# What You Didn't Know About NEC Spinwriter

*The Spinwriter's graphics features have remained hidden too long. Learn how you can generate matchless business graphics that are "better looking than output produced by dedicated pen plotters costing several times as much."*

By Timothy Stryker

Nippon Electric Co. (NEC) makes a fantastic thimblewheel printer called the Spinwriter. Many writers and small businessmen, myself among them, have found the Commodore CBM 8032, the NEC Spinwriter and the Wordpro4+ software package from Professional Software the ideal system for general word processing and business use.

But many Spinwriter users don't know how powerful a printer they have purchased—certainly features like 55 characters per second, unbeatable print quality and total reliability justify the price of the machine without looking any further. The Spinwriter, however, has special graphics features that you can easily exploit, once you know how. By using these graphics features in conjunction with the normal printing capabilities of the machine, you can generate graphic output that's better looking than output produced by dedicated pen plotters costing several times as much. This is because the Spinwriter can integrate fully-formed character output with plotted output in a single drawing. Let's see how this works.

**Basic Functions**

Fig. 1 shows a sample histogram generated by a program running on a Commodore computer. As you can imagine, it would be difficult, if not impossible, to produce output of this quality using an ordinary pen plotter. You'll need seven basic routines to generate a plot of this type:

1. *Initialize Printer*—This is the routine you would call before beginning to generate output. You would use this to define the origin of your coordinate system and to perform any necessary opening of logical devices, setup of printer modes, etc.

2. *Move Absolute to specified coordinates*—You would use this to move the print head to a particular spot on the paper when you (or your program) know in advance the absolute coordinates of the spot you wish to move to. The move is done with the "pen up"; i.e., no line is drawn on the paper as the print head moves to its new position.

3. *Move Relative by a specified (x,y) offset*—You would use this to move the print head by a given amount in x and y, relative to where the head was at the beginning of the move. As with Move Absolute, no line is drawn on the paper as the move is made.

4. *Draw Absolute to specified coordinates*—This routine behaves like Move Absolute except that the move

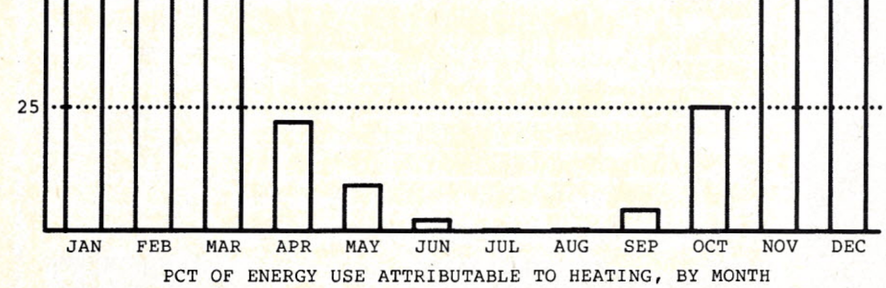



*Fig. 1. A sample histogram generated by the NEC Spinwriter and Commodore computer.*

*Address correspondence to Timothy Stryker, Samurai Software, PO Box 2902, Pompano Beach, FL 33062.*

is done with the "pen down"; i.e., a straight line is drawn from the current head position to the destination position.

5. *Draw Relative by a specified (x,y) offset*—As you might imagine, this routine draws a straight line from where it is to where it ends up (where it ends up being given in terms of a relative offset from where it started out).

6. *Print Text*—Given a character string, this routine just prints that string out, starting from wherever the print head is at the time the routine is called.

7. *Quit*—This routine performs whatever final cleanup may be necessary, such as closing logical devices, restoring the Spinwriter to its default mode, etc.

**Why RPL?**

Routines to perform these functions could be written in any computer language. However, Basic is too slow to be of use for this purpose—lines are drawn from place to place on the Spinwriter using vast numbers of individual overlapping dots, and considerable amounts of program decision-making go into the precise location of each dot. To draw a single eight-inch-long line between two points would take Basic more than a minute. A faster programming language is needed.

I have developed a language called RPL, reviewed in this magazine by Robert Baker last February (p. 10). RPL is much faster than Basic—and it is a structured language. RPL thus has many features which suit it well to this sort of work: for example, RPL lets you name your routines with reasonable mnemonics, like DRWABS for Draw Absolute, and MOVREL for Move Relative. This makes remembering how to call these routines much easier than it is in Basic, where you must remember to GOSUB 500 to do one thing, and GOSUB 750 to do another (not to mention what happens when you renumber the lines of your program).

Listing 1 shows a set of RPL routines designed to fulfill the requirements laid out above. If you're not familiar with RPL, the listing will not make much sense to you at first, but perhaps you can pick up some of the flavor of the language as you go along. As you can see, there are various "labels" followed by double-colons in the listing, whose names suggest the functions we talked about before (e.g., Initialize Printer becomes INITNEC, Move Absolute becomes MOVABS, etc.). The lines with nothing but the word REM on them are used to separate the English-language remarks from the segments of the RPL program code.

Listing 1 represents nothing more than a "subroutine library" in RPL; that is, the routines in this listing do nothing by themselves, but instead represent the seven basic, necessary functions discussed above. RPL allows you to set up a collection of routines like this independently of whatever programs you write to make use of them. Once compiled, these routines remain available for your use until you take action to eliminate them. Thus, Listing 2 shows the RPL program which, using these routines, creates the plot in Fig. 1.

Let's trace through the first few statements of Listing 2 to see how this works. The printer is initialized with −42 0 INITNEC &, which tells the printer to consider itself to be starting off at the (x,y) coordinate location (−42,0). The Spinwriter coordinate system uses units of 1/120th of an inch in the x, or horizontal, direction, and units of 1/48th of an inch in the y, or vertical, direction. Thus, this first subroutine call sets the origin at 42/120ths, or 7/20ths, of an inch from the left margin.

Next, 0 0 MOVABS & moves the print head to the origin just defined. 840 0 DRWABS & tells the printer to draw a line from where it is (the origin) to the point (840,0); this draws the seven-inch line that is the baseline of the histogram. Then 0 0 MOVABS & is repeated, which brings us back to the origin, and then 0 200 DRWABS & draws the y-axis (the solid vertical line on the left side of the graph).

In a similar manner, lines 20 and 30 of Listing 2 label the y-axis with the numbers 25 through 100 and draw the horizontal dotted lines. (Each dot is formed by moving the print head to where the dot is needed and then entering 0 0 DRWREL & to draw a "line" of zero length.) Lines 40 and 50 of Listing 2 write the month names down below the x-axis. Note the use of the PRTTXT routine to output the actual text strings to the printer. Line 60 uses this same routine to output

```
1000             NEC SPINWRITER PLOTTING PACKAGE
1010    FOR USE BY ANY V3.0 OR V4.0 PET OR CBM RUNNING RPL
1020                TIMOTHY STRYKER
1030 *
1040    INITNEC: EXPECTS INITIAL-POSITION X AND Y COORDS ON STACK;
1050             POPS THEM AND INITIALIZES PRINTER.
1060 REM
1070 INITNEC:: NECY ! NECX ! OPENPRTR & HARDCOPY SYS
1080 PIXCTL PUSHST & PRINT RETURN
1090 REM
1100    MOVABS AND MOVREL: EXPECT X AND Y DESTINATION INFO ON STACK
1110            (MOVABS EXPECTS ABSOLUTE DESTINATION COORDS; MOVREL EXPECTS
1120            OFFSETS FROM CURRENT POSITION); BOTH ARE POPPED, AND PRINT
1130            HEAD IS MOVED TO SPECIFIED LOCATION (NO LINE IS DRAWN).
1140 REM
1150 MOVABS:: NECY @ - % NECX @ - % MOVREL & RETURN
1160 MOVREL:: # IF # NECY @ + NECY ! # 0 > IF 1 FOR RVSLF ESC 2 PRINT NEXT
1170 THEN 0 % - 1 FOR LF 1 PRINT NEXT END 0 END . # IF # NECX @ + NECX !
1180 # 0 > IF SPC THEN 0 % - BCKSPC END % 1 FOR # 1 PRINT NEXT END . RETURN
1190 REM
1200    DRWABS AND DRWREL: EXPECT X AND Y COORDINATE INFO ON STACK;
1210            SIMILAR TO MOVABS AND MOVREL EXCEPT THAT A LINE IS DRAWN AS
1220            THE PRINT HEAD MOVES TO THE SPECIFIED DESTINATION POSITION.
1230 REM
1240 DRWABS:: NECY @ - % NECX @ - % DRWREL & RETURN
1250 DRWREL:: # NECY @ + NECY ! # 0 > IF 1 THEN NOT 1 + 0 END DIRY POKE #
1260 DISTY ! ; NECX @ + NECX ! ; 0 > IF 1 THEN % NOT 1 + % 0 END DIRX POKE
1270 ; DISTX ! ; ; ; > IF . # END # 2 / #
1280 DRLOOP: BCKSPC DOT 2 PRINT DISTY @ DISTX @ OR IF 4 ^ - # 0 < IF 3 ^ +
1290 DISTY @ 1 - DISTY ! DIRY PEEK IF RVSLF ESC 2 THEN LF 1 END PRINT END
1300 % 5 ^ - # 0 < IF 3 ^ + DISTX @ 1 - DISTX ! DIRX PEEK IF SPC
1310 THEN BCKSPC END 1 PRINT END % DRLOOP GOTO END . . . . . RETURN
1320 REM
1330    PRTTXT: EXPECTS A CHARACTER STRING ON THE STACK; POPS IT AND
1340             PRINTS IT OUT STARTING AT THE CURRENT PRINT HEAD POSITION.
1350 REM
1360 PRTTXT:: NRMCTL PUSHST & PRINT # 12 * NECX @ + NECX ! PRINT
1370 PIXCTL PUSHST & PRINT RETURN
1380 REM
1390    QUITNEC: CALL THIS ROUTINE WHEN ALL FINISHED WITH THE SPINWRITER.
1400 REM
1410 QUITNEC:: 13 1 PRINT NRMCTL PUSHST & PRINT CLOSEALL SYS RETURN
1420 PIXCTL:(6,27,93,65,27,93,80) NRMCTL:(6,27,93,76,27,93,87)
1430 OPENPRTR: 0 217 POKE 1279 211 ! 4 210 POKE 65473 @ 3 + SYS RETURN
1440 HARDCOPY::(162,4,76)[65481] :SOFTCOPY::65484: :CLOSEALL:65511:
1450 :LF:10: :RVSLF:57: :SPC:32: :BCKSPC:8: :ESC:27: :DOT:46:
1460 NECX::## NECY::## DIRX:# DIRY:# DISTX:## DISTY:##
```

*Listing 1. RPL routines for creating Spinwriter graphics.*

the histogram title.

Finally, lines 70 and 80 generate the histogram bars themselves. The height of each bar is given by the corresponding entry in ENERGYTBL. The whole process, once you know RPL, is remarkably simple and straightforward. Imagine the difficulty of generating a graph like this using assembly language!

### Another Example

Another common business use of computer graphics is generation of pie charts such as that shown in Fig. 2. Since RPL does not directly support floating-point mathematics, you might suppose that it is difficult to generate circles, angles and so forth using RPL. Not so. Listing 3 shows the program which gave rise to the plot in Fig. 2. In order to derive the sines and cosines needed for circles and angles, you just make use of the fact that Basic and RPL programs can coexist in memory: Let Basic handle the generation of sine/cosine information, and use RPL for printer control.

The REM in line 20 of Listing 3 acts as a separator between the Basic part of the program and the RPL part. The RPL compiler thus considers the Basic statements in line 10 to be, as far as it is concerned, a set of English-language remarks. By the same token, Basic never has to deal with the RPL part from line 30 on, because it never gets that far. When you enter the Run command from Basic, line 10 simply pokes the values of the sines of the angles from 0 to 90 degrees into some scratchpad memory, starting at address 634, and then does a Go. (Go is a command that RPL adds to your normal Basic vocabulary, saying, in effect, go execute the current RPL program, whatever it may be.)

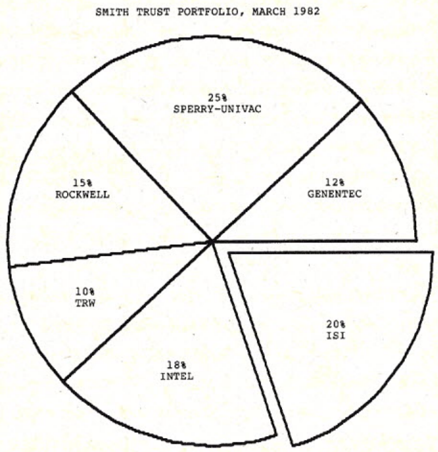



*Fig. 2. A sample pie chart.*

Once the RPL part of the program begins executing, finding the sine of an angle between 0 and 90 degrees requires nothing more than adding the angle size to the number 634 and doing a peek to that location. To find sines and cosines of angles outside this range, use the fact that $\sin(x) = \sin(360 + x)$, $\sin(x + 180) = -\sin(x)$, $\sin(x) = \sin(180 - x)$ if $90 < x < 180$, and that $\cos(x) = \sin(90 + x)$. The COS and SIN routines in lines 500 and 510 of Listing 3 handle all of these transformations, returning sines and cosines, scaled by a factor of 240, when given angles in units of degrees.

The routine labelled PIECHART, starting on line 200 of Listing 3, is the main pie-chart generator routine. It expects to be passed the address of a table of the sort found in lines 100 through 150, and it automatically generates the complete chart, including centering of legends and offsetting the last "piece" of the pie. In line 40 of Listing 3, the phrase PORTFOLIO PIECHART & is all that's needed to call for the generation of the chart—the rest of the program

```
10 -42 0 INITNEC & 0 0 MOVABS & 840 0 DRWABS & 0 0 MOVABS & 0 200 DRWABS &
20 4 1 FOR -36 50 FN * 2 - MOVABS & 25 FN * STR$ 2 = IF 32 END 3 PRTTXT &
30 0 50 FN * MOVABS & 105 1 FOR 8 0 MOVREL & 0 0 DRWREL & NEXT NEXT
40 MONTHTBL 12 1 FOR 70 FN * 43 - -8 MOVABS & # PUSHST & PRTTXT &
50 # PEEK + 1 + NEXT . 124 -20 MOVABS &
60 "PCT OF ENERGY USE ATTRIBUTABLE TO HEATING, BY MONTH" PRTTXT &
70 20 0 MOVABS & ENERGYTBL 12 1 FOR 0 ; @ 2 * ; ; DRWREL & 36 0 DRWREL &
80 0 % - DRWREL & 34 0 MOVREL & 2 + NEXT . QUITNEC & STOP
100 MONTHTBL:["JAN","FEB","MAR","APR","MAY","JUN"]
110          ["JUL","AUG","SEP","OCT","NOV","DEC"]
120 ENERGYTBL:[77,82,54,22,9,2,0,0,4,25,54,65]
```

*Listing 2. Program that uses RPL routines to plot Fig. 1.*

```
10 FORI=0TO90:POKE634+I,INT(SIN(I*3.141592654/180)*240+.5):NEXT:GO
20 REM
30 -450 0 INITNEC & -198 160 MOVABS & "SMITH TRUST PORTFOLIO, MARCH 1982"
40 PRTTXT & PORTFOLIO PIECHART & QUITNEC & STOP
100 PORTFOLIO: (12,"GENENTEC")
110            (25,"SPERRY-UNIVAC")
120            (15,"ROCKWELL")
130            (10,"TRW")
140            (18,"INTEL")
150            (20,"ISI")
200 PIECHART: 0 0 MOVABS & 0 ANGXY & DRWABS & 0 SLICE & # 0 DRWARC &
210 ; PEEK 2 / ; + ANGXY & 10 / % 10 / % MOVABS & 0 NECX ! 0 NECY !
220 # ANGXY & DRWABS & ; PEEK ; + ; DRWARC & LSLICE & . . RETURN
230 SLICE: ; PEEK ; + 100 < IF LSLICE: ; PEEK 2 / ; + ANGXY & 2 * 3 / 4 +
240 % 2 * 3 / 18 - % MOVABS & ; PEEK STR$ 1 = IF 32 END 2 PRTTXT &
250 "%" PRTTXT & ; 1 + PEEK 6 * -18 % - -8 MOVREL & ; 1 + PUSHST& PRTTXT &
260 0 0 MOVABS & ; PEEK + # ANGXY & DRWABS & % 1 + # PEEK + 1 + %
270 SLICE GOTO END RETURN
300 DRWARC: # ANGXY & MOVABS & FOR FN ANGXY & DRWABS & NEXT RETURN
400 ANGXY: 36 * 5 + 10 / # COS & 3 * 2 / % SIN & 3 * 5 / RETURN
500 COS: 90 + SIN: 360 \ 0 ; 180 > IF . 180 - 1 END % # 90 >
510 IF 180 % - END 634 + PEEK % IF 0 % - END RETURN
```

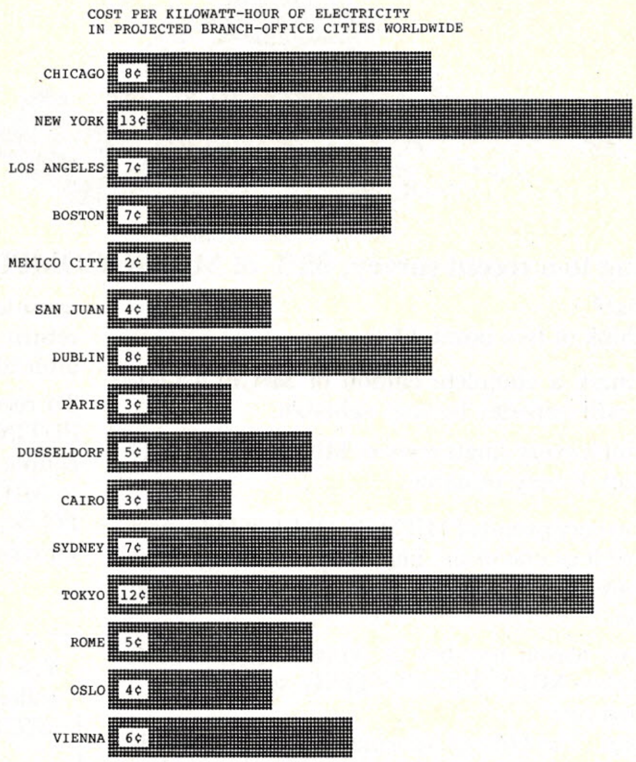
*Listing 3. Program in RPL that generates Fig. 2.*

Graphics applications are not limited to business.

mainline in lines 30 and 40 merely initializes the printer, prints out the pie chart title and calls QUITNEC when done.

This example brings out a couple of points of interest in connection with the use of the Spinwriter for graphics. Note the decreasing resolution in the perimeter of the pie toward the top and bottom of the circle, and the contrast between the line separating the Rockwell slice for the TRW slice and the others. Because the vertical unit of distance in the Spinwriter is so much greater than the horizontal unit, lines with small nonzero slopes will have a distinctly grainy quality to them.

You should also take this difference between the horizontal and vertical units of distance into account if you want an aspect ratio of 1:1 between the vertical and horizontal axes, as is the case here. (If you didn't allow for this, the pie chart would come out elliptical.) The ratio of




*Another sample graphic generated by the Spinwriter.*

```
10 -450 0 INITNEC & 7 0 FOR 25 FN * 412 - # 0 MOVABS &
20 10 FN * 5 + 0 ; HLFWAY & DRWREL & MIRRORX & ; 0 % - 0 DRWABS &
30 0 % 0 % - HLFWAY & DRWREL & MIRRORX & 0 DRWABS & NEXT -212 0 MOVABS &
40 0 -85 DRWABS & 212 0 DRWABS & 0 85 DRWABS & -212 0 DRWABS &
50 16 9 FOR 25 FN * 412 - # 0 MOVABS & 10 FN * 5 + 0 ; ; ; HLFWAY &
60 MOVREL & DRWABS & ; 0 % - 0 HLFWAY & DRWREL & MIRRORY & 0 % 0 % -
70 DRWABS & 0 HLFWAY & DRWREL & NEXT -126 -23 MOVABS &
80 CAPITALR PRTLET & CAPITALP PRTLET & CAPITALL PRTLET & QUITNEC & STOP
100 HLFWAY: NECY @ - 2 / % NECX @ - 2 / % RETURN
110 MIRRORX: 0 NECX @ - NECY @ MOVABS & RETURN
130 MIRRORY: NECX @ 0 NECY @ - MOVABS & RETURN
400 PRTLET: # 1 + % PEEK 1 FOR GETCOORDS & MOVREL & 5 + # 1 - PEEK 1
410 FOR GETCOORDS & DRWREL & 4 + NEXT NEXT GETCOORDS & MOVREL & . RETURN
450 GETCOORDS: # @ SCALEX * ; 2 + @ SCALEY * RETURN
460 :SCALEX:3: :SCALEY:1:
500 CAPITALR:(2)
510 [0,0] (11)[0,45,17,0,8,-8,0,-9,-8,-8,8,-20,-6,0,-8,20,-6,0,0,-20,-5,0]
520 [5,25] (6)[0,15,10,0,5,-5,0,-5,-5,-5,-10,0] [25,-25]
550 CAPITALP:(2)
560 [0,0] (8)[0,45,17,0,8,-8,0,-9,-8,-8,-12,0,0,-20,-5,0]
570 [5,25] (6)[0,15,10,0,5,-5,0,-5,-5,-5,-10,0] [25,-25]
600 CAPITALL:(1)
610 [0,0] (6)[0,45,5,0,0,-40,20,0,0,-5,-25,0] [30,0]
```

*Listing 4. The recreational program.*

1/48th to 1/120th is the same as a ratio of 5 to 2: thus, in order to generate true squares, circles, etc., the horizontal coordinates used must be a factor of 5/2 what they would otherwise be if the horizontal and vertical units of the Spinwriter were equal.

**Not Just Business**

Naturally the graphics applications of the NEC Spinwriter are not limited to business purposes. Computer-generated art and big-letter sign making are two recreational application areas that come quickly to mind. Fig. 3 shows the output of the program in Listing 4, just to give you a taste of the possibilities. With some additional effort, you could probably develop arbitrarily complex 3-D perspective projections, contour maps, automatic software flowcharting utilities and what-have-you, all using these same basic graphical-output routines. You could easily generate color plots by using ribbons of various colors, having your program pause at appropriate points to allow you to change ribbons from one color to the next.

The NEC Spinwriter is a truly sophisticated output device. I hope I've given you some idea of its capabilities, and that I've inspired you to learn more. If you have a Spinwriter but cannot use RPL because you don't have a Commodore computer, you can find technical details on the byte sequences required in Section 4 and Appendix D of the manual supplied with your printer. If you have both a Spinwriter and a Commodore computer, you are surely fortunate... feel free to write me for further information regarding the availability of RPL and more advanced Spinwriter-control software.■

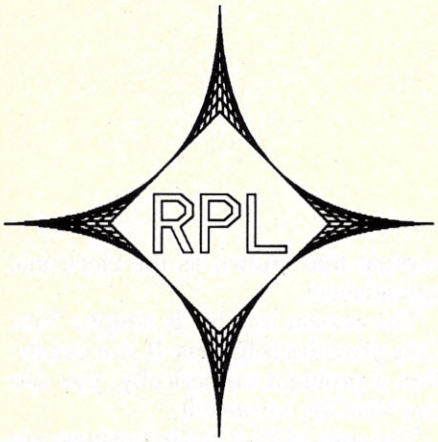



*Fig. 3. Recreational NEC graphics.*

```
10 REM BASIC PROGRAM TO GENERATE A 1"X1" SQUARE ON THE SPINWRITER
20 REM INITIALIZE PRINTER
30 OPEN 4,4 : FOR I = 1 TO 6 : READ X : PRINT#4,CHR$(X); : NEXT
40 DATA 27,93,65,27,93,80
50 REM MOVE AWAY FROM LEFT MARGIN
60 FOR I = 1 TO 240 : PRINT#4,CHR$(32); : NEXT
70 REM DRAW LEFT SIDE OF SQUARE GOING UP
80 FOR I = 1 TO 48 : PRINT#4,CHR$(46);CHR$(27);CHR$(57);CHR$(8); : NEXT
90 REM DRAW TOP SIDE OF SQUARE GOING RIGHT
100 FOR I = 1 TO 120 : PRINT#4,CHR$(46); : NEXT
110 REM DRAW RIGHT SIDE OF SQUARE GOING DOWN
120 FOR I = 1 TO 48 : PRINT#4,CHR$(46);CHR$(10);CHR$(8); : NEXT
130 REM DRAW BOTTOM SIDE OF SQUARE GOING LEFT
140 FOR I = 1 TO 120 : PRINT#4,CHR$(46);CHR$(8);CHR$(8); : NEXT
150 REM RETURN PRINTER TO NORMAL AND CLOSE DEVICE
160 FOR I = 1 TO 7 : READ X : PRINT#4,CHR$(X); : NEXT : CLOSE 4
170 DATA 13,27,93,76,27,93,87
180 END
```

*This Basic program for the Commodore computer draws a simple square box. It illustrates the complexity involved in implementing Spinwriter graphics in Basic.*