```
PPPPPPPPPPPP     LLL                 IIIIIIIII     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPPPPPPPPPPP     LLL                 IIIIIIIII     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPPPPPPPPPPP     LLL                 IIIIIIIII     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPP         PPP  LLL                    III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                    III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                    III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                    III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                    III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                    III        RRR         RRR        TTT        LLL
PPPPPPPPPPPP     LLL                    III        RRRRRRRRRRRR           TTT        LLL
PPPPPPPPPPPP     LLL                    III        RRRRRRRRRRRR           TTT        LLL
PPPPPPPPPPPP     LLL                    III        RRRRRRRRRRRR           TTT        LLL
PPP              LLL                    III        RRR   RRR              TTT        LLL
PPP              LLL                    III        RRR   RRR              TTT        LLL
PPP              LLL                    III        RRR   RRR              TTT        LLL
PPP              LLL                    III        RRR      RRR           TTT        LLL
PPP              LLL                    III        RRR      RRR           TTT        LLL
PPP              LLL                    III        RRR      RRR           TTT        LLL
PPP              LLLLLLLLLLLLLLL     IIIIIIIII     RRR         RRR        TTT        LLLLLLLLLLLLLLL
PPP              LLLLLLLLLLLLLLL     IIIIIIIII     RRR         RRR        TTT        LLLLLLLLLLLLLLL
PPP              LLLLLLLLLLLLLLL     IIIIIIIII     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

FILEID**PLIGETBUF

```
PPPPPPPP    LL            IIIIII      GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  BBBBBBBB    UU      UU  FFFFFFFFFF
PPPPPPPP    LL            IIIIII      GGGGGGGG  EEEEEEEEEE  TTTTTTTTTT  BBBBBBBB    UU      UU  FFFFFFFFFF
PP      PP  LL              II      GG          EE              TT      BB      BB  UU      UU  FF
PP      PP  LL              II      GG          EE              TT      BB      BB  UU      UU  FF
PP      PP  LL              II      GG          EE              TT      BB      BB  UU      UU  FF
PP      PP  LL              II      GG          EE              TT      BB      BB  UU      UU  FF
PPPPPPPP    LL              II      GG          EEEEEEEE        TT      BBBBBBBB    UU      UU  FFFFFFFF
PPPPPPPP    LL              II      GG          EEEEEEEE        TT      BBBBBBBB    UU      UU  FFFFFFFF
PP          LL              II      GG  GGGGGG  EE              TT      BB      BB  UU      UU  FF
PP          LL              II      GG  GGGGGG  EE              TT      BB      BB  UU      UU  FF
PP          LL              II      GG      GG  EE              TT      BB      BB  UU      UU  FF
PP          LL              II      GG      GG  EE              TT      BB      BB  UU      UU  FF
PP          LLLLLLLLLL    IIIIII      GGGGGG    EEEEEEEEEE      TT      BBBBBBBB    UUUUUUUUUU  FF
PP          LLLLLLLLLL    IIIIII      GGGGGG    EEEEEEEEEE      TT      BBBBBBBB    UUUUUUUUUU  FF


LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
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
```

```
0000     1         .title  pli$getbuffer - pl1 runtime input buffer manipulation
0000     2         .ident  /1-002/                                         ; Edit WHM1002
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; facility:
0000    31 ;
0000    32 ;       VAX/VMS PL1 runtime library.
0000    33 ;
0000    34 ; abstract
0000    35 ;
0000    36 ;       This module contains the pl1 runtime routines for buffer manipulation
0000    37 ;       used for input stream files.
0000    38 ;
0000    39 ; author: c. spitz 4-oct-79
0000    40 ;
0000    41 ; modified:
0000    42 ;
0000    43 ;
0000    44 ;       1-002   Bill Matthews   29-September-1982
0000    45 ;
0000    46 ;               Invoke macros $defdat and rtshare instead of $defopr and share.
0000    47 ;
0000    48 ;--
0000    49
0000    50 ;
0000    51 ; external definitions
0000    52 ;
0000    53         $deffcb                         ;define phile control block
0000    54         $defstr                         ;define stream block offsets
0000    55         $defcvtind                      ;define convert indices
0000    56         $rmsdef                         ;define rms error codes
0000    57         $defrtscan                      ;define runtime scan masks
```

```
                                0000     58          $devdef                         ;define device bits in fab
                                0000     59          $fabdef                         ;define fab offsets
                                0000     60          $rabdef                         ;define rms rab offsets
                                0000     61
                                0000     62 ;
                                0000     63 ; local definitions
                                0000     64 ;
                                0000     65
                   00000013     0000     66 terminator = pli$c_blank+pli$c_comma ;stream input field terminator
                                0000     67
                                0000     68          rtshare                         ;shareable
                                0000     69
                                0000     70 ;
                                0000     71 ; local data
                                0000     72 ;
                                0000     73
                          01 00 0000     74 revbit: .byte   0,1                     ;reversed bits for b1
                       03 01 02 00 0002     75          .byte   0,2,1,3                 ;reversed bits for b2
                 07 03 05 01 06 02 04 00 0006     76          .byte   0,4,2,6,1,5,3,7         ;reversed bits for b3
           09 01 0E 06 0A 02 0C 04 08 00 000E     77          .byte   0,8,4,12,2,10,6,14,1,9  ;reversed bits for b4
                    0F 07 0B 03 0D 05 0018     78          .byte   5,13,3,11,7,15          ;
                                001E     79
                                001E     80 ;++
                                001E     81 ; local commentary
                                001E     82 ;
                                001E     83 ;       The stream input buffer manipulation routines operate on 2 entities.
                                001E     84 ;These are the buffer and the field.  The buffer contains the characters re-
                                001E     85 ;turned by an rms $get.  The field contains the characters that the conversion
                                001E     86 ;routines process. They are different to handle quoted strings, bit strings,
                                001E     87 ;and fields that cross buffer boundaries.  They are accessed by three pointers,
                                001E     88 ;the starting address, the address of the next character to be processed, and
                                001E     89 ;the address one byte past the current end of the buffer or field.  e.g.
                                001E     90 ;
                                001E     91 ;       |<----------------------buffer--data----------------->|-|
                                001E     92 ;       |this is the data in the|%|buffer. 123, 12345,  12345679|  |
                                001E     93 ;       |-----------------------|-|------------------------------|-|
                                001E     94 ;        ^                       ^                              ^
                                001E     95 ;    fcb_l_buf              fcb_l_buf_pt                   fcb_l_buf_end
                                001E     96 ;
                                001E     97 ;                |<----------------------field--data--------------------->|-|
                                001E     98 ; str_l_field:   |this is the data in the|                               |  |
                                001E     99 ;                |-----------------------|-------------------------------|-|
                                001E    100 ;                                        x                               ^
                                001E    101 ;                                    str_l_fld_pt                 str_l_fld_end
                                001E    102 ;
                                001E    103 ;       Note that str_b_field(r11) is the first character in the field,
                                001E    104 ;while fcb_l_buf(ap) is the address of the first character in the buffer.
                                001E    105 ;
                                001E    106 ;       The following properties are used throughout this module:
                                001E    107 ;1. The length of the string "this is the data in the" is:
                                001E    108 ;       fcb_l_buf_pt - fcb_l_buf for the buffer and
                                001E    109 ;       str_l_fld_pt - addr(str_l_field) for the field
                                001E    110 ;2. The length of the un-processed characters in the buffer (from the
                                001E    111 ;       character pointed to by fcb_l_buf_pt (the "%") to the end of
                                001E    112 ;       the buffer (the "9") is: fcb_l_buf_end - fcb_l_buf_pt
                                001E    113 ;3. Field overflow occurs when (str_l_fld_pt + <the length of the
                                001E    114 ;       string to move to the field>) is gequ str_l_fld_end.
```

```
001E    115 ;
001E    116 ;--
001E    117
```

```
                                001E   119 ;++
                                001E   120 ; pli$$getnlis_r6 -- get next list field from the input stream
                                001E   121 ;
                                001E   122 ; functional description:
                                001E   123 ;
                                001E   124 ; This routine gets elements from a pl1 input stream. It is called by
                                001E   125 ; the pli$getl*** routines to return the next list field in the
                                001E   126 ; stream buffer.  It returns the address and length of the character
                                001E   127 ; string or bit string that contains the value of the next field in the
                                001E   128 ; stream.  It handles the allocation and filling of the stream buffer.
                                001E   129 ;
                                001E   130 ; calling sequence:
                                001E   131 ;       jsb     pli$$getnlis_r6
                                001E   132 ;
                                001E   133 ; inputs:
                                001E   134 ;       ap = address of file control block
                                001E   135 ;       r11 = address of stream block
                                001E   136 ; outputs: - the file control block and stream block are updated.
                                001E   137 ;       r0 = address of field or 0 if the list element is to be skipped
                                001E   138 ;       r1 = length of field
                                001E   139 ;       r4 = data type of field set for src case index for pli$cvrt_cg_r3
                                001E   140 ;--
                                001E   141
                                001E   142         .enabl  lsb
                                001E   143
                                001E   144 pli$$getnlis_r6::                       ;get next list field
   18 0C AC    00    E1         001E   145         bbc     #atr_v_eof,fcb_l_attr(ap),20$ ;if end of file
   0A 0C AC    17    E1         0023   146 10$:    bbc     #atr_v_string,fcb_l_attr(ap),15$ ;if its reall end of string
50    00000000'8F    D0         0028   147         movl    #pli$_endstring,r0      ;set endstring error
                0363  31         002F   148 13$:    brw     fail                    ;and fail
50    00000000'8F    D0         0032   149 15$:    movl    #pli$_endfile,r0        ;set endfile condition
                F4    11         0039   150         brb     13$                     ;and fail
   10 AB    18 AB    9E         003B   151 20$:    movab   str_b_field(r11),str_l_fld_pt(r11) ;start at beginning of field
                                0040   152 ;
                                0040   153 ; skip blanks and tabs
                                0040   154 ;
50    18 AC    1C AC    C3      0040   155 30$:    subl3   fcb_l_buf_pt(ap), -     ;get length left in buffer
                                0046   156                 fcb_l_buf_end(ap),r0    ;
                   0C    13     0046   157         beql    40$                     ;if eql, get a new buffer
00000000'GF    1C BC    50      2B     0048   158         spanc   r0,@fcb_l_buf_pt(ap), - ;scan past blanks and tabs
                   11           0051   159                 g^pli$b_scan,#pli$c_blank ;
                   1D    12     0052   160         bneq    50$                     ;if neq, then not end of buffer, cont
   0E 0C AB    05    E5         0054   161 40$:    bbcc    #str_v_null_line,str_l_fs(r11),45$ ;if not null line, cont
      0C AC    04    CA         0059   162         bicl    #atr_m_comma_exp,fcb_l_attr(ap) ;set comma not expected
      50    18 AB    9E         005D   163         movab   str_b_field(r11),r0     ;set addr of field
            51       D4         0061   164         clrl    r1                      ;set zero length
         54    2D    D0         0063   165         movl    #cvt_k_src_char,r4      ;set char data type
               05    05         0066   166         rsb                             ;return
            02E0     30         0067   167 45$:    bsbw    pli$$get_rec            ;end of buffer, so get new record
   D1 0C AC    00    E1         006A   168         bbc     #atr_v_eof,fcb_l_attr(ap),30$ ;if not end of file, go again
               B2    11         006F   169         brb     10$                     ;signal eof
      1C AC    51    D0         0071   170 50$:    movl    r1,fcb_l_buf_pt(ap)     ;copy address of next char in buffer
                                0075   171 ;
                                0075   172 ; if the next char is a comma then see if it is expected. if it is, then
                                0075   173 ; ignore it, and skip any succeeding blanks and tabs. if it is not expected,
                                0075   174 ; we will ignore this field.
                                0075   175 ;
```

```
                1C BC    2C    91  0075   176          cmpb    #^x2c,@fcb_l_buf_pt(ap) ;is next character a comma?
                       0B      12  0079   177          bneq    60$                     ;if neq, then no
                     1C AC     D6  007B   178          incl    fcb_l_buf_pt(ap)        ;skip comma in buf
              B8 0C AC   02     E4  007E   179          bbsc    #atr_v_comma_exp,fcb_l_attr(ap),20$ ;if comma expected
                         50    D4  0083   180          clrl    r0                      ;clear address
                         05    05  0085   181          rsb                             ;return
                               0086   182 ;
                               0086   183 ; if the next char is a quote then try to parse as a string constant. double
                               0086   184 ; quotes are converted to single quotes and inserted in field. after we find
                               0086   185 ; the trailing quote, the next char must be either a valid terminator (space,
                               0086   186 ; tab, or comma) or a B or b. if it is a terminator, process as a char string.
                               0086   187 ; if it is a B or b, process as a bit string. otherwise, raise the error con-
                               0086   188 ; dition.
                               0086   189 ;
                1C BC    27    91  0086   190 60$:     cmpb    #^x27,@fcb_l_buf_pt(ap) ;is next char a quote?
                         03    13  008A   191          beql    70$                     ;if eql, then yes
                       0174    31  008C   192          brw     300$                    ;go process as unquoted string
                     1C AC     D6  008F   193 70$:     incl    fcb_l_buf_pt(ap)        ;skip the quote
          50    18 AC   1C AC    C3  0092   194          subl3   fcb_l_buf_pt(ap), -     ;get length remaining
                                     0098   195                  fcb_l_buf_end(ap),r0    ;
                         12    14  0098   196          bgtr    90$                     ;if gtr, then continue
                       02AD    30  009A   197 80$:     bsbw    pli$$get_rec            ;else get new record
             0A 0C AC    00    E1  009D   198          bbc     #atr_v_eof,fcb_l_attr(ap),90$ ;if not end of file, continue
          50   00000000'8F    D0  00A2   199          movl    #pli$_cnverr,r0         ;set error code
                       02E9    31  00A9   200          brw     fail                    ;and fail
                               00AC   201 ;
                               00AC   202 ; find next quote
                               00AC   203 ;
        1C BC    50    27    3A  00AC   204 90$:     locc    #^x27,r0,@fcb_l_buf_pt(ap) ;look for a quote
                         07    12  00B1   205          bneq    100$                    ;if neq, then we found quote
             02 0C AC    15    E1  00B3   206          bbc     #atr_v_app_comma,fcb_l_attr(ap),100$ ;if append comma
                         51    D7  00B8   207          decl    r1                      ;then don't copy the comma
                         50    DD  00BA   208 100$:    pushl   r0                      ;save length remaining in buf
                       0192    30  00BC   209          bsbw    copy_chars              ;copy intervening chars to the field
                         8E    D5  00BF   210          tstl    (sp)+                   ;did we find a quote
                         D7    13  00C1   211          beql    80$                     ;if eql, no so go again
                               00C3   212 ;
                               00C3   213 ; quote found.
                               00C3   214 ;
                     1C AC     D6  00C3   215 110$:    incl    fcb_l_buf_pt(ap)        ;skip the quote
             18 AC   1C AC     D1  00C6   216          cmpl    fcb_l_buf_pt(ap), -     ;end of buffer?
                                     00CB   217                  fcb_l_buf_end(ap)       ;
                         0B    1F  00CB   218          blssu   120$                    ;if lssu, then no
                       027A    30  00CD   219          bsbw    pli$$get_rec            ;get new record
             03 0C AC    00    E1  00D0   220          bbc     #atr_v_eof,fcb_l_attr(ap),120$ ;if not eof continue
                       014C    31  00D5   221          brw     310$                    ;treat eof as terminator
                               00D8   222 ;
                               00D8   223 ;check for double quotes
                               00D8   224 ;
                1C BC    27    91  00D8   225 120$:    cmpb    #^x27,@fcb_l_buf_pt(ap) ;is next char a quote?
                         1A    12  00DC   226          bneq    140$                    ;if neq, then no
             14 AB    10 AB    D1  00DE   227          cmpl    str_l_fld_pt(r11), -    ;space left in field?
                                     00E3   228                  str_l_fld_end(r11)      ;
                         0A    1F  00E3   229          blssu   130$                    ;if lssu, yes, continue
          50   00000000'8F    D0  00E5   230          movl    #pli$_strovfl,r0        ;set field overvflow
                       02A6    31  00EC   231          brw     fail                    ;and fail
                  10 BB    27    90  00EF   232 130$:    movb    #^x27,@str_l_fld_pt(r11) ;copy a single quote to field
```

```
                     10 AB   D6  00F3   233          incl    str_l_fld_pt(r11)       ;include quote in field
                        97   11  00F6   234          brb     70$                     ;go again
                             00F8   235 ;
                             00F8   236 ;routine to check for valid terminator. if char pointed to by fcb_l_buf_pt(ap)
                             00F8   237 ;is a blank or tab or comma, r0 is non-zero. if @fcb_l_buf_pt(ap) is a comma,
                             00F8   238 ;comma_exp is set false. else, it is set to true.
                             00F8   239 ;
                             00F8   240 ;
                             00F8   241 ;check for bit constant
                             00F8   242 ;
00000000'GF    1C BC    01   2B  00F8   243 140$:   spanc   #1,@fcb_l_buf_pt(ap), - ;is next char a blank,
                        13        0101   244                  g^pli$b_scan,#terminator ;tab or comma?
                        03   12  0102   245          bneq    150$                    ;if neq, no, look for b/B
                     011D   31  0104   246          brw     310$                    ;if neq, then we found terminator
               1C BC    42 8F   91  0107   247 150$:   cmpb    #^x42,@fcb_l_buf_pt(ap) ;next char a B?
                        11   13  010C   248          beql    170$                    ;if eql, then yes
               1C BC    62 8F   91  010E   249          cmpb    #^x62,@fcb_l_buf_pt(ap) ;next char a b?
                        0A   13  0113   250          beql    170$                    ;if eql, then yes
            50   00000000'8F   D0  0115   251 160$:   movl    #pli$_cnverr,r0         ;set invalid conversion
                      0276   31  011C   252          brw     fail                    ;and fail
                             011F   253 ;
                             011F   254 ; found a b/B.
                             011F   255 ;
                  1C AC      D6  011F   256 170$:   incl    fcb_l_buf_pt(ap)        ;skip b/B in buf
            18 AC 1C AC      D1  0122   257          cmpl    fcb_l_buf_pt(ap), -     ;buf exhausted?
                             0127   258                  fcb_l_buf_end(ap)       ;
                        08   1F  0127   259          blssu   180$                    ;if lssu, no
                     021E   30  0129   260          bsbw    pli$$get_rec            ;get a new buffer
               0C 0C AC    00   E0  012C   261          bbs     #atr_v_eof,fcb_l_attr(ap),190$ ;if eof, treat as terminator
00000000'GF    1C BC    01   2B  0131   262 180$:   spanc   #1,@fcb_l_buf_pt(ap), - ;is next char a blank,
                        13        013A   263                  g^pli$b_scan,#terminator ;tab or comma?
                        05   12  013B   264          bneq    200$                    ;if neq, no, look for 1-4
                     50 01   D0  013D   265 190$:   movl    #1,r0                   ;treat as b1
                        31   11  0140   266          brb     230$                    ;continue
                  50    1C BC   9A  0142   267 200$:   movzbl  @fcb_l_buf_pt(ap),r0     ;get next char
                     50 30   C2  0146   268          subl    #^x30,r0                ;normalize char to bin
                        CA   15  0149   269          bleq    160$                    ;if leq, then invalid char, fail
                     04 50   D1  014B   270          cmpl    r0,#4                   ;is it greater that 4?
                        C5   14  014E   271          bgtr    160$                    ;if gtr, then invalid char, fail
                        50   DD  0150   272          pushl   r0                      ;save radix
                     1C AC      D6  0152   273          incl    fcb_l_buf_pt(ap)        ;skip the 1-4
               18 AC 1C AC      D1  0155   274          cmpl    fcb_l_buf_pt(ap), -     ;buf exhausted?
                             015A   275                  fcb_l_buf_end(ap)       ;
                        08   1F  015A   276          blssu   210$                    ;if lssu, no, continue
                     01EB   30  015C   277          bsbw    pli$$get_rec            ;get new record
               0C 0C AC    00   E0  015F   278          bbs     #atr_v_eof,fcb_l_attr(ap),220$ ;if eof, treat as terminator
00000000'GF    1C BC    01   2B  0164   279 210$:   spanc   #1,@fcb_l_buf_pt(ap), - ;look for a terminator
                        13        016D   280                  g^pli$b_scan,#terminator ;
                        A5   12  016E   281          bneq    160$                    ;if neq, then no terminator, error
                        50 8ED0  0170   282 220$:   popl    r0                      ;restore radix
                             0173   283 ;
                             0173   284 ; allocate room for the bit string on the stack. convert the char string to
                             0173   285 ; a bit string, based upon the character after the b/B.
                             0173   286 ;
                             0173   287 ; local reg usage
                             0173   288 ;       r0 - radix, number of bits per char
                             0173   289 ;       r1 - pointer to next char in field
```

```
                         0173   290 ;       r2 - number of bytes to allocate for bit temp
                         0173   291 ;       r3 - number of bytes in field
                         0173   292 ;       r4 - 2**(r0-1)-1 max valid integer in field
                         0173   293 ;       r5 - current char
                         0173   294 ;       r6 - offset into bit temp
                         0173   295 ;
    00000230'EF   9F     0173   296 230$:   pushab  320$                        ;set return address
                         0179   297 pli$$chrbitn_r6::                          ;entry for getformat
      51   18 AB   9E     0179   298         movab   str_b_field(r11),r1        ;get addr of start of field
53    10 AB   51   C3     017D   299         subl3   r1,str_l_fld_pt(r11),r3    ;get length of field
   52   53   50   C5     0182   300         mull3   r0,r3,r2                   ;get number of bits required
        52   07   C0     0186   301         addl    #7,r2                      ;round up
52   52   FD 8F   78     0189   302         ashl    #-3,r2,r2                  ;get number of bytes to allocate
        5E   52   C2     018E   303         subl    r2,sp                      ;allocate temp
        FF AE42   94     0191   304         clrb    -1(sp)[r2]                 ;clear last byte
   54   01   50   78     0195   305         ashl    r0,#1,r4                   ;get the max bit value
             54   D7     0199   306         decl    r4                         ;
             56   D4     019B   307         clrl    r6                         ;start at bit offset 0
             4A   11     019D   308         brb     280$                       ;enter loop
        55   81   9A     019F   309 240$:   movzbl  (r1)+,r5                   ;pick up next char
        55   30   82     01A2   310         subb    #^x30,r5                   ;find bit equiv
             0A   18     01A5   311         bgeq    260$                       ;if geq then continue
50   00000000'8F   D0     01A7   312 250$:   movl    #pli$_cnverr,r0            ;else set invalid conversion
           01E4   31     01AE   313         brw     fail                       ;and fail
        09   55   91     01B1   314 260$:   cmpb    r5,#9                      ;is it a number?
             1C   15     01B4   315         bleq    270$                       ;if leq, then yes, cont
        10   55   91     01B6   316         cmpb    r5,#16                     ;is it between 9 and A
             EC   15     01B9   317         bleq    250$                       ;if leq, then yes, fail
        55   07   82     01BB   318         subb    #7,r5                      ;try for cap A-F
        0F   55   91     01BE   319         cmpb    r5,#15                     ;is it A-F?
             0F   15     01C1   320         bleq    270$                       ;if leq, then yes, cont
        29   55   91     01C3   321         cmpb    r5,#41                     ;is it between F and a?
             DF   15     01C6   322         bleq    250$                       ;if leq, then yes, fail
        55   20   82     01C8   323         subb    #32,r5                     ;try for a-f
        0F   55   91     01CB   324         cmpb    r5,#15                     ;is it a-f?
             02   15     01CE   325         bleq    270$                       ;if leq, then yes
             D5   11     01D0   326         brb     250$                       ;it's past f, fail
        54   55   91     01D2   327 270$:   cmpb    r5,r4                      ;is it in range?
             D0   14     01D5   328         bgtr    250$                       ;if gtr then fail
     55   FE23 CF45   9E     01D7   329         movab   <revbit-1>[r5],r5          ;get addr of table entry
          55   6544   90     01DD   330         movb    (r5)[r4],r5                ;get the reversed bits of the value
   6E   50   56   55   F0     01E1   331         insv    r5,r6,r0,(sp)              ;insert reversed bits in string
             56   50   C0     01E6   332         addl    r0,r6                      ;address next offset
               B3 53   F4     01E9   333 280$:   sobgeq  r3,240$                   ;go again
   18 AB   6E   52   28     01EC   334 290$:   movc3   r2,(sp),str_b_field(r11) ;copy temp to field
        5E   51   D0     01F1   335         movl    r1,sp                      ;clean stack
 54   00000048 8F   D0     01F4   336         movl    #cvt_k_src_abit,r4         ;set data type
      50   18 AB   9E     01FB   337         movab   str_b_field(r11),r0        ;set data address
        51   56   D0     01FF   338         movl    r6,r1                      ;set data size
             05   0202   339         rsb                                        ;return
                  0203   340 ;
                  0203   341 ; not a quoted string. find the terminator, copy intervening chars to field
                  0203   342 ;
50   18 AC   1C AC   C3   0203   343 300$:   subl3   fcb_l_buf_pt(ap), -      ;get length left in buf
                  0209   344                 fcb_l_buf_end(ap),r0       ;
00000000'GF   1C BC   50   2A   0209   345         scanc   r0,@fcb_l_buf_pt(ap), - ;look for a blank,
                   13        0212   346                 g^pli$b_scan,#terminator ;tab or comma
```

```
                     50   DD  0213   347          pushl   r0                              ;save result
                   0039   30  0215   348          bsbw    copy_chars                      ;copy chars to field
                     8E   D5  0218   349          tstl    (sp)+                           ;did we find the terminator?
                     08   12  021A   350          bneq    310$                            ;if neq, yes, all done
                   012B   30  021C   351          bsbw    pli$$get_rec                    ;get a new record
           DF 0C AC   00   E1  021F   352          bbc     #atr_v_eof,fcb_l_attr(ap),300$ ;if not eof, go again
                          0224   353 ;
                          0224   354 ; finish up. set size in r1, addr in r0, char src case index in r4, update
                          0224   355 ; column in fcb.
                          0224   356 ;
        50    18 AB   9E  0224   357 310$:   movab   str_b_field(r11),r0              ;set address
     51 10 AB    50   C3  0228   358          subl3   r0,str_l_fld_pt(r11),r1         ;set size
           54    2D   D0  022D   359          movl    #cvt_k_src_char,r4              ;set data type
        0C AC    04   C8  0230   360 320$:   bisl    #atr_m_comma_exp,fcb_l_attr(ap) ;set comma expected
        1C BC    2C   91  0234   361          cmpb    #^x2c,@fcb_l_buf_pt(ap)         ;was the terminating char a comma?
                 04   12  0238   362          bneq    330$                            ;if neq, then no
        0C AC    04   CA  023A   363          bicl    #atr_m_comma_exp,fcb_l_attr(ap) ;clear comma expected
     0D 0C AC    00   E0  023E   364 330$:   bbs     #atr_v_eof,fcb_l_attr(ap),340$ ;if eof, don't touch column
           1C AC    D6  0243   365          incl    fcb_l_buf_pt(ap)                ;skip the terminating character
  52 1C AC    14 AC   C3  0246   366          subl3   fcb_l_buf(ap), -                ;update column
                          024C   367                  fcb_l_buf_pt(ap),r2             ;
        2E AC    52   B0  024C   368          movw    r2,fcb_w_column(ap)             ;
                 05   0250   369 340$:   rsb                                     ;return
                          0251   370
                          0251   371          .dsabl  lsb
                          0251   372
                          0251   373 ;++
                          0251   374 ; copy_chars
                          0251   375 ;
                          0251   376 ; copy chars from buffer to field.
                          0251   377 ;--
                          0251   378 copy_chars:
  50    51    1C AC   C3  0251   379          subl3   fcb_l_buf_pt(ap),r1,r0  ;get length to move
  51    10 AB   50   C1  0256   380          addl3   r0,str_l_fld_pt(r11),r1 ;will this put us
        14 AB   51   D1  025B   381          cmpl    r1,str_l_fld_end(r11)   ;past end of field?
                0A   1F  025F   382          blssu   10$                     ;if lssu, then no
  50  00000000'8F   D0  0261   383          movl    #pli$_strovfl,r0        ;set field overflow
              012A   31  0268   384          brw     fail                    ;and fail
10 BB 1C BC   50   28  026B   385 10$:     movc3   r0,@fcb_l_buf_pt(ap), - ;copy to field
                          0271   386                  @str_l_fld_pt(r11)      ;
        1C AC   51   D0  0271   387          movl    r1,fcb_l_buf_pt(ap)     ;update buf pointer
        10 AB   53   D0  0275   388          movl    r3,str_l_fld_pt(r11)    ;update field pointer
                05   0279   389          rsb                             ;return
                          027A   390
                          027A   391 ;pli$$getnedi_r6
                          027A   392 ; inputs:
                          027A   393 ;        r0 - width of field to get
                          027A   394 ; outputs:
                          027A   395 ;        r0 - addr of str_b_field(r11)
                          027A   396 ;        r1 - length of field
                          027A   397 ; side effects:
                          027A   398 ;        stream block and fcb updated
                          027A   399 ;        r2-ap preserved
                          027A   400 pli$$getnedi_r6::
     0A 0C AC   00   E1  027A   401          bbc     #atr_v_eof,fcb_l_attr(ap),2$ ;if end of file
  50  00000000'8F   D0  027F   402 1$:      movl    #pli$_endfile,r0        ;get endfile code
              010C   31  0286   403          brw     fail                    ;fail
```

```
                 007C 8F    BB  0289    404 2$:     pushr   #^m<r2,r3,r4,r5,r6>      ;save regs
          10 AB    18 AB    9E  028D    405         movab   str_b_field(r11),str_l_fld_pt(r11) ;start at beginning of field
                      50    DD  0292    406         pushl   r0                        ;save requested width
                   56 50    D0  0294    407         movl    r0,r6                     ;copy width
          50       18 AC    1C AC C3    0297    408 5$:     subl3   fcb_l_buf_pt(ap), -       ;get length left in buffer
                                        029D    409         fcb_l_buf_end(ap),r0      ;
          02 0C AC    15    E1  029D    410         bbc     #atr_v_app_comma,fcb_l_attr(ap),6$ ;if append comma set
                      50    D7  02A2    411         decl    r0                        ;don't count comma
                      50    D5  02A4    412 6$:     tstl    r0                        ;set cond code
                      19    14  02A6    413         bgtr    10$                       ;if gtr, cont
          2A 0C AB    05    E4  02A8    414 7$:     bbsc    #str_v_null_line,str_l_fs(r11),155$ ;if null line, return it
                    009A    30  02AD    415 8$:     bsbw    pli$$get_rec              ;get a new record
          E2 0C AC    00    E1  02B0    416         bbc     #atr_v_eof,fcb_l_attr(ap),5$ ;if not eof, cont
          C5 0C AC    15    E1  02B5    417         bbc     #atr_v_app_comma,fcb_l_attr(ap),1$ ;if not append comma fail
                   6E 56    D1  02BA    418         cmpl    r6,(sp)                   ;was there anything copied yet?
                      C0    13  02BD    419         beql    1$                        ;if eql, no, endfile
                      16    11  02BF    420         brb     155$                      ;use what we copied
                   56 50    D1  02C1    421 10$:    cmpl    r0,r6                     ;is there enough in the buffer?
                      1B    18  02C4    422         bgeq    20$                       ;if geq, then yes
                   56 50    C2  02C6    423         subl    r0,r6                     ;get length not processed
                51    18 AC    D0  02C9    424         movl    fcb_l_buf_end(ap),r1      ;set end addr for copy
          02 0C AC    15    E1  02CD    425         bbc     #atr_v_app_comma,fcb_l_attr(ap),15$ ;if append comma set
                      51    D7  02D2    426         decl    r1                        ;don't copy comma
                    FF7A    30  02D4    427 15$:    bsbw    copy_chars                ;copy from buf to field
          D1 0C AC    15    E1  02D7    428 155$:   bbc     #atr_v_app_comma,fcb_l_attr(ap),8$ ;if not app comma, get a rec
                   6E 56    C2  02DC    429         subl    r6,(sp)                   ;get length of what we did
                      08    11  02DF    430         brb     30$                       ;cont
       51    1C AC    56    C1  02E1    431 20$:    addl3   r6,fcb_l_buf_pt(ap),r1    ;get end addr for copy
                    FF68    30  02E6    432         bsbw    copy_chars                ;copy from buf to field
             2E AC    56    A0  02E9    433 30$:    addw    r6,fcb_w_column(ap)       ;update column
             50    18 AB    9E  02ED    434         movab   str_b_field(r11),r0       ;set address of field
                   51 8ED0    02F1    435         popl    r1                        ;set length
                 007C 8F    BA  02F4    436         popr    #^m<r2,r3,r4,r5,r6>       ;restore registers
                      05    02F8    437         rsb                               ;return
                            02F9    438
                            02F9    439
                            02F9    440 ;++
                            02F9    441 ; pli$$getskip_r2 - skip records for stream input
                            02F9    442 ; pli$$getskp1_r2 - skip 1 record for stream input
                            02F9    443 ;
                            02F9    444 ; functional description:
                            02F9    445 ;
                            02F9    446 ; This routine processes the skip option and skip formats of get statements.
                            02F9    447 ; It is called by pli$$getnlis_r6 and pli$$getnedi_r6. It will read enough
                            02F9    448 ; records to satisfy the skip request and update the line number and column
                            02F9    449 ; number of the stream file's fcb.
                            02F9    450 ;
                            02F9    451 ; inputs: r2 = number of skips to process
                            02F9    452 ; outputs: none
                            02F9    453 ;--
                            02F9    454         .enabl  lsb
                            02F9    455
                            02F9    456 pli$$getskp1_r2::                         ;
                      52    01  D0  02F9    457         movl    #1,r2                     ;set to skip 1
                            02FC    458 pli$$getskip_r2::                         ;
       18 AB    00000A0D 8F  D0  02FC    459         movl    #^x0a0d,str_b_field(r11) ;set to prompt with <cr><lf>
             0096 CC    02  90  0304    460         movb    #2,<fcb_b_rab+rab$b_psz>(ap) ;set prompt size in rab
```

```
                C092 CC   18 AB   9E  0309   461          movab   str_b_field(r11),<fcb_b_rab+ - ;set prompt addr
                                    030F   462                   rab$l_pbf>(ap)          ;in rab
            66 AC   40000000 8F   C8  030F   463          bisl    #rab$m_pmt,<fcb_b_rab+rab$l_rop>(ap) ;set to prompt in rab
            0C AC   00040000 8F   CA  0317   464          bicl    #atr_m_currec,fcb_l_attr(ap) ;clear currec to do skips
                  08 0C AC    19   E1  031F   465          bbc     #atr_v_virgin,fcb_l_attr(ap),10$ ;skip first rec if virgin
                  17 0C AC    00   E0  0324   466 5$:      bbs     #atr_v_eof,fcb_l_attr(ap),20$ ;if eof, signal endfile
                           001E   30  0329   467          bsbw    pli$$get_rec            ;skip a record
                         F5 52    F4  032C   468 10$:     sobgeq  r2,5$                   ;dec count and go again
            0C AC   00040000 8F   C8  032F   469          bisl    #atr_m_currec,fcb_l_attr(ap) ;set currec to turn off skipping
            66 AC   40000000 8F   CA  0337   470          bicl    #rab$m_pmt,<fcb_b_rab+rab$l_rop>(ap) ;clear prompt in rab
                                    05  033F   471          rsb                             ;return
               50   00000000'8F   D0  0340   472 20$:     movl    #pli$_endfile,r0        ;set endfile condition
                           004B   31  0347   473          brw     fail                    ;and fail
                                  034A   474          .dsabl  lsb
                                  034A   475
                                  034A   476 ;++
                                  034A   477 ; pli$$get_rec
                                  034A   478 ;
                                  034A   479 ; functional description:
                                  034A   480 ;
                                  034A   481 ; This routine reads records from a stream file. It will append a blank
                                  034A   482 ; to each record if atr_m_app_comma is set in fcb_l_attr. null records are
                                  034A   483 ; are ignored unless append comma is 1 or atr_m_currec is 0. if atr_m_currec
                                  034A   484 ; is 0, we are processing a skip function so null records count. if the
                                  034A   485 ; end of file is encountered, atr_m_eof is set, and a null record is
                                  034A   486 ; returned. if atr_m_eof is set when this routine is called, the endfile
                                  034A   487 ; condition ; is raised.
                                  034A   488 ;
                                  034A   489 ; inputs:       atr_m_currec = 0 to process a skip.
                                  034A   490 ;                atr_m_app_comma = 1 will append a blank unless we are
                                  034A   491 ;                      parsing a quoted string
                                  034A   492 ; outputs:      r0 = number of chars in record
                                  034A   493 ;                atr_m_virgin = 0 (file has been touched)
                                  034A   494 ;                fcb_w_column is set to 1
                                  034A   495 ;                the buffer pointed to by fcb_l_buf gets the next non-null
                                  034A   496 ;                      record.
                                  034A   497 ;--
                                  034A   498          .enabl  lsb
                                  034A   499
                                  034A   500 pli$$get_rec::
                  0C AB    20   CA  034A   501          bicl    #str_m_null_line,str_l_fs(r11) ;assume not a null line
               13 0C AC    17   E1  034E   502          bbc     #atr_v_string,fcb_l_attr(ap),10$ ;if string
                        04 AB   D5  0353   503          tstl    str_l_fp(r11)           ;is this get edit?
                           05   12  0356   504          bneq    1$                      ;if neq, yes, cont
                  0C AC    01   C8  0358   505          bisl    #atr_m_eof,fcb_l_attr(ap) ;set end of file
                                  035C   506          rsb                             ;return
            50   00000000'8F   D0  035D   507 1$:       movl    #pli$_endstring,r0      ;set endstring condition
                           2F   11  0364   508          brb     fail                    ;and fail
            50   00000000'8F   D0  0366   509 10$:      movl    #pli$_endfile,r0        ;assume eof
               23 0C AC    00   E0  036D   510          bbs     #atr_v_eof,fcb_l_attr(ap),fail ;if eof already, fail
                                  0372   511 11$:      $get    fcb_b_rab(ap)           ;get the record
                        3B 50   E8  037C   512          blbs    r0,30$                  ;if lbs, cont
            0001827A 8F   50   D1  037F   513 15$:      cmpl    r0,#rms$_eof            ;end of file encountered?
                           06   12  0386   514          bneq    20$                     ;if neq, then no
                  0C AC    01   C8  0388   515          bisl    #atr_m_eof,fcb_l_attr(ap) ;set end of file in fcb
                           51   11  038C   516          brb     40$                     ;continue
            50   00000000'8F   D0  038E   517 20$:      movl    #pli$_rmsr,r0           ;set rms rab error
```

```
                5C   DD  0395   518 fail:   pushl   ap                          ;set fcb addr
 00000000'8F    50   D1  0397   519          cmpl    r0,#pli$_endfile            ;endfile condition?
                10   13  039E   520          beql    25$                         ;if eql, yes, cont
                5C   DD  03A0   521          pushl   r0                          ;set error code
      00000000'8F   DD  03A2   522          pushl   #pli$_error                 ;set error condition
 00000000'GF    03   FB  03A8   523 23$:     calls   #3,g^pli$io_error           ;signal error
                04   03AF   524          ret                                 ;return
                00   DD  03B0   525 25$:     pushl   #0                          ;set no error code
      00000000'8F   DD  03B2   526          pushl   #pli$_endfile               ;set endfile condtion
                EE   11  03B8   527          brb     23$                         ;signal endfile
        50    0084 CC  3C  03BA   528 30$:     movzwl  <fcb_b_rab+rab$w_rsz>(ap),r0 ;get size of record
                04   12  03BF   529          bneq    35$                         ;if neq, cont
          0C AB   20   C8  03C1   530          bisl    #str_m_null_line,str_l_fs(r11) ;indicate null line
    14 AC    008A CC   D0  03C5   531 35$:     movl    <fcb_b_rab+rab$l_rbf>(ap),fcb_l_buf(ap) ;set buffer address
 18 AC   14 AC   50   C1  03CB   532          addl3   r0,fcb_l_buf(ap), -         ;set end of buffer
                      03D1   533                  fcb_l_buf_end(ap)           ;
      09 0C AC   15   E1  03D1   534          bbc     #atr_v_app_comma,fcb_l_attr(ap),40$ ;if append comma set
         18 BC   20   90  03D6   535          movb    #^x20,@fcb_l_buf_end(ap)    ;then append a comma
            18 AC   D6  03DA   536          incl    fcb_l_buf_end(ap)           ;update end of buffer
                50   D6  03DD   537          incl    r0                          ;update number of chars in rec
         2E AC   01   B0  03DF   538 40$:     movw    #1,fcb_w_column(ap)         ;set column to 1
      1C AC   14 AC   D0  03E3   539          movl    fcb_l_buf(ap),fcb_l_buf_pt(ap) ;update buf pointer
      18 AC   14 AC   D1  03E8   540          cmpl    fcb_l_buf(ap),fcb_l_buf_end(ap) ;null record?
                0D   12  03ED   541          bneq    60$                         ;if neq, then no
      08 0C AC   12   E1  03EF   542          bbc     #atr_v_currec,fcb_l_attr(ap),60$ ;if currec set (not skipping)
      03 0C AC   00   E0  03F4   543          bbs     #atr_v_eof,fcb_l_attr(ap),60$ ;if eof, return
              FF4E   31  03F9   544          brw     pli$$get_rec                ;then go again
0C AC   02000000 8F   CA  03FC   545 60$:     bicl    #atr_m_virgin,fcb_l_attr(ap) ;file has been touched
                05   0404   546          rsb                                 ;return
                     0405   547
                     0405   548          .dsabl  lsb
                     0405   549          .end
```

```
$$.TMP1                 =  00000001
$$.TMP2                 =  000000AC
ATR_M_COMMA_EXP         =  00000004
ATR_M_CURREC            =  00040000
ATR_M_EOF               =  00000001
ATR_M_VIRGIN            =  02000000
ATR_V_APP_COMMA         =  00000015
ATR_V_COMMA_EXP         =  00000002
ATR_V_CURREC            =  00000012
ATR_V_EOF               =  00000000
ATR_V_STRING            =  00000017
ATR_V_VIRGIN            =  00000019
COPY_CHARS                 00000251 R      02
CVT_K_SRC_ABIT          =  00000048
CVT_K_SRC_CHAR          =  0000002D
FAIL                       00000395 R      02
FCB_B_ENVIR                000001C2
FCB_B_ESA                  0000012E
FCB_B_EXTRA                0000003D
FCB_B_FAB                  000000A6
FCB_B_IDENT                00000040
FCB_B_IDENT_NAM            00000042
FCB_B_NAM                  000000F6
FCB_B_NUMKCBS              0000003C
FCB_B_RAB                  00000062
FCB_C_LEN                  000001C2
FCB_C_STRLEN               00000034
FCB_L_ATTR                 0000000C
FCB_L_BUF                  00000014
FCB_L_BUF_END              00000018
FCB_L_BUF_PT               0000001C
FCB_L_CNDADDR              000001B2
FCB_L_CONDIT               000001AE
FCB_L_DTTR                 00000010
FCB_L_ERROR                00000008
FCB_L_KCB                  00000038
FCB_L_NEXT                 00000000
FCB_L_PREVIOUS             00000004
FCB_L_PRN                  00000034
FCB_Q_RFA                  00000020
FCB_W_COLUMN               0000002E
FCB_W_IDENT_LEN            00000040
FCB_W_LINE                 00000030
FCB_W_LINESIZE             0000002A
FCB_W_PAGE                 00000032
FCB_W_PAGESIZE             0000002C
FCB_W_REVISION             00000028
PLI$$CHRBITN_R6            00000179 RG     02
PLI$$GETNEDI_R6            0000027A RG     02
PLI$$GETNLIS_R6            0000001E RG     02
PLI$$GETSKIP_R2            000002FC RG     02
PLI$$GETSKP1_R2            000002F9 RG     02
PLI$$GET_REC               0000034A RG     02
PLI$B_SCAN                 ********   X    02
PLI$C_BLANK             =  00000011
PLI$C_COMMA             =  00000002
PLI$IO_ERROR               ********   X    02
PLI$_CNVERR                ********   X    02
PLI$_ENDFILE               ********   X    02
PLI$_ENDSTRING             ********   X    02
PLI$_ERROR                 ********   X    02
PLI$_RMSR                  ********   X    02
PLI$_STROVFL               ********   X    02
RAB$B_PSZ               =  00000034
RAB$L_PBF               =  00000030
RAB$L_RBF               =  00000028
RAB$L_ROP               =  00000004
RAB$M_PMT               =  40000000
RAB$W_RSZ               =  00000022
REVBIT                     00000000 R      02
RMS$_EOF                =  0001827A
SIZ...                  =  00000001
STR_B_FIELD                00000018
STR_C_LEN                  00000C08
STR_L_FLD_END              00000014
STR_L_FLD_PT               00000010
STR_L_FP                   00000004
STR_L_FS                   0000000C
STR_L_PARENT               00000008
STR_L_SP                   00000000
STR_L_STACK                00000C04
STR_L_STACK_END            00000408
STR_M_NULL_LINE         =  00000020
STR_V_NULL_LINE         =  00000005
SYS$GET                    ********  GX    02
TERMINATOR              =  00000013
```

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	00000C08 (3080.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_PLI$CODE	00000405 (1029.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.05	00:00:00.37
Command processing	75	00:00:00.53	00:00:02.89
Pass 1	227	00:00:09.37	00:00:19.41
Symbol table sort	0	00:00:00.97	00:00:02.13
Pass 2	109	00:00:02.09	00:00:04.27
Symbol table output	9	00:00:00.08	00:00:00.18
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	431	00:00:13.12	00:00:29.28

The working set limit was 1200 pages.
51821 bytes (102 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 783 non-local and 63 local symbols.
549 source lines were read in Pass 1, producing 12 object records in Pass 2.
22 pages of virtual memory were used to define 20 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1	5
_255DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	16

880 GETS were required to define 16 macros.

There were no errors, warnings or information messages.

```
MACRO/ENABLE=SUPPRESSION/DISABLE=TRACEBACK/LIS=LIS$:PLIGETBUF/OBJ=OBJ$:PLIGETBUF MSRC$:PLIGETBUF/UPDATE=(ENH$:PLIGETBUF)+LIB$:PLIRTM
```

PLIFORMAT
LIS

PLIGETBUF
LIS

PLIMSGTXT
LIS

PLIGETEDI
LIS

PLIHEEP
LIS

PLIPUTFIL
LIS

PLIRMSBIS
LIS

PLIRECOPT
LIS

PLIOPEN
LIS

PLIREAD
LIS

PLIPROTEC
LIS

PLIREWRIT
LIS

PLIGETLIS
LIS

PLIPUTEDI
LIS

PLIPKDIVL
LIS

PLIPUTLIS
LIS

PLIMSGPTR
LIS

PLIPKDIVS
LIS

PLIPUTBUF
LIS

PLIGETFIL
LIS