UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

FLOWCHART

A COMPUTER PROGRAM FOR PLOTTING FLOWCHARTS

by Bernice Bender

Open File Report 82-999

1982

FLOWCHART
A Computer Program for Plotting Flowcharts

by Bernice Bender

Abstract

The computer program FLOWCHART can be used to very quickly and easily produce flowcharts of high quality for publication.

FLOWCHART centers each element or block of text that it processes on one of a set of (imaginary) vertical lines. It can enclose a text block in a rectangle, circle or other selected figure. It can draw a line connecting the midpoint of any side of any figure with the midpoint of any side of any other figure and insert an arrow pointing in the direction of flow. It can write "yes" or "no" next to the line joining two figures.

FLOWCHART creates flowcharts using some basic plotting subroutines[1] which permit plots to be generated interactively and inspected on a Tektronix compatible graphics screen or plotted in a deferred mode on a Houston Instruments 42" pen plotter. The size of the plot, character set and character height in inches are inputs to the program. Plots generated using the pen plotter can be up to 42" high--the larger size plots being directly usable as visual aids in a talk.

FLOWCHART centers each block of text on an imaginary column line. (The number of columns and column width are specified as input.) The midpoint of the longest line of text within the block is defined to be the center of the block and is placed on the column line. The spacing of individual words within the block is not altered when the block is positioned.

The program writes the first block of text in a designated column and continues placing each subsequent block below the previous block in the same column. A block of text may be placed in a different column by specifying the number of the column and an earlier block of text with which the new block is to be aligned. If block zero is given as the earlier block, the new text is placed in the new column continuing down the page below the previous block. Optionally a column and number of inches from the top of the page may be given for positioning the next block of text.

The program will normally draw one of five types of figure to enclose a block of text: a rectangle, circle, diamond, eight sided figure or figure with parallel sides and rounded ends. It can connect the figure with a line to the preceeding figure, and place an arrow pointing toward the second figure. Text blocks not in sequence can also be connected and "yes" or "no" written next to any line to indicate branching.

Figure 1 illustrates the various types of figures that can be drawn, spacings, connecting lines and the like.

[1] The plotting package employed is Buplot available on the VAX and PDP-1170 computers at the USGS Office of Earthquake Studies, Golden, Colo. Calls to the plotting subroutines must be adjusted if some other plotting package is used.

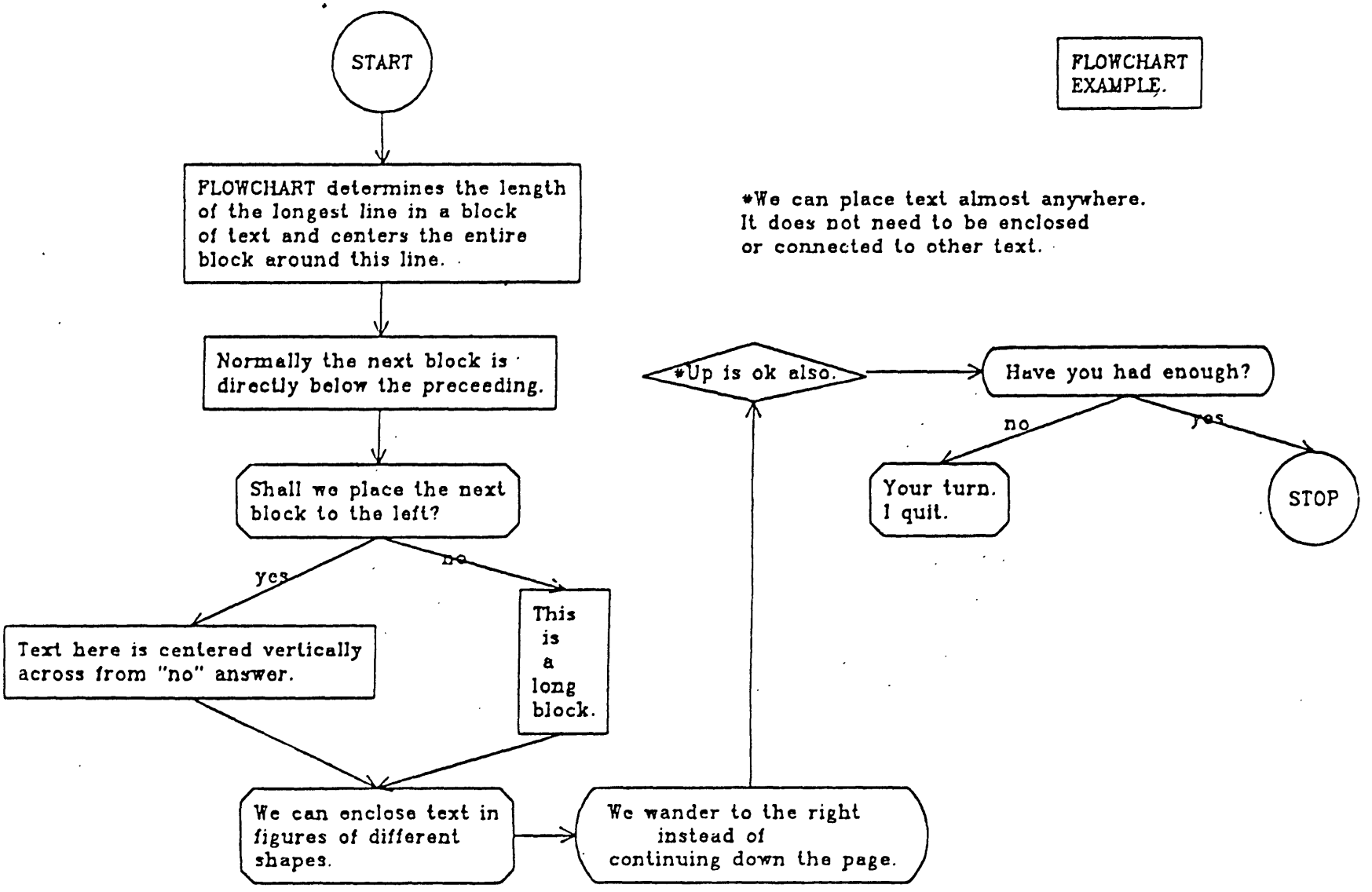


Figure 1. Example of a flowchart created using the program FLOWCHART

Inputs

FLOWCHART obtains the text to be diagrammed, positioning instructions and the like from a data file. The inputs used to generate the flowchart in figure 1 are given in Appendix A and illustrate the instructions defined below.

All inputs are free field. An input that is "blank" may be omitted. Although normally on the computer employed free field character inputs must be enclosed with a single quote (for example, the character s would normally be input as 's'), quotes are not necessary here. One or more blanks or commas can be used to separate inputs on a line but delimiters are required only between two successive numbers, not between two letters or numbers followed by or preceeding letters (for example, both 1ip and 1 i p are acceptable.)

First Input Line: (size and spacing initialization)

icol, colw, y, ht

icol: column on which first block of text is to be centered
colw: width of one column in inches
y: page length (or height) in inches
ht: character height in inches (this text is .10 inches in height).

(Figure 2 illustrates the parameters icol, colw and ht.)

Subsequent inputs are of two types, either text type or instruction type. Inputs generally consist of two or more lines.

Text Type Input:

The first line of this type of instruction tells how many lines of text follow, whether to enclose the text in a box, rectangle, circle or some other figure, whether to connect this block of text to the preceeding block, and (optionally) provides a number to identify this block of text. Subsequent lines contain the text to be plotted.

(Line 1)

ict, ibcd, ln, nument

ict: number of lines in block of text that follows
ibcd: type of figure in which this text is to be enclosed (see figure 1).

- b: box or rectangle
- c: circle
- d: diamond
- r: figure with parallel sides and rounded ends
- o: eight sided figure
- blank: no enclosure

The current block of text will normally be connected with a straight line to the last block read in, and an arrow head will point to the current block.

ln:

- x: omit line.
- y: write "yes" on the line joining this block with the preceeding
- n: write "no"
- a: join this block of text to preceeding block, but omit arrow head (writing "yes" or "no" is not permitted in this case)
- blank: line and arrow are drawn, but neither "yes" nor "no" is written.

(The vertical dashed lines were not done by the program.
They were inserted later to indicate column spacings.)

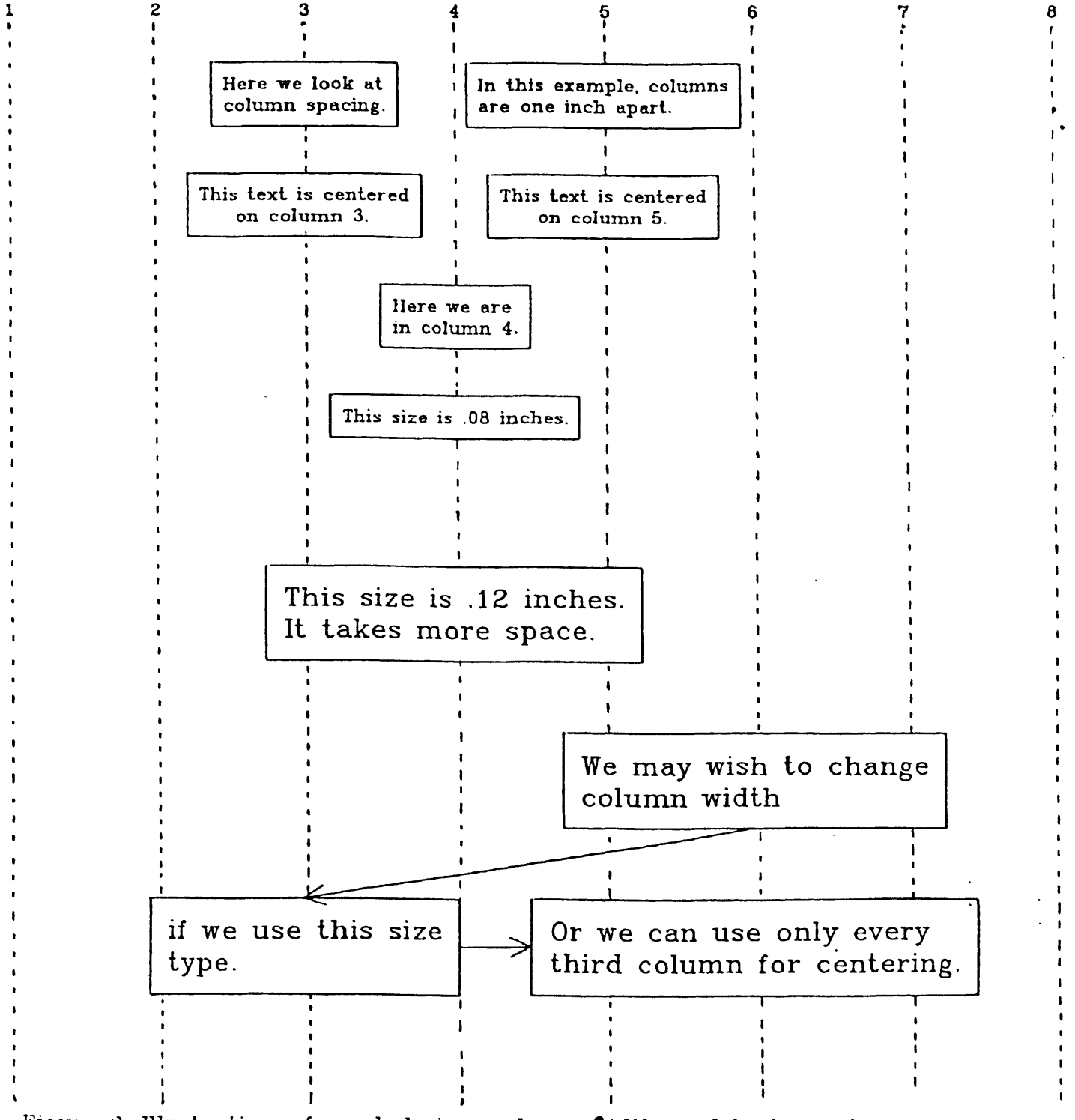


Figure 2. Illustration of symbol size, column width, and text spacing.

nument:

num: number that identifies this block of text.

blank: number blocks sequentially.

Numbers are used for identifying blocks in instructions for positioning and connecting text. Note that if the default (sequential) numbering is used and blocks are inserted or deleted, the remaining text blocks will be assigned new numbers.

Next ict lines:

Block of text (ict lines) to be inserted. Blank text lines are permitted; a single blank line (ict=1) may be used to define a position for one end of a straight line segment. (For examples, see figure 3 and figure 4).

Text will be reproduced in the flowchart as it appears in the data file; no spacing adjustments will be made within blocks.

Instruction Type Input:

Normally subsequent blocks of text will be positioned in a single column, with each subsequent block of text placed below the preceeding block. The "instruction" lines enable one to place a new block of text in a different postion (say in a new column across from a block previously entered), or to selectively connect blocks that were not entered in sequence. It should be noted that the program performs operations as it encounters them. Therefore, it cannot, for example, connect blocks of text that have not already been entered as input.

An "instruction" input also enables one to change character size or advance to a new page (begin a new flowchart).

(Line 1)

ict, ibcd, ln, yn

ict: Not used. ict=0 will terminate this run, so set ict=1 or some other integer.

ibcd:

i: indicates this is an "instruction" rather than "text" type input.

ln: type of instruction (further explained below)

- c: connect two figures with a line
- p: position the next block of text across from a previous block
- q: position the next block of text the designated number of inches from the top of the page.
- a: advance to the next page
- s: change character size

yn: (used only with the c(onnect) instruction)

- y: write "yes " on the line joining the two figures specified
- n: write "no" on the line joining the two figures
- a: omit arrow head on line (cannot write "yes" or "no" in this case)

blank: do not write anything

All but the "advance" instruction require a second line. The second line of input for an instruction is as follows:

If ln=c: (connect two blocks with a line) next input is:

p1 n11 p2 n12

p1: n(north), e(east), s(south), w(west)

n11: number identifying first block of text

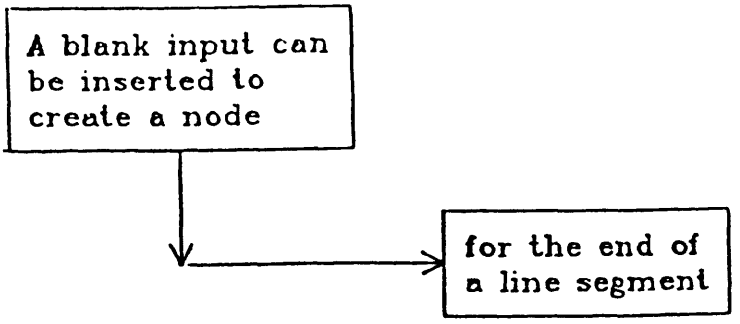
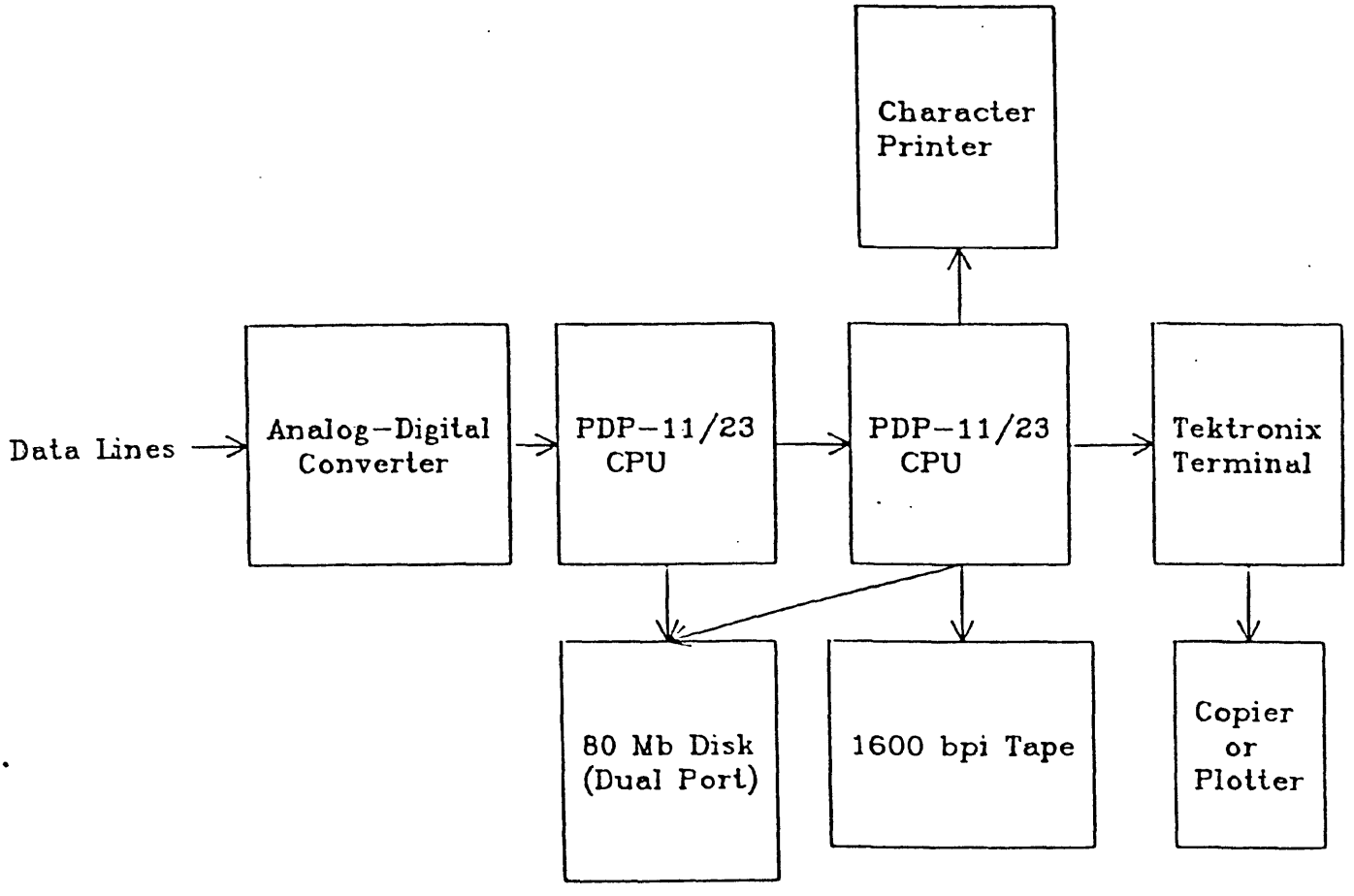


Figure 3. The data file for this illustration contains four blocks of text. The second block consists of a single blank line and is used to create an end point or node for a line segment. This explanation is the fourth block.



PROPOSED SYSTEM FOR PANAMA NETWORK

Figure 4. Blank lines are inserted in blocks of text to obtain the spacings.

p2: n(north), east(east), s(south), w(west)
n12: number identifying second block of text
Connect a line from the center of the p1 edge of the figure containing text labeled n11 with the center of the p2 edge of the figure identified by n12. An arrow head on the line points to figure n12 unless input yn=a.

If ln=p (positioning instruction) next input is:
newcol n12
newcol: column in which next text entry is to be positioned
n12: center text in column newcol across from text entry labeled n12.

If ln=q (positioning) next input is:
newcol yval
newcol: column in which next text entry is to be positioned
yval: place top of next block of text yval inches down from the top of the page.

If ln=a (advance):
No second line. This is an instruction to advance the frame or begin a new page. The top of the next block is placed at the top of the new page in the same column that it would have occupied on the preceeding page. All positioning and connecting instructions relating to blocks on the current page must be completed before advancing to a new page.

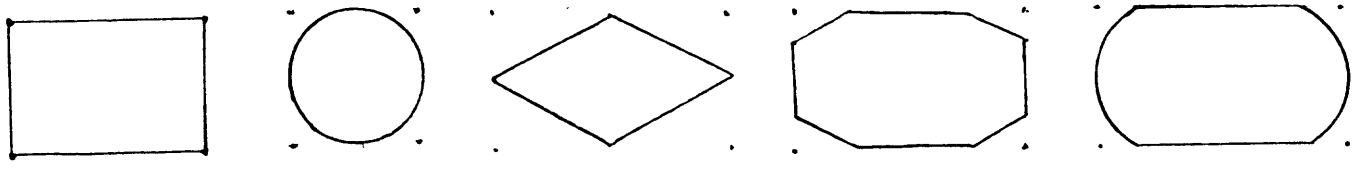
The program will automatically begin a new page if a text entry would have been positioned below the bottom of the current page. Placing text on a new page leads to trouble if one tries to connect text blocks that are on separate pages.

If ln=s (change character size) next input is:
ht
ht: new character size in inches for all text that follows.

The input is terminated by setting ict=0.

Debugging:

It is possible to run the program in such a manner that text inputs and spacing instructions are printed out, but no plots are drawn. This debugging mode is helpful in locating input errors. If the program terminates due to an input error, the last line printed should be within one line of the error. This mode can also provide information that may be used in selecting column width and positioning text. Printed are the block of text, and
nument: number identifiying block of text
xval: center of block (inches)
ysav: top of this block of text in inches (from top of page)
rmax: length of longest line of text in this block
xmi,xpl,ymi,ypl: coordinates of the corner points of the smallest rectangle containing the figure surrounding this block of text.



Running the program.

The input text and instructions are assumed to be available on an input data file. You will be asked at the terminal to:

"Enter name of file containing flowchart input"

Several questions must also be answered at the terminal at run time. These are:

"Is this a debug run" answer y(es) or n(o)"
(Debug run gives only printout as described above in discussion under debugging.)

The next two questions are asked only if this not a debug run.
"Enter print type--use 1 unless final copy--then 2"
In the plotting package used, print type 1 is the standard character set; it is faster and more compact than type 2 which features shading and seraphs, and is suitable for publication.
Print type 1 should be used for all but the final copy.

"Plot mode (i=immediate, d=deferred, e=edit)"
i=immediate gives immediate output on the Tektronix
d=defers output to file pltfl.dat
e: edit, gives immediate output which can optionally be saved on pltfl.dat.

A listing of FLOWCHART is provided in Appendix B. The questions that must be answered at the terminal may need to be altered within the program depending on the plotting package employed and graphics hardware available.

Appendix A

Inputs on for010.dat for flowchart shown in Figure 1

```
4 1. 7. .07
1 c 1
START
4 b 2
FLOWCHART determines the length
of the longest line in a block
of text and centers the entire
block around this line.
2 b 3
Normally the next block is
directly below the preceeding.
2 o 4
Shall we place the next
block to the left?
1 i p
5  0
5 b 5
This
 is
 a
long
block.
1 i c n
s 4 n 5
1 i p
4  0
3 o 6
We can enclose text in
figures of different
shapes.
1 i p
6  6
3 r 7
We wander to the right
      instead of
continuing down the page.
1 i p
6 3
1 d 8
*Up is ok also.
1 i p
8 3
1 r 9
Have you had enough?
1ip
9 4
1 c x 10
STOP
1 i c y
s 9 n 10
1 i p
7 10
2 o x 11
Your turn.
I quit.
1 i c n
s 9 n 11
1 i p
7 2
3  x 12
*We can place text almost anywhere.
It does not need to be enclosed
or connected to other text.
1 i q
8 0.
2 b x
FLOWCHART
EXAMPLE.
1 i p
3 5
2 b x
Text here is centered vertically
across from "no" answer.
1 i c y
s 4 n 14
1 i c
s 14 n 6
0
```

Appendix B

FLOWCHART: Computer Program Listing

```
c                    FLOWCHART
c           Program for drawing flowcharts
c
      character*80 title,buf(100),ro(80)
      common/ara/iara,jchar
       common/jnum/jnum(6)
       character iara(80),jchar(6)
      character*1 ibcd,debug,ln,yn,p1,p2,p3,p4,tlen
      dimension xplsv(100),yplsv(100),xmisv(100),ymisv(100)
      common/xypm/xpl,xmi,ypl,ymi,ht
      dimension icols(100),yss(100)
      character*20 filen
       common/pp/tlen(80)
      print 10
   10 format(' Enter name of file containing flowchart input')
      read 20, filen
   20 format(a20)
      open(unit=10,file=filen,status='old',err=30)
      go to 50
   30 print 40
   40 format(' error accessing file-check filename')
      call exit
   50 print 60
   60 format(' Is this a debug run-answer y(es) or n(o)')
      read 90,debug
       if(debug.eq.'y') go to 80
      print 70
   70 format(' Enter print type--use 1 unless final copy-then 2')
      read*,iptyp
   80 read(10,*) icol,colw,yb,ht
   90 format(a1)
      ybotm=0.
      deltax=1.1*ht
      deltah=1.8*ht
      delth2=deltah/2.
      arrow=deltax
      arrow2=arrow/2.
      iadj=0
      nent=0
c     nent=number of entries thus far--records positions,etc
c      first read starting column and col width
      xcent=colw+(icol-1)*colw
      xcents=colw
      yypp=yb+1.
      yvalst=yb-10.*ht
  100 if (debug.ne.'y') go to 110
      call plots (1,yypp)
      go to 120
  110 call plots(5,yypp)
      call ctype(iptyp)
  120 call plot(.5,.5,-3)
  130 yval=yvalst
      isw=0
  140 read (10,440) (iara(iq),iq=1,80)
      call decod
      ict=jnum(1)
```

```
      ibcd=jchar(1)
      ln=jchar(2)
      yn=jchar(3)
      nument=jnum(2)
      if(ibcd.eq.'i')go to 400
      if (iadj.eq.1) yval=yval+ict*delth2
      ysav=yval
      iadj=0
      if(yval .gt.ybotm) go to 160
      if(debug.eq.'y') go to 150
      call hldplt(.6,.6,1,ro)
      call advplt
      call plots(0,yypp)
  150 yval=yvalst
      isw=0
  160 if(ict.eq.0) go to 390
      numsav=nent
      nent=nent+1
      if(nument.ne.0) nent=nument
      lsav=0
      rmax=0.
      do 190 i=1,ict
      read(10,170,end=390) buf(i)
      l=length(buf(i))
      if(l.gt.lsav) lsav=l
  170 format(a80)
      call symlen(ht,buf(i),l,rmin,rsav)
      if(rsav.gt.rmax) rmax=rsav
      if(debug.ne.'y') go to 190
      print 180,nent,(tlen(ii),ii=1,l)
  180 format(i5,1x,80a1)
  190 continue
      xval=xcent-rmax/2.
      do 200 i=1,ict
      if(debug.ne.'y') call symbol(xval,yval,ht,buf(i),0,lsav)
  200 yval=yval-deltah
c      yrt=right most symbol location
c      Set limits for enclosure around text thru
      if(rmax.ne.0.) go to 210
      xleft=xval
      xrt=xval
      go to 220
c           xleft,xrt,yleft,yrt
  210 xleft=xval-deltax
      xrt=rmax+xval+deltax
c    correct spacing for border around text
  220 yup=ysav+deltah+ht/2.
      ylow=yval+ht/2.
      icols(nent)=ict
      yss(nent)=ysav
      if(debug.ne.'y') go to 240
       print 230,nent,xval,ysav,rmax
  230 format(' nent,xval,ysav,rmax='i4,3f7.3)
  240 if(ibcd.ne.'r') go to 250
      call rndend(xleft,xrt,yup,ylow)
      go to 300
```

```
  250 if(ibcd.ne.'b' ) go to 260
c     enclose text in box
      call box(xleft,xrt,yup,ylow)
      go to 300
  260 if(ibcd.ne.'c') go to 270
c     enclose text in circle
      call circle(xleft,xrt,yup,ylow)
      go to 300
  270 if(ibcd.ne.'d') go to 280
c      enclose in diamond shaped figure.
      call diam(xleft,xrt,yup,ylow)
      go to 300
  280 if(ibcd.ne.'o') go to 290
c      enclose text in eight sided figure
      call oct(xleft,xrt,yup,ylow)
      go to 300
c     no enclosure--must still set  x,y values
  290 call nothing(xleft,xrt,yup,ylow)
c     this position was blank supposedly--so we must correct
      ln=jchar(1)
      yn=jchar(2)
c     save current positions for future spacing
  300 xplsv(nent)=xpl
      xmisv(nent)=xmi
      yplsv(nent)=ypl
      ymisv(nent)=ymi
      if(rmax.ne.0) go to 310
c      only entry was a blank line--used for later positioning
      yplsv(nent)=(ypl+ymi)/2.
      ymisv(nent)=yplsv(nent)
      ypl=yplsv(nent)
  310 if(debug.ne.'y') go to 330
      print 320,nent,xmi,xpl,ymi,ypl
  320 format(' nent='i3' limits at xmi,xpl,ymi,ypl='4f7.3)
       go to 380
  330 if(ln.eq.'x') go to 380
      if(isw.eq.0) go to 380
      if(isw.eq.2) go to 410
c     the default with iline=blank is to draw line
      call plot(xcent,ymisv(numsav),3)
      call plot(xcent,ypl,2)
c      draw arrow
      ycenta=ypl+arrow
      xcenta=xcent+arrow2
      if(ln.eq.'a') go to 340
      call plot(xcenta,ycenta,2)
      xcenta=xcent-arrow2
      call plot(xcent,ypl,3)
      call plot(xcenta,ycenta,2)
  340 continue
c      write 'yes' or 'no' for question branch?
  350 if(ln.ne.'y') go to 360
      title='yes'
      go to 370
  360 if(ln.ne.'n') go to 380
      title='no'
```

```
  370  ymid=(ymisv(numsav)+yp1)/2.
       xcentp=xcent+deltax
       call symbol(xcentp,ymid,ht,title,0,3)
  380 continue
       isw=1
       yval=ymi-3.5*deltah
       go to 140
  390 if(debug.ne.'y')call endplt
       call exit
c    change columns
  400 continue
       if(ln.ne.'a') go to 430
        if(debug.eq.'y') go to 100
       call hldplt(.6,.6,1,ro)
       call advplt
       go to 100
c     here we connect text in different columns
  410 isw=1
       yval=ymi-3.5*deltah
       n11=numsav
       n12=nent
       if(debug.ne.'y') go to 450
       print 420,n11,n12,p1,p2
  420 format('n11,n12,p1,p2='2i3,1x,a1,1x,a1)
       go to 450
c     change column, connect two boxes with a line or yes or no
  430 if(ln.ne.'c') go to 570
c    position
       read (10,440) (iara(iq),iq=1,80)
       call decod
       p1=jchar(1)
       p2=jchar(2)
       n11=jnum(1)
       n12=jnum(2)
c      blocks to connect (get number from printout if necessary)
  440 format(80a1)
  450 call sxy(xval1,yval1,xplsv,yplsv,xmisv,ymisv,n11,p1)
       if(debug.ne.'y') go to 470
       print 460,n11,xval1,yval1
       go to 480
  460 format(' n11,xval1,yval1='i5,2f7.3)
  470 call plot(xval1,yval1,3)
  480 call sxy(xval2,yval2,xplsv,yplsv,xmisv,ymisv,n12,p2)
       if(debug.ne.'y') go to 490
       print 460,n12,xval2,yval2
       go to 140
  490 call plot(xval2,yval2,2)
       if(yn.eq.'a') go to 500
       call aro(xval1,yval1,xval2,yval2,arrow)
  500 if(yn.eq.' ') go to 140
       if(yn.ne.'y') go to 510
       title='yes'
       go to 520
  510 if(yn.ne.'n') go to 140
       title='no '
  520 xv12=(xval1+xval2)/2.+deltax
```

```
      yv12=(yval1+yval2)/2.
       if(xval1.ne.xval2) xv12=xv12-3.*deltax
      call symbol(xv12,yv12,ht,title,0,3)
      go to 140
c     new y-position on page (in inches)--new column
  530 if(ln.ne.'q') go to 550
      read(10,*) newcol,yval
      xold=xcent
      xcent=xcents+colw*(newcol-1)
      yval=yvalst-yval
      if(debug.ne.'y') go to 600
      print 540,newcol,yval
  540 format(' position at col 'i2' at 'f6.3' inches')
      go to 600
  550 if (ln.ne.'s') go to 140
c     change character height
      read(10,*) ht
      deltah=1.8*ht
      deltax=1.1*ht
      delth2=deltah/2.
      arrow=deltax
      arrow2=arrow/2
      if(debug.ne.'y') go to 140
      print 560,ht
  560 format(' change height to ' f5.2' inches')
      go to 140
  570 continue
      if(ln.ne.'p') go to 530
c    position--next line gives column and line number to position
      read(10,*) newcol,n12
      if(debug.eq.'y') print 580,newcol,n12
  580 format(' position, col 'i3' across from entry 'i3)
      xold=xcent
      xcent=xcents+colw*(newcol-1)
      if(n12.ne.0) yval=yss(n12)-icols(n12)*delth2
      if (n12.ne.0) iadj=1
c     if n12=0 continue downward
c     set up parameters for connecting to next block with a line
  590 if(n12.eq.0) go to 630
  600 if(xcent-xold) 640,620,610
  610 p2='w'
      p1='e'
       go to 650
  620 if(yval.lt.yss(nent)) go to 630
      p1='n'
      p2='s'
      goto 650
  630 p1='s'
      p2='n'
      go to 650
  640 p2='e'
      p1='w'
  650 isw=2
      go to 140
      end
c*************************************************************
```

```
c    find length of non blanks in array
       function length(title)
       character*80 title
        common/pp/tlen(80)
        character*1 tlen
       decode(80,10,title) tlen
    10 format(80a1)
       l=80
       do 20 i=1,40
       if(tlen(l).ne.' ' ) go to 30
    20 l=l-1
    30 length=l
       return
       end
c*************************************************************
c      draw figure with circular ends
       subroutine rndend(xleft,xrt,yup,ylow)
       common/xypm/xpl,xmi,ypl,ymi,ht
       disx=xrt-xleft
       xcent=xleft+disx/2.
       disy=yup-ylow
       ycent=ylow+disy/2.
       disx=disx/2.
       call plot(xleft,yup,3)
       call plot(xrt,yup,2)
       ystart=yup
       deltay=disy/20.
       ymid=disy/2.
c       find radius of circle from center to end
       yyp=ymid
       rsq=2.*ymid*ymid
       r=sqrt(rsq)
       do 10 i=1,20
       x=sqrt(rsq-yyp*yyp)
       xplt=xrt+x-ymid
       call plot(xplt,ystart,2)
       yyp=yyp-deltay
       ystart=ystart-deltay
    10 continue
       call plot(xrt,ylow,2)
       call plot(xleft,ylow,2)
       ystart=ylow
       yyp=ymid
       do 20 i=1,21
       x=sqrt(rsq-yyp*yyp)
       xplt=xleft-x+ymid
       call plot(xplt,ystart,2)
       ystart=ystart+deltay
       yyp=yyp-deltay
    20 continue
        xad=r-ymid
       xpl=xrt+xad
       xmi=xleft-xad
       ypl=yup
       ymi=ylow
       return
```

```
      end
c*************************************************************
      subroutine box(xleft,xrt,yup,ylow)
      common/xypm/xpl,xmi,ypl,ymi,ht
      xpl=xrt
      xmi=xleft
      ypl=yup
      ymi=ylow
      call plot(xleft,yup,3)
      call plot(xrt,yup,2)
      call plot(xrt,ylow,2)
      call plot(xleft,ylow,2)
      call plot(xleft,yup,2)
      return
      end
c*************************************************************
      subroutine circle(xleft,xrt,yup,ylow)
      common/xypm/xpl,xmi,ypl,ymi,ht
      common/d/debug
      character*1 debug
      disx=(xrt-xleft)/2.
      disy=(yup-ylow)/2
      xcent=xleft+disx
      ycent=ylow+disy
      r=sqrt(disx*disx+disy*disy)
      xr=xcent+r
      if (debug.eq.'y') go to 30
      call plot(xr,ycent,3)
      dtheta=3.14159265/20.
      theta=dtheta
      do 20 i=1,40
      yplt=ycent+r*sin(theta)
      xplt=xcent+r*cos(theta)
c     print 88,xplt,yplt,theta
   10 format('  xplt,yplt,theta='3f7.3)
      call plot(xplt,yplt,2)
   20 theta=theta+dtheta
   30 xpl=xr
      xmi=xcent-r
      ypl=ycent+r
      ymi=ycent-r
      return
      end
c*************************************************************
      subroutine diam(xleft,xrt,yup,ylow)
      common/xypm/xpl,xmi,ypl,ymi,ht
      common/d/debug
      character*1 debug
      disx=(xrt-xleft)/2.
      disy=(yup-ylow)/2.
      xpl=xrt+.2*disx
      xmi=xleft-.2*disx
      ypl=yup+.2*disy
      ymi=ylow-.2*disy
      xcent=xleft+disx
      ycent=ymi+disy
```

```
      if(debug.eq.'y') return
      call plot(xcent,ypl,3)
      call plot(xmi,ycent,2)
      call plot(xcent,ymi,2)
      call plot(xpl,ycent,2)
      call plot(xcent,ypl,2)
      return
      end
c*************************************************************
      subroutine sxy(xval,yval,xplsv,yplsv,xmisv,ymisv,nl1,p1)
      dimension xplsv(100),yplsv(100),xmisv(100),ymisv(100)
      character*1 p1
      if(p1.ne.'n') go to 10
      xval=(xmisv(nl1)+xplsv(nl1))/2.
      yval=yplsv(nl1)
      go to 40
   10 if(p1.ne.'e')go to 20
      xval=xplsv(nl1)
      yval=(yplsv(nl1)+ymisv(nl1))/2.
      go to 40
   20 if(p1.ne.'s') go to 30
      xval=(xmisv(nl1)+xplsv(nl1))/2.
      yval=ymisv(nl1)
      go to 40
   30 if(p1.ne.'w') go to 50
      xval=xmisv(nl1)
      yval=(yplsv(nl1)+ymisv(nl1))/2.
   40 continue
      return
   50 print 60
   60 format(' no line drawn--wrong call ')
      return
      end
c*************************************************************
      subroutine nothing(xleft,xrt,yup,ylow)
      common/xypm/xpl,xmi,ypl,ymi,ht
      xpl=xrt
      xmi=xleft
      ypl=yup
      ymi=ylow
      return
      end
c*************************************************************
      subroutine oct(xleft,xrt,yup,ylow)
      common/d/debug
      character*1 debug
      common/xypm/xpl,xmi,ypl,ymi,ht
      xpl=xrt
      ypl=yup
      ymi=ylow
      xmi=xleft
      xp=xpl-ht
      yp=ypl-ht
      xm=xmi+ht
      ym=ymi+ht
      if(debug.eq.'y') return
```

```
      call plot(xm,yup,3)
      call plot(xp,yup,2)
      call plot(xpl,yp,2)
      call plot(xpl,ym,2)
      call plot(xp,ymi,2)
      call plot(xm,ymi,2)
      call plot(xmi,ym,2)
      call plot(xmi,yp,2)
      call plot(xm,yup,2)
      return
      end
c*************************************************************
      subroutine aro(x1,y1,x3,y3,arrow)
      xd=x3-x1
      yd=y3-y1
      d=sqrt(xd**2+yd**2)
      ard=arrow/d
      arsiph=yd*ard/2.
      arcoph=xd*ard/2.
      x2=x3-ard*xd
      y2=y3-ard*yd
      x4=x2-arsiph
      y4=y2+arcoph
      x5=x2+arsiph
      y5=y2-arcoph
      call plot(x4,y4,3)
      call plot(x3,y3,2)
      call plot(x5,y5,2)
      return
      end
c*************************************************************
      subroutine decod
c    This subroutine separates letters (a thru z) and integers
c    that have been read into a character array.  It allows
c    inputs in flowchart program to be read in under an (80a1)
c    format but appear to be free field to the user.
      character iara(80),jchar(6),jc(10)
      common/jnum/jnum(6)
      common/ara/iara,jchar
      data jc/'0','1','2','3','4','5','6','7','8','9'/
      do 10 i=1,6
      jchar(i)=' '
   10 jnum(i)=0
      ist=1
      n=0
      nchar=0
   20 do 60 i=ist,80
c     print *,i,iara(i)
      if(iara(i).le.'9') go to 40
      if(iara(i).gt.'z') go to 60
      if(iara(i).lt.'a') go to 50
c      this is a character
   30 nchar=nchar+1
      jchar(nchar)=iara(i)
      go to 60
c      is this a number?
```

```
   40 if(iara(i).ge.'0') go to 70
      go to 60
c********** Both upper and lower case characters are assumed
c    possible but only lower case characters are used
c**********
   50 if(iara(i).gt.'Z') go to 60
      if(iara(i).lt.'A') go to 60
      ic=ichar(iara(i))
      ic=ic+32
      iara(i)=char(ic)
      go to 30
   60 continue
      return
   70 ist=i
      num=0
      do 90 i=ist,80
      do 80 j=1,10
      if(iara(i).ne.jc(j)) go to 80
      num=10*num+j-1
      go to 90
   80 continue
      go to 100
   90 continue
      i=81
  100 ist=i
      n=n+1
      jnum(n)=num
      if(ist.le.80) go to 20
       return
      end
```