OKAYAMA UNIVERSITY
Earth Science Report,
Vol. 1, No. 1, 37-48, (1994)

CALCOMP から LIPS3 への変換ライブラリの作成

CALCOMP to LIPS3 convert library

山川 純次 (Junji Yamakawa)*

CALLIP, a CALCOMP compatible Fortran 77 library using the Canon LIPS3 page description language, has been written. As a result, the graphic output of programs with CALCOMP plotting routines used for crystal structure analyses, such as ORTEP2, Rietveld analyses and others can be used under the printer equipped with LIPS3 PDL. The details of the library was given with some graphic examples.

Keywords: CALLIP, CALCOMP, LIPS3, Fortran 77

I. はじめに

鉱物学や結晶学の研究者が使用する計算機プログラムの多くは作図の際に米 Nebraska-Lincoln 大学によって開発された CALCOMP ライブラリを利用することを前提に作成されている。したがって作図結果を出力する際には CALCOMP プロッタまたはその互換機が必要になる。しかし、この CALCOMP ライブラリが備えている作図命令をレーザプリンタ等が備えている作図命令 (ページ記述言語, Page Description Language, PDL と呼ばれる) に変換するライブラリを作成すれば、これらのプログラムの作図結果をそのプリンタで行うことができる。このような働きをするプログラムを CALCOMP 互換ライブラリと呼ぶ。今回、Canon 製レーザプリンタの PDL である LIPS3 (LBP Image Processing System: Canon, 1990) を使った CALCOMP 互換ライブラリ CALLIP を作成したのでこれを報告する。

II. CALLIP の概要

CALLIP は、Fortran 77 で記述された約 30 のサブルーチンから構成されている。各サブルーチンをライブラリの中での役割により分類すると次の様になる。

1. ユーザーサービスサブルーチン
   (a) CALCOMP 互換サブルーチン
   (b) CALCOMP 準拠サブルーチン
   (c) 特殊サブルーチン
2. 内部サブルーチン
   (a) LIPS3 サブルーチン
   (b) 絶対座標管理サブルーチン

これらのサブルーチンは CALCOMP ライブラリを使用するように設計されたプログラムから呼び出された場合は CALCOMP ライブラリと同様に機能する一方で、CALCOMP プリンタに対する作図命令を LIPS3 の命令に変換する。

1. のユーザーサービスサブルーチンは CALCOMP との互換性を提供するサブルーチンである。(a) は引数の個数と順番を CALCOMP ライブラリに完全に合わせてある。(b) は引数の持つ意味が CALCOMP ライブラリと異なっている。そのため純正 CALCOMP ライブラリを使用しているプログラムで使用する場合、修正が必要なサブルーチンである。(c) はこのライブラリ独自のルーチンでオリジナルの CALCOMP ライブラリには備わっていなかった幾つかの新しい機能を提供している。

2. の内部サブルーチンは、このライブラリでの内部処理、すなわちプリンタの初期設定や終了処理、LIPS3 制御コードの出力、数値の LIPS3 表現への変換、絶対座標管理などをおこなうルーチンである。通常の使用においてこれらのサブルーチンがユーザから直接呼び出されることは無い。

*岡山大学理学部地学科, 〒 700 岡山市津島中 3-1-1

III. CALLIP の利用

このライブラリを利用して作図プログラムを移植、あるいは開発する方法には次のものがある。一つは作図プログラムのソースコード中にこのライブラリのソースコードを連結し、一気にコンパイルとリンクを実行する方法である。この方法は手軽な反面、コンパイルやリンクに余計な時間がかかる。もう一つは、このライブラリをあらかじめオブジェクト形式に変換しておき、作図プログラムのオブジェクト形式とリンクする方法である。コンパイルやリンクの時間は減少するが、ソースコードの分割コンパイルを伴うので、Makefile 等でコンパイルやリンクを管理したり、ライブラリ管理プログラムを使う必要が生じる。多少扱いが煩雑になり一般的ではないが、熟練したプログラマにはこちらの方法が好まれるようである。

このライブラリを使用した作図プログラムによる出力は "scratch.lps" という名前のファイルに LIPS3 PDL として書き出されるので、これを LIPS3 が利用可能なプリンタに転送すれば作図が実行される。

なお、CALLIP は著者によりフリーソフトウェアとして提供される。著者に連絡していただければ、ソースコードを無償で提供する。ただし CALLIP の利用に伴って発生するいかなる責任も著者は負わない。

謝　　辞

本解説記事は指導教官の岡山大学理学部地学科の河原昭先生に勧めていただきました。LIPS3 PDL についての資料の一部と米 Nebraska-Lincoln 大学による CALCOMP の解説書の貸与、CALLIP の評価と改良は岡山大学教養部計算機統計学教室の垂水共之先生にご指導いただきました。、無機材質研究所の泉富士夫先生からは FAT-RIETAN パッケージおよび CALCOMP to PostScript ライブラリ (無機材研、山本昭二先生作成) をいただきました。本ライブラリの中の NUMBER, AXIS, SCALE はこのライブラリを利用させていただきました。以上の方々に感謝いたします。

参考文献

DeLorm, R. T. and Kersten, L., "CALCOMP PROGRAMMING FOR THE DIGITAL PLOTTERS," *Univ. of Nebraska press.*

Johnson, C. K. (1965). *ORTEP*. Report OR-3794. Oak Ridge National Laboratory, Tennessee, EU.

(株)CANON (1990), "LASER SHOT B406 S/B406 D プログラマーズ マニュアル、ソフトウェア概説書、コマンドリファレンス" *CANON INC.*

```
*************************************************************************
* Callip.f : A (super) subset of CALCOMP to LIPS3 convert Library.
*            by JyAM in 1993.5
*            $Revision: 1.36 $

*            NDP Fortran-386/486 Version. See the documentation
*            by the author.

*************************************************************************
* PLOTS : CALCOMP
*         Start the LIPS3 Vector-mode.

*         ident : Bannar
*         scale : Scale factor.
*                 If scale = 1.0 then 1.0 is equal to 1.0 cm.

      subroutine PLOTS(ident, scale)

      real          scale, orgscl
      character*80 ident

      common       /mag/ orgscl

      real         sfact
      integer      corg(2), cpenp(2)
      common       /lips/ corg(2), cpenp(2), sfact

      character    lps(100), is2
      data         is2 /z'1e'/

      orgscl     = scale

      write (unit=31, fmt=6000) is2
 6000 format ('#',a1)                    ! Start a Picture.

      call selvd(-2,1)                   ! select coordinate parameter.
                                         ! -2 : 1/100 mm
                                         !  1 : magnification factor

      write (unit=31, fmt=6010) is2
 6010 format ('$',a1)                    ! Start Drawing instructions

      call setvd(0, 0, 30000, 20000)     ! Set to negative angle.
      call movvd(1000, 2000)             ! Move VD origine.
      call setvd(1000, 2000, 30000, 20000)
                                         ! Reset Drawing area.
      write (unit=31, fmt=6020) is2
 6020 format(')F2',a1)                   ! Set interpolate type to 2

      sfact = scale * 1000               ! Set initial scale factor.

*Font definition

      write (unit=31, fmt=6030) is2
 6030 format ('T1',a1)                   ! Allocate G0 font set.

      write (unit=31, fmt=6040) is2
 6040 format ('|1',a1)                   ! Allocate G0 graphic set.

      write (unit=31,fmt=6050) is2
 6050 format ('U2',a1)                   ! Set font arrtibute.
                                         !  2 : Stroke unit

      Do 10 i = 1,2                      ! Set initial penpos and origine.
         cpenp(i)  = 0
         corg(i)   = 0
 10   continue

      call FACTOR ( 1. / scale )
      call SYMBOL (-1., -1., .3,ident, 0., 0 )
      call FACTOR ( 1.0 )

      return
      end

*************************************************************************
* PLOTE : CALCOMP

* Terminate the current instructions.

      subroutine PLOTE

      character    is2, FF
      data         is2 /z'1e'/
      data         FF  /z'0c'/

      write (unit=31, fmt=6000) is2
 6000 format ('%',a1)                    ! End a Picture.

      write (unit=31, fmt=6020) FF
 6020 format (a1)                        ! Eject the page.

      return
      end

*************************************************************************
* XYOPEN :
* Preparate the Graphic. Change to the LIPS3 Vector-mode.

      subroutine XYOPEN

      character    esc, isl, is2
      data         esc /o'033'/
      data         isl /z'1f'/
      data         is2 /z'1e'/

      open (unit=31, file='scratch.lps')

      call l3saj

      write (unit=31, fmt=6000) esc
 6000 format(a1, '[0&)')                 ! Change to the Vector Mode.

      write (unit=31, fmt=6002) isl, is2
 6002 format (' <Swiss',a1, 'Dutch-Roman',a1)  ! Set font set.

      write (unit=31, fmt=6004) isl, is2
 6004 format (' :1J',a1, '1I',a1)        ! Set graphic set.

      write (unit=31, fmt=6010) is2
 6010 format (' @1',a1)                  ! Page Orientation
                                         ! rotate to 90deg

      return
      end

*************************************************************************
* XYCLOS :
* Terminate the graphic. Return to the TEXT-mode

      subroutine XYCLOS

      character    is2
      data         is2 /z'1e'/

      write(unit=31, fmt=6000) is2
 6000 format ('}p',a1)                   ! Change to the Text Mode

      call l3eoj

      close(unit=31)

      return
      end

*************************************************************************
* FACTOR : CALCOMP
*          Magnify all over scale.

      subroutine FACTOR(magfact)

      real         magfact

      common        /mag/ scale

      real         sfact
      integer      corg(2), cpenp(2)
      common       /lips/ corg(2), cpenp(2), sfact

      sfact = scale * 1000 * magfact

      return
      end

*************************************************************************
* PLOT : CALCOMP

*        x, y, : Position from current origine
*        n     :     2 : Draw a line.
*                    3 : Move with pen up.
*                   -2 : Draw and reset the current origine.
```

```
*                -3 : Move and reset the current origine.
* Bugs : Ignore PLOT ( x, y, 999 ) to terminate plot instructions.
      subroutine PLOT(x, y, n)

      real        x, y
      integer     n, digit, xy(2), workx, worky

      real        sfact
      integer     corg(2), cpenp(2)
      common      /lips/ corg(2), cpenp(2), sfact

      character   lps(100), is2
      data        is2 /z'1e'/

      xy(1)  =  int( x * sfact)
      xy(2)  =  int( y * sfact)

      call relpos( xy )

      workx = xy(1) - cpenp(1)
      worky = xy(2) - cpenp(2)
      if ( ( abs(workx) .lt. 10 ) .and. ( abs(worky) .lt. 10 ) ) then
         workx = 10
         worky = 10
      endif

      ntype = abs( n )

      if( ntype .eq. 2 ) then

         lps(1) = '1'
         digit  =  1

         call  declps(cpenp(1), lps, digit)
         call  declps(cpenp(2), lps, digit)
         call  declps(workx,    lps, digit)
         call  declps(worky,    lps, digit)

         digit      = digit + 1
         lps(digit) = is2

         write(unit=31, fmt='(100A)') (lps(i), i=1, digit)

         call stpenp( xy )      ! set current penpos

      else if( ntype .eq. 3 ) then

         call stpenp( xy )      ! set current penpos

      endif

      if ( n .lt. 0 ) then
         call stcorg( xy )
      endif

      return
      end

*************************************************************************
* DASHNU : CALCOMP
*          Draw a broken line.

*          x1, y1 : Coordinate of beginning of dashed line.
*          x2, y2 : Coordinate of ending of dashed line.
*          st, en : Code for beginning and ending condition of line.

* Bugs : Ignore the begining and ending condition.

      subroutine DASHNU (x1, y1, x2, y2, st, en )

      real        x1, y1, x2, y2
      integer     st, en

      real        cpenw
      common      /pen/ cpenw

      call plot ( x1, y1, 3 )          ! Move to start position

      if ( cpenw .lt. 0 ) then         ! To activate this instruction.
         call newpen ( 0.01 )
      endif

      call ltype(-2)                   ! Set line type to broken line.
      call plot ( x2, y2, 2 )          ! Draw a line with broken line
      call ltype(0)                    ! Reset the line type.

      return
      end

*************************************************************************
* CENLIN : CALCOMP
*          Center line style. Lining from current pen position.

      subroutine CENLIN ( x1, y1 )

      real        x1, y1

      real        cpenw
      common      /pen/ cpenw

      if ( cpenw .lt. 0 ) then         ! To activate this instruction.
         call newpen ( 0.01 )
      endif

      call ltype(3)                    ! Set line type to 1 pt broken line.
      call PLOT ( x1, y1, 2 )          ! Draw a line with broken line
      call ltype(0)                    ! Reset the line type.

      return
      end

*************************************************************************
* NEWPEN : CALCOMP ( Modified )
*          Select a new pen.

*          In this routine, the pen number convert to the line width.
*          If pcolor = -1 then reset to 1 dot line,
*          else a (pcolor)cm width line will be selected.

*          Ex) A newpen (1.0) instruction means a 1.0 cm width.

      subroutine NEWPEN(pcolor)

      real        pcolor
      integer     width, digit

      real        sfact
      integer     corg(2), cpenp(2)
      common      /lips/ corg(2), cpenp(2), sfact

      character   lps(100), is2
      data        is2 /z'1e'/

      cpenw     = pcolor

      if ( pcolor .lt. 0 ) then
         width = int (pcolor)          ! Probably reset ?
      else
         width  = int ( pcolor * sfact )  ! in cm
      endif

      lps(1) = 'P'
      lps(2) = '1'
      digit  =  2

      call  declps(width, lps, digit)

      digit      = digit + 1
      lps(digit) = is2

      write(unit=31, fmt='(100A)') (lps(i), i=1, digit)

      return
      end

*************************************************************************
* SYMBOL : CALCOMP
*      Letter output

*      x, y : Position of starting the string.
*      h    : Caracter height in cm.
*      text : String. Horelis, quote and character*n variable are available.
*      angle: String angle in degrees.
*      n    : Character digits.

* Bugs : Unsupport some (many?) CALCOMP characters.

      subroutine SYMBOL (x, y, h, text, angle, n)
```

```
      real            x, y, h, angle, xy(2), radian
      character       text(80)
      integer         n, digit, workx, worky

      real              lend(2)
      common /symbols/ lend(2)

      real          sfact
      integer       corg(2), cpenp(2)
      common        /lips/ corg(2), cpenp(2), sfact

      character     lps(100), is2
      data          is2 /z'1e'/

      call rotfnt ( angle )              ! Set font orientation vector.

      call fhigh( h )                    ! Set font height in cm

      lps(1)    =  '4'
      lps(2)    =  '0'
      digit     =   2

      workx     =  int ( x * sfact ) + corg(1)
      worky     =  int ( y * sfact ) + corg(2)

      call  declps(workx,     lps, digit)
      call  declps(worky,     lps, digit)

      digit      = digit + 1
      lps(digit) = '1'

      ldigit     = 0

      if ( n .eq. 0 ) then ! Check the numbers of the Letters
                           ! uses a '$' character.
         Do 10 I = 1, 80
            if ( ichar( text(i) ) .eq. 36 ) goto 99
            ldigit = ldigit + 1
10       continue
99       continue
      else
         ldigit = n
      endif

      write(unit=31, fmt=6000)  (lps(i), i=1, digit),
     $                          (text(i), i=1, ldigit), is2
6000  format(100A, 100A, a1)

      wllen   = ldigit * h * 0.6
      lend(1) = x + int ( wllen * cos ( angle ) )
      lend(2) = y + int ( wllen * sin ( angle ) )

      return
      end

*********************************************************************
* CIRCLE : CALCOMP

*          x,  y    :  Center coordinate.
*          ab, ae   :  Angle of Beginnig and End.  In DEGREES.
*          ra, rb   :  Radius of Beginning and End.  Currently rb = ra.
*          c        :  Line type.
*                       0.0  : solid line   0.5 : broken line
*          n        :  Center line or not.
*                       2   : with center line
*                       3   : without center line

* Bugs : Ignore the variable-radius and center lines (Sorry...).

      subroutine CIRCLE(x, y, ab, ae, ra, rb, c, n)

      real          x, y, ab, ae, ra, rb, c
      integer       xy(2), iradiu, n, digit

      real          cpenw
      common        /pen/ cpenw

      real          sfact
      integer       corg(2), cpenp(2)
      common        /lips/ corg(2), cpenp(2), sfact

      character     lps(100), is2
      data          is2 /z'1e'/

      xy(1)  = int (  x * sfact )
      xy(2)  = int (  y * sfact )

      iab    = int ( ab )
      iae    = int ( ae ) - iab
      iradiu = int ( ra * sfact )

      call relpos( xy )

      lps(1) = '}'
      lps(2) = '6'
      lps(3) = '0'      ! always a open circle.
      digit  =  3

      call  declps(xy(1),    lps, digit)
      call  declps(xy(2),    lps, digit)
      call  declps(iradiu,   lps, digit)
      call  declps(iab,      lps, digit)
      call  declps(iae,      lps, digit)

      digit      = digit + 1
      lps(digit) = is2

      if ( c .gt. 0.1 ) then
         if ( cpenw .lt. 0 ) then ! To activate this instruction.
            call newpen ( 0.01 )
         endif
         call ltype(-2)          ! Set broken line
      endif

      write(unit=31, fmt='(100A)') (lps(i), i=1, digit)

      if ( c .gt. 0.1 ) then
         call ltype(0)           ! Reset broken line
      endif

      return
      end

*********************************************************************
* RECT : CALCOMP
*      Drawing a rectangular form.

*      x, y,   : starting point.
*      ht, wd  : height and width
*      angle   : The angle of base vector relative to ( 1, 0 ) vector.

      subroutine RECT ( x, y, ht, wd, angle )

      real          x, y, ht, wd, a, angle, radian
      integer       xyl(2), xy(4,2), digit
      integer       wx, wy

      real          sfact
      integer       corg(2), cpenp(2)
      common        /lips/ corg(2), cpenp(2), sfact

      character     lps(100), is2
      data          is2 /z'1e'/

      radian     = 3.141592654 / 180.

      angle      = mod ( angle, 360. ) * radian

      xy(1,1) = int ( x * sfact )
      xy(1,2) = int ( y * sfact )

      wx         = int( wd  * sfact)
      xy(2,1) = cos ( angle ) * wx + xy(1,1)
      xy(2,2) = sin ( angle ) * wx + xy(1,2)

      wx         = int( wd * sfact)
      wy         = int( ht * sfact)
      xy(3,1) = cos ( angle ) * wx - sin ( angle ) * wy + xy(1,1)
      xy(3,2) = sin ( angle ) * wx + cos ( angle ) * wy + xy(1,2)

      wy         = int( ht * sfact)
      xy(4,1) = -1. * sin ( angle ) * wy + xy(1,1)
      xy(4,2) =       cos ( angle ) * wy + xy(1,2)

      Do 10, i = 1, 4
         xyl(1)   = xy(i,1)
         xyl(2)   = xy(i,2)
         call       relpos ( xyl )
         xy(i,1) = xyl(1)
         xy(i,2) = xyl(2)
10    continue
```

```
      lps(1) = '2'
      digit  =  1

      wx = xy(1, 1)
      wy = xy(1, 2)
      call declps(wx,  lps,  digit)
      call declps(wy,  lps,  digit)

      Do  20 i = 2,  4
         j = i - 1
         wx = xy(i, 1)  - xy(j, 1)
         wy = xy(i, 2)  - xy(j, 2)
         call declps(wx,  lps,  digit)
         call declps(wy,  lps,  digit)
 20   continue

      digit      = digit + 1
      lps(digit) = is2

      write(unit=31, fmt='(100A)')  (lps(i), i=1, digit)

      return
      end

*************************************************************************
* NUMBER : CALCOMP  ( from CALCOMP.F in Izumi's RIETAN)

c     x, y : coordinates of the starting point of the number in cm
c     h    : hight of the font in cm
c     num : the value of the number to be printed
c     angle : angle in degree
c     ndec : the number of characters after '.'
c

      subroutine NUMBER(x, y, h, num, angle, ndec)

      real              x,  y,  h,  num,  angle
      integer           ndec

      character*80      text,  otext
      integer           name,  ldev

      real              lend(2)
      common /symbols/  lend(2)

      data pi/3.141592/

      if(ndec.ge.0)  then
*           unit = text ==> text for MacFortran II                    ,
            if(ndec.eq.0)  write(text,  '(f12.0)')  num
            if(ndec.eq.1)  write(text,  '(f12.1)')  num
            if(ndec.eq.2)  write(text,  '(f12.2)')  num
            if(ndec.eq.3)  write(text,  '(f12.3)')  num
            if(ndec.eq.4)  write(text,  '(f12.4)')  num
            if(ndec.eq.5)  write(text,  '(f12.5)')  num
            if(ndec.eq.6)  write(text,  '(f12.6)')  num
            if(ndec.eq.7)  write(text,  '(f12.7)')  num
            if(ndec.eq.8)  write(text,  '(f12.8)')  num
            if(ndec.eq.9)  write(text,  '(f12.9)')  num
      else
            inum = num*10**(ndec+1)
           write(text,  '(i12)')  inum
      end if

*     Compress space

      j = 0
      do 100 i=1, 12
      if(text(i:i).ne.' ')  then
           j = j + 1
           otext(j:j) = text(i:i)
      end if
 100  continue
      otext(j+1:j+1) = char(0)

      if  ( ( x .eq. 999. )  .and.  ( y .eq. 999. ) )  then
         x = lend(1)
         y = lend(2)
      endif

      call SYMBOL(x, y, h, otext, angle, j)

      return
      end

*************************************************************************
```

```
* LINE : CALCOMP

*     xarray, yarray : Data Array
*     npts           : Number of points.
*     k              : Ignored.
*     lintype        : 0      : line only
*                      Other  : Set symbol type. From -4 to 4.
*     inteqv         : Symbol size in cm dimention. Original meaning is
*                      lost.

* Bugs : All positive and negative lintype will be
*                ignored by this routine.

      subroutine LINE ( xarray, yarray, npts, k, lintyp, inteqv )

      real         xarray(10000), yarray(10000), inteqv
      integer      npts, k, digit, xy(2), workx, worky
      integer      lintyp, svpenp(2)

      real         sfact
      integer      corg(2), cpenp(2)
      common       /lips/ corg(2), cpenp(2), sfact

      character    lps(50000), is2
      data         is2 /z'1e'/

* Save the current pen position for line connection
      svpenp(1) = cpenp(1)
      svpenp(2) = cpenp(2)

      if ( lintyp .eq. 0 ) goto 99    ! Skip Mark Plot

      call setmks ( inteqv )
      call setmkt ( lintyp )

      lps(1) = '0'
      digit  =  1

      Do  10  i = 1, npts

         xy(1)  =  int( xarray(i) * sfact ) + corg(1)
         xy(2)  =  int( yarray(i) * sfact ) + corg(2)

         call relpos( xy )

         workx = xy(1)  - cpenp(1)
         worky = xy(2)  - cpenp(2)
         if ( (abs(workx) .lt. 10) .and. (abs(worky) .lt. 10) ) then
            workx = 10
            worky = 10
         endif

         call  declps(workx,     lps, digit)
         call  declps(worky,     lps, digit)

         call stpenp( xy )       ! set current penpos

 10   continue

      digit      = digit + 1
      lps(digit) = is2

      write(unit=31, fmt='(50000A)')  (lps(i), i=1, digit)

 99   continue

* Draw lines.
* Resave the current pen position.
      cpenp(1) = svpenp(1)
      cpenp(2) = svpenp(2)

      lps(1) = '1'
      digit  =  1

      Do  20  i = 1, npts

         xy(1)  =  int( xarray(i) * sfact ) + corg(1)
         xy(2)  =  int( yarray(i) * sfact ) + corg(2)

         call relpos( xy )

         workx = xy(1)  - cpenp(1)
         worky = xy(2)  - cpenp(2)
         if ( (abs(workx) .lt. 10) .and. (abs(worky) .lt. 10) ) then
            workx = 10
            worky = 10
```

```
      endif

      call  declps(workx,     lps, digit)
      call  declps(worky,     lps, digit)

      call stpenp( xy )       ! set current penpos

 20   continue

      digit      = digit + 1
      lps(digit) = is2

      write(unit=31, fmt='(50000A)')  (lps(i), i=1, digit)

* Resave the current pen position.
      cpenp(1) = svpenp(1)
      cpenp(2) = svpenp(2)

      return
      end

*************************************************************************
* Set line type : LIB internal.

      subroutine ltype(n)

      integer     n, wn, digit

      character   lps(100), is2
      data        is2 /z'1e'/

      lps(1) = 'E'
      lps(2) = '1'
      digit  =  2

      call declps(n, lps, digit)

      digit      = digit + 1
      lps(digit) = is2

      write(unit=31, fmt='(100A)')  (lps(i), i=1, digit)

      return
      end

*************************************************************************
* Set current origine : LIB internal.

      subroutine stcorg(xy)

      integer     xy(2), work

      real        sfact
      integer     corg(2), cpenp(2)
      common      /lips/ corg(2), cpenp(2), sfact

      Do 10 i = 1, 2
         work    = xy(i)
         corg(i) = work
 10   continue

      return
      end


*************************************************************************
* Set pen position : LIB internal.
      subroutine stpenp(xy)

      integer     xy(2), work

      real        sfact
      integer     corg(2), cpenp(2)
      common      /lips/ corg(2), cpenp(2), sfact

      Do 10 i = 1, 2
         work    = xy(i)
         cpenp(i) = work
 10   continue

      return
      end

*************************************************************************
* Convert to relarive position : LIB internal.

      subroutine relpos(xy)

      integer     xy(2), work

      real        sfact
      integer     corg(2), cpenp(2)
      common      /lips/ corg(2), cpenp(2), sfact

      Do 10  i = 1, 2
         xy(i)  = xy(i) + corg(i)
 10   continue

      return
      end

*************************************************************************
* Set Vector-mode origine. : LIB internal

      subroutine setvd(x1, y1, x2, y2)

      integer     x1, y1, x2, y2, digit

      character   lps(100), is2
      data        is2 /z'1e'/

      lps(1) = '('
      digit  =  1

      call  declps(x1, lps, digit)
      call  declps(y1, lps, digit)
      call  declps(x2, lps, digit)
      call  declps(y2, lps, digit)

      digit      = digit + 1
      lps(digit) = is2

      write(unit=31, fmt='(100A)')  (lps(i), i=1, digit)

      return
      end

*************************************************************************
* Move Vector-mode origine. : LIB internal

      subroutine movvd(x1, y1)

      integer     x1, y1, digit

      character   lps(100), is2
      data        is2 /z'1e'/

      lps(1) = 'L'
      lps(2) = '
      digit  =  2

      call  declps(x1, lps, digit)
      call  declps(y1, lps, digit)

      digit      = digit + 1
      lps(digit) = is2

      write(unit=31, fmt='(100A)')  (lps(i), i=1, digit)

      return
      end

*************************************************************************
* Convert the decinal numbers to LIPS3 internal expression.

      subroutine declps(num, lps, digit)

      integer       num, decnum, tcode, ccode, bope, digit, idigit

      character     lps(100), work(10)
      data          ccode /64/

      decnum = num
      digit  = digit + 1

      if ( decnum .eq. 0 ) then
         lps(digit) = char(48)    ! 30h (+0)
         digit      = digit + 1
         goto 999
      else if ( decnum .gt. 0 ) then
         tcode  = 48   ! Positive terminate code.
      else
```

```
         tcode  = 32
      endif

      decnum = abs(decnum)
      idigit = 1

      bope          = and(decnum, 15)
      bope          = bope + tcode
      work(idigit)  = char(bope)
      idigit        = idigit + 1

      decnum = decnum / 16   ! shift to right by 4 bits

      Do 10  i = 2, 4
         bope          = and(decnum, 63)
         bope          = bope + ccode
         work(idigit)  = char(bope)
         decnum        = decnum / 64 ! Shift to right by 6 bits.
         if ( decnum .eq. 0 ) goto 998
         idigit        = idigit + 1
 10   continue

 998  continue

      Do 20 i = idigit, 1, -1
         lps(digit) = work(i)
         digit      = digit + 1
 20   continue

 999  continue

      digit = digit - 1

      return
      end

*************************************************************************
*  LIPS3 Start a Job : LIB internal.

      subroutine l3saj
      character esc
      data      esc /o'033'/

      write (unit=31, fmt=5990) esc
 5990 format (a1, '[%@')                          ! To Text mode from the emulator
      write (unit=31, fmt=6000) esc
 6000 format (a1, 'P31;300;1J')                   ! Mode setup
      write (unit=31, fmt=6010) esc
 6010 format('CALCOMP',a1, '¥')                   ! Job Identifire
      write (unit=31, fmt=6020) esc
 6020 format(a1, '<')                             ! Software Reset
      write (unit=31, fmt=6022) esc
 6022 format(a1, '[0p')                           ! Select current page format.
      write (unit=31, fmt=6030) esc, esc
 6030 format(a1, 'P1y', 'CALCOMP to LIPS', a1, '¥')    ! Display a string
      write (unit=31, fmt=6040) esc, esc
 6040 format(a1, 'Pz', 'ALP1125 ROMA', a1, '¥')        ! Select a charact set name
      write (unit=31, fmt=6050) esc
 6050 format(a1, '[4 L')                          ! Select the line pitch to 8 lines
r inch

      return
      end

*************************************************************************
* LIPS3 End of Job : LIB internal

      subroutine l3eoj
      character esc
      data      esc /o'033'/

      write  (unit=31,fmt=6000) esc, esc
 6000 format (a1, 'POJ', a1, '¥')

      return
      end

*************************************************************************
* Determin font orientation : LIB internal.
*          (Up vector),  (Base vector)

      subroutine forien ( x1, y1, x2, y2 )

      real        x1, y1, x2, y2
      integer     wx1, wy1, wx2, wy2
      integer     digit
      character   lps(100), is2
      data        is2 /z'1e'/

      f(x)    =  int ( x * 100 )

      lps(1) = 'Z'
      digit  =  1

      wx1 = f(x1)
      wy1 = f(y1)
      wx2 = f(x2)
      wy2 = f(y2)

      call declps(wx1, lps, digit)
      call declps(wy1, lps, digit)
      call declps(wx2, lps, digit)
      call declps(wy2, lps, digit)

      digit      = digit + 1
      lps(digit) = is2

      write(unit=31, fmt='(100A)')  (lps(i), i=1, digit)

      return
      end

*************************************************************************
* Define Font rotation vector : LIB internal

*   angle : in degrees

      subroutine rotfnt ( angle )

      real        x1, y1, x2, y2, angle, radian

      radian    = 3.141592654 / 180.

      angle     = mod ( angle, 360. )
      angle     =  ( angle * radian )

      x1 = cos ( angle ) *  100.       ! Base vector ( 1, 0 )
      y1 = sin ( angle ) *  100.
      x2 = sin ( angle ) * -100.       ! UP vector ( 0, 1 )
      y2 = cos ( angle ) *  100.

      call forien ( x2, y2, x1, y1 )     ! (Up vector), (Base vector)

      return
      end

*************************************************************************
* Select a vector-mode coordinate system : LIB internal.

      subroutine selvd(basic, mag)

      integer     basic, mag, digit

      character   is2, lps(100)
      data        is2 /z'1e'/

      lps(1) = '1'
      lps(2) = '0'
      digit  =  2

      call  declps(basic, lps, digit)
      call  declps(mag,   lps, digit)

      digit      = digit + 1
      lps(digit) = is2

      write(unit=31, fmt='(100A)')  (lps(i), i=1, digit)

      return
      end

*************************************************************************
* ASYMBOL : Special feature of this library.
*           Write a affin tranceformed string.

*       bangle :  base vector angle in degrees
*       uangle :  up vector offset
*                 angle from base vector
*       ratio  : (up vector)/(base vector)

      subroutine ASYMBOL (x, y, h, text, bangle, uangle, ratio, n)
```

```
      real            x, y, h, bangle, uangle, xy(2), ratio
      character       text(80)
      integer         n, digit, workx, worky

      real        sfact
      integer     corg(2), cpenp(2)
      common      /lips/ corg(2), cpenp(2), sfact

      character   lps(100), is2
      data        is2 /z'1e'/

      call faffin( bangle, uangle, ratio )
                                    ! Set font affin transformation

      call fhigh( h )               ! Set font height in cm

      lps(1)    =  '4'
      lps(2)    =  '0'
      digit     =   2

      workx     =  int ( x * sfact ) + corg(1)
      worky     =  int ( y * sfact ) + corg(2)

      call  declps(workx,     lps, digit)
      call  declps(worky,     lps, digit)

      digit       = digit + 1
      lps(digit)  = '1'

      ldigit      = 0

      if ( n .eq. 0 ) then ! Check the numbers of the Letters.
                           ! uses a '$' character.
         Do 10 I = 1, 80
            if ( ichar( text(i) ) .eq. 36 ) goto 99
            ldigit = ldigit + 1
10       continue
99       continue
      else
         ldigit = n
      endif

      write(unit=31, fmt=6000) (lps(i), i=1, digit),
     $                         (text(i), i=1, ldigit), is2
6000  format(100A, 100A, a1)

      return
      end

*************************************************************************
* Define Font Affin Transformation.

*     bangle : base vector (1, 0) angle in degrees.
*     uangle : up vector (0, 1) angle in degrees
*              offset from the base vector.

      subroutine faffin ( bangle, uangle, ratio )

      real        bangle, uangle, radian, ratio
      real        x1, y1, x2, y2

      radian    = 3.141592654 / 180.

      uangle     = uangle + bangle - 90

      bangle     = mod ( bangle, 360. ) * radian
      uangle     = mod ( uangle, 360. ) * radian

      x1 = cos ( bangle ) *  100.          ! Base vector ( 1, 0 )
      y1 = sin ( bangle ) *  100.
      x2 = sin ( uangle ) * -100. * ratio ! UP vector ( 0, 1 )
      y2 = cos ( uangle ) *  100. * ratio

      call forien ( x2, y2, x1, y1 )     ! (Up vector), (Base vector)

      return
      end

*************************************************************************
* Set Font height : LIB internal

      subroutine fhigh ( h )

      real        h
      integer     wh, digit

      character   lps(100), is2
      data        is2 /z'1e'/

      real        sfact
      integer     corg(2), cpenp(2)
      common      /lips/ corg(2), cpenp(2), sfact

      wh      =  int ( h * sfact )          ! in cm

      lps(1) = 'Y'
      digit  =  1

      call declps(wh, lps, digit)

      digit       = digit + 1
      lps(digit)  = is2

      write(unit=31, fmt='(100A)') (lps(i), i=1, digit)

      return
      end

*************************************************************************
* Set Marker SIZE : LIB internal.

*     dim : in cm

      subroutine setmks ( dim )

      real        dim
      integer     digit, wkdim

      real        sfact
      integer     corg(2), cpenp(2)
      common      /lips/ corg(2), cpenp(2), sfact

      character   lps(100), is2
      data        is2 /z'1e'/

      wkdim  = int ( dim * sfact )

      lps(1) = 'B'
      digit  =  1

      call  declps(wkdim, lps, digit)

      digit       = digit + 1
      lps(digit)  = is2

      write(unit=31, fmt='(100A)') (lps(i), i=1, digit)

      return
      end


*************************************************************************
* Set Marker TYPE : LIB internal

*     1  : Plus Sign          2  : Aster
*     3  : Open circle        4  : Cross
*    -1  : Open Rect.        -2  : Solid Rect.
*    -3  : Open triangle     -4  : Solid triangle

      subroutine setmkt ( spc )

      integer     spc, digit

      character   lps(100), is2
      data        is2 /z'1e'/

      lps(1) = 'A'
      digit  =  1

      digit  = digit + 1

      if ( spc .gt. 0 ) then
         spc = spc + 48
      else
         spc = abs(spc) + 32
      endif

      lps(digit) = char(spc)

      digit       = digit + 1
      lps(digit)  = is2
```

```
      write(unit=31, fmt='(100A)')  (lps(i), i=1, digit)

      return
      end

*************************************************************************
* AXIS : CALCOMP
*   From Izumi's Fat-Rietan package.
c   Write a scale for an axis

c         xmin, ymin : coordinates of the first scale
c         leng       : length of the axis
c         ang        : angle from hrizontal line (deg)
c         delta      : increments of scale

      subroutine AXIS(xmin, ymin, leng, delta, ang, iside)

      real      ax(4), ay(4), leng
      data      pi2 /6.28318/

      ax(3) = 0.0
      ay(3) = 0.0
      ax(4) = 1.0
      ay(4) = 1.0
      im    = leng / delta + 0.001

      Do 60 i = 1, im
           ax(1) = xmin + delta * float(i) * cos(pi2*ang/360)
           ay(1) = ymin + delta * float(i) * sin(pi2*ang/360)
           ax(2) = ax(1) -0.1*sin(pi2*ang/360)*sign(1, iside)
           ay(2) = ay(1) +0.1*cos(pi2*ang/360)*sign(1, iside)
 60        call LINE ( ax, ay, 2, 1, 0, 0 )

      return
      end

*************************************************************************
* SCALE : CALCOMP

      subroutine SCALE ( array, axislg, nrpnts, k, i )
      end
```

付録 B. CALLIP を使った出力例

図 B.1 は CALLIP を使って LIPS3 プリンタ用に移植した ORTEP2 (Johnson, 1965) による出力である。使用プリンタは CANON LBP-B406S (LIPS3, 300DPI)、原寸大である。

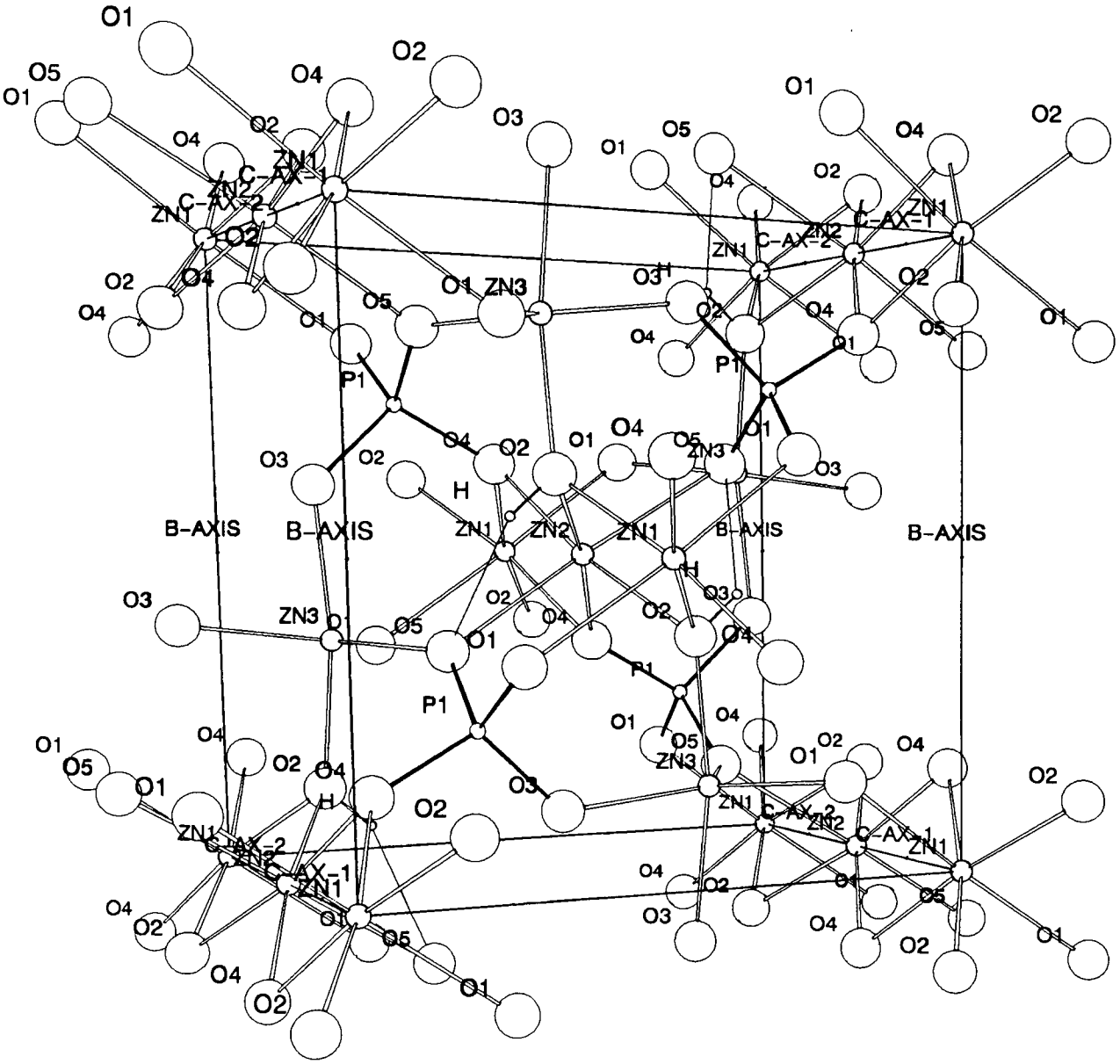

図 B.1