THE GURU'S LAIR

Don Lancaster

PostScript™ Flutterwumpers

Pick a peck of PostScript PIC's.

With a proper "flutfile" partitioning, all the raw power of real PostScript can easily get applied to stunningly low cost PIC robotic systems.

I have long been a rabid fan of that incredibly superb and totally general purpose PostScript computing language. For years, I have chosen PostScript as my first and foremost weapon for nearly all of my columns and stories, all of my primary engineering designs, and all of my technical illustrations. Plus bunches of other stuff ranging from investment models to my new magic sinewave research to Smith Charts to shaft encoders to web site traffic analyzers to hot tub controllers.

And, as we shall shortly see, to explore some utterly incredible new low end *robotics* opportunities.

Until recently, you really had to want to use PostScript-as-language in the worst sort of way to be able to do so. In particular, needing a PostScript printer to serve as the host computer was horribly limiting. The PostScript hard drives were somewhat few and far between. As was the lack of a real time visual display. And even more brutally crippling was that one-way parallel port that nearly all early PC systems had irrationally and stupidly forced upon PostScript users.

But these limitations are ancient history. PostScript is now literally busting out all over with brand new capabilities and unique powers.

Ferinstance, the free GhostScript 5.01 clone gives you full host based PostScript computing combined with full screen visuals. GhostScript has gotten a lot better lately.

Variations on GhostScript can be customized for new commercial apps. As an example, the *Videonics* people now offer the *PowerScript 1000* fully animated video character generator. Thus turning zillions of fonts and all of the usual desktop publishing tools loose on video editing.

Original PostScript lacked most transparency features. The (literally) brilliant Videonics workaround was to redefine their CMYK color space, with "K" now serving as a new alpha transparency channel.

Host-to-printer interactions have dramatically improved. With higher speed two-way parallel comm newly added to the readily available options of Serial, AppleTalk, and Ethernet. Program apps such as that *Download Mechanic* from *Acquired Knowledge* simplify PostScript-as-language and printer interactions. With freebie eval demos available.

Meanwhile, Adobe has improved their *Acrobat 3.01* PostScript variant. Acrobat very greatly revolutionizes the distribution of information. Both off and on line. I do feel that Acrobat completely blows HTML away on all counts. Ludicrously so. By offering an exact control of what the end user will see, with *full screen* magnifiable displays *while still online*, and vastly improved antialiased visibility.

Combined with *url* hot linking, interactive forms, wipes, full text or entire library searching, bookmarks, thumbnails, byte range loading, scan capture, plus powerful handicapped access provisions. At typical single file sizes of 11K per fancy illustrated page. While printing equally well to either PostScript or certain newer non-PostScript printers.

One element of Acrobat is called the Adobe *Acrobat Distiller* 3.01.

Distiller's intended purpose is to let you convert complex PostScript code into simpler Acrobat .PDF files. But don't let the name fool you. *The Acrobat Distiller is really a speedy* *genuine Adobe host based general purpose PostScript computer!*

One that makes PostScript fast and fun to play with.

ACROBAT DISTILLER AS A POSTSCRIPT COMPUTER

A PC, UNIX, or Macintosh based *Acrobat Distiller 3.01* makes a fairly powerful general purpose PostScript computer. At its input, the Distiller accepts PostScript code from a disk text file. Distiller can also easily be taught ahead of time *to read nearly any file format in any language!* At the output, Distiller can generate its (usually intended) .PDF file. Or could write messages or calculated results to its (usually ignored) .LOG file. Or can write any custom disk based text file for you in almost any format.

As examples, Distiller easily can analyze web site log files or extract special content from private search engine internals. Or can convert file and font formats. Or can filter all the Fourier transforms from millions of magic sinewaves, keeping only those scant few good ones. Or can design digital filters or solve higher order linear equations.

For final results, you could view your screen displays using Acrobat Exchange, Reader or GhostScript. And use either Exchange or Reader to print. Doing so to both PostScript and non-PostScript printers.

Let's look at one simple example that shows how fast and easy it is to use PostScript as language. Let's find the sine of 35.4 degrees. First, create this textfile by using your favorite editor or WP...

```
%!
% Find sine of 35.4 degrees
35.4 sin ==
```

Drag and drop this file into your Acrobat Distiller. The magic value of 0.579281 should pop right up on your log file display.

Distiller obeys the rules that any *printing* (*print* or *"=="*) commands go to the .LOG file; *writing* commands (such as *write* or *writestring*) output to your custom .TXT disk output file; and any *marking* commands (such as *show*, *stroke*, or *showpage*) go to the generated .PDF file.

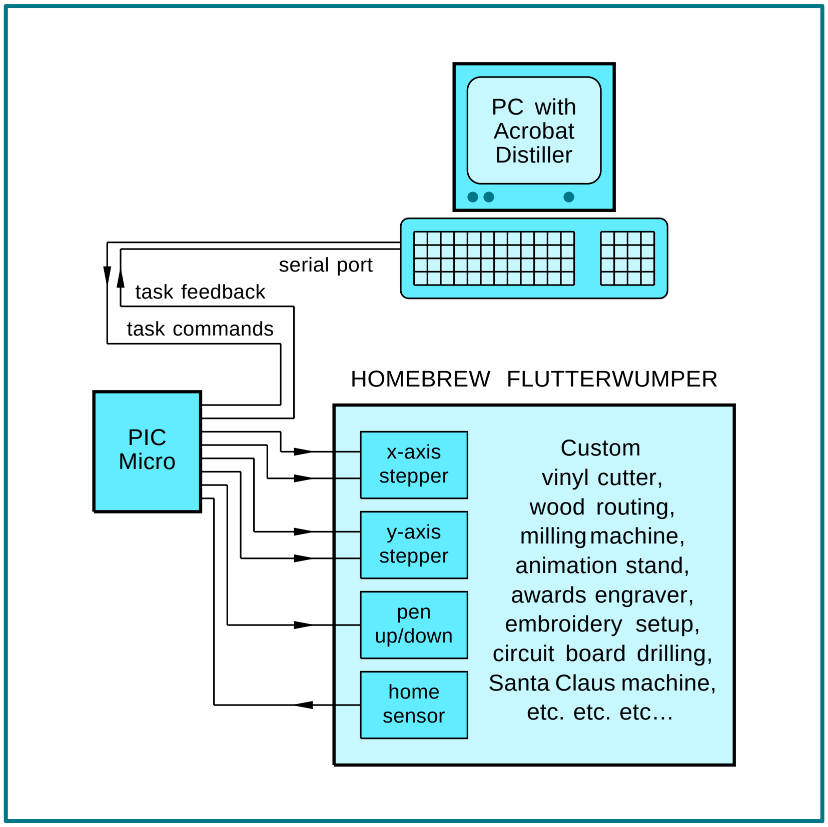


Figure 1 – *How flutfile partitioning brings genuine PostScript to PIC fluterwumpers.*

Do *not* end your file with *quit* or control-[D] when using Distiller.

While the log file is intended for error and status messages, you could write nearly anything you want to it at any time for any reason. Sadly, the log file is *not quite* transparent and has a few hidden restrictions. So you will have to decide whether to pick the quick and dirty "free" log file or write a custom output file.

Some log file limitations: Adobe Distiller does a *flushfile* after every newline. Which nicely lets you see problems as they occur. But sure can slow you down. While causing a lot of disk clatter on extended outputs. Long strings are also truncated to 200 chars in the log file. Carriage returns are tricky. And you will get a mix of status, error, and data messages.

One really big time gotcha when writing *any* PostScript as language code that needs any PC disk access: *Always use "\\" whenever you mean "\" inside of a PostScript string!* The confusion comes about because that "\" acts as a directory boundary in Windows file names, but is a fully reserved PostScript string character.

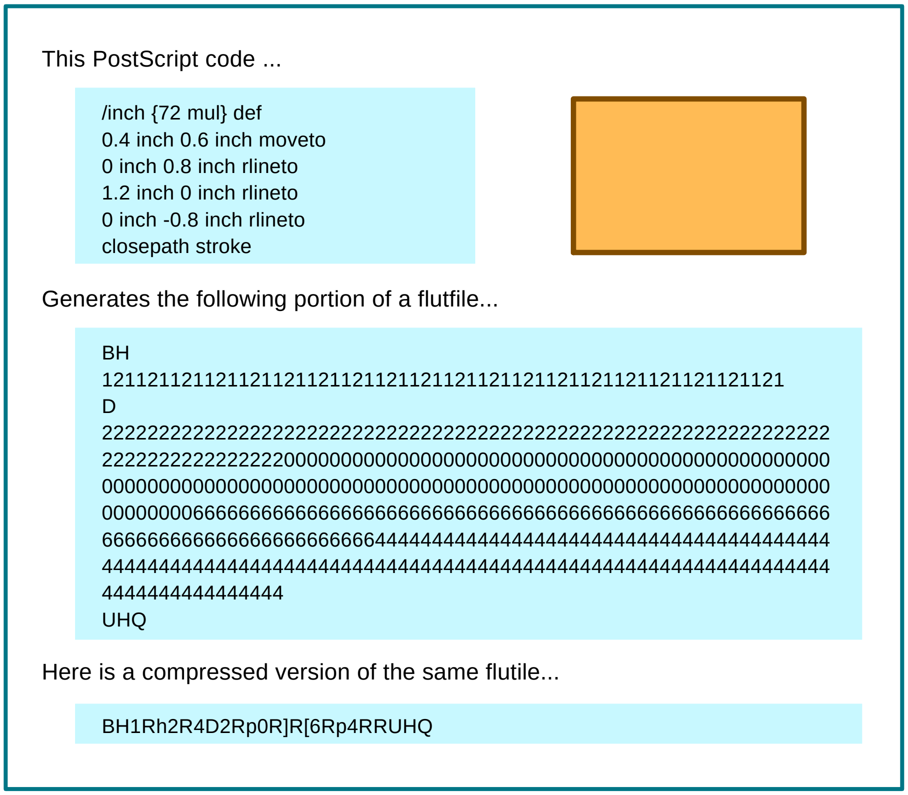
GhostScript gets around this by substituting "/" for "\" in filenames.

Do not be alarmed when you see a *"Warning: No PDF file produced"* error. If your primary goal is writing your own custom disk file, the oft wanted .PDF viewable file may not be needed at all. Forget the baby; you are after the wash water here.

Note that any Gerber or HPGL to flutfile format disk conversions are a trivial task for PostScript.

"0"	=	Go one step due east
"1"	=	Go one step northeast
"2"	=	Go one step due north
"3"	=	Go one step northwest
"4"	=	Go one step due west
"5"	=	Go one step southwest
"6"	=	Go one step due south
"7"	=	Go one step southeast
"B"	=	Begin sequence
"D"	=	Pen down or cutter on
"H"	=	Reset to home position
"Q"	=	Quit sequence
"Rn"	=	Repeat last n-32 times
"U"	=	Pen up or cutter off
"X"	=	Breaking debugger
"%"	=	Ignore comments till cr

Figure 2 – *One possible flutfile command set. Single characters do elemental tasks.*



This PostScript code ...

```
/inch {72 mul} def
0.4 inch 0.6 inch moveto
0 inch 0.8 inch rlineto
1.2 inch 0 inch rlineto
0 inch -0.8 inch rlineto
closepath stroke
```

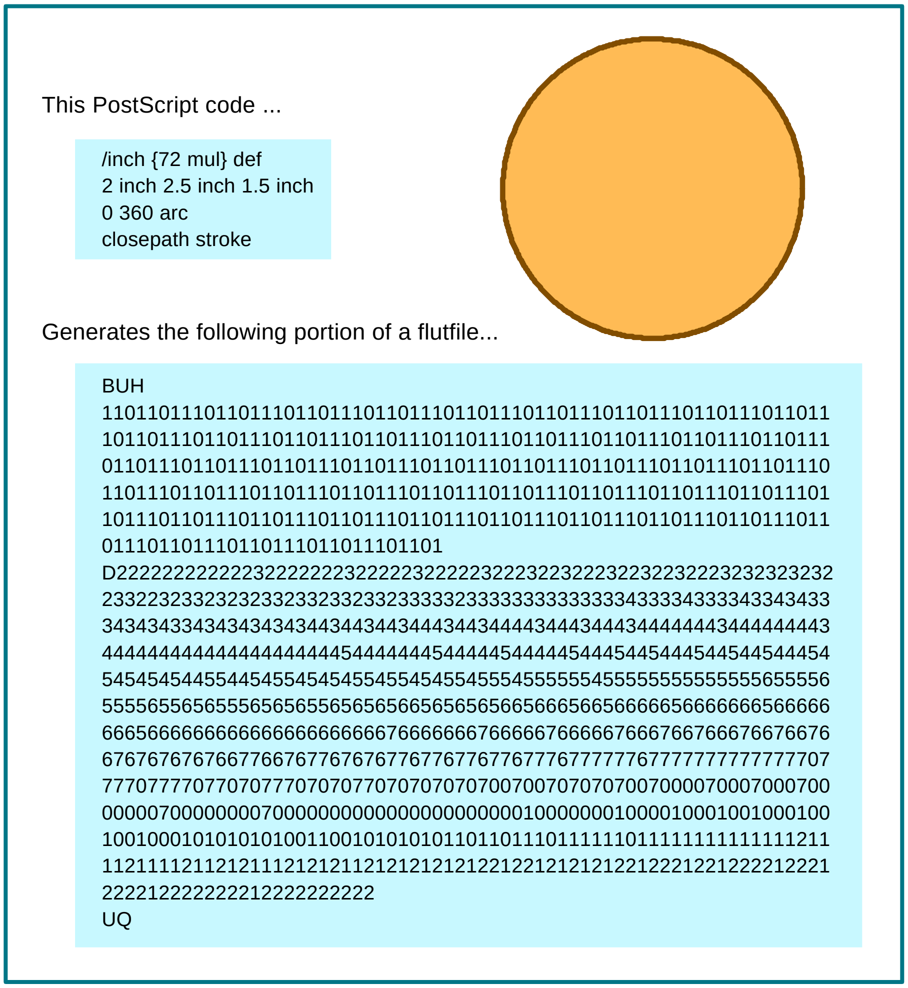
Generates the following portion of a flutfile...

```
BH
121121121121121121121121121121121121121121121121121121121121
D
2222222222222222222222222222222222222222222222222222222222222222
2222222222222222000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000066666666666666666666666666666666666666666666666666666666
6666666666666666666666664444444444444444444444444444444444444444
4444444444444444444444444444444444444444444444444444444444444444
4444444444444444
UHQ
```

Here is a compressed version of the same flutile...

```
BH1Rh2R4D2Rp0R]R[6Rp4RRUHQ
```

Figure 3 – *A flutfile example: simple rectangle.*



This PostScript code ...

```
/inch {72 mul} def
2 inch 2.5 inch 1.5 inch
0 360 arc
closepath stroke
```

Generates the following portion of a flutfile...

```
BUH
1101101110110111011011101101110110111011011101101110110111011011
1011011101101110110111011011101101110110111011011101101110110111
0110111011011101101110110111011011101101110110111011011101101110
1101110110111011011101101110110111011011101101110110111011011101
1011101101110110111011011101101110110111011011101101110110111011
011101101110110111011011101101
D222222222222322222223222223222223222322322232232232223232323232
2332232332323233233233233233332333333333333334333343333343343433
3434343343434343434434434434443443444434443444434444444434444444443
4444444444444444444444454444444454444454444445444544544454454454444454
5454545445544545545454545545545454554555455555545555555555555655556
5555655565655565656556565656656565656566566656665666665666666666566666
6665666666666666666666666667666666667666667666667666766676667667667667676676
6767676767667766767677676767767767767767776777777767777777777777707
7770777707707077707070770707070707007007070707007000070007000700
0000070000000007000000000000000000000001000000010000100010010001000100
1001000101010101001100101010101101101110111111011111111111111211
1121111211212111212121121212121212212212121212212221221222212221
2222122222222212222222222
UQ
```

Figure 4 – *A flutfile example: large 100 DPI circle.*

DISTILLER VERSUS GHOSTSCRIPT

Distiller is amazingly fast and is 100% authentic genuine Adobe. But GhostScript is slow and klutzy and is not at all friendly. GS clearly flaunts its UNIX heritage. But a new *GS View* add-on gives you a GUI interface.

Both Distiller and GhostScript do offer visual displays. Even with those two mouse clicks required to switch to Exchange, Distiller displays faster and considerably clearer.

GhostScript lacks certain of the latest PostScript features and has got unofficial bugs in other ones. But GS can be customized to make anything you like. GhostScript font additions are a hassle. GhostScript source code is easily gotten. Distiller costs you money; GhostScript does not.

There is one debugging advantage to GhostScript: Distiller gags on any error, outputting only a stack dump. GhostScript gives you a visual screen output up to the point your error was made. Thus showing you where to go to pin down any obtuse bug.

I personally prefer Distiller. But then again, I am an Adobe developer and beta tester.

The use of Acrobat Distiller for PostScript-as-language that I am the most excited about involves...

POSTSCRIPT FLUTTERWUMPERS

I like to call a *flutterwumper* any lower cost robotic which either spits or chomps. Typical would be printed circuit drillers, wood routers, fabric shears, Santa Claus machines, panel engravers, animation stands, cutters for vinyl, embroidery machines, shop mills, automated bartending setups, laser trimmers or box makers.

As figure one shows, PostScript gives us a unique way of partitioning any flutterwumper. Greatly reducing your costs of dedicated hardware.

Your key secret here of is to *let stock PostScript do all of the "hard" higher level stuff*. Such as managing disks and other resources, interfacing design software or clip art libraries, dealing with fonts, doing all of fancy coordinate transformations, making tool path corrections, or converting curves to individual machine space steps. After all the hard stuff is done, you have your on-board PIC handle only the "easy" or simplest possible tasks. Done in the dumbest possible manner.

To do this, we will use transfer documents that I will call *flutfiles*. Two obvious languages for flutfile documents would be Gerber Format or HPGL. But let's think cheap and scungy here. Let's instead create an ultra simple language. A language that asks as little as possible in the way of flutterwumper smarts.

We will limit flutfile commands to printable ASCII text characters. That use ordinary higher Baud rate serial comm. This turns out to be more than fast enough for the max chomping speed of pretty near any mechanical flutterwumper..

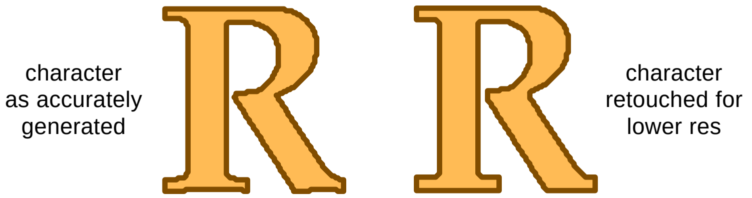
We'll initially restrict our flutfile language to one that is immediately interpreted. We will use no loops, no subroutines, nothing fancy. But we will allow for comments, a breaking debugger, and a simple repeat.

Let's initially assume a "2-1/2D" flutterwumper system. One which requires precise steps in the X and Y directions, but only a simple up and down action in the Z direction. The flutterwumper mechanism itself can use any coordinate system it wants to. PostScript will take care of auto creating X-Y equivalent motions.

A suitable flutterwumper meta language command set appears for you in figure two.

We can use the numerals 0-7 for cardinal directions. Let us stick with the PostScript convention of due east being zero and positive degrees going counterclockwise. Thus, your ASCII numeral "0" means to go due east by generating one positive step in the X axis stepper direction.

The numeral "1" means to shift northeast by generating positive X and Y axis steps. Your "2" means to head north with a single positive Y step. And similarly around all of the compass points. To the southeast "7" which needs a positive X step and a negative Y step.



This PostScript code ...

```
/Palatino-Bold findfont 100 scalefont setfont
 100 200 moveto
 (R) show
```

Generates the following portion of a flutfile...

```
BUH
121121211212112121121211212112121121211212112121121211212112121121211
211212112121121211212112121121211212112121121211212112121121211212112
121121121211212112121121211212112121121211212112121121211212121121121
211212112112121121211212112121121211212112121121211212112121211211212
112121121121211212112121121211212112121121211212112121121211212121121
1212112
D
010101010111111111212122212222222223223233323343434344434444444444
444444444444444444444444444444444444444444444444444444444666660000000
767666666666666666666666666666666666666666666666666666666666666666666
666666666666666565445444444666666000000000000000000000000000070022
222243433222222222222222222222222222222222222222222222222222222222222
222222222222222222222222100000000000000700707077766766666666666565
656555554544444546576776776767767767767767767767767767767767767767
767600000000010000222222443433333323323323323323323323323323323323
323323
UHQ
```

Figure 5 – *A flutfile example: Palatino "R".*

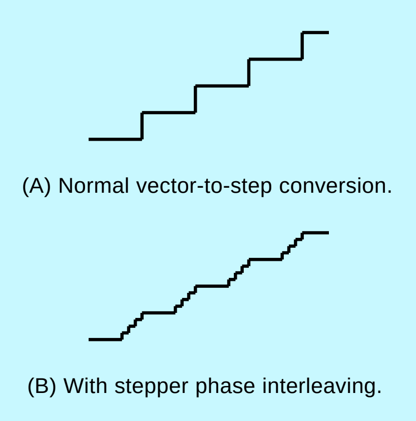
Other single letters can designate needed simple actions. Such as a "U" for up, "D" for down, a "H" for home, "X" for a breaking debugger, and "Q" for quit. We can also include flutfile comments. By using that PostScript convention of anything between a "%" and the next carriage return or line feed being a comment.

A simple repeat could get done by letting "R" and a following value *n* mean "repeat your last command *n*-32 times". The 32 occurs because we restrict ourselves to the printing ASCII characters. That ASCII dollar sign has a code of decimal 36. Thus a "R$" would mean "repeat your last command 36-32=4 times.

Note that it is inefficient to use "R" for *fewer than four* repeats. For more than 95 steps, you just repeat repeat. The repeat command could significantly shorten a flutfile which has mostly cardinal direction moves and actions involved.

A formal flutfile spec would also include handshaking. On one hand, you do not want to baud rate limit. On the other, you must not overrun your mechanism. Especially on time intensive homing or repeats.

Handshaking details are system specific, but buffered XON-XOFF is often an optimum choice.



(A) Normal vector-to-step conversion.

(B) With stepper phase interleaving.

Figure 6 – *How stepper phase interleaving can improve apparent resolution.*

SOME FLUTFILE EXAMPLES

Your new PostScript routines can generate flutfiles. Your flutfiles then get used by your flutterwumper.

All in real time, or in a deferred production mode where the same file gets resent once per piece. Again, the big advantage of such a partitioning is that you'll get by with extremely simple and cheap dedicated hardware on the flutterwumper itself.

PostScript is also especially adept in doing the oddball transformations you will need for nontraditional axis flutterwumpers. Such as rectangular to polar or az-el.

Any old terminal program can do your actual comm for you. Plain old *Hyperterm* works just fine.

Let's take a look at three simple PostScript flutfile examples to scope out what they might look like. Since flutfiles are all resolution dependent, let's assume that our flutterwumper resolution is 100 steps per inch.

Or ten mils per step.

In figure three, we make a simple rectangle. First we home with pen or cutter up. Then we shift to the lower left corner of the rectangle and drop the pen. We then trace out the entire rectangular path, raise the pen, and return home.

To get from PostScript into your flutfile, you'll need a PS routine that intercepts PS commands, finds their paths, converts the paths to straight line vectors, resolves these vectors down into machine dependent steps, and then creates a flut file.

All trivial PostScript tasks.

The usual way a PS routine does its flutfile building is by intercepting PostScript's *moveto*, *lineto*, *curveto* and *closepath* operators. Since nearly anything PostScript can be redefined at any time for any purpose, all these operators are redirected to generate the output flutfile commands.

Adobe Acrobat automatically can reduce complex PostScript code into these fundamental *moveto*, *curveto*, *lineto* and *closepath* elements. Your usual way to deal with fancy artwork is to send it on through the *Acrobat Distiller* first. Then print to disk to extract those "just the facts ma'am" elemental PostScript commands.

Those commands could then get pasted into a PS flutfile generator. Or else get read and interpreted directly from a disk textfile.

Flutfiles can be sent in real time or else saved to disk and reused for production. You can also host create elemental flutfile libraries.

Several ready-to-use examples of suitable tools are in FLUTOOLS.PS on the *Flutterwumper Library Shelf* of *http://www.tinaja.com*.

Note that *any* existing PostScript code application can generate all of your desired paths. Although I very much prefer to write custom code in raw PostScript, use of *Illustrator* or any PS-compatible drawing program will work just fine. In fact, *nearly any* existing PS or Acrobat program or code can be used to automatically generate robotic flutfiles.

Thus, *your flutterwumper will automatically speak PostScript from day one.* Royalty free.

Using nothing but a PIC or two.

In this example, our PostScript code is simply...

```
/inch 72 def
0.4 inch 0.6 inch moveto
0 inch 0.8 inch rlineto
1.2 inch 0 inch rlineto
0 inch -0.8 inch rlineto
closepath
stroke
```

A rectangle version using repeats and removal of unneeded characters is also shown. Which dramatically shortens this particular flutfile. Any positioning commands now do take slightly longer to execute since they are limited to 0-7 cardinal moves.

If you are always using inches, you can simply do a *72 dup scale* at the start of your file instead.

Insider secrets of vector-to-step conversion are in HACK83.PDF.

In figure four, we create a three inch circle. The PostScript code is even simpler...

```
/inch 72 def
2 inch 2.5 inch 1.5 inch
0 360 arc
closepath
stroke
```

Finally, in figure five we generate a fancy Palatino "R". Again using any old PostScript artwork creator or else these raw PostScript commands...

```
/Palatino-Bold findfont
100 scalefont setfont
100 200 moveto
(R) show
```

One minor gotcha: The font you pick *must* already be ATM resident in your host. If not, *Courier* will get substituted for misspellings or for a non-Adobe font, while a rectangular bounding box will be substituted for any real but unavailable Adobe font. More details in DISTPALO.HTML on that *PostScript Library Shelf* of my *www.tinaja.com*

A second minor point: *Notepad* has problems with newline control characters, so use *Wordpad* or some other WP or editor instead.

Do note that your flutterwumper electronics could not care less about all this fancy stuff. All your custom hardware has to do is convert ASCII characters into simple steps.

Something that any old PIC can do in its sleep.

IMPROVING RESOLUTION

Flutterwumper resolution tends to be much poorer than what you'd see in any laser printer. There is not much point in routing out any large wooden display sign closer than, say, a sixteenth of an inch. And it is real tricky to force better than ten mils resolution from any larger homebrew positioning hardware.

A flutfile can be set to most any resolution, so there is no real limit to how fine you slice things. But on any coarser systems, there are a few tricks you can pull that will greatly improve your results.

The first is to *retouch* any grungy flutfile elements so they look best at your chosen resolution. For instance, our initial Palatino "R" looks a little jaggy and unbalanced. The retouched version also shown seems somewhat "less Palatinoish" but more suited to any lower resolution flutterwumper. With some practice, you can easily "sight read" any flutfile. And make these adjustments on the fly.

"Retouching" is slightly different than the usual font hinting processes for coarse bitmaps. Because diagonal paths are permitted. But *Black Book Hinting* techniques might lead to a semi-automated solution.

Retouching is especially useful for character or libraries of forms or other locations where standardized code is going to see lots of reuse.

A second resolution enhancing trick is *stepper phase interleaving*. Most steppers employ four *phases* or increments to complete a step. Each phase moves you one quarter of the way towards your goal.

So, as figure six shows us, instead of jumping a whole X step and then a whole Y step, on any 1-3-5-7 slant moves you first do a quarter X step followed by a quarter y step. All your paths end up much smoother and the jaggies are much smaller.

For low cost larger systems, it is possible to convert any automotive alternator into a cheap power stepper motor. Alternator conversions tend to use three phase steps rather than four. But the same concept of stepper phase interleaving remains.

FOR MORE INFORMATION

A PostScript-as-language tutorial appears as DISTLANG.HTML on my *Acrobat Library Shelf*. Find this one at *www.tinaja.com/acrob01.html*. Some flutfile generators and utilities similarly should appear over on the *Flutterwumper Library Shelf* at my *www.tinaja.com/flut01.html*. Lots of fully working PostScript-as-language examples are included. Also offered are PIC sourcecode examples, custom consulting services, and various new design *Infopacks*.

Hot links to both *Acrobat* and *GhostScript* are also found here. As are extensive PIC tutorials and links.

More details on converting car alternators to power steppers in my HACK47.PDF A superb alternator conversion video is available from *John Rees*. His address and other mentioned resources appear in the adjacent sidebar.

Your normal starting points for learning PostScript as language are that "red" book, otherwise known as the *PostScript Reference Manual* and the "blue" book, called the *PostScript Tutorial and Cookbook*. I now stock these and also have the *Whole Works* package of everything important by all major PostScript Authors. Details per my nearby *Synergetics* ad.

Contest: A free *Incredible Secret Money Machine II* for your best new aps for PIC's and PostScript.

Microcomputer pioneer and guru Don Lancaster is now the author of 35 books and countless articles. Don maintains his US technical helpline you'll find at (520) 428-4073, besides offering all his own books, reprints and consulting services.

Don has a free new catalog full of his latest insider secrets waiting for you. Your best calling times are 8-5 MST on weekdays.

Don is also the webmaster of his Guru's Lair at www.tinaja.com You could also reach Don at Synergetics, Box 809, Thatcher, AZ 85552. Or you can email don@tinaja.com

SOURCES

Adobe Acrobat
1585 Charleston Rd
Mountain View CA 94039
(800) 833-6687

Download Mechanic
3655 Nobel Drive
La Jolla CA 92037
(619) 587-4668

Ghostscript/Aladdin
Box 60264
Palo Alto CA 94306
(415) 322-0103

John Rees
81 Rees Road
Sautee GA 30571
(706) 865-5495

Synergetics
Box 809
Thatcher AZ 85552
(520) 428-4073

Videonics
1370 Dell Avenue
Campbell CA 95008
(408) 866-8300