                    169B ;
                                    169B ;        R1/R2 = DESCRIPTOR OF ASCII STRING
                                    169B ;
                                    169B ; REGISTER USAGE:
                                    169B ;
                                    169B ;        R0,R1,R2,R3,R7,R8
                                    169B ;-
                                    169B
                               00   169B NULL:   .BYTE   0
                           3D  00'  169C EQUAL:  .ASCIC  /=/
                               01   169C
                      44 45 57 52   169E ACCESS: .ASCII  /RWED/
                                    16A2 PROTECTMSG:
44 41 21 43 41 21 4D 45 54 53 59 53 16A2         .ASCII  /SYSTEM!AC!AD, OWNER!AC!AD, GROUP!AC!AD, WORLD!AC!AD/
41 21 43 41 21 52 45 4E 57 4F 20 2C 16AE
21 43 41 21 50 55 4F 52 47 20 2C 44 16BA
43 41 21 44 4C 52 4F 57 20 2C 44 41 16C6
                         44 41 21   16D2
                                    16D5 PROTECTEND:
                                    16D5
                                    16D5 FORMAT_PROT:
                    53   5E   D0    16D5         MOVL    SP,R3                   ;START A SCRATCH AREA ON THE STACK
                    5E   30   C2    16D8         SUBL    #4*12,SP                ;ALLOCATE FAO ARGUMENT DESCRIPTORS
                                    16DB                                         ;AND ASCIC ADDRESS DESCRIPTORS
                                    16DB
                    58   0C   9A    16DB         MOVZBL  #12,R8                  ;SET OUTER LOOP INDEX
                         7E   D4    16DE 10$:    CLRL    -(SP)                   ;ALLOCATE SPACE FOR FAO ARG BUFFER
                         73   7C    16E0         CLRQ    -(R3)                   ;ZERO FAO ARG DESCRIPTOR
                  73  B6 AF   9E    16E2         MOVAB   NULL,-(R3)              ;INIT ASCIC POINTER, ASSUME NO ACCESS
          51    54   04   58   EF   16E6         EXTZV   R8,#4,R4,R1             ;EXTRACT ONE PROTECTION FIELD
                    52   51   D2    16EB         MCOML   R1,R2                   ;COMPLEMENT IT
                    52   0F   D3    16EE         BITL    #^XF,R2                 ;ALL ACCESS DENIED?
                         1F   13    16F1         BEQL    40$                     ;IF EQL, YES
                  63  A6 AF   9E    16F3         MOVAB   EQUAL,(R3)              ;SET POINTER TO ASCIC =
                         57   D4    16F7         CLRL    R7                      ;CLEAR INNER LOOP INDEX
               08 A3     6E   9E    16F9         MOVAB   (SP),8(R3)              ;PUT ADDR OF BUFFER IN DESCRIPTOR
                      04 A3   D4    16FD         CLRL    4(R3)                   ;CLEAR BUFFER LENGTH
                                    1700
               0A 51     57   E0    1700 20$:    BBS     R7,R1,30$               ;IF SET, ACCESS DENIED
         04 B34E    96 AF47   90    1704         MOVB    ACCESS[R7],@4(R3)[SP]   ;MOVE ACCESS CHARACTER INTO BUFFER
                      04 A3   D6    170B         INCL    4(R3)                   ;INCREMENT BUFFER LENGTH
               EE 57     04   F2    170E 30$:    AOBLSS  #4,R7,20$               ;CHECKED ALL FOUR BITS?
                                    1712
      FFC5 58   FC 8F    00   9D    1712 40$:    ACBB    #0,#-4,R8,10$           ;CHECKED ALL FOUR MODES?
                                    1719
```

```
              86 AF   9F  1719  4225          PUSHAB  PROTECTMSG                  ;PUSH DESCRIPTOR FOR CONTROL STRING
                 33   DD  171C  4226          PUSHL   #PROTECTEND-PROTECTMSG
            F486 CA   DD  171E  4227          PUSHL   WRK_L_EXPANDPTR(R10)        ;PUSH DESCRIPTOR TO RECEIVE FAO OUTPUT
                 37   DD  1722  4228          PUSHL   #PROTECTEND-PROTECTMSG+4
              50 5E   D0  1724  4229          MOVL    SP,R0                       ;SAVE POINTER TO TOP OF STACK
                          1727  4230          $FAOL_S CTRSTR=8(R0),-              ;GENERATE FORMATTED ASCII STRING
                          1727  4231                  OUTBUF=(R0),-
                          1727  4232                  OUTLEN=(R0),-
                          1727  4233                  PRMLST=(R3)
                          1737  4234
              51 6E   7D  1737  4235          MOVQ    (SP),R1                     ;MOVE OUTPUT DESCRIPTOR INTO R1/R2
    5E  00000050 8F   C0  173A  4236          ADDL    #20*4,SP                    ;RESTORE STACK
                      05  1741  4237          RSB
```

```
                                    1742        .SBTTL  CONVERT FILE ID TO ASCII STRING
                                    1742  ;+
                                    1742  ; FORMAT_FILEID - CONVERT FILE ID TO ASCII STRING
                                    1742  ;
                                    1742  ; INPUTS:
                                    1742  ;
                                    1742  ;       R4 = ADDRESS OF 3-WORD FILE ID
                                    1742  ;
                                    1742  ; OUTPUTS:
                                    1742  ;
                                    1742  ;       R1/R2 = DESCRIPTOR OF ASCII STRING
                                    1742  ;-
                                    1742
55 21 2C 57 55 21 2C 57 55 21 28 00' 1742  FILEID: .ASCIC  '(!UW,!UW,!UB)'
                              29 42 174E
                                 0D 1742
                                    1750
                                    1750  FORMAT_FILEID:
             51    EF AF     9E     1750        MOVAB   FILEID,R1                       ;GET ADDRESS OF ASCIC FAO STRING
                   50    81  9A     1754        MOVZBL  (R1)+,R0                        ;CREATE DESCRIPTOR OF FAO STRING
                   7E    50  7D     1757        MOVQ    R0,-(SP)                        ;PUSH ONTO STACK
                   F486  CA  DD     175A        PUSHL   WRK_L_EXPANDPTR(R10)            ;CREATE DESCRIPTOR OF SCRATCH BUFFER
     7E   F48E CA  F486  CA  C3     175E        SUBL3   WRK_L_EXPANDPTR(R10),WRK_L_CHARPTR(R10),-(SP)
                   50    5E  D0     1766        MOVL    SP,R0
                                    1769        $FAO_S  CTRSTR=8(R0),-                  ;FORMAT INTO ASCII STRING
                                    1769                OUTBUF=(R0),-                   ;ADDRESS OF RESULT BUFFER
                                    1769                OUTLEN=(R0),-                   ;WORD TO RECEIVE RESULT LENGTH
                                    1769                P1=(R4),-                       ;FILE NUMBER
                                    1769                P2=2(R4),-                      ;SEQUENCE NUMBER
                                    1769                P3=4(R4)                        ;RELATIVE VOLUME NUMBER
                         1E  BA     177F        POPR    #^M<R1,R2,R3,R4>                ;GET DESCRIPTOR OF STRING
                                    1781                                                ;AND CLEAN STACK
                             05     1781        RSB
```

```
                             1782       .SBTTL  CONVERT BINARY TO BOOLEAN STRING
                             1782  ;+
                             1782  ; FORMAT_BOOLEAN - CONVERT BINARY TO ASCII BOOLEAN STRING
                             1782  ;
                             1782  ; INPUTS:
                             1782  ;
                             1782  ;       R1 = STATUS (ONLY BOTTOM BIT CHECKED)
                             1782  ;
                             1782  ; OUTPUTS:
                             1782  ;
                             1782  ;       R1/R2 = DESCRIPTOR OF ASCII STRING (TRUE OR FALSE)
                             1782  ;-
                             1782
                             1782  FORMAT_BOOLEAN:
52    00002263'EF    9E      1782       MOVAB   FALSE,R2                ;ASSUME FALSE
            07 51    E9      1789       BLBC    R1,10$                  ;BRANCH IF FALSE
52    00002269'EF    9E      178C       MOVAB   TRUE,R2                 ;SET TRUE
         51    82    9A      1793  10$: MOVZBL  (R2)+,R1                ;SET DESCRIPTOR OF BOOLEAN STRING
                     05      1796       RSB
                             1797
                             1797       .END
```

```
$$.TMP1              = 00000001
$$.TMP2              = 00000065
$$T1                 = 00000001
$$T2                 = 00000006
$ARGS$               = 00000044
$BITS$               = 00000008
$XX$                 = 00000004
$ZZ$                 = 00000004
ACCESS                 0000169E R       03
ACCESS_TABLE           0000204C R       02
ACP$V_READCHK        = 00000000  G
ACP$V_SWAPGRP        = 00000001  G
ACP$V_SWAPMAG        = 00000003  G
ACP$V_SWAPPRV        = 00000002  G
ACP$V_SWAPSYS        = 00000000  G
ACP$V_WRITECHK       = 00000001  G
ACP_TABLE              00001E5E R       02
ADD_PRIV_STATE         000014C6 R       03
ARG_C_ANY            = 00000003
ARG_C_BINARY         = 00000000
ARG_C_ENTSIZE        = 00000008
ARG_C_STRING         = 00000001
ARG_C_SYMREF         = 00000002
ARG_S_TYPE           = 00000002
ARG_V_KEYWORD        = 00000003
ARG_V_OPTIONAL       = 00000002
ARG_V_TYPE           = 00000000
BATCH                  00002218 R       02
BIN                  = 00000000
BIT                  = 00000004
BIT...               = 00000001
BUFOVF                 0000160C R       03
CLI$_ABFNAM          = 000381B8
CLI$_ABKEYW          = 00038010
CLI$_ARGREQ          = 00038268
CLI$_BUFOVF          = 00038018
CLI$_EXPSYN          = 00038038
CLI$_INVRANGE        = 00038228
CLI$_IVATIME         = 00038290
CLI$_IVDTIME         = 00038298
CLI$_IVFNAM          = 000381C0
CLI$_IVKEYW          = 00038060
CLI$_STRTOOLNG       = 000388FA
CLI$_SYMDEL          = 00038130
CLI$_SYMOVF          = 00038138
CTL$AG_CLIDATA         ********   X     03
CTL$GB_MSGMASK         ********   X     03
CVTIME_ITEMS           000020B8 R       02
CVTIME_TYPES           0000209A R       02
DCL$ALLDYNMEM          ********   X     03
DCL$ALLOCSYM           ********   X     03
DCL$CBTA_DEC           ********   X     03
DCL$CBTA_HEX           ********   X     03
DCL$CBTA_HEXSTR        ********   X     03
DCL$CNVASCBIN          ********   X     03
DCL$CVT_BINARY         ********   X     03
DCL$CVT_STRING         ********   X     03
DCL$DEADYNMEM          ********   X     03
DCL$DISABLE            ********   X     03
DCL$EDIT               ********   X     03
DCL$ERRORMSG           ********   X     03
DCL$EXPRESS            ********   X     03
DCL$FLUSH              ********   X     03
DCL$GETOKEN            ********   X     03
DCL$LEXIF              00000000 RG      03
DCL$MARK               ********   X     03
DCL$MARKEDTOKEN        ********   X     03
DCL$MOVCHAR            ********   X     03
DCL$PARSERR            00001651 RG      03
DCL$PUTCHAR            ********   X     03
DCL$RESTART            ********   X     03
DCL$SEARCH             ********   X     03
DCL$SETNBLK            ********   X     03
DCL$SETVERIFY_IMAGE    ********   X     03
DCL$SET_STATUS         ********   X     03
DCL$SHOWDEFINT         ********   X     03
DCL$SQUEEZE            ********   X     03
DCL$TRIM               ********   X     03
DCL$UPCASE             ********   X     03
DELCTX                 000013BF R       03
DEV$V_EXISTS         = FFFFFFFF
DEV$V_SPL            = 00000006
DISPATCH               000001BB R       03
DVI$C_ACP_F11V1      = 00000001
DVI$C_ACP_F11V2      = 00000002
DVI$C_ACP_JNL        = 00000006
DVI$C_ACP_MTA        = 00000003
DVI$C_ACP_NET        = 00000004
DVI$C_ACP_REM        = 00000005
DVI$_ACPPID          = 00000040
DVI$_ACPTYPE         = 00000042
DVI$_ALL             = 0000006C
DVI$_ALLDEVNAM       = 000000EC
DVI$_ALLOCLASS       = 000000F2
DVI$_ALT_HOST_AVAIL  = 000000F4
DVI$_ALT_HOST_NAME   = 000000F6
DVI$_ALT_HOST_TYPE   = 000000F8
DVI$_AVL             = 00000062
DVI$_CCL             = 00000048
DVI$_CLUSTER         = 0000003A
DVI$_CONCEALED       = 00000044
DVI$_CYLINDERS       = 00000028
DVI$_DEVBUFSIZ       = 00000008
DVI$_DEVCHAR         = 00000002
DVI$_DEVCHAR2        = 000000E6
DVI$_DEVCLASS        = 00000004
DVI$_DEVDEPEND       = 0000000A
DVI$_DEVDEPEND2      = 0000001C
DVI$_DEVLOCKNAM      = 000000F0
DVI$_DEVNAM          = 00000020
DVI$_DEVSTS          = 000000E4
DVI$_DEVTYPE         = 00000006
DVI$_DIR             = 0000004C
DVI$_DMT             = 00000068
```

```
DVI$_DUA                         = 0000005C          DVI$_SQD                         = 00000050
DVI$_ELG                         = 0000006A          DVI$_STS                         = 000000E2
DVI$_ERRCNT                      = 00000014          DVI$_SWL                         = 00000070
DVI$_FOD                         = 0000005A          DVI$_TRACKS                      = 00000026
DVI$_FOR                         = 0000006E          DVI$_TRANSCNT                    = 00000036
DVI$_FREEBLOCKS                  = 0000002A          DVI$_TRM                         = 0000004A
DVI$_FULLDEVNAM                  = 000000E8          DVI$_TT_ALTYPEAHD                = 000000B8
DVI$_GEN                         = 00000060          DVI$_TT_ANSICRT                  = 000000D6
DVI$_HOST_AVAIL                  = 000000FA          DVI$_TT_APP_KEYPAD               = 000000D2
DVI$_HOST_COUNT                  = 000000FC          DVI$_TT_AUTOBAUD                 = 000000AE
DVI$_HOST_NAME                   = 000000FE          DVI$_TT_AVO                      = 000000DC
DVI$_HOST_TYPE                   = 00000100          DVI$_TT_BLOCK                    = 000000DA
DVI$_IDV                         = 00000072          DVI$_TT_BRDCSTMBX                = 000000B4
DVI$_LOCKID                      = 000000EA          DVI$_TT_CRFILL                   = 00000092
DVI$_LOGVOLNAM                   = 0000002C          DVI$_TT_DCL_MAILBX               = 000000BC
DVI$_MAXBLOCK                    = 0000001A          DVI$_TT_DECCRT                   = 000000E0
DVI$_MAXFILES                    = 0000003C          DVI$_TT_DECCRT2                  = 00000114
DVI$_MBX                         = 00000066          DVI$_TT_DIALUP                   = 000000C4
DVI$_MEDIA_ID                    = 0000011A          DVI$_TT_DISCONNECT               = 000000C8
DVI$_MEDIA_NAME                  = 00000116          DVI$_TT_DMA                      = 000000B6
DVI$_MEDIA_TYPE                  = 00000118          DVI$_TT_DRCS                     = 000000CE
DVI$_MNT                         = 00000064          DVI$_TT_EDIT                     = 000000DE
DVI$_MOUNTCNT                    = 00000038          DVI$_TT_EDITING                  = 000000BE
DVI$_NET                         = 00000058          DVI$_TT_EIGHTBIT                 = 0000009A
DVI$_NEXTDEVNAM                  = 00000034          DVI$_TT_ESCAPE                   = 00000084
DVI$_ODV                         = 00000074          DVI$_TT_FALLBACK                 = 000000C2
DVI$_OPCNT                       = 00000016          DVI$_TT_HALFDUP                  = 000000A4
DVI$_OPR                         = 00000054          DVI$_TT_HANGUP                   = 000000B0
DVI$_OWNUIC                      = 00000010          DVI$_TT_HOSTSYNC                 = 00000086
DVI$_PID                         = 0000000E          DVI$_TT_INSERT                   = 000000C0
DVI$_RCK                         = 0000007A          DVI$_TT_LFFILL                   = 00000094
DVI$_RCT                         = 00000056          DVI$_TT_LOCALECHO                = 000000AC
DVI$_REC                         = 00000046          DVI$_TT_LOWER                    = 0000008C
DVI$_RECSIZ                      = 00000018          DVI$_TT_MBXDSABL                 = 0000009C
DVI$_REFCNT                      = 0000001E          DVI$_TT_MECHFORM                 = 000000A2
DVI$_REMOTE_DEVICE               = 00000102          DVI$_TT_MECHTAB                  = 0000008E
DVI$_RND                         = 00000076          DVI$_TT_MODEM                    = 000000A6
DVI$_ROOTDEVNAM                  = 00000032          DVI$_TT_MODHANGUP                = 000000B2
DVI$_RTM                         = 00000078          DVI$_TT_NOBRDCST                 = 0000009E
DVI$_SDI                         = 0000004E          DVI$_TT_NOECHO                   = 00000080
DVI$_SECTORS                     = 00000024          DVI$_TT_NOTYPEAHD                = 00000082
DVI$_SERIALNUM                   = 0000003E          DVI$_TT_OPER                     = 000000A8
DVI$_SERVED_DEVICE               = 00000104          DVI$_TT_PAGE                     = 000000AA
DVI$_SHDW_CATCHUP_COPYING        = 00000106          DVI$_TT_PASSALL                  = 0000007E
DVI$_SHDW_MASTER                 = 00000108          DVI$_TT_PASTHRU                  = 000000CA
DVI$_SHDW_MASTER_NAME            = 0000010A          DVI$_TT_PHYDEVNAM                = 00000112
DVI$_SHDW_MEMBER                 = 0000010C          DVI$_TT_PRINTER                  = 000000D0
DVI$_SHDW_MERGE_COPYING          = 0000010E          DVI$_TT_READSYNC                 = 000000A0
DVI$_SHDW_NEXT_MBR_NAME          = 00000110          DVI$_TT_REGIS                    = 000000D8
DVI$_SHDW_SPARE_BIT_1            = 0000011C          DVI$_TT_REMOTE                   = 00000098
DVI$_SHDW_SPARE_BIT_2            = 0000011E          DVI$_TT_SCOPE                    = 00000096
DVI$_SHDW_SPARE_INTEGER_1        = 00000124          DVI$_TT_SCRIPT                   = 0000008A
DVI$_SHDW_SPARE_INTEGER_2        = 00000126          DVI$_TT_SECURE                   = 000000C6
DVI$_SHDW_SPARE_STRING_1         = 00000120          DVI$_TT_SETSPEED                 = 000000BA
DVI$_SHDW_SPARE_STRING_2         = 00000122          DVI$_TT_SIXEL                    = 000000CC
DVI$_SHR                         = 0000005E          DVI$_TT_SYSPWD                   = 000000D4
DVI$_SPL                         = 00000052          DVI$_TT_TTSYNC                   = 00000088
```

```
DVI$_TT_WRAP                     = 00000090                FSPARSE                          0000101D R      03
DVI$_UNIT                        = 0000000C                F$PID                            00001181 R      03
DVI$_VOLCOUNT                    = 00000030                F$PRIVILEGE                      00001210 R      03
DVI$_VOLNAM                      = 00000022                F$PROCESS                        00001289 R      03
DVI$_VOLNUMBER                   = 0000002E                F$SEARCH                         000012D3 R      03
DVI$_VOLSETMEM                   = 000000EE                F$SETPRV                         000013F0 R      03
DVI$_VPROT                       = 00000012                F$STRING                         00001516 R      03
DVI$_WCK                         = 0000007C                F$TIME                           00001527 R      03
EDIT_B_FLAGS                       00000000                F$TRNLNM                         00000DF3 R      03
EDIT_M_COLLAPSE                  = 00000004                F$TYPE                           0000154E R      03
EDIT_M_COMPRESS                  = 00000002                F$USER                           0000159B R      03
EDIT_M_EOL                       = 00000040                F$VERIFY                         000015DC R      03
EDIT_M_LOWERCASE                 = 00000020                FAB$B_BID                      = 00000000
EDIT_M_TRIM                      = 00000008                FAB$B_BKS                      = 0000003E
EDIT_M_UNCOMMENT                 = 00000001                FAB$B_BLN                      = 00000001
EDIT_M_UPCASE                    = 00000010                FAB$B_DNS                      = 00000035
EDIT_TABLE                         00001E9E R      02      FAB$B_FAC                      = 00000016
EDIT_V_COLLAPSE                  = 00000002                FAB$B_FNS                      = 00000034
EDIT_V_COMPRESS                  = 00000001                FAB$B_FSZ                      = 0000003F
EDIT_V_EOL                       = 00000006                FAB$B_ORG                      = 0000001D
EDIT_V_LOWERCASE                 = 00000005                FAB$B_RAT                      = 0000001E
EDIT_V_TRIM                      = 00000003                FAB$B_RFM                      = 0000001F
EDIT_V_UNCOMMENT                 = 00000000                FAB$B_SHR                      = 00000017
EDIT_V_UPCASE                    = 00000004                FAB$C_BID                      = 00000003
ENVIRON_TABLE                      00001ED7 R      02      FAB$C_BLN                      = 00000050
EQUAL                              0000169C R      03      FAB$C_FIX                      = 00000001
ERROR                              00001613 R      03      FAB$C_IDX                      = 00000020
EXE$C_SYSEFN                       ********   X    03      FAB$C_REL                      = 00000010
EXE$V_BRK_DISUSER                = 00C00003                FAB$C_SEQ                      = 00000000
EXE$V_BRK_TERM                   = 00000002                FAB$C_STM                      = 00000004
EXE$V_CLASS_PROT                 = 00000000                FAB$C_STMCR                    = 00000006
EXE$V_DISMOUMSG                  = 00000001  G             FAB$C_STMLF                    = 00000005
EXE$V_MOUNTMSG                   = 00000000  G             FAB$C_UDF                      = 00000000
EXE$V_OPAO                       = 00000000  G             FAB$C_VAR                      = 00000002
EXE$V_REBLDSYSD                  = 00000001                FAB$C_VFC                      = 00000003
EXE$V_WRITESYSPARAMS             = 00000001                FAB$L_ALQ                      = 00000010
EXE$V_XQP_RESIDENT               = 00000000                FAB$L_CTX                      = 00000018
F$CVSINTEGER                       00000202 R      03      FAB$L_DNA                      = 00000030
F$CVTIME                           0000024D R      03      FAB$L_FNA                      = 0000002C
F$CVUINTEGER                       00000207 R      03      FAB$L_FOP                      = 00000004
F$DIRECTORY                        00000408 R      03      FAB$L_MRN                      = 00000038
F$EDIT                             00000413 R      03      FAB$L_NAM                      = 00000028
F$ELEMENT                          00000487 R      03      FAB$L_STV                      = 0000000C
F$ENVIRONMENT                      000004D3 R      03      FAB$L_XAB                      = 00000024
F$EXTRACT                          000006D9 R      03      FAB$M_CR                       = 00000002
F$FAO                              000006FC R      03      FAB$M_DEL                      = 00000004
F$FILE                             0000073C R      03      FAB$M_FTN                      = 00000001
F$GETDVI                           0000092E R      03      FAB$M_GET                      = 00000002
F$GETJPI                           00000A6F R      03      FAB$M_PRN                      = 00000004
F$GETSYI                           00000BEA R      03      FAB$M_PUT                      = 00000001
F$IDENTIFIER                       00000CD2 R      03      FAB$M_UPD                      = 00000008
F$INTEGER                          00000D47 R      03      FAB$M_UPI                      = 00000040
F$LENGTH                           00000D58 R      03      FAB$V_CBT                      = 00000015
F$LOCATE                           00000D5F R      03      FAB$V_CTG                      = 00000014
F$LOGICAL                          00000D76 R      03      FAB$V_KFO                      = 0000001E
F$MESSAGE                          00000FA9 R      03      FAB$V_PPF                      = 00000012
F$MODE                             00000FC5 R      03      FAB$V_RCK                      = 00000017
```

```
FABSV_WCK              = 00000009
FABSW_BLS              = 0000003C
FABSW_DEQ              = 00000014
FABSW_MRS              = 00000036
FACILITY                 00002255 R     02
FALSE                    00002263 R     02
FILEID                   00001742 R     03
FILE_TABLE               00001513 R     02
FORMAT_BOOLEAN           00001782 R     03
FORMAT_FILEID            00001750 R     03
FORMAT_PROT              000016D5 R     03
FORMAT_UIC               0000166F R     03
GETDVI_TABLE             00001683 R     02
GETDVI_VALUE             00000A00 R     03
GETJPI_LOGINTIM          00000B9A R     03
GETJPI_STATE             00000BB4 R     03
GETJPI_TABLE             00000100 R     02
GETJPI_UIC               00000BC3 R     03
GETJPI_VALUE             00000B3B R     03
GETSYISW               = 00000000
GETSYI_TABLE             00000480 R     02
GET_FILE_ITEM            0000083C R     03
HEX                    = 00000002
HEXSTR                 = 00000005
IDENT                    0000223D R     02
IDENT_TABLE              000021D2 R     02
IDF_B_OUTFLAGS           00000038
IDF_C_LENGTH             00000074
IDF_K_LENGTH             00000074
IDF_L_FILENAME           00000068
IDF_L_INPRABCTX          0000000C
IDF_L_LNK                00000000
IDF_L_ONCTLY             00000060
IDF_L_ONERROR            00000008
IDF_L_OUTRABCTX          00000024
IDF_L_SEARCHCTX          00000064
IDF_Q_LABEL              00000018
IDF_Q_LOCAL              00000010
IDF_T_INPDVI             0000003C
IDF_T_OUTDVI             00000028
IDF_W_FLAG               0000005E
IDF_W_INPDID             00000052
IDF_W_INPFID             0000004C
IDF_W_INPIFI             00000004
IDF_W_INPRFA             00000058
IDF_W_ONLEVEL            00000006
IDF_W_OUTIFI             00000020
IDF_W_OUTISI             00000022
INTEGER                  00002275 R     02
INTERACTIVE              0000221E R     02
JPI$_ACCOUNT           = 00000203
JPI$_APTCNT            = 0000030A
JPI$_ASTACT            = 00000300
JPI$_ASTCNT            = 0000030E
JPI$_ASTEN             = 00000301
JPI$_ASTLM             = 00000409
JPI$_AUTHPRI           = 00000418
JPI$_AUTHPRIV          = 00000412
JPI$_BIOCNT            = 0000030F
JPI$_BIOLM             = 00000310
JPI$_BUFIO             = 0000040C
JPI$_BYTCNT            = 00000311
JPI$_BYTLM             = 0000031A
JPI$_CLINAME           = 0000020A
JPI$_CPULIM            = 0000040D
JPI$_CPUTIM            = 00000407
JPI$_CREPRC_FLAGS      = 0000020C
JPI$_CURPRIV           = 00000400
JPI$_DFPFC             = 00000406
JPI$_DFWSCNT           = 00000403
JPI$_DIOCNT            = 00000312
JPI$_DIOLM             = 00000313
JPI$_DIRIO             = 0000040B
JPI$_EFCS              = 00000317
JPI$_EFCU              = 00000318
JPI$_EFWM              = 00000316
JPI$_ENQCNT            = 0000031F
JPI$_ENQLM             = 00000320
JPI$_EXCVEC            = 00000100
JPI$_FILCNT            = 00000314
JPI$_FILLM             = 0000040F
JPI$_FINALEXC          = 00000101
JPI$_FREP0VA           = 00000404
JPI$_FREP1VA           = 00000405
JPI$_FREPTECNT         = 00000415
JPI$_GPGCNT            = 0000030C
JPI$_GRP               = 00000308
JPI$_IMAGECOUNT        = 0000041A
JPI$_IMAGNAME          = 00000207
JPI$_IMAGPRIV          = 00000413
JPI$_JOBPRCCNT         = 0000031E
JPI$_JOBTYPE           = 00000323
JPI$_LOGINTIM          = 00000206
JPI$_MASTER_PID        = 00000325
JPI$_MAXDETACH         = 0000020E
JPI$_MAXJOBS           = 0000020F
JPI$_MEM               = 00000307
JPI$_MODE              = 00000322
JPI$_MSGMASK           = 00000209
JPI$_OWNER             = 00000303
JPI$_PAGEFLTS          = 0000040A
JPI$_PAGFILCNT         = 00000414
JPI$_PAGFILLOC         = 00000419
JPI$_PGFLQUOTA         = 0000040E
JPI$_PHDFLAGS          = 0000041B
JPI$_PID               = 00000319
JPI$_PPGCNT            = 0000030D
JPI$_PRCCNT            = 0000031B
JPI$_PRCLM             = 00000408
JPI$_PRCNAM            = 0000031C
JPI$_PRI               = 00000302
JPI$_PRIB              = 00000309
JPI$_PROCPRIV          = 00000204
JPI$_PROC_INDEX        = 00000324
```

```
JPI$_SHRFILLM                    = 00000210
JPI$_SITESPEC                    = 00000208
JPI$_STATE                       = 00000306
JPI$_STS                         = 00000305
JPI$_SWPFILLOC                   = 00000321
JPI$_TABLENAME                   = 0000020B
JPI$_TERMINAL                    = 0000031D
JPI$_TMBU                        = 0000030B
JPI$_TQCNT                       = 00000315
JPI$_TQLM                        = 00000410
JPI$_UAF_FLAGS                   = 0000020D
JPI$_UIC                         = 00000304
JPI$_USERNAME                    = 00000202
JPI$_VIRTPEAK                    = 00000200
JPI$_VOLUMES                     = 00000205
JPI$_WSAUTH                      = 00000401
JPI$_WSAUTHEXT                   = 00000417
JPI$_WSEXTENT                    = 00000416
JPI$_WSPEAK                      = 00000201
JPI$_WSQUOTA                     = 00000402
JPI$_WSSIZE                      = 00000411
L                                = 00000003
LIB$CVT_DTIME                      ********     X   03
LIB$CVT_TIME                       ********     X   03
LNM$M_CASE_BLIND                 = 02000000
LNM$V_CONCEALED                  = 00000008
LNM$V_CONFINE                    = 00000001
LNM$V_CRELOG                     = 00000002
LNM$V_NO_ALIAS                   = 00000000
LNM$V_TABLE                      = 00000003
LNM$V_TERMINAL                   = 00000009
LNM$_ACMODE                      = 00000006
LNM$_ATTRIBUTES                  = 00000003
LNM$_INDEX                       = 00000001
LNM$_LENGTH                      = 00000005
LNM$_MAX_INDEX                   = 00000007
LNM$_STRING                      = 00000002
LNM$_TABLE                       = 00000004
LOGATTR_TABLE                      00002029 R       02
LOGICAL_TABLE                      00001FA8 R       02
LXFUNTABLE                         00000000 R       02
LXFUNTABLEN                      = 00000100
MAX_ARGS                         = 00000010
MAX_DEPTH                          ********     X   03
MODE_TABLE                         000021F3 R       02
MONTH_NO_TABLE                     00002172 R       02
MONTH_TABLE                        00002142 R       02
NAM$B_BID                        = 00000000
NAM$B_BLN                        = 00000001
NAM$B_DEV                        = 00000039
NAM$B_DIR                        = 0000003A
NAM$B_ESL                        = 0000000B
NAM$B_ESS                        = 0000000A
NAM$B_NAME                       = 0000003B
NAM$B_NODE                       = 00000038
NAM$B_NOP                        = 00000008
NAM$B_RSL                        = 00000003
NAM$B_RSS                        = 00000002
NAM$B_TYPE                       = 0000003C
NAM$B_VER                        = 0000003D
NAM$C_BID                        = 00000002
NAM$C_BLN                        = 00000060
NAM$C_MAXRSS                     = 000000FF
NAM$L_DEV                        = 00000044
NAM$L_DIR                        = 00000048
NAM$L_ESA                        = 0000000C
NAM$L_FNB                        = 00000034
NAM$L_NAME                       = 0000004C
NAM$L_NODE                       = 00000040
NAM$L_RLF                        = 00000010
NAM$L_RSA                        = 00000004
NAM$L_TYPE                       = 00000050
NAM$L_VER                        = 00000054
NAM$L_WCC                        = 00000030
NAM$M_NOCONCEAL                  = 00000010
NAM$M_SYNCHK                     = 00000008
NAM$T_DVI                        = 00000014
NAM$V_NOCONCEAL                  = 00000004
NAM$V_SYNCHK                     = 00000003
NAM$V_WILDCARD                   = 00000008
NAM$W_DID                        = 0000002A
NAM$W_FID                        = 00000024
NARGS                            = 00000002
NETWORK                            0000222A R       02
NLAO                               0000228D R       02
NOALL                              0000225E R       02
NPARM                            = 00000000
NULL                               0000169B R       03
OTHER                              00002232 R       02
OUTLEN                           = 00000004
P1                               = 00000004
P1_ERROR                           00001618 R       03
P2                               = 0000000C
P2_ERROR                           0000161E R       03
P3                               = 00000014
P3_ERROR                           00001624 R       03
P4                               = 0000001C
P4_ERROR                           0000162A R       03
P5                               = 00000024
P5_ERROR                           00001630 R       03
P6                               = 0000002C
P6_ERROR                           00001636 R       03
PARSE_FIELDS                       00001634 R       02
PARSE_TYPES                        00001669 R       02
PIO$GT_DDSTRING                    ********     X   03
PN_ERROR                           0000163A R       03
PPD$B_NPROCS                       0000001C
PPD$C_LENGTH                       00000168
PPD$K_LENGTH                       00000168
PPD$L_INPDEV                       00000044
PPD$L_LGI                          00000014
PPD$L_LSTSTATUS                    00000018
PPD$L_OUTDEV                       00000064
PPD$L_PRC                          00000008
```

```
PPD$Q_CLIREG                00000004            PRC_L_SAVAP                 00000000
PPD$Q_CLISYMTBL             0000000C            PRC_L_SAVFP                 00000004
PPD$T_FILENAME              00000068            PRC_L_SEVERITY              00000050
PPD$T_INPDVI                00000028            PRC_L_SPWN                  000000C0
PPD$T_OUTDVI                00000048            PRC_L_STACKLM               000000A4
PPD$V_CAPTIVE             = 00000003            PRC_L_STACKPT               000000A0
PPD$W_FLAGS                 00000002            PRC_L_STATUS                00000054
PPD$W_INPCHAN               0000001E            PRC_L_STS                   00000084
PPD$W_INPDID                0000003E            PRC_L_STV                   00000088
PPD$W_INPFID                00000038            PRC_L_SYMBOL                00000060
PPD$W_INPIFI                00000020            PRC_L_TMBX                  00000074
PPD$W_INPISI                00000022            PRC_L_TRMLIST               00000010
PPD$W_OUTDID                0000005E            PRC_Q_ALLOCREG              00000020
PPD$W_OUTFID                00000058            PRC_Q_COMMAND               000000E0
PPD$W_OUTIFI                00000024            PRC_Q_FLUSHTIME             000000D0
PPD$W_OUTISI                00000026            PRC_Q_GLOBAL                00000028
PPD$W_SIZE                  00000000            PRC_Q_IMAGENAME             000000D8
PRC_B_CONTINUE              000000F3            PRC_Q_KEYPAD                00000040
PRC_B_DEFRADIX              000000AE            PRC_Q_LABEL                 00000030
PRC_B_EXMDEPMOD             000000AD            PRC_Q_LOCAL                 00000038
PRC_B_EXMDEPWID             000000AC            PRC_Q_SAVEPRIV              000000E8
PRC_B_EXONLYL               0000012D            PRC_T_OUTDVI                0000011C
PRC_B_FLAGS2                000000AF            PRC_V_CARRCNTL            = 00000000
PRC_B_IMGFLAG               00000078            PRC_V_CTRLT               = 00000014
PRC_B_OUTFLAGS              0000012C            PRC_V_CTRLY               = 00000019
PRC_B_PROMPTLEN             000000F0            PRC_V_MODE                = 00000006
PRC_C_LENGTH                00000534            PRC_V_VERIFY              = 00000007
PRC_G_COMMANDS              00000133            PRC_V_VERIMAGE            = 00000007
PRC_G_PROMPT                000000F4            PRC_W_ASTIOSB               000000C6
PRC_K_DEC                 = 00000001            PRC_W_ASTRETN               000000C8
PRC_K_HEX                 = 00000000            PRC_W_ASTSTATUS             000000C4
PRC_K_LENGTH                00000534            PRC_W_ATTMBX                0000007A
PRC_L_CURRKEY               00000048            PRC_W_FLAGS                 00000068
PRC_L_EXMDEPADR             000000A8            PRC_W_INPCHAN               00000064
PRC_L_EXTARG                00000094            PRC_W_ONLEVEL               0000006A
PRC_L_EXTBLK                0000008C            PRC_W_OUTIFI                00000114
PRC_L_EXTCOD                0000009C            PRC_W_OUTISI                00000116
PRC_L_EXTHND                00000090            PRC_W_OUTMBXCHN             000000CA
PRC_L_EXTPRM                00000098            PRC_W_OUTMBXREF             000000CE
PRC_L_IDFLNK                000000BC            PRC_W_OUTMBXSIZ             000000CC
PRC_L_IMGACTSTS             00000080            PRC_W_PMPTCTRL              000000F1
PRC_L_INDCLOCK              0000007C            PRC_W_WAITIOSB              00000066
PRC_L_INDEPTH               0000005C            PROTECTEND                  000016D5 R     03
PRC_L_INDFAB                0000001C            PROTECTMSG                  000016A2 R     03
PRC_L_INDINPRAB             00000014            PRV                       = 00000003
PRC_L_INDOUTRAB             00000018            PRV$KEYWORD                 ********   X   03
PRC_L_INPRAB                00000008            PRV$PRIVBIT                 ********   X   03
PRC_L_LASTKEY               0000004C            PRV$SETPRIV                 ********   X   03
PRC_L_LSTSTATUS             000000B0            PRV$_NOTUNQ                 ********   X   03
PRC_L_ONCTLY                000000B8            PSL$C_USER                = 00000003
PRC_L_ONERROR               0000006C            S                         = 00000006
PRC_L_OUTOFBAND             000000B4            SEVERITY                    0000224C R     02
PRC_L_OUTRAB                0000000C            SEVERITY_TABLE              00002072 R     02
PRC_L_OUTRABCTX             00000118            SGN$V_LOADCHKPRT          = 00000001  G
PRC_L_PPFLIST               00000070            SGN$V_LOADERAPAT          = 00000000  G
PRC_L_RECALLPTR             0000012F            SGN$V_LOADMTACCESS        = 00000002  G
PRC_L_RESTART               00000058            SIZ...                    = 00000001
```

```
SS$_BUFFEROVF                    ********  X    03
SS$_IVDEVNAM                     ********  X    03
SS$_NOLOGNAM                     ********  X    03
SS$_NOMOREPROC                   ********  X    03
SS$_NONEXPR                      ********  X    03
SS$_NOPRIV                       ********  X    03
SS$_NOSUCHDEV                    ********  X    03
SS$_NOSUCHNODE                   ********  X    03
SS$_NOTRAN                       ********  X    03
SS$_SUSPENDED                    ********  X    03
STATE_TABLE                      00001491 R     02
STDTIM                         = 00000006
STR                            = 00000001
STRING                           0000226E R     02
STS$K_ERROR                    = 00000002
STS$K_SEVERE                   = 00000004
STS$K_WARNING                  = 00000000
STS_MODE                         00000BCA R     03
SYI$_ACP_BASEPRIO              = 000010B5
SYI$_ACP_DATACHECK             = 000010B4
SYI$_ACP_DINDXCACHE            = 00001101
SYI$_ACP_DIRCACHE              = 000010AA
SYI$_ACP_EXTCACHE              = 000010AD
SYI$_ACP_EXTLIMIT              = 000010AE
SYI$_ACP_FIDCACHE              = 000010AC
SYI$_ACP_HDRCACHE              = 000010A9
SYI$_ACP_MAPCACHE              = 000010A8
SYI$_ACP_MAXREAD               = 000010B1
SYI$_ACP_MULTIPLE              = 00002005
SYI$_ACP_QUOCACHE              = 000010AF
SYI$_ACP_REBLDSYSD             = 00002029
SYI$_ACP_SHARE                 = 0000200C
SYI$_ACP_SWAPFLGS              = 000010B6
SYI$_ACP_SYSACC                = 000010B0
SYI$_ACP_WINDOW                = 000010B2
SYI$_ACP_WORKSET               = 000010AB
SYI$_ACP_WRITEBACK             = 000010B3
SYI$_ACP_XQP_RES               = 00002025
SYI$_ALLOCLASS                 = 000010E5
SYI$_ARCHFLAG                  = 000010DA
SYI$_AWSMIN                    = 00001038
SYI$_AWSTIME                   = 00001039
SYI$_BALSETCNT                 = 00001012
SYI$_BJOBLIM                   = 000010B9
SYI$_BOOTTIME                  = 000010BF
SYI$_BORROWLIM                 = 00001056
SYI$_BUGCHECKFATAL             = 00002004
SYI$_BUGREBOOT                 = 00002001
SYI$_CHANNELCNT                = 00001024
SYI$_CHARACTER_EMULATED        = 0000201E
SYI$_CJFLOAD                   = 00002019
SYI$_CJFSYSRUJ                 = 0000201A
SYI$_CLASS_PROT                = 0000201D
SYI$_CLISYMTBL                 = 0000105B
SYI$_CLUSTER_FSYSID            = 000010CD
SYI$_CLUSTER_FTIME             = 000010CE
SYI$_CLUSTER_MEMBER            = 000010CF
SYI$_CLUSTER_NODES             = 000010CA
SYI$_CLUSTER_QUORUM            = 000010CC
SYI$_CLUSTER_VOTES             = 000010CB
SYI$_CONCEAL_DEVICES           = 00002012
SYI$_CPU                       = 00002000
SYI$_CRDENABLE                 = 00002002
SYI$_CTLIMGLIM                 = 00001027
SYI$_CTLPAGES                  = 00001026
SYI$_DEADLOCK_WAIT             = 0000105E
SYI$_DECIMAL_EMULATED          = 0000201F
SYI$_DEFMBXBUFQUO              = 00001050
SYI$_DEFMBXMXMSG               = 00001051
SYI$_DEFMBXNUMMSG              = 00001052
SYI$_DEFPRI                    = 000010B7
SYI$_DEFQUEPRI                 = 000010E2
SYI$_DISK_QUORUM               = 000010DC
SYI$_DISMOUMSG                 = 00002015
SYI$_DLCKEXTRASTK              = 00001011
SYI$_DORMANTWAIT               = 000010F1
SYI$_DUMPBUG                   = 00002003
SYI$_D_FLOAT_EMULATED          = 00002020
SYI$_EXTRACPU                  = 0000104C
SYI$_EXUSRSTK                  = 0000101B
SYI$_FREEGOAL                  = 00001054
SYI$_FREELIM                   = 00001053
SYI$_F_FLOAT_EMULATED          = 00002021
SYI$_GBLPAGES                  = 00001007
SYI$_GBLPAGFIL                 = 00001008
SYI$_GBLSECTIONS               = 00001006
SYI$_GROWLIM                   = 00001055
SYI$_G_FLOAT_EMULATED          = 00002022
SYI$_H_FLOAT_EMULATED          = 00002023
SYI$_IJOBLIM                   = 000010B8
SYI$_IMGIOCNT                  = 00001028
SYI$_INTSTKPAGES               = 00001010
SYI$_IOTA                      = 0000103D
SYI$_IRPCOUNT                  = 00001013
SYI$_IRPCOUNTV                 = 00001014
SYI$_KFILSTCNT                 = 00001005
SYI$_LAMAPREGS                 = 00001059
SYI$_LGI_BRK_DISUSER           = 00002028
SYI$_LGI_BRK_LIM               = 000010E8
SYI$_LGI_BRK_TERM              = 00002027
SYI$_LGI_BRK_TMO               = 000010E9
SYI$_LGI_HID_TIM               = 000010EA
SYI$_LGI_PWD_TMO               = 000010EB
SYI$_LGI_RETRY_LIM             = 000010E6
SYI$_LGI_RETRY_TMO             = 000010E7
SYI$_LNMPHASHTBL               = 00001072
SYI$_LNMSHASHTBL               = 00001071
SYI$_LOADCHKPRT                = 00002017
SYI$_LOADERAPT                 = 00002016
SYI$_LOADMTACCESS              = 00002024
SYI$_LOCKDIRWT                 = 000010EF
SYI$_LOCKIDTBL                 = 0000105C
SYI$_LOCKIDTBL_MAX             = 000010C0
SYI$_LOCKRETRY                 = 00001057
```

SYI$_LONGWAIT	= 0000103E	SYI$_PFRATL	= 00001033
SYI$_LRPCOUNT	= 0000101C	SYI$_PFRATS	= 00001035
SYI$_LRPCOUNTV	= 0000101D	SYI$_PHYSICALPAGES	= 00001032
SYI$_LRPMIN	= 0000101F	SYI$_PIOPAGES	= 00001025
SYI$_LRPSIZE	= 0000101E	SYI$_PIXSCAN	= 0000100A
SYI$_MAXBUF	= 0000104F	SYI$_POOLPAGING	= 00002009
SYI$_MAXPROCESSCNT	= 00001009	SYI$_PQL_DASTLM	= 0000108E
SYI$_MAXQUEPRI	= 000010E3	SYI$_PQL_DBIOLM	= 00001090
SYI$_MAXSYSGROUP	= 0000104D	SYI$_PQL_DBYTLM	= 00001092
SYI$_MINWSCNT	= 0000100C	SYI$_PQL_DCPULM	= 00001094
SYI$_MOUNTMSG	= 00002014	SYI$_PQL_DDIOLM	= 00001096
SYI$_MPW_HILIMIT	= 0000102B	SYI$_PQL_DENQLM	= 000010A6
SYI$_MPW_LOLIMIT	= 0000102C	SYI$_PQL_DFILLM	= 00001098
SYI$_MPW_PRIO	= 0000102D	SYI$_PQL_DJTQUOTA	= 000010EC
SYI$_MPW_THRESH	= 0000102F	SYI$_PQL_DPGFLQUOTA	= 0000109A
SYI$_MPW_WAITLIMIT	= 00001030	SYI$_PQL_DPRCLM	= 0000109C
SYI$_MPW_WRTCLUSTER	= 0000102A	SYI$_PQL_DTQELM	= 0000109E
SYI$_MVTIMEOUT	= 0000104E	SYI$_PQL_DWSDEFAULT	= 000010A0
SYI$_NJOBLIM	= 000010BA	SYI$_PQL_DWSEXTENT	= 000010A4
SYI$_NOAUTOCONFIG	= 00002006	SYI$_PQL_DWSQUOTA	= 000010A2
SYI$_NOCLOCK	= 00002007	SYI$_PQL_MASTLM	= 0000108F
SYI$_NOCLUSTER	= 00002008	SYI$_PQL_MBIOLM	= 00001091
SYI$_NODENAME	= 000010D9	SYI$_PQL_MBYTLM	= 00001093
SYI$_NODE_AREA	= 0000201B	SYI$_PQL_MCPULM	= 00001095
SYI$_NODE_CSID	= 000010D0	SYI$_PQL_MDIOLM	= 00001097
SYI$_NODE_HWTYPE	= 000010D7	SYI$_PQL_MENQLM	= 000010A7
SYI$_NODE_HWVERS	= 000010D8	SYI$_PQL_MFILLM	= 00001099
SYI$_NODE_NUMBER	= 0000201C	SYI$_PQL_MJTQUOTA	= 000010ED
SYI$_NODE_QUORUM	= 000010D2	SYI$_PQL_MPGFLQUOTA	= 0000109B
SYI$_NODE_SWINCARN	= 000010D4	SYI$_PQL_MPRCLM	= 0000109D
SYI$_NODE_SWTYPE	= 000010D5	SYI$_PQL_MTQELM	= 0000109F
SYI$_NODE_SWVERS	= 000010D6	SYI$_PQL_MWSDEFAULT	= 000010A1
SYI$_NODE_SYSTEMID	= 000010D3	SYI$_PQL_MWSEXTENT	= 000010A5
SYI$_NODE_VOTES	= 000010D1	SYI$_PQL_MWSQUOTA	= 000010A3
SYI$_NPAGEDYN	= 00001016	SYI$_PRCPOLINTERVAL	= 00001069
SYI$_NPAGEVIR	= 00001017	SYI$_PROCSECTCNT	= 0000100B
SYI$_PAGEDYN	= 00001018	SYI$_QDSKINTERVAL	= 000010E4
SYI$_PAGEFILE_FREE	= 000010F4	SYI$_QDSKVOTES	= 000010F0
SYI$_PAGEFILE_PAGE	= 000010F2	SYI$_QUANTUM	= 00001029
SYI$_PAGFILCNT	= 0000100D	SYI$_QUORUM	= 000010BC
SYI$_PAGTBLPFC	= 00001003	SYI$_REALTIME_SPTS	= 0000105A
SYI$_PAMAXPORT	= 000010E0	SYI$_RECNXINTERVAL	= 000010BE
SYI$_PANOPOLL	= 000010F8	SYI$_RESALLOC	= 00002010
SYI$_PANUMPOLL	= 0000106C	SYI$_RESHASHTBL	= 0000105D
SYI$_PAPOLLINTERVAL	= 0000106D	SYI$_RJOBLIM	= 000010BB
SYI$_PAPOOLINTERVAL	= 0000106E	SYI$_RMS_DFMBC	= 00001084
SYI$_PASANITY	= 000010E1	SYI$_RMS_DFMBFHSH	= 0000108A
SYI$_PASTDGBUF	= 0000106B	SYI$_RMS_DFMBFIDX	= 00001089
SYI$_PASTIMOUT	= 0000106A	SYI$_RMS_DFMBFREL	= 00001088
SYI$_PE1	= 000010F9	SYI$_RMS_DFMBFSDK	= 00001085
SYI$_PE2	= 000010FA	SYI$_RMS_DFMBFSMT	= 00001086
SYI$_PE3	= 000010FB	SYI$_RMS_DFMBFSUR	= 00001087
SYI$_PE4	= 000010FC	SYI$_RMS_DFNBC	= 00001100
SYI$_PE5	= 000010FD	SYI$_RMS_EXTEND_SIZE	= 0000108C
SYI$_PE6	= 000010FE	SYI$_RMS_FILEPROT	= 0000108D
SYI$_PFCDEFAULT	= 00001002	SYI$_RMS_GBLBUFQUO	= 000010FF
SYI$_PFRATH	= 00001034	SYI$_RMS_PROLOGUE	= 0000108B

```
SYI$_SAVEDUMP              = 00002013
SYI$_SBIERRENABLE          = 0000200A
SYI$_SCSBUFFCNT            = 0000105F
SYI$_SCSCONNCNT            = 00001060
SYI$_SCSFLOWCUSH           = 00001064
SYI$_SCSMAXDG              = 00001062
SYI$_SCSMAXMSG             = 00001063
SYI$_SCSNODE               = 00001067
SYI$_SCSRESPCNT            = 00001061
SYI$_SCSSYSTEMID           = 00001065
SYI$_SCSSYSTEMIDH          = 00001066
SYI$_SCS_EXISTS            = 000010DB
SYI$_SETTIME               = 0000200B
SYI$_SID                   = 00001001
SYI$_SPTREQ                = 0000101A
SYI$_SRPCOUNT              = 00001020
SYI$_SRPCOUNTV             = 00001021
SYI$_SRPMIN                = 00001023
SYI$_SRPSIZE               = 00001022
SYI$_SSINHIBIT             = 00002011
SYI$_STARTUP_P1            = 000010C2
SYI$_STARTUP_P2            = 000010C3
SYI$_STARTUP_P3            = 000010C4
SYI$_STARTUP_P4            = 000010C5
SYI$_STARTUP_P5            = 000010C6
SYI$_STARTUP_P6            = 000010C7
SYI$_STARTUP_P7            = 000010C8
SYI$_STARTUP_P8            = 000010C9
SYI$_SWAPFILE_FREE         = 000010F5
SYI$_SWAPFILE_PAGE         = 000010F3
SYI$_SWPALLOCINC           = 0000103C
SYI$_SWPFAIL               = 0000103F
SYI$_SWPFILCNT             = 0000100E
SYI$_SWPOUTPGCNT           = 0000103B
SYI$_SWPRATE               = 0000103A
SYI$_SWP_PRIO              = 0000102E
SYI$_SYSMWCNT              = 0000100F
SYI$_SYSPAGING             = 0000200D
SYI$_SYSPFC                = 00001004
SYI$_TAILORED              = 000010C1
SYI$_TBSKIPWSL             = 00001031
SYI$_TIMEPROMPTWAIT        = 0000106F
SYI$_TTY_ALTALARM          = 0000107D
SYI$_TTY_ALTYPAHD          = 0000107C
SYI$_TTY_AUTOCHAR          = 000010F7
SYI$_TTY_BUF               = 00001078
SYI$_TTY_CLASSNAME         = 00001081
SYI$_TTY_DEFCHAR           = 00001079
SYI$_TTY_DEFCHAR2          = 0000107A
SYI$_TTY_DEFPORT           = 00001083
SYI$_TTY_DIALTYPE          = 00001074
SYI$_TTY_DMASIZE           = 0000107E
SYI$_TTY_OWNER             = 00001080
SYI$_TTY_PARITY            = 00001077
SYI$_TTY_PROT              = 0000107F
SYI$_TTY_RSPEED            = 00001076
SYI$_TTY_SCANDELTA         = 00001073
SYI$_TTY_SILOTIME          = 00001082
SYI$_TTY_SPEED             = 00001075
SYI$_TTY_TIMEOUT           = 000010F6
SYI$_TTY_TYPAHDSZ          = 0000107B
SYI$_UAFALTERNATE          = 0000200E
SYI$_UDABURSTRATE          = 00001070
SYI$_USER3                 = 0000104A
SYI$_USER4                 = 0000104B
SYI$_USERD1                = 00001048
SYI$_USERD2                = 00001049
SYI$_VAXCLUSTER            = 000010EE
SYI$_VERSION               = 00001000
SYI$_VIRTUALPAGECNT        = 00001019
SYI$_VMS5                  = 00001044
SYI$_VMS6                  = 00001045
SYI$_VMS7                  = 00001046
SYI$_VMS8                  = 00001047
SYI$_VMSD1                 = 00001040
SYI$_VMSD2                 = 00001041
SYI$_VMSD3                 = 00001042
SYI$_VMSD4                 = 00001043
SYI$_VOTES                 = 000010BD
SYI$_WRITABLESYS           = 0000200F
SYI$_WRITESYSPARAMS        = 00002026
SYI$_WSDEC                 = 00001037
SYI$_WSINC                 = 00001036
SYI$_WSMAX                 = 00001015
SYI$_WS_OPA0               = 0000202A
SYI$_XFMAXRATE             = 00001058
SYM_B_FLAGS                  0000000B
SYM_B_NONUNIQUE              0000000B
SYM_B_TYPE                   0000000A
SYM_K_BINARY               = 00000002
SYM_L_BL                     00000004
SYM_L_FL                     00000000
SYM_T_SYMBOL                 0000000C
SYM_W_SIZE                   00000008
SYS$ASCTIM                   ********  GX    03
SYS$ASCTOID                  ********  GX    03
SYS$CLOSE                    ********  GX    03
SYS$FAO                      ********   X    03
SYS$FAOL                     ********  GX    03
SYS$GETDVIW                  ********  GX    03
SYS$GETJPIW                  ********  GX    03
SYS$GETMSG                   ********  GX    03
SYS$GETSYIW                  ********  GX    03
SYS$IDTOASC                  ********  GX    03
SYS$OPEN                     ********  GX    03
SYS$PARSE                    ********  GX    03
SYS$SEARCH                   ********  GX    03
SYS$SETDFPROT                ********   X    03
SYS$SETPRV                   ********  GX    03
SYS$TRNLNM                   ********  GX    03
SYS$TRNLOG                   ********  GX    03
TABDEF                       0000227D R      02
TEXT                         00002238 R      02
TIME_DELIMS                  00000249 R      03
```

```
TRUE                    00002269 R      02      XAB$L_EBK               = 00000010
UIC                     0000166B R      03      XAB$Q_BDT               = 00000024
WEEKDAY_TABLE           0000218C R      02      XAB$Q_CDT               = 00000014
WRK_B_CMDOPT            FFFFFFC3                XAB$Q_EDT               = 0000001C
WRK_B_MAXPARM           FFFFFFD0                XAB$Q_RDT               = 0000000C
WRK_B_MINPARM           FFFFFFD1                XAB$W_FFB               = 00000014
WRK_B_PARMCNT           FFFFFFCE                XAB$W_GRP               = 0000000E
WRK_B_PARMSUM           FFFFFFCF                XAB$W_MBM               = 0000000C
WRK_B_RECALLCNT         FFFFFFC5                XAB$W_PRO               = 00000008
WRK_B_VALLEV            FFFFFFC4                XAB$W_PVN               = 0000000A
WRK_B_VERBTYP           FFFFFFC2                XAB$W_RVN               = 00000008
WRK_C_LENGTH            FFFFF486                XAB_INFO                  00001509 R      02
WRK_G_BUFFER            FFFFF492                _$$_                    = 000000EF
WRK_G_INPBUF            FFFFF896
WRK_G_RESULT            FFFFF9B6
WRK_K_LENGTH            FFFFF486
WRK_L_CHARPTR           FFFFF48E
WRK_L_DISALLOW          FFFFFFE6
WRK_L_ERRORRTN          FFFFF9AE
WRK_L_EXPANDPTR         FFFFF486
WRK_L_IMAGE             FFFFFFE2
WRK_L_MARKPTR           FFFFF48A
WRK_L_PAROUT            FFFFFFD2
WRK_L_PMPTADDR          FFFFF9A2
WRK_L_PROMPTRTN         FFFFF9A6
WRK_L_PROPTR            FFFFFFC6
WRK_L_QUABLK            FFFFFFCA
WRK_L_READRTN           FFFFF9AA
WRK_L_RECALLPTR         FFFFFFEA
WRK_L_RSLEND            FFFFFFB6
WRK_L_RSLNXT            FFFFFFBA
WRK_L_SAVAP             FFFFFFF8
WRK_L_SAVFP             FFFFFFFC
WRK_L_SAVSP             FFFFFFF4
WRK_L_SIGNALRTN         FFFFFFD6
WRK_L_SPECRTN           FFFFF9B2
WRK_L_TAB_VEC           FFFFFFDE
WRK_L_VERB              FFFFFFBE
WRK_V_COMMENT         = 0000000C
WRK_W_FLAGS             FFFFFFF0
WRK_W_FLAGS2            FFFFFFF2
WRK_W_IMGCHAN           FFFFFFEE
WRK_W_PMPTLEN           FFFFF99E
XAB$B_BLN             = 00000001
XAB$B_COD             = 00000000
XAB$B_NOA             = 00000008
XAB$B_NOK             = 00000009
XAB$C_ALL             = 00000014
XAB$C_ALLLEN          = 00000020
XAB$C_DAT             = 00000012
XAB$C_DATLEN          = 0000002C
XAB$C_FHC             = 0000001D
XAB$C_FHCLEN          = 0000002C
XAB$C_PRO             = 00000013
XAB$C_PROLEN          = 00000058
XAB$C_SUM             = 00000016
XAB$C_SUMLEN          = 0000000C
```

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

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (   0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           FFFFFFFC  (     0.)  01 (   1.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
DCL$ZDATA                       00002294  (  8852.)  02 (   2.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD   NOWRT NOVEC BYTE
DCL$ZCODE                       00001797  (  6039.)  03 (   3.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD   NOWRT NOVEC BYTE

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

Phase                       Page faults    CPU Time        Elapsed Time
-----                       -----------    --------        ------------
Initialization                       15    00:00:00.04     00:00:01.87
Command processing                   96    00:00:00.69     00:00:07.52
Pass 1                             2302    00:01:56.74     00:05:37.29
Symbol table sort                     0    00:00:03.35     00:00:08.09
Pass 2                              776    00:00:25.46     00:01:17.72
Symbol table output                   0    00:00:00.78     00:00:02.97
Psect synopsis output                 2    00:00:00.02     00:00:00.33
Cross-reference output                0    00:00:00.00     00:00:00.00
Assembler run totals               3191    00:02:27.09     00:07:15.79

The working set limit was 3750 pages.
632877 bytes (1237 pages) of virtual memory were used to buffer the intermediate code.
There were 120 pages of symbol table space allocated to hold 2003 non-local and 328 local symbols.
4291 source lines were read in Pass 1, producing 78 object records in Pass 2.
242 pages of virtual memory were used to define 89 macros.

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

Macro library name                           Macros defined
------------------                           --------------
_$255$DUA28:[SYSLIB]SYSBLDMLB.MLB;1                       7
_$255$DUA28:[DCL.OBJ]DCL.MLB;1                           13
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                            2
_$255$DUA28:[SYSLIB]STARLET.MLB;2                        41
TOTALS (all libraries)                                   63

5663 GETS were required to define 63 macros.

There were no errors, warnings or information messages.

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

0071 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

INQUIRE
LIS

LEXICON
LIS

KEYPAD
LIS

LOGICAL
LIS