```
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**PLIFORMAT   B 1   PLI
1-(

```
PPPPPPPP    LL            IIIIII    FFFFFFFFFF    000000    RRRRRRRR    MM      MM      AAAAAA    TTTTTTTTTT
PPPPPPPP    LL            IIIIII    FFFFFFFFFF    000000    RRRRRRRR    MM      MM      AAAAAA    TTTTTTTTTT
PP      PP  LL              II      FF          00      00  RR      RR  MMMM  MMMM    AA      AA      TT
PP      PP  LL              II      FF          00      00  RR      RR  MMMM  MMMM    AA      AA      TT
PP      PP  LL              II      FF          00      00  RR      RR  MM  MM  MM    AA      AA      TT
PP      PP  LL              II      FF          00      00  RR      RR  MM  MM  MM    AA      AA      TT
PPPPPPPP    LL              II      FFFFFFFF    00      00  RRRRRRRR    MM      MM    AA      AA      TT
PPPPPPPP    LL              II      FFFFFFFF    00      00  RRRRRRRR    MM      MM    AA      AA      TT
PP          LL              II      FF          00      00  RR  RR      MM      MM    AAAAAAAAAA      TT
PP          LL              II      FF          00      00  RR  RR      MM      MM    AAAAAAAAAA      TT
PP          LL              II      FF          00      00  RR    RR    MM      MM    AA      AA      TT          ....
PP          LL              II      FF          00      00  RR    RR    MM      MM    AA      AA      TT          ....
PP          LLLLLLLLLL    IIIIII    FF            000000    RR      RR  MM      MM    AA      AA      TT          ....
PP          LLLLLLLLLL    IIIIII    FF            000000    RR      RR  MM      MM    AA      AA      TT          ....

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
```

```
0000          .title  pli$format
0000          .ident  /1-006/                                                 ; Edit CGN1006
0000                                                                          ; Edit CGN1005
0000                                                                          ; Edit CGN1004
0000                                                                          ; Edit WHM1003
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
0000    ;++
0000    ; iacility:
0000    ;
0000    ;       VAX/VMS PL1 runtime library
0000    ;
0000    ; abstract:
0000    ;
0000    ;       This module contains the pl1 runtime routines for getting the next
0000    ;       format item in a format list.
0000    ;
0000    ;
0000    ; author: c. spitz 28-nov-79
0000    ;
0000    ; modified:
0000    ;
0000    ;       1-002   Chip Nylander   7-September-1982
0000    ;
0000    ;               Modified GETCOL to conform to ANSI X3.53 page 259
0000    ;               step 1.2.3.2.3: if a column request cannot be satisfied for
0000    ;               any reason, do an implicit getskip;  if the column request can
0000    ;               now be satisfied, perform the column positioning, otherwise
0000    ;               do nothing.
0000    ;
0000    ;
0000    ;       1-003   Bill Matthews   29-September-1982
0000    ;
```

```
             0000  ;                Invoke macros $defdat and rtshare instead of $defopr and share.
             0000  ;
             0000  ;        1-004   Chip Nylander   03-Februaruy-1983
             0000  ;
             0000  ;                Save the parent frame pointer in R1 instead of R0 when calling
             0000  ;                an expression routine.
             0000  ;
             0000  ;        1-005   Chip Nylander   23-February-1983
             0000  ;
             0000  ;                Make fixed-point edited output of floating binary values
             0000  ;                round instead of truncate, per the ANSI Standard and our
             0000  ;                own published documentation.
             0000  ;
             0000  ;        1-006   Chip Nylander   08-August-1983
             0000  ;
             0000  ;                Solve problem with uplevel references to automatic variables
             0000  ;                in remote formats by using the parent pointer of the frame
             0000  ;                containing the remote format.  Use the parent pointer instead
             0000  ;                of -4(fp) for all vfe calls.
             0000  ;
             0000  ;--
             0000
             0000  ;
             0000  ; external definitions
             0000  ;
             0000          $defstr                         ;define stream block offsets
             0000          $defdat                         ;define data types
             0000          $defcvtind                      ;define convert indices
             0000          $deffcb                         ;define file control block
             0000          $defpic                         ;define picture node offsets
             0000          $sfdef                          ;define stack offsets
             0000
             0000  ;
             0000  ; local data
             0000  ;
             0000
             0000          rtshare                         ;sharable
             0000
             0000  bformattab:                             ;table of chars for B-radix conversion
      31 30  0000          .byte   ^a\0\,^a\1\             ;entries for B1
33 31 32 30  0002          .byte   ^a\0\,^a\2\,^a\1\,^a\3\ ;entries for B2
36 32 34 30  0006          .byte   ^a\0\,^a\4\,^a\2\,^a\6\ ;entries for B3
37 33 35 31  000A          .byte   ^a\1\,^a\5\,^a\3\,^a\7\ ;
43 34 38 30  000E          .byte   ^a\0\,^a\8\,^a\4\,^a\C\ ;entries for B4
45 36 41 32  0012          .byte   ^a\2\,^a\A\,^a\6\,^a\E\ ;
44 35 39 31  0016          .byte   ^a\1\,^a\9\,^a\5\,^a\D\ ;
46 37 42 33  001A          .byte   ^a\3\,^a\B\,^a\7\,^a\F\ ;
             001E
             001E  ;++
             001E  ; pli$$getfmt_r6
             001E  ;
             001E  ; functional description:
             001E  ;       control formats are processed and the next item is transmitted from the
             001E  ;       file buffer via edit directed input. for data formats, the general
             001E  ;       flow is: the compiled code jsb's to pli$gete**** routine. that
             001E  ;       routine saves the source address and precision and jsb's to this
             001E  ;       routine to get the next input item. this routine processes interceding
```

```
                        001E  ;       control formats until a data format is encountered. the data format
                        001E  ;       evaluates its parameters and gets the proper number of characters by
                        001E  ;       jsb'ing to pli$$getnedi. pli$$getnedi returns a character string
                        001E  ;       in the field area of the current format. the data format routine then
                        001E  ;       converts this character string to the a temporary, whose data type is
                        001E  ;       based on the format. it returns with the address, precision and data
                        001E  ;       type of this temporary. the pli$gete**** routine then restores the
                        001E  ;       target information and calls pli$cvrt_cg_r3 to finish processing.
                        001E  ;       note that the common control formats for input and output are located
                        001E  ;       in this section. all output control formats MUST PRESERVE R5, which is
                        001E  ;       used to store the offset of unaligned bit sources.
                        001E  ;
                        001E  ; inputs:
                        001E  ;       r11 - address of stream block
                        001E  ;       ap - address of fcb
                        001E  ; outputs:
                        001E  ;       r0 - address of field in stream block
                        001E  ;       r1 - precision / scale of temp in stream block
                        001E  ;       r4 - case index of temp as the source to any
                        001E  ; side effects:
                        001E  ;       r0-r6 are destroyed
                        001E  ;--
                        001E
                        001E  pli$$getfmt_r6::
  50   04 BB     90     001E          movb    @str_l_fp(r11),r0       ;get format type
       04 AB     D6     0022          incl    str_l_fp(r11)           ;update format pointer
                        0025          case    type=b,r0,limit=#1,< -  ;case on format type
                        0025                  getbiter, -             ;1 byte constant iteration
                        0025                  getwiter, -             ;2 word constant iteration
                        0025                  getliter, -             ;3 long constant iteration
                        0025                  invfrm, -               ;4 pc relative iter (invalid)
                        0025                  getexpriter, -          ;5 expression iteration (Version 1)
                        0025                  geteof, -               ;6 end of format
                        0025                  getexpriter_v2, -       ;7 expression iteration (Version 2)
                        0025                  invfrm, -               ;8 invalid format
                        0025                  invfrm, -               ;9 invalid format
                        0025                  geta, -                 ;10 alphanumeric format
                        0025                  getb1, -                ;11 bit (1) format
                        0025                  getb1, -                ;12 bit 1 format
                        0025                  getb2, -                ;13 bit 2 format
                        0025                  getb3, -                ;14 bit 3 format
                        0025                  getb4, -                ;15 bit 4 format
                        0025                  getcol, -               ;16 column format
                        0025                  getcol, -               ;17 column format
                        0025                  gete, -                 ;18 exp format
                        0025                  getf, -                 ;19 fixed format
                        0025                  invfrm, -               ;20 line format invalid for get
                        0025                  getp, -                 ;21 picture format
                        0025                  invfrm, -               ;22 page format invalid for get
                        0025                  getr, -                 ;23 remote format (PL/I version 1)
                        0025                  getskip, -              ;24 skip format
                        0025                  invfrm, -               ;25 tab format invalid for get
                        0025                  getx, -                 ;26 blank format
                        0025                  invfrm, -               ;27 left paren (no longer used)
                        0025                  getrparen, -            ;28 right paren
                        0025                  getr_v2>                ;29 remote format (PL/I version 2)
                        0063
```

```
        0136   31  0063                  brw     invfrm                  ;none of the above, invalid format
                   0066
                   0066  ; process an iteration factor. the iteration factor is stored on the format
                   0066  ; stack as a count and the address of its first item. if the iteration factor
                   0066  ; is less than or equal to 0, we will skip the format item(s) between the
                   0066  ; iteration and its matching right paren.
                   0066
                   0066  getbiter:                                       ;byte constant iteration
        B5 AF      9F  0066              pushab  pli$$getfmt_r6          ;set return addr
           06      11  0069              brb     biter                   ;cont in common
                   006B  putbiter:                                       ;
   0000053C'EF     9F  006B              pushab  pli$$putfmt_r6          ;set return addr
   51    04 BB     98  0071  biter:      cvtbl   @str_l_fp(r11),r1       ;get iteration count
         04 AB     D6  0075              incl    str_l_fp(r11)           ;update format pointer
            5E     11  0078              brb     getitercom              ;cont in common
                   007A
                   007A  getwiter:                                       ;word constant iteration
        A1 AF      9F  007A              pushab  pli$$getfmt_r6          ;set return addr
           06      11  007D              brb     witer                   ;cont in common
                   007F  putwiter:                                       ;
   0000053C'EF     9F  007F              pushab  pli$$putfmt_r6          ;set return addr
   51    04 BB     32  0085  witer:      cvtwl   @str_l_fp(r11),r1       ;get iteration count
   04 AB    02     C0  0089              addl    #2,str_l_fp(r11)        ;update format pointer
            49     11  008D              brb     getitercom              ;cont in common
                   008F
                   008F  getliter:                                       ;long constant iteration
        8C AF      9F  008F              pushab  pli$$getfmt_r6          ;set return addr
           06      11  0092              brb     liter                   ;cont in common
                   0094  putliter:                                       ;
   0000053C'EF     9F  0094              pushab  pli$$putfmt_r6          ;set return addr
   51    04 BB     D0  009A  liter:      movl    @str_l_fp(r11),r1       ;get iteration count
   04 AB    04     C0  009E              addl    #4,str_l_fp(r11)        ;update format pointer
            34     11  00A2              brb     getitercom              ;cont in common
                   00A4
                   00A4  getexpriter_v2:                                 ;expression iteration (Version 2)
      FF76 CF      9F  00A4              pushab  pli$$getfmt_r6          ;set return addr
           06      11  00A8              brb     exiter_v2               ;cont in common
                   00AA  putexpriter_v2:                                 ;
   0000053C'EF     9F  00AA              pushab  pli$$putfmt_r6          ;set return addr
                   00B0  exiter_v2:
   51    08 AB     D0  00B0              movl    str_l_parent(r11),r1    ;get parent frame pointer
            10     11  00B4              brb     exiter_common           ;join common code
                   00B6
                   00B6  getexpriter:                                    ;expression iteration
      FF64 CF      9F  00B6              pushab  pli$$getfmt_r6          ;set return addr
           06      11  00BA              brb     exiter                  ;cont in common
                   00BC  putexpriter:                                    ;
   0000053C'EF     9F  00BC              pushab  pli$$putfmt_r6          ;set return addr
   51    FC AD     D0  00C2  exiter:     movl    -4(fp),r1               ;get parent frame pointer
                   00C6  exiter_common:
   50    04 BB     D0  00C6              movl    @str_l_fp(r11),r0       ;get rel addr
   04 AB    04     C0  00CA              addl    #4,str_l_fp(r11)        ;update format pointer
   50    04 AB     C0  00CE              addl    str_l_fp(r11),r0        ;get absolute addr
      60    00     FB  00D2              calls   #0,(r0)                 ;call the routine
      51    50     D0  00D5              movl    r0,r1                   ;set iteration factor
                   00D8
                   00D8  getitercom:                                     ;process iteration factor
```

```
              52   6B   D0  00D8           movl    str_l_sp(r11),r2           ;get format stack pointer
          50  0410 CB   9E  00DB           movab   <str_l_stack_end+8>(r11),r0 ;get last place for an iteration
              52   50   D1  00E0           cmpl    r0,r2                      ;is there room for another iteration?
                   0A   1B  00E3           blequ   10$                        ;if lequ, yes continue
       50  00000000'8F  D0  00E5           movl    #pli$_formatovfl,r0        ;set format stack overflow
                 0437   31  00EC           brw     fail                       ;and fail
          72    04 AB   D0  00EF   10$:    movl    str_l_fp(r11),-(r2)        ;push fp on stack
              72   51   D0  00F3           movl    r1,-(r2)                   ;push iter count on stack
              6B   52   D0  00F6           movl    r2,str_l_sp(r11)           ;store stack pointer
              4A   62   F4  00F9           sobgeq  (r2),30$                   ;do an iteration
                            00FC   ; the format iteration is < 0, so we must skip all format items until the
                            00FC   ; matching right paren is found.
                   62   D4  00FC           clrl    (r2)                       ;skip this iteration, clear paren count
          50    04 BB   90  00FE   20$:    movb    @str_l_fp(r11),r0          ;get next format
                04 AB   D6  0102           incl    str_l_fp(r11)              ;increment format pointer
                            0105           case    type=B,r0,limit=#1,< -     ;case on format type
                            0105                   70$, -                     ;1 byte iter
                            0105                   80$, -                     ;2 word iter
                            0105                   90$, -                     ;3 long iter
                            0105                   invfrm, -                  ;4 pc rel cons
                            0105                   90$, -                     ;5 expression iter (Version 1)
                            0105                   invfrm, -                  ;6 end of format (not expected)
                            0105                   90$, -                     ;7 expression iter (Version 2)
                            0105                   invfrm, -                  ;8 unused
                            0105                   invfrm, -                  ;9 unused
                            0105                   50$, -                     ;10 a
                            0105                   50$, -                     ;11 b1
                            0105                   50$, -                     ;12 b1
                            0105                   50$, -                     ;13 b2
                            0105                   50$, -                     ;14 b3
                            0105                   50$, -                     ;15 b4
                            0105                   50$, -                     ;16 col
                            0105                   50$, -                     ;17 col
                            0105                   40$, -                     ;18 e
                            0105                   40$, -                     ;19 f
                            0105                   50$, -                     ;20 lin
                            0105                   50$, -                     ;21 pic
                            0105                   20$, -                     ;22 page
                            0105                   50$, -                     ;23 rem (PL/I version 1)
                            0105                   50$, -                     ;24 skip
                            0105                   50$, -                     ;25 tab
                            0105                   50$, -                     ;26 x
                            0105                   invfrm, -                  ;27 left paren
                            0105                   60$, -                     ;28 right paren
                            0105                   45$>                       ;29 rem (PL/I version 2)
                            0143
                 0056   31  0143           brw     invfrm                     ;invalid format
                        05  0146   30$:    rsb                                ;process next format item
                 0356   30  0147   40$:    bsbw    get_format_parm            ;get first parm
                 0353   30  014A   45$:    bsbw    get_format_parm            ;get second parm
                 0350   30  014D   50$:    bsbw    get_format_parm            ;get last parm
                   AC   11  0150           brb     20$                        ;go again
                   62   D7  0152   60$:    decl    (r2)                       ;decrement paren count
                   A8   18  0154           bgeq    20$                        ;if geq, then go again
              6B   08   C0  0156           addl    #8,str_l_sp(r11)           ;clean stack
                        05  0159           rsb                                ;process next format item
                04 AB   D6  015A   70$:    incl    str_l_fp(r11)              ;skip iteration
```

```
               0A   11 015D           brb     100$                      ;continue
         04 AB 02   C0 015F   80$:    addl    #2,str_l_fp(r11)          ;skip iteration
               04   11 0163           brb     100$                      ;continue
         04 AB 04   C0 0165   90$:    addl    #4,str_l_fp(r11)          ;skip iteration
               62   D6 0169   100$:   incl    (r2)                      ;increment paren count
             FF90   31 016B           brw     20$                       ;go again
                       016E
                       016E   ; end of format - if processing remote format, return to 'caller'. otherwise
                       016E   ; repeat format.
          FEAC CF   9F 016E   geteof: pushab  pli$$getfmt_r6            ;set return addr
               06   11 0172           brb     comeof                    ;cont in com
      0000053C'EF   9F 0174   puteof: pushab  pli$$putfmt_r6            ;set return addr
       50  0C04 CB  9E 017A   comeof: movab   str_l_stack(r11),r0       ;get addr of top of stack
         50    6B   D1 017F           cmpl    str_l_sp(r11),r0          ;anything on the stack?
               07   1F 0182           blssu   10$                       ;if lssu, yes, its end of remote
    04 AB  0C04 CB  D0 0184           movl    str_l_stack(r11),str_l_fp(r11) ;restart the format
               05      018A           rsb                               ;go again
     08 AB  00 BB   D0 018B   10$:    movl    @str_l_sp(r11),str_l_parent(r11) ;reset parent pointer
         6B    04   C0 0190           addl    #4,str_l_sp(r11)          ;
     04 AB  00 BB   D0 0193           movl    @str_l_sp(r11),str_l_fp(r11) ;reset format pointer
         6B    04   C0 0198           addl    #4,str_l_sp(r11)          ;clean stack
               05      019B           rsb                               ;go again
                       019C
 50   00000000'8F   D0 019C   invfrm: movl    #pli$_invformat,r0        ;set invalid format
             0380   31 01A3           brw     fail                      ;and fail
                       01A6
                       01A6   invfrmprm:
 50   00000000'8F   D0 01A6           movl    #pli$_invfmtparm,r0       ;set invalid format parameter
             0376   31 01AD           brw     fail                      ;and fail
                       01B0   invstrfmt:
 50   00000000'8F   D0 01B0           movl    #pli$_invstrfmt,r0        ;set invalid stream format
             036C   31 01B7           brw     fail                      ;and fail
                       01BA   ; a format, input. get the width, get that number of chars and return.
             02E3   30 01BA   geta:   bsbw    get_format_parm           ;get the parameter
               E7   15 01BD           bleq    invfrmprm                 ;if leq, then invalid format
         50    51   D0 01BF           movl    r1,r0                     ;set width
      00000000'GF   16 01C2           jsb     g^pli$$getnedi_r6         ;get the item
         54    2D   D0 01C8           movl    #cvt_k_src_char,r4        ;set case index
               05      01CB           rsb                               ;return
                       01CC
                       01CC   ; b format, input
                       01CC   ; set the radix factor
               01   DD 01CC   getb1:  pushl   #1                        ;push radix
               0A   11 01CE           brb     getb                      ;continue in common
               02   DD 01D0   getb2:  pushl   #2                        ;push radix
               06   11 01D2           brb     getb                      ;continue in common
               03   DD 01D4   getb3:  pushl   #3                        ;push radix
               02   11 01D6           brb     getb                      ;continue in common
               04   DD 01D8   getb4:  pushl   #4                        ;push radix
                       01DA   ; get the width and that number of characters
             02C3   30 01DA   getb:   bsbw    get_format_parm           ;get the parameter
               C7   15 01DD           bleq    invfrmprm                 ;if leq, then invalid format
         50    51   D0 01DF           movl    r1,r0                     ;set width
      00000000'GF   16 01E2           jsb     g^pli$$getnedi_r6         ;get the item
                       01E8   ; skip leading blanks. there must be at least 1 non-blank.
  18 AB    51  20   3B 01E8           skpc    #^x20,r1,str_b_field(r11) ;skip leading blanks
               0A   12 01ED           bneq    20$                       ;if neq, non-blank found
```

```
 50    00000000'8F   D0  01EF   10$:    movl    #pli$_cnverr,r0         ;set conversion error
               032D  31  01F6           brw     fail                    ;and fail
             56  50  D0  01F9   20$:    movl    r0,r6                   ;save length left
             55  51  D0  01FC           movl    r1,r5                   ;save addr of 1st non-blank
                         01FF   ; locate trailing blanks. we won't convert them
      61   50  20  3A  01FF           locc    #^x20,r0,(r1)           ;find next blank
                 09  13  0203           beql    30$                     ;if eql, not found, cont
             56  50  C2  0205           subl    r0,r6                   ;get new length for string
      61   50  20  3B  0208           skpc    #^x20,r0,(r1)           ;anything left other than blanks?
                 E1  12  020C           bneq    10$                     ;if neq, then yes, error
                         020E   ; copy the non-blank chars to the beginning of the field
  18 AB    65  56  28  020E   30$:    movc3   r6,(r5),str_b_field(r11) ;copy to beginning of field
        10 AB  53  D0  0213           movl    r3,str_l_fld_pt(r11)    ;set field pointer
                         0217   ; convert the chars to a bit string based on the radix factor
             50 8ED0     0217           popl    r0                      ;restore radix
       00000000'GF   16  021A           jsb     g^pli$$chrbitn_r6       ;convert to a bit string
 54    00000048 8F   D0  0220           movl    #cvt_k_src_abit,r4      ;set case index
                 05      0227           rsb                             ;return
                         0228
                         0228   ; column format, input
                         0228   ; if the requested column is after current column, and before the end of the
                         0228   ; line, we just position the buffer pointer to the requested place.
                         0228   ; otherwise, perform a getskip;  if the column can now be positioned as
                         0228   ; requested, then do so, otherwise give up.
                         0228
    03 0C AC    17   E1  0228   getcol: bbc     #atr_v_string,fcb_l_attr(ap),5$ ;if string i/o
               FF80  31  022D           brw     invstrfmt               ;fail with invalid string format
               026D  30  0230   5$:     bsbw    get_format_parm         ;get the parameter
                 07  14  0233           bgtr    20$                     ;if gtr, cont
                 03  13  0235           beql    10$                     ;if eql, use 1
               FF6C  31  0237           brw     invfrmprm               ;its lss, invalid format
                 51  D6  023A   10$:    incl    r1                      ;use 1 instead of 0
                 52  D4  023C   20$:    clrl    r2                      ;say that this is first time through
          2E AC  51  B1  023E           cmpw    r1,fcb_w_column(ap)     ;already past requested column?
                 1D  19  0242           blss    30$                     ;if lss, then yes
  50    51  2E AC    A3  0244   25$:    subw3   fcb_w_column(ap),r1,r0  ;get number to move
          2E AC  51  B0  0249           movw    r1,fcb_w_column(ap)     ;update column
             50  50  3C  024D           movzwl  r0,r0                   ;make it long
  53    1C AC  50  C1  0250           addl3   r0,fcb_l_buf_pt(ap),r3  ;make updated buffer pointer
          18 AC  53  D1  0255           cmpl    r3,fcb_l_buf_end(ap)    ;past end of this line?
                 06  18  0259           bgeq    30$                     ;if geq, then yes
          1C AC  53  D0  025B           movl    r3,fcb_l_buf_pt(ap)     ;update buffer pointer
                 1D  11  025F           brb     40$                     ;exit
                 52  D5  0261   30$:    tstl    r2                      ;is this first time through?
                 19  14  0263           bgtr    40$                     ;if no, give up trying
0C AC   02000000 8F   CA  0265           bicl    #atr_m_virgin,fcb_l_att,(ap) ;deflower file (so we don't skip
                         026D                                           ; first record)
             53  51  D0  026D           movl    r1,r3                   ;save requested column
       00000000'GF   16  0270           jsb     g^pli$$getskp1_r2       ;do a skip
             51  53  D0  0276           movl    r3,r1                   ;restore request
             52  01  D0  0279           movl    #1,r2                   ;say that this is second time through
                 C6  11  027C           brb     25$                     ;go try to position on new line
               FD9D  31  027E   40$:    brw     pli$$getfmt_r6          ;go again
                         0281
                         0281   ; e format, input
                         0281   ; get the parameters. w,d and s are supplied, but s is ignored.
               021C  30  0281   gete:   bsbw    get_format_parm         ;get width
```

```
            03   18  0284          bgeq    10$                     ;if geq, cont
          FF1D   31  0286          brw     invfrmprm               ;if lss, then invalid format
            03   12  0289  10$:    bneq    20$                     ;if neq, then cont
          003C   31  028B          brw     zero                    ;make result zero
       52   51   D0  028E  20$:    movl    r1,r2                   ;save width
          020C   30  0291          bsbw    get_format_parm         ;get fractional digits
            03   18  0294          bgeq    30$                     ;if geq, cont
          FF0D   31  0296          brw     invfrmprm               ;if lss, then invalid format
       54   51   D0  0299  30$:    movl    r1,r4                   ;set frac digits for pli$fchrfltd_r6
          0201   30  029C          bsbw    get_format_parm         ;get scale factor
       50   52   D0  029F          movl    r2,r0                   ;set width
                     02A2  ; get the required number of chars
   00000000'GF   16  02A2          jsb     g^pli$$getnedi_r6       ;get the field
          0838   30  02A8          bsbw    char_flt_ctx            ; get the float context
    52   18 AB   9E  02AB          movab   str_b_field(r11),r2     ; addr field as target
       7E   52   7D  02AF          movq    r2,-(sp)                ; save destination
00000000'GF 00   FB  02B2          calls   #0,g^pli$fchrfltd_r6    ; convert fractioned char to float dec
       50   8E   7D  02B9          movq    (sp)+,r0                ; use destination of cvt as src
       54   24   D0  02BC          movl    #cvt_k_src_fltd,r4      ; set case index for fltd src
            05       02BF          rsb                             ; return
                     02C0
                     02C0          .enabl lsb
                     02C0  ; f format, input
                     02C0  ; get w,d,s. s is ignored
          01DD   30  02C0  getf:   bsbw    get_format_parm         ;get width
            03   18  02C3          bgeq    10$                     ;if geq, cont
          FEDE   31  02C5          brw     invfrmprm               ;its lss, invalid format
            13   12  02C8  10$:    bneq    20$                     ;if neq, then cont
          01D3   30  02CA  zero:   bsbw    get_format_parm         ;get next parm
          01D0   30  02CD          bsbw    get_format_parm         ;get last parm
    50   18 AB   9E  02D0          movab   str_b_field(r11),r0     ;set addr of result
            60   D4  02D4          clrl    (r0)                    ;clear result
       51   1F   D0  02D6          movl    #31,r1                  ;set precision
       54   09   D0  02D9          movl    #cvt_k_src_fixb,r4      ;set case index for fixb
            05       02DC          rsb                             ;return
       52   51   D0  02DD  20$:    movl    r1,r2                   ;save width
          01BD   30  02E0          bsbw    get_format_parm         ;get fractional digits
            03   18  02E3          bgeq    30$                     ;if geq, ok
          FEBE   31  02E5          brw     invfrmprm               ;its lss, invalid format
       53   51   D0  02E8  30$:    movl    r1,r3                   ;save fractional digits
          01B2   30  02EB          bsbw    get_format_parm         ;get scale factor
       50   52   D0  02EE          movl    r2,r0                   ;set width
                     02F1  ; get the required number of chars
   00000000'GF   16  02F1          jsb     g^pli$$getnedi_r6       ;get the field
       56   50   D0  02F7          movl    r0,r6                   ;save start addr
       52   51   D0  02FA          movl    r1,r2                   ;save length read
                     02FD  ; if there is no decimal point in the input, we use the specified d to imply one
  60   51   2E   3A  02FD          locc    #^x2e,r1,(r0)           ;find the decimal point
            59   13  0301          beql    70$                     ;if eql, then use fractional digits
       55   51   D0  0303          movl    r1,r5                   ;save addr of point
                     0306  ; make sure there is nothing but trailing blanks
  61   50   20   3A  0306          locc    #^x20,r0,(r1)           ;find trailing blank
            13   13  030A          beql    40$                     ;if eql, then none
       52   50   C2  030C          subl    r0,r2                   ;get new length of field
  61   50   20   3B  030F          skpc    #^x20,r0,(r1)           ;anything left other than blanks?
            0A   13  0313          beql    40$                     ;if eql, then no, ok
50  00000000'8F  D0  0315          movl    #pli$_cnverr,r0         ;set conversion error
```

```
             0207   31  031C   457            brw     fail                     ;and fail
                        031F   458 ; pli$charfixd_r6 allows an exponent, but f format does not. we will append
                        031F   459 ; an exponent of 0, which will cause charfixd to signal error if the input
                        031F   460 ; already has an exponent.
     50   52   56   C1  031F   461 40$:   addl3   r6,r2,r0                 ;get addr of end of field
     53   50   55   C3  0323   462        subl3   r5,r0,r3                 ;get number of fractional digits
               53   D7  0327   463        decl    r3                       ;
     60 3045   8F   B0  0329   464        movw    #^x3045,(r0)             ;append 'E0', (its not allowed in f)
     51   52   02   C1  032E   465        addl3   #2,r2,r1                 ;set length for convert
          1F   52   D1  0332   466        cmpl    r2,#31                   ;length > max fixd prec?
               03   15  0335   467        bleq    50$                      ;if leq, then no, cont
          52   1F   D0  0337   468        movl    #31,r2                   ;use max prec
     53   53   08   78  033A   469 50$:   ashl    #8,r3,r3                 ;set scale of temp = # digs in frac
          53   52   88  033E   470        bisb    r2,r3                    ;put in the prec
               53   DD  0341   471        pushl   r3                       ;save prec,scale
      50   18  AB   9E  0343   472        movab   str_b_field(r11),r0      ;set addr of src
          52   50   D0  0347   473        movl    r0,r2                    ;set addr of dst
   00000C00'GF 00   FB  034A   474        calls   #0,g^pli$charfixd_r6     ;convert to fixd
             51   8ED0  0351   475        popl    r1                       ;restore prec,scale
      50   18  AB   9E  0354   476 60$:   movab   str_b_field(r11),r0      ;set addr of temp
          54   1B   D0  0358   477        movl    #cvt_k_src_fixd,r4       ;set case index
                    05  035B   478        rsb                              ;return
                        035C   479 ; there was no decimal point in the input string, so we will convert to a non-
                        035C   480 ; scaled fixd, and fix up the scale after the conversion.
      50   18  AB   9E  035C   481 70$:   movab   str_b_field(r11),r0      ;get addr of field
     60   52   20   3B  0360   482        skpc    #^x20,r2,(r0)            ;skip leading blanks
     61   50   20   3A  0364   483        locc    #^x20,r0,(r1)            ;find trailing blank
               18   13  0368   484        beql    90$                      ;if eql, no trail blanks, cont
               51   DD  036A   485        pushl   r1                       ;save start of blanks
          52   50   C2  036C   486        subl    r0,r2                    ;don't count blanks in len
     61   50   20   3B  036F   487        skpc    #^x20,r0,(r1)            ;skip trail blanks.
               0A   13  0373   488        beql    80$                      ;if eql, ok
  50    00000000'8F D0  0375   489        movl    #pli$_cnverr,r0          ;set conversion error (non blank found)
             01A7   31  037C   490        brw     fail                     ;and fail
             51   8ED0  037F   491 80$:   popl    r1                       ;get start of blanks
     61 3045   8F   B0  0382   492 90$:   movw    #^x3045,(r1)             ;append 'E0' (its not allowed in f)
     51   52   02   C1  0387   493        addl3   #2,r2,r1                 ;set len of src
          1F   52   D1  038B   494        cmpl    r2,#31                   ;length > max fixd prec?
               03   15  038E   495        bleq    100$                     ;if leq, then no, cont
          52   1F   D0  0390   496        movl    #31,r2                   ;use max prec
     54   53   08   78  0393   497 100$:  ashl    #8,r3,r4                 ;set scale = numb digs in frac
          54   52   88  0397   498        bisb    r2,r4                    ;set len
               54   DD  039A   499        pushl   r4                       ;save prec,scale
      50   18  AB   ..  039C   500        movab   str_b_field(r11),r0      ;set addr of src
          53   52   D0  03A0   501        movl    r2,r3                    ;set len of dst
          52   50   D0  03A3   502        movl    r0,r2                    ;set addr of dst
   00000000'GF 00   FB  03A6   503        calls   #0,g^pli$charfixd_r6     ;convert to fixd
             51   8ED0  03AD   504        popl    r1                       ;restore prec,scale
               A2   11  03B0   505        brb     60$                      ;continue
                        03B2   506        .dsabl  lsb
                        03B2   507
                        03B2   508 ; picture format, input
                        03B2   509 ; get the addr of the picture descriptor
             00EB   30  03B2   510 getp:  bsbw    get_format_parm          ;get the parm
               03   12  03B5   511        bneq    10$                      ;if neq, cont
             FDE2   31  03B7   512        brw     invfrm                   ;
          56   51   D0  03BA   513 10$:   movl    r1,r6                    ;save picture descr addr
```

```
                          03BD   ; get the required chars
     50   04 A1     9A    03BD           movzbl  pic$b_byte_size(r1),r0  ;set size to read
   00000000'GF      16    03C1           jsb     g^pli$$getnedi_r6       ;get the field
                          03C7   ; validate the picture. note that p format requires that the chars read be
                          03C7   ; a valid picture string. this differs from list input of a picture variable
                          03C7   ; where the input must be a valid fixed decimal number.
               50   DD    03C7           pushl   r0                      ;set addr
               51   DD    03C9           pushl   r1                      ;set length read
               56   DD    03CB           pushl   r6                      ;set picture desc addr
 00000000'GF   03   FB    03CD           calls   #3,g^pli$valid_pic      ;validate picture
           0A 50    E8    03D4           blbs    r0,20$                  ;if lbs, cont
50   00000000'8F    D0    03D7           movl    #pli$_cnverr,r0         ;set conversion error
             0145   31    03DE           brw     fail                    ;and fail
     50   18 AB     9E    03E1   20$:    movab   str_b_field(r11),r0     ;set addr
       51   56      D0    03E5           movl    r6,r1                   ;set pic desc addr
       54   00      D0    03E8           movl    #cvt_k_src_pic,r4       ;set src data type
               05         03EB           rsb                             ;return
                          03EC
                          03EC   ; version 2 remote format.  a remote format is processed by using the nesting
                          03EC   ; level difference passed as the first format param to calculate the parent
                          03EC   ; pointer of the remote format.  this calculated parent pointer is then set
                          03EC   ; info r1 for all vfe calls that occur in the remote format, and the vfes
                          03EC   ; use r1 for uplevel references to automatic variables.
          FC2E CF   9F    03EC   getr_v2:pushab  pli$$getfmt_r6          ;set return addr
               06   11    03F0           brb     comr_v2                 ;cont in common
   0000053C'EF      9F    03F2   putr_v2:pushab  pli$$putfmt_r6          ;set return addr
             00A5   30    03F8   comr_v2:bsbw    get_format_parm         ;get nesting level relative to referencer
               03   18    03FB           bgeq    10$                     ;if geq, continue
             FD9C   31    03FD           brw     invfrm                  ;else invalid format
     53   08 AB     D0    0400   10$:    movl    str_l_parent(r11),r3    ;get parent pointer of referencer
               51   D7    0404   20$:    decl    r1                      ;decrement relative nesting level
               16   19    0406           blss    remcom                  ;if lss then have correct parent pointer
     53   0C A3     D0    0408           movl    sf$l_save_fp(r3),r3     ;else get next higher parent pointer
               F6   11    040C           brb     20$                     ;and go back
                          040E
                          040E
                          040E   ; remote format. a remote format is processed by pushing the address of the
                          040E   ; next item in the original format onto the format stack. when the remote
                          040E   ; formats end of format is encountered, this address is popped, and control
                          040E   ; returns to the original format.
          FC0C CF   9F    040E   getr:   pushab  pli$$getfmt_r6          ;set return addr
               06   11    0412           brb     comr                    ;cont in common
   0000053C'EF      9F    0414   putr:   pushab  pli$$putfmt_r6          ;set return addr
     53   08 AB     D0    041A   comr:   movl    str_l_parent(r11),r3    ;pickup default parent
             007F   30    041E   remcom: bsbw    get_format_parm         ;get the remote format
               03   12    0421           bneq    10$                     ;if neq, continue
             FD76   31    0423           brw     invfrm                  ;invalid format
   50   0410 CB     9E    0426   10$:    movab   <str_l_stack_end+8>(r11),r0 ;get addr of last place for remote
       52   6B      D0    042B           movl    str_l_sp(r11),r2        ;get format stack pointer
       52   50      D1    042E           cmpl    r0,r2                   ;room for this remote?
               0A   1B    0431           blequ   20$                     ;if lequ, then yes
50   00000000'8F    D0    0433           movl    #pli$_formatovfl,r0     ;set format stack overflow
             00E9   31    043A           brw     fail                    ;and fail
     72   04 AB     D0    043D   20$:    movl    str_l_fp(r11),-(r2)     ;push addr of next item in this format
     72   08 AB     D0    0441           movl    str_l_parent(r11),-(r2) ;push parent pointer for this format
       6B   52      D0    0445           movl    r2,str_l_sp(r11)        ;store stack pointer
    04 AB   51      D0    0448           movl    r1,str_l_fp(r11)        ;set format pointer to remote format
```

```
      08 AB    53   D0  044C          movl    r3,str_l_parent(r11)    ;set parent pointer to remote format
               05       0450          rsb                             ;go with remote format
                        0451
                        0451  ; skip format, input
   03 0C AC    17   E1  0451  getskip:bbc     #atr_v_string,fcb_l_attr(ap),5$ ;if string i/o
             FD57   31  0456          brw     invstrfmt               ;fail with invalid string format
             003F   30  0459  5$:     bsbw    get_format_parm_1       ;get the number of records to skip
               03   18  045C          bgeq    10$                     ;if geq, ok
             FD45   31  045E          brw     invfrmprm               ;its leq, invalid format
          52   51   D0  0461  10$:    movl    r1,r2                   ;set number to skip
      00000000'GF   16  0464          jsb     g^pli$$getskip_r2       ;skip em
             FBB1   31  046A          brw     pli$$getfmt_r6          ;go again
                        046D
                        046D  ; x format, input
             002B   30  046D  getx:   bsbw    get_format_parm_1       ;get the number of chars to skip
               09   13  0470          beql    10$                     ;if eql, ignore it
          50   51   D0  0472          movl    r1,r0                   ;set width
      00000000'GF   16  0475          jsb     g^pli$$getnedi_r6       ;get that number of chars
             FBA0   31  047B  10$:    brw     pli$$getfmt_r6          ;go again
                        047E
                        047E  ; right paren - end of iteration. the iteration count on the format stack is
                        047E  ; decremented. if it is <= 0, we go on to the next format item. otherwise, we
                        047E  ; repeat the iterated items.
                        047E  getrparen:
          FB9C CF   9F  047E          pushab  pli$$getfmt_r6          ;set return addr
               06   11  0482          brb     comrparen               ;cont in common
                        0484  putrparen:
      0000053C'EF   9F  0484          pushab  pli$$putfmt_r6          ;set return addr
                        048A  comrparen:
          52   6B   D0  048A          movl    str_l_sp(r11),r2        ;get format sp
            05 62   F4  048D          sobgeq  (r2),10$                ;do an iteration
      6B    08 A2   9E  0490          movab   8(r2),str_l_sp(r11)     ;clean stack
               05       0494          rsb                             ;process next format item
   04 AB    04 A2   D0  0495  10$:    movl    4(r2),str_l_fp(r11)     ;restart this format
               05       049A          rsb                             ;process next format item
                        049B
                        049B  ;get_format_parm_1 - get a format parm. if the parm is missing, 1 is supplied
                        049B  ; as default.
                        049B  ; inputs:
                        049B  ;       r11 - address of stream block
                        049B  ;       ap - address of fcb
                        049B  ; outputs:
                        049B  ;       r1 - value of parameter or 1 if item missing
                        049B  ;
                        049B  get_format_parm_1:
          51   01   D0  049B          movl    #1,r1                   ;set missing parm value
               02   11  049E          brb     get_format_com          ;cont in common
                        04A0
                        04A0  ;get_format_parm - get a format parm. if the parm is missing, 0 is supplied
                        04A0  ; as default.
                        04A0  ; inputs:
                        04A0  ;       r11 - address of stream block
                        04A0  ;       ap - address of fcb
                        04A0  ; outputs:
                        04A0  ;       r1 - value of parameter or 0 if item missing
                        04A0  ;
                        04A0  get_format_parm:
```

```
         51  D4 04A0         clrl    r1                        ;set missing parm value
                04A2 get_format_com:
   0C AB  01  CA 04A2         bicl    #str_m_missing,str_l_fs(r11) ;clear missing parameter
                04A6 ; get the parameter type and case on it
   50  04 BB  90 04A6         movb    @str_l_fp(r11),r0         ;get parm type
       04 AB  D6 04AA         incl    str_l_fp(r11)             ;increment format pointer
                04AD         case    type=B,r0,< -             ;case on parameter type
                04AD                 10$, -                    ;missing
                04AD                 20$, -                    ;byte constant
                04AD                 30$, -                    ;word constant
                04AD                 40$, -                    ;long constant
                04AD                 50$, -                    ;pc relative long constant
                04AD                 60$, -                    ;pc relative long entry point (V1)
                04AD                  5$, -                    ;invalid format
                04AD                 70$>                      ;pc relative long entry point (V2)
       FCD8  31 04C1 5$:      brw     invfrm                    ;none of the above, invalid format
                04C4 ; missing
   0C AB  01  C8 04C4 10$:    bisl    #str_m_missing,str_l_fs(r11) ;set missing parameter
          51  D5 04C8         tstl    r1                        ;set default value condition code
          05     04CA         rsb                               ;return
                04CB ; byte constant
   51  04 BB  98 04CB 20$:    cvtbl   @str_l_fp(r11),r1         ;get the parm
       04 AB  D6 04CF         incl    str_l_fp(r11)             ;increment format pointer
          51  D5 04D2         tstl    r1                        ;set cond codes
          05     04D4         rsb                               ;return
                04D5 ; word constant
   51  04 BB  32 04D5 30$:    cvtwl   @str_l_fp(r11),r1         ;get the parm
   04 AB  02  C0 04D9         addl    #2,str_l_fp(r11)          ;increment format pointer
          51  D5 04DD         tstl    r1                        ;set cond codes
          05     04DF         rsb                               ;return
                04E0 ; long constant
   51  04 BB  D0 04E0 40$:    movl    @str_l_fp(r11),r1         ;get the parm
   04 AB  04  C0 04E4         addl    #4,str_l_fp(r11)          ;increment format pointer
          51  D5 04E8         tstl    r1                        ;set cond codes
          05     04EA         rsb                               ;return
                04EB ; pc relative long constant, used for remote and picture formats
   51  04 BB  D0 04EB 50$:    movl    @str_l_fp(r11),r1         ;get the parm
   04 AB  04  C0 04EF         addl    #4,str_l_fp(r11)          ;increment format pointer
   51  04 AB  C0 04F3         addl    str_l_fp(r11),r1          ;make addr absolute
          05     04F7         rsb                               ;return
                04F8 ; version 1 pc relative entry point, used for expressions in format items
   50  04 BB  D0 04F8 60$:    movl    @str_l_fp(r11),r0         ;get the addr
   04 AB  04  C0 04FC         addl    #4,str_l_fp(r11)          ;update format pointer
   50  04 AB  C0 0500         addl    str_l_fp(r11),r0          ;make addr absolute
   51  FC AD  D0 0504         movl    -4(fp),r1                 ;set parent frame pointer
     60   00  FB 0506         calls   #0,(r0)                   ;call it
     51   50  D0 050B         movl    r0,r1                     ;set parm
          05     050E         rsb                               ;return
                050F
                050F ; version 2 pc relative entry point, used for expressions in format items
                050F 70$:
   50  04 BB  D0 050F         movl    @str_l_fp(r11),r0         ;get the addr
   04 AB  04  C0 0513         addl    #4,str_l_fp(r11)          ;update format pointer
   50  04 AB  C0 0517         addl    str_l_fp(r11),r0          ;make addr absolute
   51  08 AB  D0 051B         movl    str_l_parent(r11),r1      ;set parent frame pointer
     60   00  FB 051F         calls   #0,(r0)                   ;call it
     51   50  D0 0522         movl    r0,r1                     ;set parm
```

```
                        05  0525          rsb                                 ;return
                            0526
         08 AC    50    D0  0526  fail:   movl    r0,fcb_l_error(ap)          ;set error in fcb
                  5C    DD  052A          pushl   ap                          ;set fcb addr
                  50    DD  052C          pushl   r0                          ;set error code
        00000000'8F     DD  052E          pushl   #pli$_error                 ;set error condition
   00000000'GF    03    FB  0534          calls   #3,g^pli$io_error           ;signal error
                        04  053B          ret                                 ;return
                            053C  ;++
                            053C  ; pli$$putfmt_r6
                            053C  ;
                            053C  ; functional description:
                            053C  ;       control formats are processed and the next item is transmitted to the
                            053C  ;       file buffer via edit directed output. for data formats, the general
                            053C  ;       flow is: the compiled code jsb's to pli$pute**** routine. that
                            053C  ;       routine pushes the address, scale and precision and the case index
                            053C  ;       for the general conversion routine for the data type of the source.
                            053C  ;       for unaligned bit targets, the offset is passed in r5. thus r5 MUST
                            053C  ;       BE PRESERVED by all output control formats or the offset is lost.
                            053C  ;       the pli$pute**** routine then jmp's to this routine. this routine
                            053C  ;       finds the next data format (processing all intervening control formats)
                            053C  ;       and then enters the data format processing code. the data formats
                            053C  ;       convert the source to a standard type based on the format. this is then
                            053C  ;       placed in the files buffer by jumping to pli$$putnedi_r6.
                            053C  ;       note that some of the common control formats are above in the getformat
                            053C  ;       section.
                            053C  ;
                            053C  ; inputs:
                            053C  ;       0(sp) - data type as a case index for pli$cvrt_cg_r3 as source
                            053C  ;       4(sp) - address of next item to put
                            053C  ;       8(sp) - prec/scale of item
                            053C  ;       12(sp) - return address
                            053C  ;       r11 - address of stream block
                            053C  ;       ap - address of fcb
                            053C  ; outputs:
                            053C  ; side effects:
                            053C  ;       r0-r6 are destroyed
                            053C  ;--
                            053C
                            053C  pli$$putfmt_r6::
      50      04 BB     90  053C          movb    @str_l_fp(r11),r0           ;get format type
              04 AB     D6  0540          incl    str_l_fp(r11)               ;update format pointer
                            0543          case    type=b,r0,limit=#1,< -      ;case on format type
                            0543                  putbiter, -                 ;1 byte constant iteration
                            0543                  putwiter, -                 ;2 word constant iteration
                            0543                  putliter, -                 ;3 long constant iteration
                            0543                  invfrm, -                   ;4 pc relative iter (invalid)
                            0543                  putexpriter, -              ;5 expression iteration (Version 1)
                            0543                  puteof, -                   ;6 end of format
                            0543                  putexpriter_v2, -           ;7 expression iteration (Version 2)
                            0543                  invfrm, -                   ;8 invalid format
                            0543                  invfrm, -                   ;9 invalid format
                            0543                  puta, -                     ;10 alphanumeric format
                            0543                  putb1, -                    ;11 bit (1) format
                            0543                  putb1, -                    ;12 bit 1 format
                            0543                  putb2, -                    ;13 bit 2 format
                            0543                  putb3, -                    ;14 bit 3 format
```

```
                              0543                        putb4, -                ;15 bit 4 format
                              0543                        putcol, -               ;16 column format
                              0543                        putcol, -               ;17 column format
                              0543                        pute, -                 ;18 exp format
                              0543                        putf, -                 ;19 fixed format
                              0543                        putline, -              ;20 line format
                              0543                        putp, -                 ;21 picture format
                              0543                        putpage, -              ;22 page format
                              0543                        putr, -                 ;23 remote format (PL/I version 1)
                              0543                        putskip, -              ;24 skip format
                              0543                        puttab, -               ;25 tab format
                              0543                        putx, -                 ;26 blank format
                              0543                        invfrm, -               ;27 left paren (no longer used)
                              0543                        putrparen, -            ;28 right paren
                              0543                        putr_v2>                ;29 remote format (PL/I version 2)
                              0581
                FC18    31    0581          brw           invfrm                  ;none of the above, invalid format
                              0584
                              0584   ; a format, output
                              0584   ; get the width
                FF19    30    0584   puta:  bsbw          get_format_parm         ;get the parameter
       1F 0C AB   00    E1    0587          bbc           #str_v_missing,str_l_fs(r11),20$ ;if parm missing then[
                              058C   ; if the width is missing, we convert the source to a char(1000) var.
 53   000003E8 8F       D0    058C          movl          #1000,r3                ;set max size for vcha in field
      52   18 AB        9E    0593          movab         str_b_field(r11),r2     ;set addr
    54    8E   06       C1    0597          addl3         #cvt_k_dst_vcha,(sp)+,r4 ;set case index for vcha dest
          50   8E       7D    059B   10$:   movq          (sp)+,r0                ;set src addr, and prec
   00000000'GF   00     FB    059E          calls         #0,g^pli$cvrt_cg_r3     ;convert src to vcha
                              05A5   ; put it out
       00000000'GF      17    05A5          jmp           g^pli$$putnedi_r6       ;put it in buffer and return]
                              05AB   ;width present
                  51    D5    05AB   20$:   tstl          r1                      ;else [set cond codes
                  29    13    05AD          beql          50$                     ;if eql, ignore this field
                  03    14    05AF          bgtr          30$                     ;if gtr, cont
                FBF2    31    05B1          brw           invfrmprm               ;its lss, invalid format
   000003E8 8F   51     D1    05B4   30$:   cmpl          r1,#1000                ;len too big for field?
                  0A    15    05BB          bleq          40$                     ;if leq, no
 50   00000000'8F       D0    05BD          movl          #pli$_strovfl,r0        ;set field overflow
                FF5F    31    05C4          brw           fail                    ;and fail
        18 AB    51     B0    05C7   40$:   movw          r1,str_b_field(r11)     ;set len in field
            53   51     D0    05CB          movl          r1,r3                   ;set dst len
      52   1A AB        9E    05CE          movab         <str_b_field+2>(r11),r2 ;set dst addr
    54    8E   05       C1    05D2          addl3         #cvt_k_dst_char,(sp)+,r4 ;set case index for char dest
                  C3    11    05D6          brb           10$                     ;cont]
      5E    0C AE       9E    05D8   50$:   movab         12(sp),sp               ;clean stack
                  05          05DC          rsb                                   ;its a(0), ignore field by returning
                              05DD
                              05DD   ; b format, output
                              05DD   ; set the radix
                  01    DD    05DD   putb1: pushl         #1                      ;set radix
                  0A    11    05DF          brb           putb                    ;cont in common
                  02    DD    05E1   putb2: pushl         #2                      ;set radix
                  06    11    05E3          brb           putb                    ;cont in common
                  03    DD    05E5   putb3: pushl         #3                      ;set radix
                  02    11    05E7          brb           putb                    ;cont in common
                  04    DD    05E9   putb4: pushl         #4                      ;set radix
                              05EB          ;stack at this point:
```

```
                          05EB        ;12(sp) prec/scale of src
                          05EB        ; 8(sp) addr of src
                          05EB        ; 4(sp) datatype of src
                          05EB        ; 0(sp) radix
             FEB2   30    05EB  putb:    bsbw    get_format_parm          ;get the width
               0F   14    05EE           bgtr    20$                      ;if gtr, cont
               03   13    05F0           beql    10$                      ;if eql, check for missing
             FBB1   31    05F2           brw     invfrmprm                ;its lss, invalid format
     05 0C AB   00  E0    05F5  10$:     bbs     #str_v_missing,str_l_fs(r11),20$ ;if parm missing, use src prec
        5E   10 AE  9E    05FA           movab   16(sp),sp                ;its a(0), so clean stack
                    05    05FE           rsb                              ;return
                          05FF  ; determine the binary precision of the src
        18 AB   5E  D0    05FF  20$:     movl    sp,str_b_field(r11)      ;save current stack addr
           56   51  D0    0603           movl    r1,r6                    ;save width
   50   04 AE   09  C7    0606           divl3   #9,4(sp),r0              ;get data type of source
                          060B           case    type=b,r0,< -            ;case on src data type
                          060B                   35$, -                   ;0 pic, not yet implemented
                          060B                   30$, -                   ;1 fixb
                          060B                   30$, -                   ;2 fltb
                          060B                   40$, -                   ;3 fixd
                          060B                   40$, -                   ;4 fltd
                          060B                   30$, -                   ;5 char
                          060B                   60$, -                   ;6 vcha
                          060B                   30$, -                   ;7 bit
                          060B                   30$>                     ;8 abit
             FB78   31    0621           brw     invfrm                   ;invalid data type, fail
        53   0C AE  D0    0624  30$:     movl    12(sp),r3                ;set dst prec eql to src prec
   53   00000080 8F CA    0628           bicl    #^x80,r3                 ;clear gfloat indicator
               58   11    062F           brb     70$                      ;cont
        51   0C AE  D0    0631  35$:     movl    12(sp),r1                ;get addr of pic descr
           53   61  9A    0635           movzbl  pic$w_pq(r1),r3          ;get src prec
        51   01 A1  9A    0638           movzbl  pic$w_pq+1(r1),r1        ;get src scale
               11   11    063C           brb     45$                      ;cont
        53   0C AE  9A    063E  40$:     movzbl  12(sp),r3                ;get src prec
   53   00000080 8F CA    0642           bicl    #^x80,r3                 ;clear gfloat indicator
 51   0C AE   F8 8F 78    0649           ashl    #-8,12(sp),r1            ;get src scale
           53   51  C2    064F  45$:     subl    r1,r3                    ;get number of digs in integer part
               1F   15    0652           bleq    50$                      ;if leq, then result is zero
   53   0000014C 8F C4    0654           mull2   #332,r3                  ;get binary precision according to rule
   53   00000063 8F C0    065B           addl    #99,r3                   ;round for ceil and fixed divide
   53   00000064 8F C6    0662           divl2   #100,r3                  ;finish (r3=ceil((p-q)*3.32)) really!)
           1F   53  D1    0669           cmpl    r3,#31                   ;prec > 31?
               1B   15    066C           bleq    70$                      ;if leq, no, continue
           53   1F  D0    066E           movl    #31,r3                   ;use max fixb prec
               16   11    0671           brb     70$                      ;cont
        51   18 AB  9E    0673  50$:     movab   str_b_field(r11),r1      ;get addr of field
           81   56  B0    0677           movw    r6,(r1)+                 ;put in width
 61   56   20   6E  00 2C 067A           movc5   #0,(sp),#^x20,r6,(r1)    ;blank it out
        5E   10 AE  9E    0680           movab   16(sp),sp                ;clean stack
                    05    0684           rsb                              ;return
        53   08 BE  3C    0685  60$:     movzwl  @8(sp),r3                ;get cur len of vcha src
                          0689  ;get size of bit temp needed, based on src prec and radix
           50   6E  D0    0689  70$:     movl    (sp),r0                  ;get radix
           53   50  C0    068C           addl    r0,r3                    ;round prec up to next multiple of radix
                53  D7    068F           decl    r3                       ;
           53   50  C6    0691           divl    r0,r3                    ;
           53   50  C4    0694           mull    r0,r3                    ;
```

```
           50   53   07   C1 0697          addl3   #7,r3,r0                ;round prec up to a byte
      50   50   FD 8F     78 069B          ashl    #-3,r0,r0               ;get number of bytes required
                     D1   13 06A0          beql    50$                     ;if eql, then result is 0
                             06A2 ;allocate temp on stack and clear last byte
                5E   50   C2 06A2          subl    r0,sp                   ;get space for temp on stack
                FF AE40   94 06A5          clrb    -1(sp)[r0]              ;clear last byte of temp
                52   5E   D0 06A9          movl    sp,r2                   ;set addr of temp
           54   18 AB     D0 06AC          movl    str_b_field(r11),r4     ;get old stack pointer
           50   08 A4     7D 06B0          movq    8(r4),r0                ;get original src
           08 A4     53   D0 06B4          movl    r3,8(r4)                ;save number of bits in temp
           0C A4     56   D0 06B8          movl    r6,12(r4)               ;save width of field
                     05   12 06BC          bneq    80$                     ;if neq, cont
      0C A4     53   64   C7 06BE          divl3   (r4),r3,12(r4)          ;use converted prec for missing width
      54   04 A4     08   C1 06C3 80$:     addl3   #cvt_k_dst_abit,4(r4),r4 ;set case index for abit dst
      00000000'GF    00   FB 06C8          calls   #0,g^pli$cvrt_cg_r3     ;convert src to abit temp
                             06CF ;convert abit temp to vcha in field using B-radix conversion
                             06CF ;local register usage for conversion:
                             06CF ; r0 - radix
                             06CF ; r1 - address of table for this radix
                             06CF ; r2 - current position in bit string
                             06CF ; r3 - output pointer
                             06CF ; r4 - current bits or char
                             06CF ; r5 - number of chars left to do
                             06CF ; r6 - requested width, number of blanks to append
           54   18 AB     D0 06CF          movl    str_b_field(r11),r4     ;get old stack pointer
                50   64   D0 06D3          movl    (r4),r0                 ;get radix
           51   08 A4     D0 06D6          movl    8(r4),r1                ;get number of bits in temp
           56   0C A4     D0 06DA          movl    12(r4),r6               ;get req width
      000003E8 8F    56   D1 06DE          cmpl    r6,#1000                ;width too big?
                     0A   15 06E5          bleq    90$                     ;if leq, no
      50   00000000'8F    D0 06E7          movl    #pli$_strovfl,r0        ;set field overflow
                   FE35   31 06EE          brw     fail                    ;and fail
           53   18 AB     9E 06F1 90$:     movab   str_b_field(r11),r3     ;get addr of start of output field
                83   56   B0 06F5          movw    r6,(r3)+                ;set length in field
           55   51   50   C7 06F8          divl3   r0,r1,r5                ;get number of bytes of output
           0C AB     08   CA 06FC          bicl    #str_m_blankend,str_l_fs(r11) ;assume we can fill req. width
                56   55   D1 0700          cmpl    r5,r6                   ;enough to fill requested width?
                     07   19 0703          blss    100$                    ;if lss, no
                             0705                                          ;if gtr and stringsize supported
                             0705                                          ;then raise it here
                55   56   D0 0705          movl    r6,r5                   ;set req width as length
                     56   D4 0708          clrl    r6                      ;set no blanks on end
                     07   11 070A          brb     110$                    ;cont
           0C AB     08   C8 070C 100$:    bisl    #str_m_blankend,str_l_fs(r11) ;remember to blank out end
                56   55   C2 0710          subl    r5,r6                   ;get number of blanks for end
           51   01   50   78 0713 110$:    ashl    r0,#1,r1                ;get table address
      51   F8E2 CF41      9E 0717          movab   bformattab-2(pc)[r1],r1 ;based on radix
                     52   D4 071D          clrl    r2                      ;start at beginning of bit string
                     54   DD 071F          pushl   r4                      ;save old stack pointer
                     0D   11 0721          brb     130$                    ;enter loop
 54   04 AE     50   52   EF 0723 120$:    extzv   r2,r0,4(sp),r4          ;get some bits
           83   6144      90 0729          movb    (r1)[r4],(r3)+          ;store resulting char in field
                52   50   C0 072D          addl    r0,r2                   ;update pos in bit string
                F0   55   F4 0730 130$:    sobgeq  r5,120$                 ;go again
                             0733 ; append blanks if necessary
      06 0C AB   03       E1 0733          bbc     #str_v_blankend,str_l_fs(r11),140$ ;if we must append blanks
 63   56   20   6E   00   2C 0738          movc5   #0,(sp),#^x20,r6,(r3)   ;append blanks
```

```
      5E   8E   10    C1  073E  140$:   addl3   #16,(sp)+,sp            ;clean stack
       00000000'GF    17  0742          jmp     g^pli$$putnedi_r6       ;put it out
                          0748
                          0748  ; column format, output
                          0748  ; if the requested column is greater than current column and less than the
                          0748  ; linesize, we put in enough blanks to position to the requested column.
                          0748  ; if the requested column is greater than linesize we do a skip. if the
                          0748  ; requested column is less than current column, we do a skip and then
                          0748  ; fill with blanks to get to the requested column
    03 0C AC   17     E1  0748  putcol: bbc     #atr_v_string,fcb_l_attr(ap),5$ ;if string i/o
            FA60      31  074D          brw     invstrfmt               ;fail with invalid string format
            FD4D      30  0750  5$:     bsbw    get_format_parm         ;get the parameter
              07      14  0753          bgtr    20$                     ;if gtr, cont
              03      13  0755          beql    10$                     ;if eql, cont
            FA4C      31  0757          brw     invfrmprm               ;parm < 0, invalid format
              51      D6  075A  10$:    incl    r1                      ;use 1 instead of 0
      50   2A AC      3C  075C  20$:    movzwl  fcb_w_linesize(ap),r0   ;get linesize
         50   51      B1  0760          cmpw    r1,r0                   ;req col > linesize
              03      15  0763          bleq    30$                     ;if leq, no, cont
         51   01      D0  0765          movl    #1,r1                   ;use 1 for col
              51      D7  0768  30$:    decl    r1                      ;get req col - 1
      2E AC   51      B1  076A          cmpw    r1,fcb_w_column(ap)     ;(requested col-1) > current col?
              12      14  076E          bgtr    50$                     ;if gtr, then yes
              0D      13  0770          beql    40$                     ;if eql, then already at right col
              51      DD  0772          pushl   r1                      ;save req col
       00000000'GF    16  0774          jsb     g^pli$$putskp1_r2       ;do a skip
            51 8ED0       077A          popl    r1                      ;restore req col
              07      14  077D          bgtr    60$                     ;if eql, just return
            FDBA      31  077F  40$:    brw     pli$$putfmt_r6          ;go again
      51   2E AC      A2  0782  50$:    subw    fcb_w_column(ap),r1     ;get number of blanks to move
            032E      31  0786  60$:    brw     blank_field             ;fill with blanks, put in buf, go again
                          0789
                          0789  ; e format, output
                          0789  ;get prec of float dec temp from src dtyp and prec
     0C AB    10      CA  0789  pute:   bicl    #str_m_gfloat,str_l_fs(r11) ;assume not g float src
      50   6E   09    C7  078D          divl3   #9,(sp),r0              ;get data type of source
                          0791          case    type=b,r0,< -           ;case on data type
                          0791                  5$, -                   ;0 pic
                          0791                  10$, -                  ;1 fixb
                          0791                  10$, -                  ;2 fltb
                          0791                  30$, -                  ;3 fixd
                          0791                  30$, -                  ;4 fltd
                          0791                  50$, -                  ;5 char
                          0791                  45$, -                  ;6 vcha
                          0791                  40$, -                  ;7 bit
                          0791                  40$>                    ;8 abit
            F9F2      31  07A7          brw     invfrm                  ;invalid data type, fail
      51   08 AE      D0  07AA  5$:     movl    8(sp),r1                ;get addr of pic descr
         53   61      9A  07AE          movzbl  pic$w_pq(r1),r3         ;get prec of pic src
              2E      11  07B1          brb     35$                     ;cont
      53   08 AE      D0  07B3  10$:    movl    8(sp),r3                ;get prec of binary src
     04 53    07      E5  07B7          bbcc    #7,r3,20$               ;if g float
     0C AB    10      C8  07BB          bisl    #str_m_gfloat,str_l_fs(r11) ;set gfloat
53   00000064 8F      C4  07BF  20$:    mull    #100,r3                 ;get pli decimal prec
53   0000014B 8F      C0  07C6          addl    #331,r3                 ;
53   0000014C 8F      C6  07CD          divl    #332,r3                 ;
         12   6E      91  07D4          cmpb    (sp),#cvt_k_src_fltb    ;float bin src?
```

```
            2B  12 07D7           bneq    60$                       ;if neq, no, cont
            53  D7 07D9           decl    r3                        ;correct prec for context computation
            27  11 07DB           brb     60$                       ;cont
    53   08 AE  9A 07DD   30$:    movzbl  8(sp),r3                  ;get prec of decimal src
    1F  53  07  E5 07E1   35$:    bbcc    #7,r3,60$                 ;if g float
    0C  AB  10  C8 07E5           bisl    #str_m_gfloat,str_l_fs(r11) ;set gfloat
            19  11 07E9           brb     60$                       ;cont
        53  1F  D0 07EB   40$:    movl    #31,r3                    ;use max fixb prec for bit
            CF  11 07EE           brb     20$                       ;cont
    50   04 AE  D0 07F0   45$:    movl    4(sp),r0                  ; get addr of string
        51  80  3C 07F4           movzwl  (r0)+,r1                  ; and size (point past 1st word)
            08  11 07F7           brb     55$
    51   08 AE  9A 07F9   50$:    movzbl  8(sp),r1                  ; get size of src
    50   04 AE  D0 07FD           movl    4(sp),r0                  ; get addr of src
          02DF  30 0801   55$:    bsbw    char_flt_ctx              ; get flt dec context
            53  DD 0804   60$:    pushl   r3                        ;save dec prec
               0806       ; get context of fltb temp
  0B 0C AB  04  E1 0806           bbc     #str_v_gfloat,str_l_fs(r11),80$ ;if g float src
            01  DD 080B           pushl   #1                        ;set for g context
 53  00000080 8F C8 080D          bisl    #128,r3                   ;set g float bit for convert
            0E  11 0814           brb     100$                      ;cont
        53  0F  D1 0816   80$:    cmpl    #15,r3                    ;is it f or d?
            07  19 0819           blss    90$                       ;if lss, no
            7E  D4 081B           clrl    -(sp)                     ;set for d context
        53  0F  D0 081D           movl    #15,r3                    ;set max prec of d
            02  11 0820           brb     100$                      ;cont
            02  DD 0822   90$:    pushl   #2                        ;set for h context
    12   08 AE  91 0824   100$:   cmpb    8(sp),#cvt_k_src_fltb     ;float bin src?
            03  12 0828           bneq    105$                      ;if neq, no, cont
         04 AE  D6 082A           incl    4(sp)                     ;correct dec prec
               082D       ;allocate fltb temp on stack
        5E  10  C2 082D   105$:   subl    #16,sp                    ;get room for temp
        52  5E  D0 0830           movl    sp,r2                     ;set temp addr for dst
    50   1C AE  7D 0833           movq    28(sp),r0                 ;set src addr and prec
 54   18 AE  04 C1 0837           addl3   #cvt_k_dst_fltd,24(sp),r4 ;set convert index, dst = fltd
               083C       ; convert src to fltd
 00000000'GF  00 FB 083C          calls   #0,g^pli$cvrt_cg_r3       ;convert to fltd
               0843       ; get w,d,s. s is ignored
          FC5A  30 0843           bsbw    get_format_parm           ;get the width
 000003E8 8F 51 D1 0846           cmpl    r1,#1000                  ;too big?
            0A  15 084D           bleq    110$                      ;if leq, no
 50  00000000'8F D0 084F          movl    #pli$_strovfl,r0          ;set field overflow
          FCCD  31 0856           brw     fail                      ;and fail
        56  51  D0 0859   110$:   movl    r1,r6                     ;save it
          FC41  30 085C           bsbw    get_format_parm           ;get the digs in frac
  08 0C AB  00  E0 085F           bbs     #str_v_missing,str_l_fs(r11),130$ ;if digs in frac not missing
        53  51  D0 0864           movl    r1,r3                     ;save digs in frac
            19  18 0867           bgeq    140$                      ;if geq, cont
          F93A  31 0869           brw     invfrmprm                 ;set invalid format
 53   14 AE  01 C3 086C   130$:   subl3   #1,20(sp),r3              ;use dec prec of src-1 as digs in frac
 50   10 AE  07 C1 0871           addl3   #7,16(sp),r0              ;get number of chars for exp,sign, dot
    50  56  50  C3 0876           subl3   r0,r6,r0                  ;get max number of digs in frac
        50  53  D1 087A           cmpl    r3,r0                     ;src-1 digs too many?
            03  15 087D           bleq    140$                      ;if leq, no, use src-1
        53  50  D0 087F           movl    r0,r3                     ;use number of digs in frac that fits
          FC1B  30 0882   140$:   bsbw    get_format_parm           ;get scale but ignore it
               0885       ; set up parms for convert routine
```

```
7E    10 AE    02   C1  0885          addl3   #2,16(sp),-(sp)          ;set number of digits in exp
               01   DD  088A          pushl   #1                       ;set number cf digits in int
               7E   D4  088C          clrl    -(sp)                    ;set scale factor
               53   DD  088E          pushl   r3                       ;set number of digits in frac
   2C AE    1A AB   9E  0890          movab   <str_b_field+2>(r11),44(sp) ;set addr of dest in dscr
     28 AE     56   D0  0895          movl    r6,40(sp)                ;set size of dest in dscr
     18 AB     56   B0  0899          movw    r6,str_b_field(r11)      ;set size in field
            28 AE   9F  089D          pushab  40(sp)                   ;set addr of dest descr
            14 AE   9F  08A0          pushab  20(sp)                   ;set addr of src
                        08A3  ; convert from fltd to char
                        08A3          case    type=b,40(sp),<160$,170$,180$> ;case on src type
50    00000000'8F   D0  08AE  150$:   movl    #pli$_invfmtparm,r0      ;set format overflow (really size)error
             FC6E   31  08B5          brw     fail                     ;and fail
00000000'GF    06   FB  08B8  160$:   calls   #6,g^FOR$CVT_D_TE        ;convert it
               10   11  08BF          brb     190$                     ;cont
00000000'GF    06   FB  08C1  170$:   calls   #6,g^FOR$CVT_G_TE        ;convert it
               07   11  08C8          brb     190$                     ;cont
00000000'GF    06   FB  08CA  180$:   calls   #6,g^FOR$CVT_H_TE        ;convert it
            DA 50   E9  08D1  190$:   blbc    r0,150$                  ;if lbc, error
            5E 24   C0  08D4          addl    #36,sp                   ;clean stack
      00000000'GF   17  08D7          jmp     g^pli$$putnedi_r6        ;put it out
                        08DD
                        08DD  ; f format, output
                        08DD  ; get w,d,s. s is ignored
                        08DD  putf:
             FBC0   30  08DD          bsbw    get_format_parm          ;get width
               03   14  08E0          bgtr    20$                      ;if gtr, cont
             F8C1   31  08E2  10$:    brw     invfrmprm                ;its leq, so invalid format
000003E8 8F    51   D1  08E5  20$:    cmpl    r1,#1000                 ;width too big?
               0A   15  08EC          bleq    30$                      ;if leq, no
50    00000000'8F   D0  08EE          movl    #pli$_strovfl,r0         ;set field overflow
             FC2E   31  08F5          brw     fail                     ;and fail
               51   DD  08F8  30$:    pushl   r1                       ;save width
             FBA3   30  08FA          bsbw    get_format_parm          ;get digits in frac
               E3   19  08FD          blss    10$                      ;if lss, invalid format
               51   DD  08FF          pushl   r1                       ;save digs in frac
             FB9C   30  0901          bsbw    get_format_parm          ;get scale, ignored for now
                        0904  ;
                        0904  ; we will convert the src to a fixd number with
                        0904  ; 1 more fractional digit than that required. then we round it to the correct
                        0904  ; number of fractional digits.
                        0904  ;
50    08 AE    09   C7  0904          divl3   #9,8(sp),r0              ;get data type of source
                        0909          case    type=b,r0,< -            ;case on data type
                        0909                  65$, -                   ;0 pic
                        0909                  90$, -                   ;1 fixb
                        0909                  40$, -                   ;2 fltb
                        0909                  70$, -                   ;3 fixd
                        0909                  70$, -                   ;4 fltd
                        0909                  90$, -                   ;5 cha.
                        0909                  90$, -                   ;6 vcha
                        0909                  90$, -                   ;7 bit
                        0909                  90$>                     ;8 abit
             F87A   31  091F          brw     invfrm                   ;invalid data type
                        0922  40$:    ;fltb
10 AE     0063 8F   B1  0922          cmpw    #99,16(sp)               ;dec prec > 30?
               40   14  0928          bgtr    90$                      ;if leq, then no, use common
```

```
                        092A      ;       movq    12(sp),r0                  ;set src addr and prec
                        092A      ;       movl    (sp),r3                    ;get digs in frac
             6E   1F  D1  092A            cmpl    #31,(sp)                   ;trying to print more than 31 digs?
                  03  18  092D            bgeq    60$                        ;if geq, no
                F874  31  092F  50$:      brw     invfrmprm                  ;invalid format
                  36  11  0932  60$:      brb     90$                        ;go output rounded format
                        0934      ;       ashl    #8,r3,r3                   ;use digs in frac as scale
                        0934      ;       movb    #31,r3                     ;use max fixd prec
                        0934      ;       subl    #16,sp                     ;get space for fixd temp
                        0934      ;       movl    sp,r2                      ;set tmp addr
                        0934      ;       calls   #0,g^pli$fltbfixd_r6       ;convert fltb to fixd
                        0934      ;       brb     110$                       ;cont
        51    10 AE   D0  0934  65$:      movl    16(sp),r1                  ;get addr of picture descr
             51   61  3C  0938            movzwl  pic$w_pq(r1),r1            ;get prec and scale
             51   1F  91  093B            cmpb    #31,r1                     ;prec >= 31?
                  04  11  093E            brb     75$                        ;cont
                          0940  70$:      ;decimal
          10 AE   1F  91  0940            cmpb    #31,16(sp)                 ;prec >= 31?
                  24  14  0944  75$:      bgtr    90$                        ;if gtr, no, use common
                  E7  19  0946            blss    50$                        ;if lss, invalid src prec
        50    0C AE   7D  0948            movq    12(sp),r0                  ;set src addr and prec
             53   6E  D0  094C            movl    (sp),r3                    ;get digs in frac
             1F   53  D1  094F            cmpl    r3,#31                     ;trying to print more than 31 digs?
                  DB  14  0952            bgtr    50$                        ;if gtr, then yes, invalid format
        53   53   08  78  0954            ashl    #8,r3,r3                   ;use digs in frac as scale
             53   1F  90  0958            movb    #31,r3                     ;use max fixd prec
             5E   10  C2  095B            subl    #16,sp                     ;get room for fixd temp
             52   5E  D0  095E            mo      sp,r2                      ;set addr of tmp
    00000000'GF   00  FB  0961            calls   #0,g^pli$fixdfixd_r6       ;convert fixd to fixd tmp
                  3D  11  0968            brb     110$                       ;cont
        50    0C AE   7D  096A  90$:      movq    12(sp),r0                  ;set src addr and prec
     54   03    08 AE C1  096E            addl3   8(sp),#cvt_k_dst_fixd,r4   ;set case index
        53   01   6E  C1  0973            addl3   (sp),#1,r3                 ;get digs in frac + 1
             1F   53  D1  0977            cmpl    r3,#31                     ;trying to print more than 31 digits?
                  03  15  097A            bleq    100$                       ;if leq, no
                F827  31  097C            brw     invfrmprm                  ;invalid format
        53   53   08  78  097F  100$:     ashl    #8,r3,r3                   ;use digs in frac + 1 as scale
             53   1F  90  0983            movb    #31,r3                     ;set max fixd prec
             5E   10  C2  0986            subl    #16,sp                     ;get room for fixd temp
             52   5E  D0  0989            movl    sp,r2                      ;set addr of tmp
    00000000'GF   00  FB  098C            calls   #0,g^pli$cvrt_cg_r3        ;convert src to fixd
             5E   10  C2  0993            subl    #16,sp                     ;get room for another temp
1F   05   10 AE   1F   FF 8F  F8  0996    ashp    #-1,#31,16(sp),#5,#31,(sp) ;round temp
                  6E      099E
          08 AE   8E  7D  099F            movq    (sp)+,8(sp)                ;copy to orig temp
          08 AE   8E  7D  09A3            movq    (sp)+,8(sp)                ;
                          09A7  110$:     ;at this point stack looks like:
                          09A7            ; 0(sp) - rounded fixd(31,digs in frac) temp
                          09A7            ; 16(sp) - digs in frac
                          09A7            ; 20(sp) - width
                          09A7            ; 24(sp) - src data type
                          09A7            ; 28(sp) - src addr
                          09A7            ; 32(sp) - src prec
                          09A7            ; 36(sp) - return addr
             50   5E  D0  09A7            movl    sp,r0                      ;set addr of fixd temp src
     51    10 AE  08  78  09AA            ashl    #8,16(sp),r1               ;use digs in frac as scale
             51   1F  90  09AF            movb    #31,r1                     ;use 31 as prec of fixd src
```

```
             5E    22   C2  09B2          subl    #34,sp                   ;get space for char temp
             52    5E   D0  09B5          movl    sp,r2                    ;set char temp addr of dst
             53    22   D0  09B8          movl    #34,r3                   ;set 34 as len
      00000000'GF  00   FB  09BB          calls   #0,g^pli$fixdchar_r6     ;convert fixd to char
          56   36 AE    D0  09C2          movl    54(sp),r6                ;get width
          18 AB    56   B0  09C6          movw    r6,str_b_field(r11)      ;set width in field
             56    22   D1  09CA          cmpl    #34,r6                   ;width < 34?
                   1C   19  09CD          blss    140$                     ;if lss, no
          54   22  56   C3  09CF          subl3   r6,#34,r4                ;get number of leading blanks
          6E   54  20   3B  09D3          skpc    #^x20,r4,(sp)            ;skip leading blanks
                   03   13  09D7          beql    120$                     ;if eql, cont
                 F7CA   31  09D9          brw     invfrmprm                ;and fail
     1A AB   61    56   28  09DC 120$:    movc3   r6,(r1),<str_b_field+2>(r11) ;copy result to field
          5E   46 AE    9E  09E1 130$:    movab   70(sp),sp                ;clean stack
         00000000'GF    17  09E5          jmp     g^pli$$putnedi_r6        ;put it out
         51   56   22   C3  09EB 140$:    subl3   #34,r6,r1                ;get number of blanks needed
1A AB   51   20   6E   00   2C  09EF          movc5   #0,(sp),#^x20,r1,<str_b_field+2>(r11) ;put in leading blanks
         63   6E   22   28  09F6          movc3   #34,(sp),(r3)            ;copy the result to field
                   E5   11  09FA          brb     130$                     ;cont
                            09FC
                            09FC ; line format
       03 0C AC    17   E1  09FC putline:bbc     #atr_v_string,fcb_l_attr(ap),5$ ;if string i/o
                 F7AC   31  0A01          brw     invstrfmt                ;fail with invalid string format
       0A 0C AC    07   E0  0A04 5$:      bbs     #atr_v_print,fcb_l_attr(ap),10$ ;if print, cont
     50   00000000'8F   D0  0A09          movl    #pli$_notprint,r0        ;set not print file
                 FB13   31  0A10          brw     fail                     ;and fail
                 FA8A   30  0A13 10$:     bsbw    get_format_parm          ;get the parm
         00000000'GF    16  0A16          jsb     g^pli$$putline_r6        ;process the line
                 FB1D   31  0A1C          brw     pli$$putfmt_r6           ;go again
                            0A1F
                            0A1F ; p format output
                 FA7E   30  0A1F putp:    bsbw    get_format_parm          ;get the pict desc
                   03   12  0A22          bneq    10$                      ;if neq, cont
                 F775   31  0A24          brw     invfrm                   ;fail
          52   18 AB    9E  0A27 10$:     movab   str_b_field(r11),r2      ;set dst addr
             53    51   D0  0A2B          movl    r1,r3                    ;set addr of pict desc
          82   04 A1    9B  0A2E          movzbw  pic$b_byte_size(r1),(r2)+ ;set size of resulting string
          54   8E  00   C1  0A32          addl3   #cvt_k_dst_pic,(sp)+,r4 ;set data type
             50    8E   7D  0A36          movq    (sp)+,r0                 ;set addr, size of src
      00000000'GF  00   FB  0A39          calls   #0,g^pli$cvrt_cg_r3      ;convert to pic
         00000000'GF    17  0A40          jmp     g^pli$$putnedi_r6        ;put it out
                            0A46
                            0A46 ; page format
       03 0C AC    17   E1  0A46 putpage:bbc     #atr_v_string,fcb_l_attr(ap),5$ ;if string i/o
                 F762   31  0A4B          brw     invstrfmt                ;fail with invalid string format
       0A 0C AC    07   E0  0A4E 5$:      bbs     #atr_v_print,fcb_l_attr(ap),10$ ;if print, cont
     50   00000000'8F   D0  0A53          movl    #pli$_notprint,r0        ;set not print file
                 FAC9   31  0A5A          brw     fail                     ;and fail
         00000000'GF    16  0A5D 10$:     jsb     g^pli$$putpage_r6        ;do a put page
                 FAD6   31  0A63          brw     pli$$putfmt_r6           ;go again
                            0A66
                            0A66 ; skip format, output
       03 0C AC    17   E1  0A66 putskip:bbc     #atr_v_string,fcb_l_attr(ap),5$ ;if string i/o
                 F742   31  0A6B          brw     invstrfmt                ;fail with invalid string format
                 FA2A   30  0A6E 5$:      bsbw    get_format_parm_1        ;get the number to skip
             52    51   D0  0A71          movl    r1,r2                    ;copy number to skip
         00000000'GF    16  0A74          jsb     g^pli$$putskip_r2        ;do the skips
```

```
              FABF   31  0A7A          brw     pli$$putfmt_r6          ;go again
                         0A7D
                         0A7D  ; tab format
     03 CC AC   17   E1  0A7D  puttab: bbc     #atr_v_string,fcb_l_attr(ap),5$ ;if string i/o
              F72B   31  0A82          brw     invstrfmt               ;fail with invalid string format
              FA13   30  0A85  5$:     bsbw    get_format_parm_1       ;get the tab stop
                05   14  0A88          bgtr    10$                     ;if gtr, cont
                25   13  0A8A          beql    30$                     ;if eql, go again
              F717   31  0A8C          brw     invfrmprm               ;its lss, invalid format
        50   2E AC   3C  0A8F  10$:    movzwl  fcb_w_column(ap),r0     ;get current column
     53   50    07   CB  0A93          bicl3   #7,r0,r3                ;round down to last tab stop
     52   51    03   78  0A97          ashl    #3,r1,r2                ;get number of blanks for req tabs
          52    53   C0  0A9B          addl    r3,r2                   ;get ending column
       52    2A AC   B1  0A9E          cmpw    fcb_w_linesize(ap),r2   ;past end of line?
                07   19  0AA2          blss    20$                     ;if lss, yes, cont
     51   52    50   C3  0AA4          subl3   r0,r2,r1                ;get number of blanks needed
              000C   31  0AA8          brw     blank_field             ;output blanks and go again
       00000000'GF   16  0AAB  20$:    jsb     g^pli$$putskp1_r2       ;do a skip
              FA88   31  0AB1  30$:    brw     pli$$putfmt_r6          ;go again
                         0AB4
                         0AB4  ; x format, output
              F9E4   30  0AB4  putx:   bsbw    get_format_parm_1       ;get the number of blanks
                         0AB7  ;       brw     blank_field             ;put out blanks and go again
                         0AB7
                         0AB7  ;+
                         0AB7  ;blank_field
                         0AB7  ; this routine puts the specified number of blanks in to the field in vcha
                         0AB7  ; format. it then calls pli$$putnedi_r6 and jumps to pli$$putfmt_r6.
                         0AB7  ; inputs:
                         0AB7  ;        r1 - number of blanks
                         0AB7  ; outputs:
                         0AB7  ;        none
                         0AB7  ; side effects:
                         0AB7  ;        r0-r4,r6 are destroyed
                         0AB7  ;        r5 is preserved for the offset to bit sources
                         0AB7  ;-
                         0AB7  blank_field:
                55   DD  0AB7          pushl   r5                      ;save r5 in case a bit src is pending
   000003E8 8F  51   D1  0AB9          cmpl    r1,#1000                ;trying to put too many blanks in?
                0A   15  0AC0          bleq    10$                     ;if leq, no
 50    00000000'8F   D0  0AC2          movl    #pli$_strovfl,r0        ;set field overflow
              FA5A   31  0AC9          brw     fail                    ;and fail
        18 AB   51   F7  0ACC  10$:    cvtlw   r1,str_b_field(r11)     ;set size of string
1A AB  51  20  6E   00   2C  0AD0      movc5   #0,(sp),#^x20,r1,str_b_field+2(r11) ;put in the blanks
       00000000'GF   16  0AD7          jsb     g^pli$$putnedi_r6       ;output the field
            55 8ED0      0ADD          popl    r5                      ;restore r5
              FA59   31  0AE0          brw     pli$$putfmt_r6          ;go on to next format
                         0AE3
                         0AE3  ;+
                         0AE3  ;
                         0AE3  ; char_flt_ctx
                         0AE3  ;
                         0AE3  ; finds the appropriate float decimal precision for a character
                         0AE3  ; string based on the number of digits in the mantissa and
                         0AE3  ; the value of the exponent.
                         0AE3  ;
                         0AE3  ; inputs:
```

```
                         0AE3  ;
                         0AE3  ;        r1 - string size
                         0AE3  ;        r0 - string addr
                         0AE3  ;        ap - addr of file control block
                         0AE3  ;
                         0AE3  ; outputs:
                         0AE3  ;
                         0AE3  ;        returns the precision in r3
                         0AE3  ;
                         0AE3  ;        all other registers preserved
                         0AE3  ;
                         0AE3  ;-
                         0AE3  char_flt_ctx:
  05 0C AC    1A    E1   0AE3          bbc     #atr_v_flttrg,fcb_l_attr(ap),4$ ;if flt target
     53    10 AE    D0   0AE8          movl    16(sp),r3               ;set fltb prec of target
                  05     0AEC          rsb                             ;return
            37    BB     0AED  4$:     pushr   #^m<r0,r1,r2,r4,r5>     ; save regs
    60    51    20    3B 0AEF          skpc    #32,r1,(r0)             ; skip leading blanks
            05    12     0AF3          bneq    5$                      ; if string not blank, br
      54    01    D0     0AF5          movl    #1,r4                   ; else set prec of 1
            7A    11     0AF8          brb     100$
                         0AFA  ;
      52    50    7D     0AFA  5$:     movq    r0,r2                   ; save new addr and length from skip
    63    50    20    3A 0AFD          locc    #32,r0,(r3)             ; throw out trailing blanks too
      52    50    C2     0B01          subl    r0,r2                   ; find the number of non-blank chars
      2B    63    91     0B04          cmpb    (r3),#^a/+/             ; check for a sign
            05    13     0B07          beql    10$                     ; br if found
      2D    63    91     0B09          cmpb    (r3),#^a/-/             ; minus?
            04    12     0B0C          bneq    20$                     ; br if no sign
            53    D6     0B0E  10$:    incl    r3                      ; point past it
            52    D7     0B10          decl    r2
      54    52    D0     0B12  20$:    movl    r2,r4                   ; make char. count the digit count
    63    52    2E    3A 0B15          locc    #^a/./,r2,(r3)          ; check for decimal point
            02    13     0B19          beql    30$                     ; br if none
            54    D7     0B1B          decl    r4                      ; deduct dec. pt. from digit count
 63    52    45 8F    3A 0B1D  30$:    locc    #^a/E/,r2,(r3)          ; look for E
            07    12     0B22          bneq    40$                     ; br if found
 63    52    65 8F    3A 0B24          locc    #^a/e/,r2,(r3)          ; e?
            49    13     0B29          beql    100$                    ; if none, that's it
      54    50    C2     0B2B  40$:    subl    r0,r4                   ; sub. exponent chars from digit count
            51    D6     0B2E          incl    r1                      ; point past the E/e
            50    D7     0B30          decl    r0
      2B    61    91     0B32          cmpb    (r1),#^a/+/             ; check for exponent sign
            05    13     0B35          beql    45$                     ; br if found
      2D    61    91     0B37          cmpb    (r1),#^a/-/             ; minus?
            04    12     0B3A          bneq    50$                     ; br if no sign
            51    D6     0B3C  45$:    incl    r1                      ; point past the sign char
            50    D7     0B3E          decl    r0
      0F    54    D1     0B40  50$:    cmpl    r4,#15                  ; is prec. huge?
            2F    14     0B43          bgtr    100$                    ; if so, that's it
                         0B45                                          ; else, get exponent value
      5E    50    C2     0B45          subl    r0,sp                   ; get a stack temp
            11    BB     0B48          pushr   #^m<r0,r4>              ; save some regs
 08 AE    61    50    28 0B4A          movc3   r0,(r1),8(sp)           ; copy exp. digits to temp
            11    BA     0B4F          popr    #^m<r0,r4>              ; restore regs
            7E    7C     0B51          clrq    -(sp)                   ; more temps
    07 AE    20    90    0B53          movb    #32,7(sp)               ; make a leading sep. string
```

```
                        50   DD  0B57  1311          pushl   r0                       ; save size
   08 AE   04   0B AE   50   09  0B59  1312          cvtsp   r0,11(sp),#4,8(sp)       ; cvrt exponent to packed
      04 AE     08 AE   04   36  0B60  1313          cvtpl   #4,8(sp),4(sp)           ; cvrt packed to long
                26   04 AE   D1  0B66  1314          cmpl    4(sp),#38                ; see if exponent is huge
                        03   15  0B6A  1315          bleq    60$                      ; if not, br
                   54   22   D0  0B6C  1316          movl    #34,r4                   ; plug max. huge prec.
                   5E   8E   C0  0B6F  1317 60$:     addl    (sp)+,sp                 ; clean off the stack
                        8E   7C  0B72  1318          clrq    (sp)+
                                 0B74  1319 100$:
                   53   54   D0  0B74  1320          movl    r4,r3                    ; return result in r3
                        37   BA  0B77  1321          popr    #^m<r0,r1,r2,r4,r5>      ; restore regs
                             05  0B79  1322          rsb
                                 0B7A  1323
                                 0B7A  1324          .end
```

```
ATR_M_VIRGIN          = 02000000              FCB_W_PAGE                 00000032
ATR_V_FLTTRG          = 0000001A              FCB_W_PAGESIZE             0000002C
ATR_V_PRINT           = 00000007              FCB_W_REVISION             00000028
ATR_V_STRING          = 00000017              FOR$CVT_D_TE               ********    X  02
BFORMATTAB              00000000 R    02      FOR$CVT_G_TE               ********    X  02
BITER                   00000071 R    02      FOR$CVT_H_TE               ********    X  02
BLANK_FIELD             00000AB7 R    02      GETA                       000001BA R     02
CHAR_FLT_CTX            00000AE3 R    02      GETB                       000001DA R     02
COMEOF                  0000017A R    02      GETB1                      000001CC R     02
COMR                    0000041A R    02      GETB2                      000001D0 R     02
COMRPAREN               0000048A R    02      GETB3                      000001D4 R     02
COMR_V2                 000003F8 R    02      GETB4                      000001D8 R     02
CVT_K_DST_ABIT        = 00000008              GETBITER                   00000066 R     02
CVT_K_DST_CHAR        = 00000005              GETCOL                     00000228 R     02
CVT_K_DST_FIXD        = 00000003              GETE                       00000281 R     02
CVT_K_DST_FLTD        = 00000004              GETEOF                     0000016E R     02
CVT_K_DST_PIC         = 00000000              GETEXPRITER                000000B6 R     02
CVT_K_DST_VCHA        = 00000006              GETEXPRITER_V2             000000A4 R     02
CVT_K_SRC_ABIT        = 00000048              GETF                       000002C0 R     02
CVT_K_SRC_CHAR        = 0000002D              GETITERCOM                 000000D8 R     02
CVT_K_SRC_FIXB        = 00000009              GETLITER                   0000008F R     02
CVT_K_SRC_FIXD        = 0000001B              GETP                       000003B2 R     02
CVT_K_SRC_FLTB        = 00000012              GETR                       0000040E R     02
CVT_K_SRC_FLTD        = 00000024              GETRPAREN                  0000047E R     02
CVT_K_SRC_PIC         = 00000000              GETR_V2                    000003EC R     02
EXITER                  000000C2 R    02      GETSKIP                    00000451 R     02
EXITER_COMMON           000000C6 R    02      GETWITER                   0000007A R     02
EXITER_V2               000000B0 R    02      GETX                       0000046D R     02
FAIL                    00000526 R    02      GET_FORMAT_COM             000004A2 R     02
FCB_B_ENVIR             000001C2              GET_FORMAT_PARM            000004A0 R     02
FCB_B_ESA               0000012E              GET_FORMAT_PARM_1          0000049B R     02
FCB_B_EXTRA             0000003D              INVFRM                     0000019C R     02
FCB_B_FAB               000000A6              INVFRMPRM                  000001A6 R     02
FCB_B_IDENT             00000040              INVSTRFMT                  000001B0 R     02
FCB_B_IDENT_NAM         00000042              LITER                      0000009A R     02
FCB_B_NAM               000000F6              PIC$B_BYTE_SIZE          = 00000004
FCB_B_NUMKCBS           0000003C              PIC$W_PQ                 = 00000000
FCB_B_RAB               00000062              PLI$$CHRBITN_R6            ********    X  02
FCB_C_LEN               000001C2              PLI$$GETFMT_R6             0000001E RG    02
FCB_C_STRLEN            00000034              PLI$$GETNEDI_R6            ********    X  02
FCB_L_ATTR              0000000C              PLI$$GETSKIP_R2            ********    X  02
FCB_L_BUF               00000014              PLI$$GETSKP1_R2            ********    X  02
FCB_L_BUF_END           00000018              PLI$$PUTFMT_R6             0000053C RG    02
FCB_L_BUF_PT            0000001C              PLI$$PUTLINE_R6            ********    X  02
FCB_L_CNDADDR           000001B2              PLI$$PUTNEDI_R6            ********    X  02
FCB_L_CONDIT            000001AE              PLI$$PUTPAGE_R6            ********    X  02
FCB_L_DTTR              00000010              PLI$$PUTSKIP_R2            ********    X  02
FCB_L_ERROR             00000008              PLI$$PUTSKP1_R2            ********    X  02
FCB_L_KCB               00000038              PLI$CHARFIXD_R6            ********    X  02
FCB_L_NEXT              00000000              PLI$CVRT_CG_R3             ********    X  02
FCB_L_PREVIOUS          00000004              PLI$FCHRFLTD_R6            ********    X  02
FCB_L_PRN               00000034              PLI$FIXDCHAR_R6            ********    X  02
FCB_Q_RFA               00000020              PLI$FIXDFIXD_R6            ********    X  02
FCB_W_COLUMN            0000002E              PLI$IO_ERROR               ********    X  02
FCB_W_IDENT_LEN         00000040              PLI$VALID_PIC              ********    X  02
FCB_W_LINE              00000030              PLI$_CNVERR                ********    X  02
FCB_W_LINESIZE          0000002A              PLI$_ERROR                 ********    X  02
```

```
PLI$_FORMATOVFL          ********   X   02
PLI$_INVFMTPARM          ********   X   02
PLI$_INVFORMAT           ********   X   02
PLI$_INVSTRFMT           ********   X   02
PLI$_NOTPRINT            ********   X   02
PLI$_STROVFL             ********   X   02
PUTA                     00000584 R     02
PUTB                     000005EB R     02
PUTB1                    000005DD R     02
PUTB2                    000005E1 R     02
PUTB3                    000005E5 R     02
PUTB4                    000005E9 R     02
PUTBITER                 0000006B R     02
PUTCOL                   00000748 R     02
PUTE                     00000789 R     02
PUTEOF                   00000174 R     02
PUTEXPRITER              000000BC R     02
PUTEXPRITER_V2           000000AA R     02
PUTF                     000008DD R     02
PUTLINE                  000009FC R     02
PUTLITER                 00000094 R     02
PUTP                     00000A1F R     02
PUTPAGE                  00000A46 R     02
PUTR                     00000414 R     02
PUTRPAREN                00000484 R     02
PUTR_V2                  000003F2 R     02
PUTSKIP                  00000A66 R     02
PUTTAB                   00000A7D R     02
PUTWITER                 0000007F R     02
PUTX                     00000AB4 R     02
REMCOM                   0000041E R     02
SF$L_SAVE_FP           = 0000000C
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_BLANKEND         = 00000008
STR_M_GFLOAT           = 00000010
STR_M_MISSING          = 00000001
STR_V_BLANKEND         = 00000003
STR_V_GFLOAT           = 00000004
STR_V_MISSING          = 00000000
WITER                    00000085 R     02
ZERO                     000002CA R     02
```

```
                                  +----------------+
                                  ! 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                       00000B7A  ( 2938.)  02 (   2.)    PIC     USR     CON     REL     LCL   SHR   EXE   RD   NOWRT NOVEC LONG

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

Phase                      Page faults    CPU Time       Elapsed Time
-----                      -----------    --------       ------------
Initialization                    9       00:00:00.08    00:00:01.23
Command processing               73       00:00:00.51    00:00:03.45
Pass 1                          210       00:00:08.45    00:00:26.43
Symbol table sort                 0       00:00:00.74    00:00:01.18
Pass 2                          244       00:00:03.09    00:00:06.34
Symbol table output              19       00:00:00.16    00:00:00.52
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            557       00:00:13.06    00:00:39.18

The working set limit was 1350 pages.
49255 bytes (97 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 338 non-local and 154 local symbols.
1324 source lines were read in Pass 1, producing 21 object records in Pass 2.
19 pages of virtual memory were used to define 17 macros.

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

Macro library name                            Macros defined
------------------                            --------------
_$255$DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1                   7
_$255$DUA28:[SYSLIB]STARLET.MLB;2                        6
TOTALS (all libraries)                                  13

295 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=TRACEBACK/LIS=LIS$:PLIFORMAT/OBJ=OBJ$:PLIFORMAT MSRC$:PLIFORMAT/UPDATE=(ENH$:PLIFORMAT)+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