Don Lancaster's

Ask the Guru

Selected reprints — volume III
Computer Shopper series (January 1990 - present)

Box 809, Thatcher, AZ 85552 (602) 428-4073

Book-on-Demand self-published using the Apple IIe computer and the LaserWriter NTX. All graphics were done in their entirety by ProDOS AppleWriter 2.1.

Introduction

And another volume bites the dust. Welcome to Volume III of our ongoing series of edited and updated reprints from my *Ask the Guru* columns that originally appeared in *Computer Shopper* magazine. This volume begins with column #59, which first ran in January of 1990.

In the fall of 1989, *Computer Shopper* was sold to Ziff-Davis and moved to New York. At that time, very serious publishing problems began. Problems that included erratic *Ask the Guru* appearances, heavy-handed (and often dead wrong) editing, disappearance of manuscripts, appalling delays, loss of key *Names & Numbers* sidebars, and really low printed figure quality.

My deepest and most profound apologies to those thousands of you readers who were run over roughshod by these problems. While the worst of these seem to be easing at present, only time will tell. Sigh.

As a workaround, I have now given *Ask the Guru* a new identity and a new primary home on the *GEnie* information service under the PSRT PostScript RoundTable. Preprints of all columns, including *exactly* what I want to say and how I want to say it, along with all of the full high quality camera-ready illustrations are directly available to you many months ahead of when the columns may or may not end up appearing in the magazine.

As usual, these reprints are about most anything I happen to end up being interested in. Recurring themes often do seem to include PostScript, laser printing, the Apple IIe and Macintosh computers, insider secrets that you are not supposed to find out about, thorough end-user long term product reviews, hardware and software hacks, emerging desktop opportunities, off-the-wall resources, *tinaja questing*, and Book-on-demand publishing.

Speaking of which, you are holding yet another example of my ongoing *Book-on-demand* publishing, a revolutionary new method in which books are produced only when and as ordered. All of the figures, text, artwork, and everything you see here was literally beat out on a brick in my backyard by using an Apple IIe computer and the *AppleWriter* word processor. And done in astonishingly high speeds with outstanding economics.

A reminder here that I have a no-charge voice helpline available to you at (602) 428-4073, and that you can get your *GEnie* PSRT voice connect info by dialing (800) 638-9636.

-- Don Lancaster
January, 1992

About the Author

As he has said in his classic *Incredible Secret Money Machine*, Don Lancaster writes books. And quests *tinajas*.

Microcomputer pioneer and guru Don Lancaster is now the author of 26 books and countless articles. He is considered by some to be the father of the personal computer, for his early ground-breaking work with hacker digital electronics and low cost video terminal displays. He is considered by others to be the patron saint of the Walter Mitties of the world. And, he is considered by yet others to be the… er, I guess we better skip that one.

His monthly columns include both the *Ask the Guru* and *LaserWriter Corner* over in *Computer Shopper*, and his *Hardware Hacker* column in *Radio Electronics* magazine. He is also the *Blatant Opportunist* in *Midnight Engineering*.

Some of his other titles include his *CMOS* and his million-seller *TTL Cookbooks*, *Micro Cookbooks* volumes *I* and *II*, *Enhancing your Apple II*, volumes *I* and *II*, the *AppleWriter Cookbook*, the *Active Filter Cookbook*, *Apple Assembly Cookbook*, his *Ask the Guru* and *Hardware Hacker* reprints, *Don Lancaster's PostScript Secrets*, and his *Intro to PostScript* video.

Don's current software offerings include his *PostScript Show and Tell*, plus a few companion disks for his various books. He is also the sysop for *GEnie PSRT*, the leading PostScript BBS in the country, and a registered developer for several leading edge technical firms.

Don is also the head honcho of *Synergetics*, a new-age design and consulting firm that specializes in Apple computing, laser printing, *PostScript* program utilities, electronic prototyping, Book-on-demand publishing, technical writing, and innovative software design. His avocations include firefighting, cave exploration, bicycling, and, of course, *tinaja* questing.

Don maintains a no charge voice helpline found at (602) 428-4073. He welcomes your calls and letters. Best calling times are 8-5 weekdays, *Mountain Standard Time*. ✦

Table of Contents

more...

Table of Contents, continued...

Don Lancaster's

ASK THE GURU

January, 1990

New Midnight Engineering
Some pad printing resources
Superstroking & Roundpath
EPS Encapsulated PostScript
Printed circuit breakthrough

Apple has now been making quite a few long-needed repairs, adjustments and corrections recently. On their IIgs, the very latest *GS/OS 5.2* operating system is now available by way of your local dealer or user group.

Besides correcting a fatal flaw that trashed disks and made *GS/OS* totally unusable, other corrections and improvements have now been added. Once again, do *not* use the older *GS/OS 5.0* ever for anything, since it fatally trashes disks.

On the *Quantum* 3-1/2 inch hard drives that used superglue instead of lubricant, the superglue is still in use, but the new ROM chip gives very much more shove to the arm moving mechanism. There's also a new "aerobic exerciser" that jumps the arm around every now and then to try and prevent lockups. Free upgrades and an extended warranty are available on this.

Apple belatedly discovered that the design for their *ImageWriter LQ* was not even wrong. They have at long last flushed this turkey for good. They are now offering all LQ owners an "upgrade" to the *LaserWriter SC* for only $500, or are providing other adjustments.

Which seems about the same to me as saying "Give me your ailing BMW and $500 cash, and we will put you in a brand new Yugo." As we have seen in past *Guru* columns, you could easily build your own fake *LaserWriter SC* for around $500 if anyone really wanted to. Since it lacks PostScript, I feel the -SC is totally useless.

The *Personal AppleLink* BBS program has been dropped completely, to disappear forever into the third party "me-too" woodwork under a name of *America On-Line*. The "real" *AppleLink* remains available and as useful as ever to developers, the dealers, user groups, and to any serious individuals.

Apple is also newly making their dealer's product training packages available directly to user groups at no charge. Contact your local club for more info.

As always, you'll find user group listings here in *Computer Shopper*, or else you can call (800) 538-9696, Ext 500 for a complete rundown of all the groups in your area.

One little-known yet outstanding Apple publication is their *Apple Library Users Group Newsletter*. These fat and free monthly volumes are chock full of all sorts of goodies.

Roger Wagner sent me a review copy of the latest upgrade to his new *HyperStudio GS*. Besides fixing a few teething bugs, this one is fully GS/OS 5.2 compatible. As an experiment, I did turn the package loose on two newer students who never even saw a IIgs before and did not have the slightest idea what hypermedia was. In just one class session, they managed to produce a

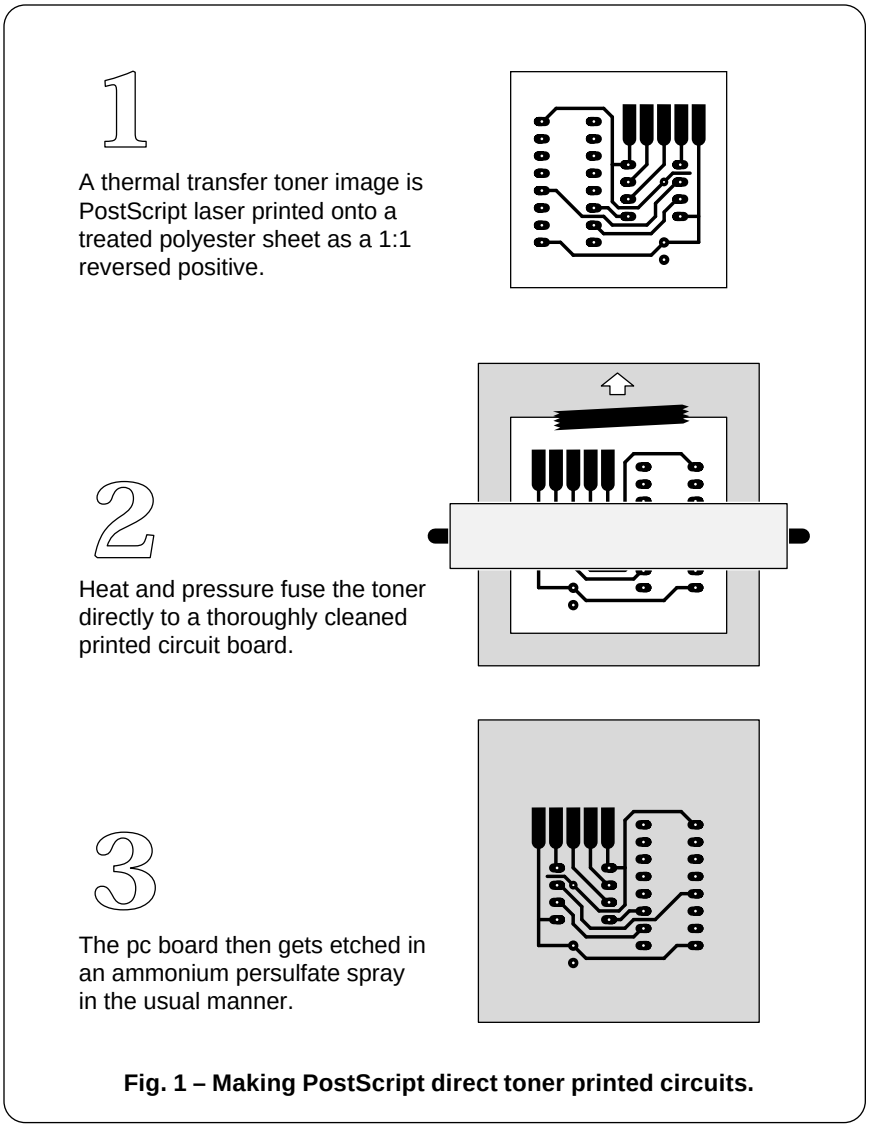


Fig. 1 – Making PostScript direct toner printed circuits.

rather sophisticated full color interactive on-line integrated circuit pinout directory. Not too shabby.

As I have said before, the triple combo of the *HyperStudio GS*, *AppleWorks 3.0*, and the *Apple Video Overlay Card* mentor programs should be able to give the IIgs new life and lead it upward and onward, whereever that might end up being.

Adobe Systems has just released a new Mac-based *Adobe Type Manager*. This incredible $99 utility lets you see real top quality PostScript fonts on your screen and then print them to your *ImageWriter* or any other dot matrix or inkjet printer. Once installed, the *ATM* is invisible and fully automatic for pretty near any popular Mac program. And since it is based on the outline-generating technology, only a single download is needed for *all* font sizes.

The quality of all the *ATM* screen images and dot matrix printouts are now ridiculously better than before. ATM is definitely in the "Golly Gee Mr. Science!" category.

There is a unique new magazine called *Midnight Engineering* which should be of crucial interest to any and all of you now developing and marketing your own hardware or software. This one has a great and no-nonsense shoot from the hip style, is crammed full of real world insider secrets, and seems very much cast into the *Incredible Secret Money Machine* mold. Free samples are available.

In addition, I've gathered most of my *Blatant Opportunist* columns into a new set of *Midnight Engineering Reprints*. Call for info.

As a big reminder, this is your column and you can get technical help and off-the-wall networking per the end blurb. I do have a new and freebie laser printing insider resources brochure for you and I am now shipping my brand new *LaserWriter Secrets* book / disk combo.

Also per usual, all of the *Names and Numbers* have been gathered together into the end appendix.

Actually, I don't really write this stuff. All I do is let it out. We seem to have a mixed bag this month...

Tell Me About Low Cost Printed Circuits

That magic of PostScript and an ancient yet little known insider's secret can gang up to dramatically simplify making all of your low-end printed circuit prototypes. In fact, you can now go from a cold start to a ready-to-etch circuit board in under a minute at a cost of a dime or less. And do so using nothing but your favorite word processor.

The little known secret is that laser printer toner can make an outstanding etch resist. The only trick is perfecting a technique to precisely bond your toner onto your copper in the first place.

Use of PostScript means that you're totally device independent, working with any old word processor on just about any brand of computer. The software investment can be *zero*, over and above your favorite word processor or comm program.

In the case of hacker pc boards in user groups or hobby magazines, each end user can get their own printed listing or a BBS download. Instead of working with a third- or fourth-generation artwork copy, you give them precise layout in the exact form the author intended.

Figure one shows you several key secrets of this direct toner process. You first get a PostScript textfile of your desired layout. You can do this on your own, by using my printed circuit utilities, or by running the PostScript driver from a commercial CAD / CAM layout program.

You then run a paper checking copy with any suitable PostScript printer. For most uses, a 300 DPI printer, such as a *LaserWriter NT*, will work just fine. Your image must be a 1:1 reversed positive, so that *left* is right and *black* is foil.

Note that PostScript will let you accurately rescale in either the X or Y direction, should your first results not end up precisely at 1:1.

The tricky part lies in getting the right kind of toner onto the correct type of transfer material. Right now, this is still wide open, so you'll want to experiment bunches here. Three good choices of transfer material are: (A) polyester (mylar) overhead transparency sheets, (B) the very same sheets lightly overcoated with some higher temperature mold release such as the *Miller-Stephenson* MS-136, or (C) a more stable and a higher temperature aramide sheet, such as *Kapton*.

There are several choices for toners. A good grade of third party refill graphics toner works well, as should those new T-Shirt thermal transfer toners. It also seems to be a good idea to use a "dry", (silicone oil

Pad printing: jack-of-all-trades in industrial decorating
Steve Duccilli, *Screen Printing*, April 89, pp 63-71.

Pad geometry
John Legat, *Screen Printing*, October 84, pp 92-96.

Pad printing equipment review
Editors, *Screen Printing*, October 84, pp 103-106

An evaluation of a unique image transfer process
Tamas Grecska, *Screen Printing*, October 84, pp 157-165.

Pad-transfer printing: soft touch in a tough market
David Karlyn, *Screen Printing*, August 82, pp 100-104.

Looking at pad transfer printing
Paul Wasserman, *Screen Printing*, October 79, pp 122-125.

Fig. 2 – Some pad printing resources.

free) fusion wiper pad for several copies before each pc image.

Your bare printed circuit board has to be incredibly clean. Start out using a thorough scouring with a chlorine-based (*Comet*) cleanser, followed by a commercial copper cleaner, and then ending up with a brief etch and a wash in distilled water. The board must be cleaned immediately before use.

Properly cleaned copper allows an unbroken stream of water to flow over it. It also will not be copper colored at all – your board will instead be a uniform *hot pink*.

The crucial step is the actual toner transfer. Tape the transfer sheet, toner side down onto your printed circuit board. While an iron could in theory be used, you'll get a far better result by using a *Kroy Kolor* machine or one of its imitators. Note that you will get improved stability using a heated point source roller than by heating the entire sheet.

Preheating the copper up to 150 degrees or so seems to be a good idea. It also seems to be a good idea to rub an ice cube on the back of the transfer sheet before peeling the sheet off the copper. An annealing bake of twenty seconds at 300 degrees also seems to improve toner adhesion and etchant resistance.

Some users do seem to get perfect results no matter what they try. With others, nothing works at all. Your goal here is to get an accurate and a dimensionally stable toner image securely bonded onto your copper, and zero toner left on the transfer sheet.

Your final step is the usual etch. You should get the best results by using ammonium persulfate at 120 degrees and agitated by sloshing or using an aquarium pump bubbler. A warming tray from a yard sale makes a good etchant heater.

Never etch face up! Always work vertically or face down. Only glass or plastic is allowed to ever contact the etchant solution. Typical etch time is twelve minutes or so. If it takes you much longer than this, your temperature is far too low.

Naturally, ammonium persulfate etchant is nasty stuff. This is an oxidant and sometimes includes a poisonous mercuric activator. Don't drink it. It slow burns any wood or other organics it comes in contact with. You have to wear gloves and safety glasses. Small quantities of spent etchant can be disposed of down the drain, provided you first dilute it using tremendous quantities of *cold* water.

Several resources on all this: A complete new low-end PostScript printed circuit layout package now appears as part of my *PostScript Show and Tell*. Lots of additional details on the direct toner method were shown in the December 1989 *Radio-Electronics*. That same issue shows you how to build a low cost homemade etching tank.

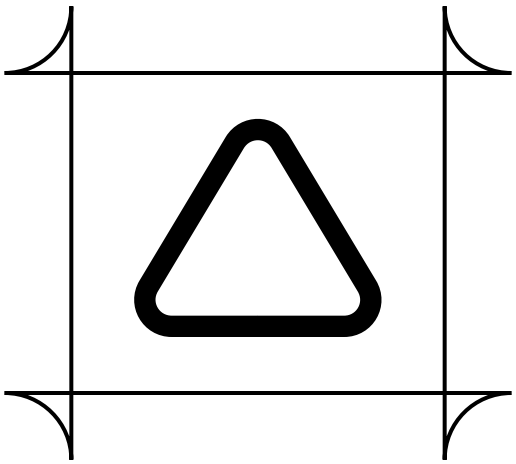
Two printed circuit trade journals include *Circuits Manufacturing* and *Electronic Packaging*.

Suitable toners are available from *Black Lightning*, *Don Thompson*, or *Lazer Products*. And, finally, printed circuit bare boards and the sodium persulfate etchants are stocked by the folks at *Kepro*.

As I've said, much of this is still experimental, so some playing around will be needed to get top results. Please let me know what does and does not work for you.

What is EPS, or "Encapsulated PostScript"?

The *Encapsulated PostScript*, or EPS file is really no big deal. All it



```
% Copyright c 1989 by Don Lancaster and Synergetics, 746 First Street,
% Box 809, Thatcher AZ 85552. (602) 428-4073. All commercial rights are
% reserved. Personal use permitted so long as this header remains present
% and intact. Show and Tell disk for Apple, Mac, or IBM costs $39.50.

/roundpath {/rpdata exch def /rprad exch def rpdata length 1 sub cvi /rppoints
exch def rpdata 0 get rpdata 1 get moveto 2 2  rppoints 2 sub {/rpvalue exch
def 0 1 3 {rpdata exch rpvalue add get } for rprad arcto pop pop pop pop} for
rpdata rppoints 1 sub get rpdata rppoints get lineto} def

%  //// demos - remove before use. ////

200 300 translate 10 dup scale

gsave 15 6 translate 1 [ 0 0 -5 0 0 8.33 5 0 0 0] roundpath
0.8 setlinewidth stroke grestore

gsave 15 3.5 translate -2.5 [0 0 -7 0 -7 12 7 12 7 0 0 0 ]
roundpath 0.15 setlinewidth stroke

showpage quit
```

Fig. 3 – A PostScript roundpath utility.

consists of is a segment of device-independent PostScript code that can be safely used by any larger PostScript application programs or other packages. This lets you move any of your PostScript figures or illustrations into your pagemaking programs, or otherwise link individual bits and pieces into the larger layout.

Let's begin right at the horse's whatever. The two key documents you need include the *Encapsulated PostScript File Specification*, Version 2.0, and that *Document Structuring Conventions Specification*, version 2.1. Both of these are available free on request through Cynthia Johnson at *Adobe Systems*, or else through one of the PostScript BBS systems.

The specs also appear in the new *Red Book II*. You should not create or use EPS files without having these two documents on hand.

In general, all of the document structuring conventions are all *voluntary*. The PostScript interpreter residing in your printer will ignore all of these comments and structure. Thus, some cooperation is needed between your exporting and your importing EPS code routines to let you decide what's genuinely needed and what is not.

As a general rule, any *conforming* document comment will begin with a *%%* or a *%!* and will be less than 255 characters long. Any comment extensions are allowed with a %+ comment line. Popular conforming comments include the PostScript version number, the title, program creator, the creation date, and so on. Other conforming comments could get included for such esoteric things as the paper weight and color, and included fonts, files, and procs.

On to an EPS file. In its simplest form, an ordinary textfile is used. The only *mandatory* requirements are this initial version line…

%!PS-Adobe-2.0 EPSF-2.0

and this size descriptor…

BoundingBox: LLx LLy URx URy

Here LLx is the distance in points from the left sheet edge to the left side of your printed image. LLy is your distance from the bottom sheet edge to the bottom of your printed image. URx is the distance from the left sheet edge to the right image side and URy is the distance from the bottom sheet edge to your top image limit.

Note that these numbers are completely independent of any scaling, translation, or rotation system in use within your EPS file. To find your bounding box, you print your file and then measure all of these dimensions in points upon your printed page. Use a standard point rule to do this.

The purpose of a bounding box is to let an applications package know how big the "standard" size of the image is, and just where it is to be located from a reference point.

These two conforming comments are the only two actually *required* by an EPS file. The additional conforming comments could get added as needed, or as requested by whatever is importing the EPS file.

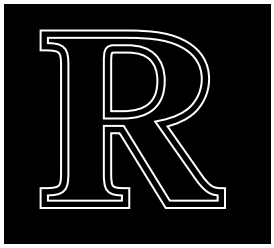
It's suggested that all conforming comments be ended by a…

%%EndComments

final place marker.

There are some restrictions to what type of PostScript code gets allowed inside an EPS file. The code must be well behaved, should be bracketed by a *save* and a *restore* and must not do anything that would corrupt what your importing package is trying to do.

The PostScript EPS code should



```
%  Copyright c 1989 by Don Lancaster and Synergetics, 746 First Street,
%  Box 809, Thatcher AZ 85552. (602) 428-4073. All commercial rights are
%  reserved. Personal use permitted so long as this header remains present
%  and intact. Show and Tell disk for Apple, Mac, or IBM costs $39.50.

/superstroke { save /sssnap exch def /sscmd exch def 0 2 sscmd length 2
div cvi 1 sub 2 mul {/aposn exch def gsave sscmd aposn get setlinewidth
sscmd aposn 1 add get setgray stroke grestore} for  sssnap restore
newpath} def

%  //// demos - remove before use. ////

200 300 translate 10 dup scale

106 45 {dup mul exch dup mul add 1.0 exch sub} setscreen

gsave 15 4 translate -6 0 moveto  12 0 rlineto 1 setlinecap [1 0 0.85 0.9]
superstroke grestore

gsave /Bookman-Demi findfont [9 0 0 9 0 0] makefont setfont
10 6 translate 0 0 moveto 10 0 rlineto 0 9 rlineto  -10 0 rlineto closepath fill 1.2
1.5 moveto  (R) false charpath [0.35 1 0.2 0] superstroke grestore

showpage quit
```

Fig. 4 – A PostScript superstroke utility.

involve one single page or smaller image. These following commands are *not* permitted, since they could corrupt the operation of your importing program...

note
initclip
copypage
erasepage
exitserver
nulldevice
initmatrix
grestoreall
banddevice
framedevice
initgraphics
renderbands
setpageparams

Several other obvious and device-specific operators are also no-no's, such as those covered in the LaserWriter white book. One obvious example is *initializedisk*.

In addition, the *setscreen* and *settransfer* operators are allowed only if you carefully do save and restore all the original screen and transfer functions. Although the *showpage* operator still is allowed in an EPS testfile, the importing program is supposed to be intelligent enough to ignore it. One detailed example appears in the EPS notes.

The rules here are simple: Don't do anything that can mess up the importing program. Do make sure the importing program can pick up exactly where it left off.

What I have just described is the *textfile* form of an EPS file. These usually will be host independent. It is also possible to include an EPS file into a device specific host program. This could get done to allow such things as non-PostScript screen images and host specific code.

Examples of these would include EPSF and PICT screen image files on the Mac, and the TIFF files on the pc clones. Once again, get the original EPS specs for further details.

Any More Details On Pad Printing?

Well, maybe a few. As you recall, *pad printing* is an arcane and sneaky process to full-color print onto such specialty objects as pens, golf balls, electronic keycaps, antique radio dials, classic car speedometers, or even on eggshells.

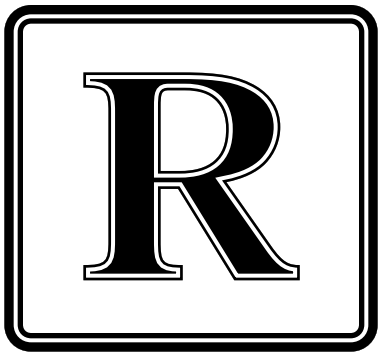
Obviously, PostScript cries to be combined with some great new and do-it-yourself home pad printing operation, to open up whole new worlds of ultra low cost, lower end specialty promotion products, sold direct and real time to the end user in sanely small quantities.

Here, briefly, is how pad printing works: An engraved plate called a *cliche* contains a flat and frontwards image of your artwork. Ink gets spread onto the plate and then doctored off, leaving an ink image. Your ink image is picked up by a special shaped silicon rubber pad and then transferred to the irregular object being printed.

This transfer occurs because ink exposed to air creates a thin and tacky film. Initially, the tack sticks to the pad. During the pad motion, a second and stronger tack builds up on the exposed surface, which lets the ink stick to the final object better than it does the pad.

A nesting carrier can hold the object you are printing. The shape of the pad is typically an *involute* of your final printing path, so that you can produce a distortion-free result. Dozens of pad shapes and sizes are common. Several colors can even be done at once.

So far, I haven't found any trade journal, ap-note or newsletter that specifically centers on pad printing.




```
% Copyright c 1989 by Don Lancaster and Synergetics, 746 First Street,
% Box 809, Thatcher AZ 85552. (602) 428-4073. All commercial rights are
% reserved. Personal use permitted so long as this header remains present
% and intact. Show and Tell disk for Apple, Mac, or IBM costs $39.50.

/superinsidestroke {save clip /sssnap exch def /sscmd exch def 0 2 sscmd
length 2 div cvi 1 sub 2 mul {/aposn exch def gsave sscmd aposn get 2 mul
setlinewidth sscmd aposn 1 add get setgray stroke grestore} for sssnap
restore newpath} def

% //// demos - remove before use. ////

200 300 translate 10 dup scale

gsave 15 3 translate 1 [0 0 -7 0  -7 13 7 13 7 0 0 0 ] roundpath
[0.7 0 0.5 1 0.35 0] superinsidestroke grestore

gsave /Bookman-Demi findfont [11 0 0 11 0 0] makefont setfont
10 5 translate 0.6 0.7 moveto (R) false charpath [1.3 0 0.25 1 0.1 0]
superinsidestroke grestore

showpage quit
```

Fig. 5 – A PostScript superinsidestroke utility.

A free *Incredible Secret Money Machine* to you if you find one.

At any rate, there is a fine *Screen Printing* trade journal that does an in-depth article on pad printing each year or two whether it needs it or not. Screen Printing's answer to me seems to be a Guru by the name of Richard Greaves. Richard kindly sent me one reprint of everything which *Screen Printing* has done on pad printing over the last decade. The list appears in figure two.

There appear to be several dozen pad printer manufacturers. Prices are, of course, outrageously high since this is an arcane specialty, and since most of these are high volume automatic machines.

Of the article listings in figure two, Steve Ducilli's shows the latest and best listing of manufacturers and prices.

One low-end pad printer system is available from *Basco*, but I do believe that this particular beast is ridiculously overpriced.

What if I Really Want to Get Arcane?

Hmmmm. Your next step above and beyond pad printing is an even more esoteric technology known as *diffusion transfer printing*. Sort of the ultimate in iron-on applique.

In diffusion transfer, full color ink is previously placed onto a transfer mylar sheet. Heat and pressure then vaporize the ink. The ink will then transfer *as a vapor* and literally diffuses onto and into whatever it is you are printing on. Since that ink actually *enters* into the intended substrate, your final image can be extremely durable and literally can not be rubbed off. On ceramics, the result can even be fired.

The diffusion transfer printing is coming on like gangbusters, but it is even harder to get any insider info on. Some ads occasionally appear in both the free *Design News* and the *Appliance* trade journals.

Tellyawhat. For this month's contest, just contribute in some way to our low end pad printing or our diffusion transfer printing dialog. Remember that our goals here are a PostScript based $50 do-it-yourself machine and a simple process that lets you get filthy rich at a swap meet. There'll be an *Incredible Secret Money Machine* book to the top dozen entries, plus an all-expense paid (FOB Thatcher, AZ) *tinaja quest* for two to the very best.

What are This Month's PostScript Utilities?

Let us have three of them this month. I will call them *roundpath*, *superstroke*, and *superinsidestroke*. They greatly simplify generating all sorts of special path and stroking effects. While these are a minor portion of my PostScript utilities 13, you can easily add them to most any package you care to.

Let's start with *roundpath*, found in figure three. This one is able to automatically round all the *internal* corners of a fancy path to a fixed radius. To use *roundpath*, you create a radius value and use an array of form *-rad- [$x_0 y_0 x_1 y_1 x_2 y_2 \ldots x_n y_n$] roundpath.*

Note that only the *internal joints* of the path are rounded. The ends are not. If you want to round all the joints of a closed figure, start at the *middle* of some side and make x_0 equal x_n and y_0 *equal* y_n, closing the path. One useful trick is to pick a negative value for the radius. This gives you some unique inverse arcs at each corner.

Rather interesting effects can get created by multiple stroking a path. For instance, a rounded-end fat black stroke overlaid by a thinner gray one creates an unusual underline for use on any letterheads and resumes. In a wiring diagram, an overwide white combined with a wide black and a thinner white will give you a "white" wire that automatically breaks any other wires or other parts it happens to cross over. And, any *charpath* font characters that are multi-stroked can lead to double outlines, litho spreads, and various "neon" effects.

The usual way to multiple stroke is to do a *gsave*, set your fattest linewidth and gray, stroke, and then get the path back again with a *grestore*. Then you repeat the process for each successively thinner line, ending up with the narrowest line. You do have to do the fattest stuff first; otherwise you'll overwrite your previous work.

The *superstroke* of figure four can automate this process. You simply create an array of *linewidth, gray* data values and then do a *[5 1 3 0 1.5 0.8] superstroke*. In this specific example, you first draw the white "background erasing" line as five points wide using a white or one setgray, followed by a three point wide black line at a zero setgray, followed finally by the thin gray fill that is 1.5 points thick.

A variation of *superstroke* is called *superinsidestroke* and is shown as figure five. This one first clips you to your original path boundary. While stroking gets attempted in exactly the same way as *superstroke*, only those portions of the lines lying *inside* your boundary will actually appear on the final printed page.

Which could result in borders having variable linewidths, or else single or multiple outlined fonts which do not ever get any fatter than their original size. Far better than the crude attempts usually tried. Note that the linewidth you set is the width you print.

While incredibly powerful, there are two main limitations to *superinsidestroke*. One is that the print time can get very long if there are curves or other complexities forming your clipping boundary. Especially with *superinsidestroke*. A second is that larger messages may have to be done a single character at a time.

Several interesting results can be gotten if you first create your path using *roundpath* and then multiple stroke your path by using either *superstroke* or *superinsidestroke*.

These three operators are now included in my PostScript Utilities version 13. You can call for a free printed listing.

Experiment to see what you can come up with on these. I think you will find this new trio an incredibly powerful new suite of PostScript everyday utilities. ❄

Don Lancaster's

ASK THE GURU

February, 1990

Secrets of Vinyl Lettering
5,000:1 PostScript Speedup
More on the Bezier Curves
A Third-Generation Printer
IBM/LaserWriter Interfacing

Meanwhile, consternation had broken out among all of those disgruntled mutineers encamped in the old Eskimo village. Er, whoops. Wrong column. Let's try again…

Plans are underfoot for a major new PostScript BBS board to open up on *GEnie*. The intent is to have at least a thousand heavy-duty free downloads, along with most of the routines found in my *Ask the Guru*, *LaserWriter Corner* and my *Hardware Hacker* columns. Its called PSRT.

Patches are at long last available for *AppleWriter* which let you have much longer resident textfiles on your IIgs. These patches are handy for downloading PostScript fonts. So far, the patches do seem to be compatible with *Don Thompson's WPL Toolkit* and many of my own patches. The package is shareware priced at $29 and is available from *Chester H. Page*.

Sadly, *Call A.P.P.L.E.* has ceased publication after all these years. My recommended substitute would be Tom Weishaar's great *A2 Central*. In yet another ominous development, *Sierra On-Line* announced they have discontinued all future IIgs product development, citing the intolerably frustrating development tools and the lack of sorely needed and long overdue IIgs hardware upgrades. Sierra, of course, was one of those Apple pioneers that long led the field in all of the high resolution graphic adventures. Sierra intends to continue to develop for all other major personal computers.

Over in the PostScript clone wars, *Freedom of the Press* has issued an upgrade that no longer breaks the tail off *Meowwrrr*, my PostScript Puss de Resistance. But *GoScript* still seems totally incapable of printing the PostScript fractal ferns that we looked at several columns ago. So much for PostScript compatibility.

That *Adobe Type Manager* is going

1. The printer shall speak genuine *Adobe* PostScript and use fully hinted downloaded Type I outline fonts.
2. The resolution is to be 415 DPI with an alternate 415/830 mode that allows 106 line halftone screens of 35 gray levels. Full histogram equalization shall be available.
3. The repeat print speed will be at least 12 pages per minute. Page makeready times shall be a minimum of 5 times faster than the *LaserWriter NTX.* A combination of new algorithms, software, and hardware is to be used for this speedup, including RISC-based or direct PostScript hardware engine chips.
4. An absolutely straight paper path will be provided as an option, allowing no-nonsense hand feeding of thicker materials. At least two 400 sheet minimum paper trays shall be provided, capable of automatically feeding heavy cover stock.
5. Envelope print quality is to be substantially better than on previous printers. The fusion rollers are to be separately accessible for use with *Kroy Kolor* and laminating materials.
6. As extra price options, an 11 x 17 capability will be offered, as shall the ability to print toner directly onto a 1/16th inch thick printed circuit board material.
7. Toner costs shall not exceed 0.5 cents per printed side, and shall be third-party reducible to under 0.2 cents per page. Toner cartridges must last a minimum of 10,000 copies and shall be conveniently end-user refillable a minimum of twelve times. The photosensitive toner drum shall have a *Mohs* scale surface hardness of not less than 9.0. Any and all parts of the toner and charging system must be low in cost, easy to get, and simple to replace.
8. The printer shall, in a single feeding, be able to duplex print on both sides of a sheet of paper. All rollers and all paper paths shall be capable of handling any previously printed material on a long-term basis. The reliability of the paper path and feeding mechanism shall be good enough to allow unattended overnight printing of book-on-demand published titles.
9. A minimum of 150 high quality, fully hinted, and outline fonts will be standard, with other fonts easily accessible as user-programmable EPROM cartridges and CD-ROM disks.
10. All descriptive font paths shall be available on all levels, including alterable plaintext representations of downloaded Type I fonts. The font paths must be easily interceptable and modifiable for such things as perspective and similar lettering or other nonlinear transformations.
11. There shall be no secrecy. *Everything* on *all* levels must be fully documented, fully unlocked, and fully available in plaintext form.
12. An optional hard disk shall have a fully documented, reliable, and user-friendly operating system that does not blow up during routine use. All disk files and formats shall be totally SCSI standard and fully documented. A mix of up to eight disks and CD ROM devices shall be allowed.
13. Provisions shall be made for direct hard disk access by the host, for the return of cached font characters to the host, and for use of the SCSI port as a direct and high speed data channel. The font cache is to be totally separable from other disk files and directories.
14. Minimum memory expansion will be 16 megabytes, and provisions are to be made for dual bitmaps that allow one page to compose while the other is printing.
15. Service manuals exceeding current *Hewlett-Packard* quality standards shall be readily available to the end user. These shall include full schematic diagrams, complete memory maps, and fully disassembled ROM listings. All source code shall be optionally available.
16. A minimum of 500,000 copies are required before minor repairs or parts replacements are needed. The machine is to last forever, simply by bolting on new parts.
17. A fully programmable video output of the final page bitmap shall be available. Any portion of any and all bitmaps shall be returnable to the host for recording, in both "open" and run-length encoded formats.
18. A compiling or pseudo-compiling provision is to be available that can dramatically speed up later reuse of any page image. A user system monitor is to be provided that gives complete and total access to all code levels.
19. Large carrying handles must be provided. Mode switches shall be rugged and easily changed. Toner density is to be adjustable from outside the machine. Unusual page formats as large as 8-1/2 x 24 shall be supported.
20. The end-user street price is to be under $3499.

Fig. 1 – An acceptable third generation PostScript printer.

gangbusters as the number one Mac utility and is now pushing the half million mark for unit sales. One discounting source is *MacWarehouse* at $53. The good news is that this gem can stunningly improve screen graphics and higher resolution dot matrix and most ink jet printouts. The bad news is that it only works on Type I downloaded fonts, and that it actually slightly degrades 10- and 12-point type on older *ImageWriter* printers. An IBM version of the ATM should be available soon.

Later versions of ATM fixed this *Imagewriter* bug.

As we have seen in our previous columns, *Hewlett-Packard* is one outstanding source of repair and service manuals for most of those Apple LaserWriter products. They have combined two of their older CX manuals into their new part #02606-90920. Their SX manual remains part #33440-90904.

A reminder about the *Midnight Engineering* magazine for all of you developing and marketing your own low end hardware or software. By way of a special arrangement, free sample copies are available to *Computer Shopper* readers.

As always, this is your column and you can get technical help and off-the-wall networking by calling (602) 428-4073. I do have a new and free laser printing *insider resources* brochure just for you and I am now shipping my brand new *LaserWriter Secrets* book and disk combo.

We seem to be into advanced topics this month. So, I thought we'd have us five really advanced contests to go with them. There'll be all the usual *Incredible Secret Money Machine* prizes for the dozen best entries, along with an all expense paid (FOB Thatcher, AZ) *tinaja quest* for two to the best of all.

For the first two contests, either (1) show me a way to use the SCSI port on the LaserWriter NTX as a ridiculously faster *AppleTalk* substitute, or else (2) show me a way to compatibly timeshare a single hard disk between an NTX and a host on a real time basis. Be sure to send your entries directly to me, and not to the *Computer Shopper* editorial. On to the goodies...

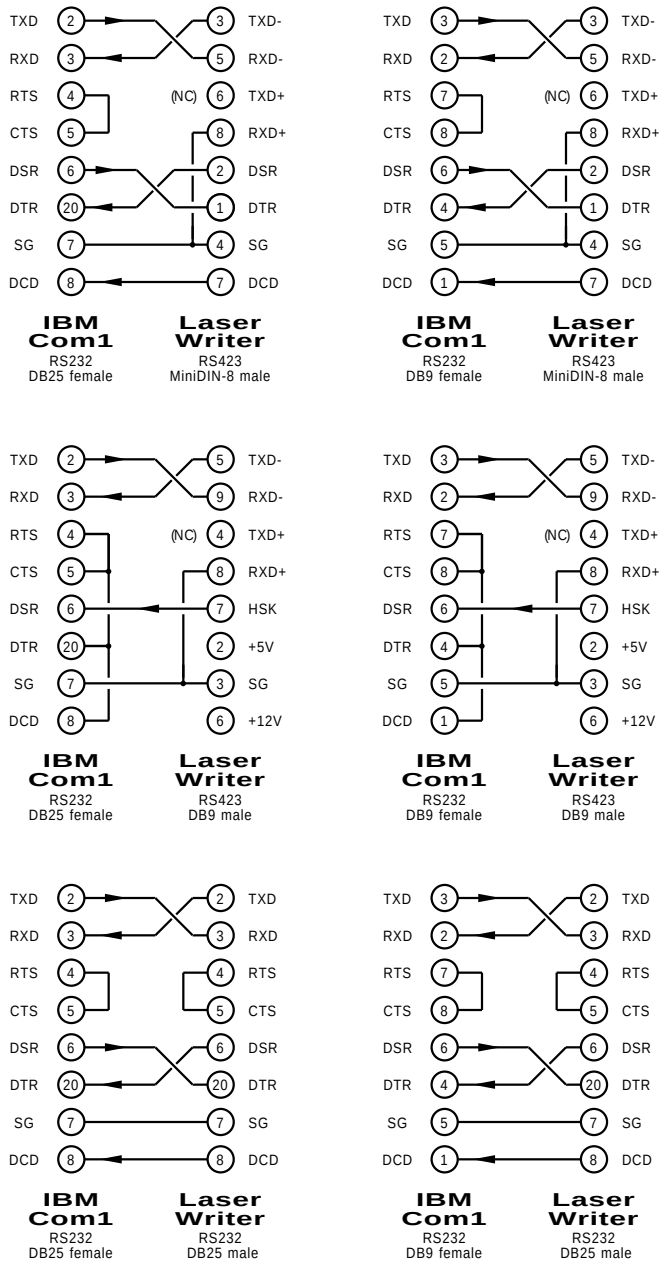


Fig. 2 – Six possible IBM to LaserWriter cables.

What do You Really Want In A Third-Generation Printer?

I never thought you would ask. Dozens of manufacturers seem to be on the verge of releasing a bunch of brand new PostScript laser printers. Sadly, most of them seem about to offer some monumentally stupid features for precisely the wrong reasons. Arrghhh.

To try and head things off at the pass, figure one gives you the bare minimum specs for an acceptable third-generation PostScript laser printer. This is based upon your

helpline calls and requests, on my own needs, and on lengthly talks with the other PostScript power users and developers. Every feature on this list is both technically and economically feasible here and now.

As I now see it, there are four main areas which severely cripple today's PostScript printers....

The toner atrocity – There's no sane reason why toner should cost more than ink, nor any reason whatsoever to exceed a toner cost of 0.2 cents per printed page. Instead, we now have these purposely undersized and overpriced hard-to-refill *Kamikaze* cartridges filled with excessively abrasive toner and an intentionally softened drum.

The font path paranoia – By depriving power users of a simple and direct access to the actual font paths used, many PostScript routines are slowed down by several orders of magnitude. Fancier effects such as perspective or star wars lettering are made unduly difficult.

The speed debacle – Many of the perceived speed problems in PostScript are caused by external hassles. Things such as the lack of display PostScript on your host. Or the sad absence of any direct SCSI replacement for unbearably slow *AppleTalk* or other serial comm channels, the lack of any general compiling routines, excessive host software overhead, and the failure to properly interact between the PostScript printer and the host.

The bitmap obstinticity – Although a device independence might be a laudable goal for PostScript, in no way should it get shoved down everyone's throat. By giving end users direct and simple access to any and all bitmaps, such things as step-and- repeat business cards can be ridiculously sped up. A video output of the bitmap as it is being generated would end up extremely handy for your debugging, and for finding any slower portions of your code. The ability to compact and save whole page bitmaps to CD-ROM would dramatically improve any Book-on-Demand publishing. And finally, some programmable line-by-line bitmap output would open up PostScript to previously unthunk of great opportunities, allowing a printer to interface to virtually any other output device, however specialized or limited its niche market.

Improving PostScript is kind of an ongoing dialog around here, so be sure to let me know what else you want to see in the way of any printer upgrades. We also offer professional services in this area.

Usually, all we need is an *accurate* sketch of what you want, sample input and output, and a written description of your problem. The turnaround time is typically a few weeks.

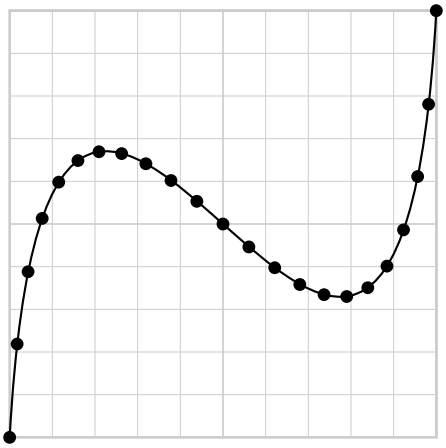
How About Another Round on IBM -to- PostScript Printer Interface Problems?

Here we go again. That helpline keeps ringing off the hook with IBM to PostScript printer interface problems. By far the overwhelming majority of these are caused by user inattention to detail...

"4073. Your PostScript helpline".

"Your #$@$#& Free Font will not

```
% Copyright c 1990 by Don Lancaster and Synergetics, 3860 West First Street, Box 809,
% Thatcher AZ 85552. (602) 428-4073. All commercial rights reserved. Personal use permitted so
% long as this header remains intact. Show and Tell disks for Apple, Mac, or IBM cost $39.50.
```



```
% Returns the approximate length of a selected Bezier curve to the host. Value /numpoints
% determines the accuracy. 100 is usually good enough. Result /blength holds the final length,
% usually to 0.1% accuracy For best results, use with the rubbergrid from my PostScript utilities.

/pcurveto {6 copy /y3 exch def /x3 exch def /y2 exch def /x2 exch def /y1 exch def /x1 exch def
currentpoint /y0 exch def /x0 exch def curveto} def

/xtt {x3 x2 3 mul sub x1 3 mul add x0 sub tt 3 exp mul x2 3 mul x1 6 mul neg add x0 3 mul add tt
dup mul mul add x1 3 mul x0 3 mul neg add tt mul add x0 add} def  % x coefficients as f(t)

/ytt {y3 y2 3 mul sub y1 3 mul add y0 sub tt 3 exp mul y2 3 mul y1 6 mul neg add y0 3 mul add tt
dup mul mul add y1 3 mul y0 3 mul neg add tt mul add y0 add} def % y coefficients as f(t)

/plotdots {0 1 numpoints 1 sub div 1.00001 {/tt exch def newpath xtt ytt moveto 0 0 0.2 0 360 arc fill}
for }def

/bezierlength {/oldx x0 def /oldy y0 def /blength 0 def 0 1 numpoints 1 sub div 1.0001 {/tt exch def
xtt ytt /newy exch def /newx exch def newx oldx sub dup mul newy oldy sub dup mul add sqrt
blength add /blength exch def /oldy newy def /oldx newx def} for }def % approximate curve with
line segments

% //// demo - remove before use. ////

200 200 moveto 10 dup scale 1 setlinewidth 0 0 moveto
1 18 9 -8 10 10 pcurveto stroke  % substitute desired curve here

/numpoints 23 def plotdots bezierlength blength == flush showpage quit
```

Fig. 3 – Approximate length of a PostScript Bezier curve.

work on my PC clone".

"What PostScript error message do you get back?"

"The lights blink for a while."

"What PostScript error message do you get back?"

"Huh?"

Back to square one. Yes, an IBM or a PC clone will easily support any PostScript laser printer, provided you do follow a few common sense rules. Firstoff, do **not**, under any circumstances **ever** use a parallel interface between your PC clone and your PostScript laser printer. To do so deprives you of crucial return error messages, severely limits what you can accomplish with your new PostScript language, and makes all of your printer drivers unbearably restrictive and klutzy.

Any software that only supports PostScript parallel interface is, by definition, defective and should be immediately flushed.

Second, do not ever, under any circumstances, try to print by copying out your COM-1 port. Instead, always use a full duplex comm channel, such as *PC Talk* shareware, that lets you view your return error messages and optionally record any PostScript output returned to your host. While the typical parameters are 9600 baud, 8 data bits, 1 stop bit, and no parity, you can later make your interface go as fast as 57,600 baud. Which is much faster than most parallel interfaces.

Third, be certain you do set up an XON / XOFF handshaking environment. If your handshaking is set wrong, errors will often be picked up after a page or two is printed. While you can, in theory, change your laser printer to do hardwire handshaking, there is absolutely no reason to ever do so. Always use XON / XOFF handshakes.

Fourth, after you have set up a reliable two-way serial communications with full error reporting, you install a printing and stack dumping error trapper. You can get one for $10 directly from *Adobe Systems*, pick one up off a BBS, find it in the green book, or else use the copy in my *PostScript Beginner Stuff*.

A shareware version of Adobe's EHANDLER.PS is also available as file #196 on *GEnie* PSRT.

Cables. It is normally considered proper form to have one between your computer and a printer. Figure two shows you the six most popular IBM to LaserWriter cable lashups.

The six combinations result since you can have a DB-25 or DB-9 connector on the host and a DB-25 using RS-232-C or (depending upon your LaserWriter model) a DB-9 or a Mini DIN-8 connector using the RS-423 interface serial standard.

These cables are available ready-to-go through the *Redmond Cable*, folks among other sources.

Some general guidelines here, in case you are trying out something special. When using RS-232 at both ends, you want to use what the IBM folks call a *null modem*, or what

Proc cacheing can give you a 12:1 up to a 3,000,000:1 speedup of your PostScript routines for any computation-intensive image that is to be reused at least once. To cache a proc, you define that proc as a character in a font, and then let the existing font machinery do the cacheing for you.

There are four possibilities…

(1) Font defined inside the job (speedup goes away with the job).
(2) Font persistently downloaded (fast so long as power applied).
(3) Font saved to NTX hard disk (stays fast until your disk blows up).
(4) NTX hard disk bitmap returned to host (can be permanently fast).

And here are some use rules…

1. Open up your font cache as wide as possible by using a *mark 1250 n setcacheparams*. The *n* value can be 48K (4 square inches) on a 3 meg NTX up to 188K (16 square inches) on a 12 meg NTX. For larger images, you can use several characters side by side.
2. A font definition is not allowed to contain an *image* operator; or a *setgray*, *setrgbcolor*, *sethsbcolor*, *settransfer*, or *setscreen* commands. Because a font dictionary ends up read only, a "floating" dictionary has to be used for any internal *save*, *restore*, or *def* commands.
3. One separate character needs to be created for each shade of gray in use. A white fill requires a separate mask as shown in figure five.
4. Use a font matrix of [1 0 0 1 0 0] and a point size of 1.0.
5. Font caches can be stored in run-length encoded format or as full bitmaps. The run-length encoding often uses much less memory with only a minor speed penalty. The *m* in *mark m n setcacheparams* decides whether full bitmaps or run length encoding will get used. To guarantee a bitmap, make *m* equal *n*. See the white book for details.
6. To verify that cacheing is taking place, run your routine twice, and measure the speed each time. Trip #2 should be ridiculously faster if the cache machinery really worked.
7. Should you get no image, this usually means you have inadvertently substituted *Courier* at 1/1000th normal size and that something is wrong with your font definitions.
8. Fonts used to define your proc cache are not in themselves cached.
9. The needed cache size can be estimated by multiplying the height in pixels by the width in pixels and adding around a thousand bytes. With enough side-by-side characters, even an entire page can be cached either in a bitmap or run length encoded form.

Fig. 4 – PostScript proc cacheing guidelines.

Apple will call a *printer cable* . Do note that your data channels and your handshaking are *crossed* in this type of cable.

IBM sometimes uses the RS-232-C auxiliary handshake on pins 4 and 5. But these are rarely used elsewhere and are best locally self-jumpered.

When using RS-232 at the host and RS-423 at the printer, note that the TXD- from the printer becomes RXD at your host, and that there is no connection to TXD+.

Note further that the transmitted data out will go to RXD- and that RXD+ must get *grounded* . Failure to ground the differential RXD+ input is far and away the most common problem on any RS-232 to RS-423 interface. Watch this detail.

Instead of any custom cables, it is often better to use a stock cable and add a terminating "pin rearranger," built up from a few short wires and a pair of *Radio Shack* male and female connectors. A pair of loops of #14 house wiring can be used to solder your two connector cases to each other.

Please note that you definitely must not ever use a "straight" or a "crossed" DB-9 to DB-9 cable, as your connections will be wrong.

Also note that some PC clone comm software may require a cold boot to change parameters. During your initial debug, always do a cold reboot and make sure your printer is a solid green before any retry.

Additional details on laser printer interface and debug appear in my *LaserWriter Secrets* , and the Apple White Book, otherwise known as the *LaserWriter Technical Reference Manual* . I do stock these.

Any More Info On Bezier Curves?

Lots of stuff. We do have a *tinaja quest* winner from our avuncular sleezoid contest. Michael Chaplin not only picked up the football on this one, but he ran out of the stadium and hasn't been seen since.

Mike's mind-blowing routines do include entire sleezoid alphabets and incredible metamorphically tweened sleezoid animation sequences. Write him for details.

It appears *Hewlett-Packard* has picked up on avuncular sleezoids in a very big way. They have upped their sleezoids to three dimensions and now call them *NURBS* instead. Leave it to HP to abandon the industry standard notation. A good introduction on this appeared in the October 1989 issue of the *Computer Graphics Review* .

I was sadly disappointed with the results from our find-the-length-of-a-Bezier-curve contest. A few math freaks claimed that a Bezier curve does not have a length, since a discontinuity is often possible. And several of those other entries pretty

```
%  Copyright c 1990 by Don Lancaster and Synergetics, 3860 West First Street, Box 809,
%  Thatcher AZ 85552. (602) 428-4073. All commercial rights are reserved. Personal use is
%  permitted so long as this header remains intact. Show and Tell disks cost $39.50.
```


```
%  This is my good old poison ivy can label, modified so it will be legal as a pair of font
%  characters.  Character (a) will be the white fill mask, while (b) will be the actual label…

/drawlabel {gsave stuffdict begin stuffdict exch /maskonly exch put stuffdict /overscan 1.0 put
/pixellineremap {0 1 overscan mul pxpwidth 300 mul 72 div cvi {stuffdict exch /sline# exch put
gsave gsave mappingproc newpath sline# 72 mul 300 div 0 moveto 0 pxpheight rlineto 0 0
rlineto 0 pxpheight neg rlineto closepath clip newpath pixelproc grestore grestore} for} stuffdict 3
1 roll put /mappingproc {sline# .24 mul dup neg exch pxpwidth 2 div sub canwide div 114.6 mul
dup sin canwide 2 div mul exch 3 1 roll add exch cos 1 exch sub tiltangle sin canwide 2 div mul
mul translate} stuffdict 3 1 roll put 94 18 translate stuffdict /tiltangle 14 put stuffdict /canhi 50
tiltangle sin div put stuffdict /canwide 200 put /NewCenturySchlbk-Bold findfont [30 0 0 33 0 0]
makefont setfont stuffdict /labelypos 8 put stuffdict /pxpwidth 238 put stuffdict /pxpheight 35 put
0 canwide 2 div tiltangle sin mul neg labelypos add translate /pixelproc { 0 0 moveto 0 pxpheight
rlineto pxpwidth 0 rlineto 0 pxpheight neg rlineto closepath maskonly {fill} {4 setlinewidth stroke
12 6 moveto 1 0 (FREE  FONT) ashow} ifelse} stuffdict 3 1 roll put pixellineremap end grestore}
def

/stuffdict 50 dict def  % a work place since internal def's are a no-no

% This is a more or less standard custom font builder, except for the matrix size…

/newfont 200 dict def newfont begin /FontType 3 def /FontMatrix [1 0 0 1 0 0] def /fontBBox
[0 0 188 54] def

/Encoding 256 array def 0 1 255 {Encoding exch /.notdef put} for Encoding dup (a) 0 get
/canwhite put dup (b) 0 get /canfont put

/Metrics 15 dict def Metrics begin /canwhite 0 def /canfont 0 def end

/BBox 8 dict def BBox begin /.notdef [0 0 0 0] def /canwhite [0 0 188 54] def /canfont
[0 0 188 54] def end

/CharacterDefs 10 dict def CharacterDefs begin /.notdef { } def /canwhite {true drawlabel} def
/canfont {false drawlabel} def end

/BuildChar {0 begin /char exch def /fontdict exch def /charname fontdict /Encoding get char get
def fontdict begin Metrics charname get 0 BBox charname get aload pop setcachedevice
CharacterDefs charname get exec end end} def

/BuildChar load 0 3 dict put /UniqueID 1111 def end /CanFont newfont definefont pop

% //// demo - remove before use. ////

/CanFont findfont [1 0 0 1 0 0] makefont setfont
/fullbitmap true def  % false for run length encoded

fullbitmap {mark 50000 23050 setcacheparams}{mark 50 23050 setcacheparams} ifelse

200 200 moveto 1 setgray (a) show 0 setgray (b) show showpage          % the slow one
200 500 moveto 1 setgray (a) show 0 setgray (b) show showpage quit     % absurdly faster
```

Fig. 5 – A 5000:1 PostScript speedup example.

much cancelled each other out.

I am still convinced that a simple closed expression for the length of a Bezier curve exists and that this is a trivially easy one to derive, given a creative enough transformation. Only I just can't seem to find it. So, for contest number three of this month, show me a closed formula for the length of a Bezier or other cubic spline curve.

Meanwhile, figure three shows you how to accurately approximate the length of a Bezier curve. You do this by using as many straight line segments as you think you need and adding them up. Sort of a *Simpson's Rule versus Pythagoras and the Teenage Mutant Ninja Night Nurses* type of thing. Since the points are closer together along all of the "more bent" portions of the curve, the results are surprisingly accurate. One hundred rapidly calculated points gets you well under an 0.1 percent error for most curves. This is more than good enough when you're plotting pixels.

Tell me all About Vinyl Lettering.

Toner sticks quite nicely to vinyl, which strongly suggests using your LaserWriter and an X-Acto knife to replace a $15,000 lettering machine.

Several tips here. Your vinyl must be carrier supported, and I would recommend trimming at least one-quarter of an inch or more off each edge before hand feeding. Your carrier itself should be precisely precut to 8-1/2 by 11 inches.

Dozens of sources for the vinyl lettering materials often advertise in *SignCraft* magazine; one typical supplier would be *Vinyl Express*. The usual pricing is around 50 cents to a dollar per square foot.

There are dozens of nice colors, including exotics which simulate etched or frosted glass, chromes, reflectives, translucents, and lots more. Some are self-adhesive, while others use application tapes.

Other options include the *Form-X Films* and the heavier *static cling* materials, such as those offered by *Joseph Struhl*. These are intended for long-term sticking to glass and to similar glossy surfaces simply by wetting the surface and squeegeeing them on.

Thicknesses vary from two to ten mils. I did seem to pick up some SX transfer assembly problems after I jammed a ten mil self-cling piece, so I would recommend either using the thinner materials, or else first creating a transfer pattern.

PostScript easily does backwards lettering by doing a *-1 1 scale*. This lets you put the letters on the inside of the glass, where they should be more vandal resistant. More details on this in *LaserWriter Secrets*.

I keep hearing persistent rumors of a magic new material which just might revolutionize both sign-making and PostScript laser pattern making in general. But every time I try to chase this down it seems to vanish. Poof even.

The story goes like this: You take a sheet of a carrier-supported and light-sensitive previnyl which is soft and only partially chemically cross-linked. You run this through your laser printer, placing toner only on the outlines needed.

Then you expose the pre-vinyl to the sun or strong light, which in turn hardens and completes the cross-linking into a vinyl polymer everywhere the toner was *absent*. Then you wipe on some glop which dissolves both your toner and the previnyl under the toner. Presto. Instant cut letters, ready for use.

A free *Incredible Secret Money Machine* if you can show me where to get this magic material.

So, What's the Cache?

I was going to show you how to double or triple your speed for this month's PostScript utility, but it appears our demo example ended up with a 5,000:1 speedup instead.

Adjectives such as "adequate" immediately come to mind.

I call my secret new technique *proc cacheing*, and it could work on any PostScript procedure that you want to reprint at least once at the same size at some future time. It is particularly good when used with any computation-intensive routines which involve irregular clipping intervals, custom logos, fancy font footwork, any extensive non-linear transformations, your curve-traced signatures, repeated randomizing, or extensive fractals.

Proc cacheing is more or less free, since all it requires is some re-thinking and a modest change or two in your programming style. While it works best on the NTX with a hard disk, you can apply it to any PostScript laser printer. You can even use your proc cacheing to capture and save your bitmap of an entire page.

PostScript has a built-in *font cache* which converts the outline descriptions of all fonts into bitmaps for later reuse. All you have to do to proc cache is *define your PostScript procedure as one or more characters in a custom font!* Any repeated use of your proc is then done as a bitmap, rather than by recalculating everything all over again. The typical run time speedups will range from 12:1 to 3,000,000:1.

Figure four summarizes the key proc cacheing rules, while a detailed example appears in figure five.

There are four major ways to use proc cacheing…

(A) You can define your font only inside your job, in which case the proc cacheing goes away with the job. This gets handy for all 12-up business cards where you do not want to permanently tie up any memory, or are using an older machine with limited memory.

(B) You can persistently download your font, so that the proc cacheing remains as long as power is applied. This does reserve some cache memory, and using lots of new fonts may in fact overwrite your proc cache.

(C) You can let the NTX hard disk automatically grab your proc cache. It will keep the bitmap for you until the next time your hard disk blows up. Your font should also be saved to disk for proper operation.

(D) You can read all your bitmaps saved to the NTX hard disk and

return them back to your host for recording. They appear on your directory as FC type textfiles. Which will give you a permanently fast version of your routine. This final route is especially attractive for Book-on-Demand publishing.

At any rate, your standard font cacheing is controlled by a *mark* 1250 12500 *setcacheparams*. Those 12500 bytes equal 12500 x 8 = 100,000 pixels, and sets a maximum possible size for any bitmap to be cached.

There are often two font cacheing mechanisms used. One generates a real bitmap, while the other uses a slower *run length encoding*, which produces a fairly fast image while using much less memory. The 1250 in the above example decides when to switch from real bitmaps into a run length encoding. The 1250 bytes will equal 1250 x 8 = 10,000 pixels.

Now, the total number of pixels per character bitmap will equal the height in pixels times the width in pixels. Assuming a square bounding box, the square root of the bitmap size gives us the height of the character. In this stock example, we could use real bitmaps up to 100 pixels high (or 24 points at 300 DPI). The run length encoding is used up to 316 pixels high (or 76 points at 300 DPI). Any of the characters larger than this will not go into the font cache, but will get recalculated each and every time.

Obviously, we need larger images than these to be really useful. The allowable maximum size for *n* in *mark m n setcacheparams* can change with the available printer memory. To find your limit, just keep on increasing *n* until you get a *limitcheck* error. On the 3 Megabyte NTX, you're allowed an *n* value around 48,000, which will translate into a bounding box of four square inches.

This quadruples to sixteen square inches on an NTX having a full 12 Megabytes installed.

These font cache size limits aren't nearly as bad as they sound. Note that you only have to proc cache your tightly trimmed slow stuff, which is often much smaller than your entire image. You are also free to use as many characters as you need to build up your full image.

Do note that as few as a mere *six* characters can proc cache an entire page bitmap when using a "full" NTX. Finally, note that you might use a full NTX to do all your proc cacheing, return the results to your host, and then rerun the final results on any old PostScript printer of any memory size.

There are several subtle gotchas, though. The *only* goal of your proc must be to make marks on the page. To qualify for font cacheing, your proc descriptions must not have any *setgray*, *sethsbcolor*, *setrgbcolor*, *settransfer*, *setscreen*, or any *image* calls present in them.

Since a font dictionary gets made read-only during its processing, you will have to provide an internal "floating dictionary" or two if you want to use a *save*, *restore*, or a *def* command. Otherwise, you will get one or more of the *invalidaccess* error messages.

So how do you handle grays or white fills? Simply by using a few additional characters. You use one character as a mask for each black, gray, or white level as needed. Our example of figure five uses the (a) character to erase the background to white and then overwrites the (b) character on top of it to create the actual label. Obviously, you do have to put the characters down in the proper sequence and in the correct colors to get the desired final result.

Two other differences between the proc cacheing and a "real" font: You use a font matrix of [1 0 0 1 0 0] and a point size of 1.0.

Because of pixel line remapping and all of the intensive non-linear transformations, the wraparound isometric label would normally print in 70 seconds on an NTX. As a run length encoded cache, it prints in 83 milliseconds. As a cached bitmap, it prints in 14 milliseconds. That's a speedup of 5,000:1. Which is not half bad for an amateur. Note that 6:1 speed penalty of your run length encoding over using the full bitmap. No big deal.

Your own speedups will depend on how calculation-intensive your proc is, and can range from a low of 12:1 for a simple 12-up business card, on up to several million to one or more for a fractal landscape.

Very handily, any other fonts used internally to create your proc font are *not* in themselves cached. Which can eliminate a major source of NTX hard disk blowups on such things as perspective letters. It can also slow down your first proc-as-a-font runtime, so watch this detail.

Some black magic does appear involved in tricking the NTX hard disk to actually cache your proc, even after it is obviously in the RAM cache in printer memory. Try (A) Persistently downloading your new proc, (B) Running your proc, along with a dozen standard font characters in an unused and oddball size; and then (C) Issuing a *control-c* to force a hard disk cache update, saving your proc.

I suspect this has something to do with Type 3 versus Type 1 Fonts. Let me know if you can come up with something better here, or at least an explanation of what is going on.

Once again, you can use several characters side by side for the larger images, and you can even cache an entire page as a nearly instant printing bitmap.

The newly announced PostScript level II does include several easy to use proc cacheing features. These are included in their forms and user path operators. More on this as the code becomes available.

Most of the PostScript routines you see here are now available on our new *GEnie* PSRT roundtable, in ready-to-run formats. You can call (800) 638-9636 for voice connect info. You can also get my *Ask the Guru* column preprints here as well.

As our final two contests for this month, either (4) show me a disgustingly sneaky new use for my proc cacheing, or else for you supergonzo PostScript folks out there, (5) find me a buyer for an 81 *VW* van with a mere 148,000 off-road miles on it. This last one really has got me stumped.

Let's hear from you. ❄

Don Lancaster's

ASK THE GURU

March, 1990

Apple Fiesta Details
Blackflashing Secrets
Understanding EEXEC
Making Flippity-Floppers
Starting out with PostScript

This year's *Apple Fiesta* will be held June 16, 17, and 18th at the Tempe, AZ *Mission Palms Resort*. As usual, this will be one of the finest club-sponsored regional shows anywhere and will include exhibits, workshops, and seminars of interest on Apple, Mac, PostScript, user groups, and on desktop publishing.

Also as usual, I'll be there talking on PostScript and Book-on-Demand publishing, while Bee might do a workshop or two on high tech opportunities for the working artist. Be sure to stop in and visit.

Cost for the weekend is around $20 and special off-season room rates are available. For any further details on this Apple Fiesta, contact Jerry Cline at (602) 992-7035.

Apple has a new slick magazine titled *Develop* that now includes a CD-ROM disk chock full of tech notes and new developer info.

Included is that new Apple II *Dynamo* code plus bunches of old and new Mac stuff. Price is $30 per year if you are not already an Apple developer. You can contact APDA for more details.

The earlier versions of the *Apple* CD-ROM drive apparently include a curious flaw. It seems the fan blew enough dirt onto the optics to foul them up. Their latest version does not have a fan, and a repair and a new upgrade policy has been announced. See your dealer for info.

Apple also offers a "one-unit-only" bargain price on these drives for user groups, dealers, and for developers. Expect the majority of future *Apple* system software and technical info to appear exclusively on CD-ROM.

A reminder that there should be a major new and highly independent PostScript board up on *Genie* by the time you read this. Page 835.

Adobe Systems just released an interesting new program called *The Distillery* and newly added to their developer's disks for both Mac and IBM. This one can do a pseudo-compiling which gets returned to your host for recording. It very substantially speeds up and often can shorten most PostScript code. Obviously, it is intended for those applications where you want to rerun your code at least once.

Word has it that there is a fairly serious teething problem in the new low end 4-PPM laser printers as now being offered by *Canon*, H-P, and, someday soon, by *Apple*. This can involve an overheating of the fusion rollers and/or the power supply. A few fires have resulted.

A fix is certainly in the works and will probably be in place by the time you read this. For now, if you own one of these printers, do not run it unattended and do keep a *Halon* fire extinguisher handy. See your dealer for further upgrade info.

Specifically, look out for an *Error 50* message on your *Hewlett-Packard* Laserjet IIP. Disconnect your power immediately should this occur.

Yes, all these smaller laser printer cartridges can be refilled. Two of the usual sources for supplies, tools, and materials include *Lazer Products* and *Don Thompson*.

Let's see. I am now shipping my new *Don Lancaster's LaserWriter Secrets* book/disk combo. Call me per the end blurb for more info.

On to a matter of great import...

What is a Flippity-Flopper?

Very rarely can *Computer Shopper* reveal any emerging grest new technological concept so profound that the entire future of our Western Civilization hinges on it. Life in the universe as we know it will never be the same.

Not to mention that you can also make a buck on it at a swap meet.

Figure one shows you the details. A *Flippity-Flopper* is a two sided disk that sticks on either side to a *velcro* dot that's itself stuck on whatever.

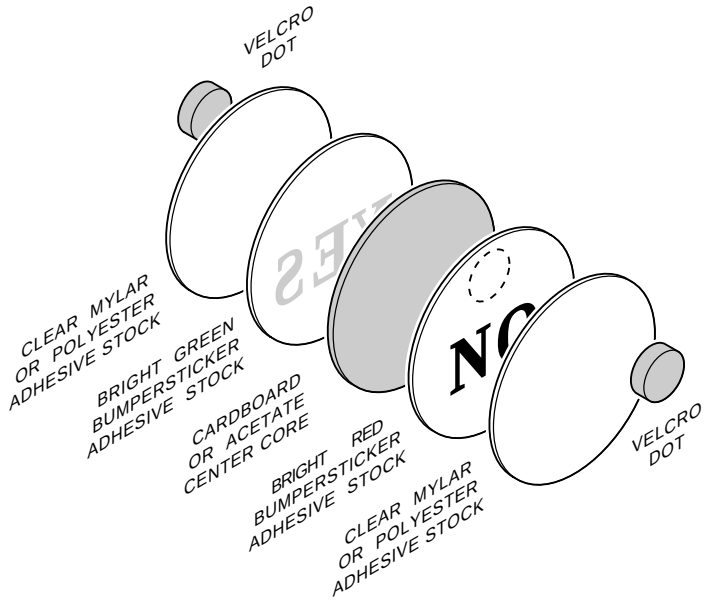


Fig. 1 – How to make a Flippity-Flopper.

Usually, there will be a "red" side with a red message and a "green" side holding a green message. I use one as a warning on my *NTX* not to use a brand new toner cartridge for anything except camera-ready copy. Obvious other places to stick your very own custom Flippity-Floppers are on refrigerators, dishwashers, medicine cabinets, and employee in/out boards.

The standard size of a *M1A1 Rev 7.03* Flippity-Floppers is *2-3/4* inches in diameter, because that's the size of the standard circle cutter that gets provided with that Badge-A-Minit badgemaking kit. Your core could be cardboard, rigid vinyl, or heavy acetate. The red and green sides are fluorescent bumpersticker sheets from Paper Plus. The optional plastic overlay can be self-stick acetate or polyester.

Naturally, you PostScript laser print all the custom text and graphic messages to suit whatever uses you care to. You might want to offer a standard and custom mix.

Let me know what other uses you can come up with for these.

How Do I Really Get Started Using PostScript?

I overwhelmingly prefer to work directly in "raw" PostScript, by using standard *ASCII* textfiles in the non-*WYSIWYG* environment. I do find this lets me explore lots of new PostScript ideas and uses which simply are not apparent in a canned pagemaking or an illustration environment.

And since what you get is what you get with raw PostScript, there's never any rude surprises along the way. Well, hardly ever.

I likewise force my students to work in raw PostScript. Apparently it works, since many of my rank beginning off-the-street students routinely end up doing award winning graphics after a few hours of hands-on time.

Just how can you get started exploring "raw" PostScript?

First and foremost, you might want to pick up Adobe's blue and red books, otherwise known as the PostScript Tutorial and Cookbook and the PostScript Reference Manual II. I'm also laboring under the delusion that my PostScript Beginner Stuff, my PostScript Show and Tell, and my LaserWriter Secrets book and disk packages might be of use.

The details on where you go from here will depend very much upon your choice of host computer. In general, you'll want to set up an interactive and two-way comm environment that lets you receive return error messages and then optionally record any PostScript code returned to your host.

From there, you'll usually want to create your own custom PostScript files by using your favorite word processor, but being very careful to always save your final files in a standard *ASCII* textfile form. Those files then get sent via your two-way comm setup.

NOTE: In the text that follows, *[T]* means *[control]-T*, etc...

For your very first experiment, I'd recommend a file consisting of a few carriage returns, the message showpage one word and all lower case, several more carriage returns, your middle name, several more carriage returns, and either a *[D]* (preferred) or else an ending quit command. Save this as an ordinary ASCII textfile under a filename of showpage.tst.

Note that quit is not nearly as good as *[D]*, since quit only works if you reach it, while *[D]* works, error or not.

This file gives you a quick check on your normal PostScript lashup. It should initially eject a blank page, followed up by the display of an error message on the screen of your host computer. Do not ever attempt any PostScript work unless you first get showpage.tst to work!

From there, begin with the simple exercises in the middle of the blue book, starting with number two. First run each excercise as is, then modify it slightly, and finally go ahead and customize it.

Let's start with the Mac. There are

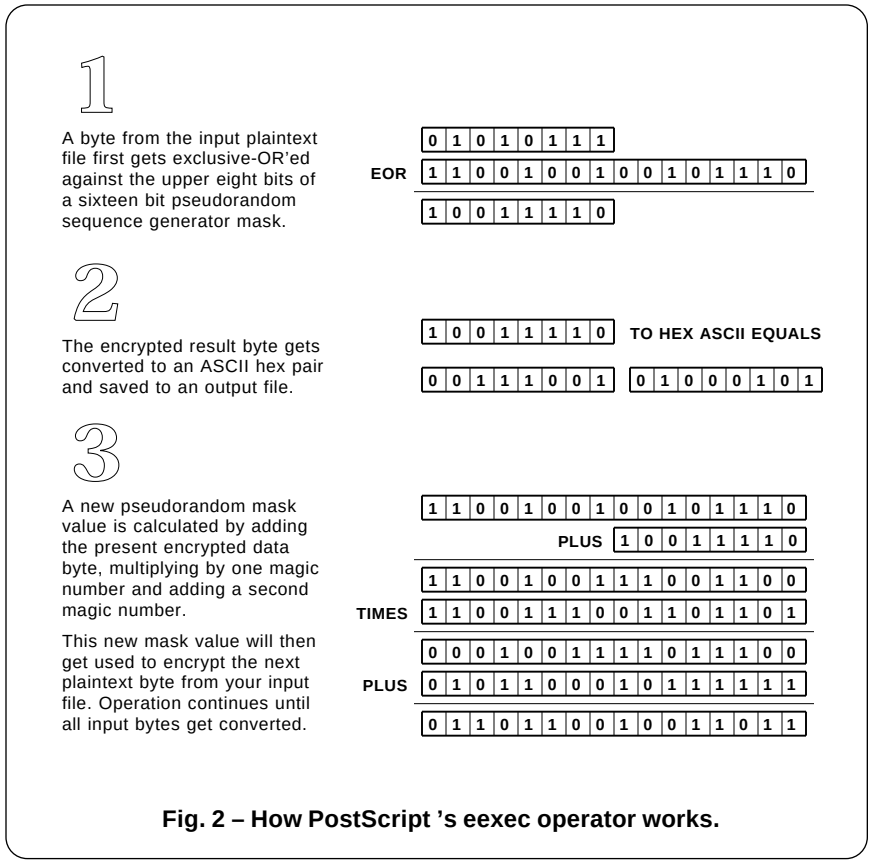


Fig. 2 – How PostScript 's eexec operator works.

two methods of communicating here, either AppleTalk or serial. Note that an ordinary printer cable can replace the AppleTalk cables, so long as you are using a single Mac and both it and your LaserWriter are plugged into the same grounded ac outlet. Note also that AppleTalk is not significantly faster than using ordinary serial comm for most PostScript users.

At any rate, if you want to use AppleTalk, get a copy of the Adobe Font Downloader, or a copy of Send PS. These are available cheaply or free from Adobe, off all of the usual bulletin boards, or from me.

Be certain to use your chooser to connect AppleTalk and then select a LaserWriter or whatever before you begin. To download a PostScript file, double click on the Adobe Font Downloader, check the Status for an idle message, then pick the Download PostScript File box. Finally, you'll double-click on your text file to transmit it. The file should do its thing, and any error messages or returned info should appear on the host screen.

Be sure to remember to use your Save As option inside your word processor to create standard *ASCII* textfiles. Also be certain to enter carriage returns at the end of each paragraph when you are importing a standard textfile back into your word processor.

Should you prefer to run serial instead, just use a two-way comm program such as FreeTerm. Select *9600* baud, *8* data bits, *1* stop bit, no parity and *XON/XOFF* handshaking activated. Just grab your textfiles and transmit them.

Serial comm is a very good choice whenever you are recording longer returned messages.

Some applications packages make it quite difficult to use Mac serial communications. One aid that can ease this problem is that *$14* Asynchronous LaserWriter Driver disk from APDA.

```
/mask 16#D971 def /mult 16#CE6D def /mult1 16#6000 def
/mult2 16#6E6D def /adder 16#58BF def /strrx (X) def
/char 32 def /trunc16 {16#FFFF and} def /hexvalues (0123456789ABCDEF)
def

/printashex {cvi /vall exch def vall 16 div cvi hexvalues exch 1 getinterval print
vall 15 and hexvalues exch 1 getinterval print flush 20 {37 sin pop} repeat
formatcount 1 add 32 eq {(n) print flush 100 {37 sin pop} repeat} if
/formatcount formatcount 1 add cvi 31 and def} def

/makeeexecfile {/formatcount 0 def 4 {char mask -8 bitshift or char mask -8
bitshift and not and /echar exch def echar printashex flush 15 {37 sin pop}
repeat mask echar add dup mult1 mul trunc16 exch mult2 mul trunc16 add
trunc16 adder add trunc16 /mask exch def} repeat {currentfile strrx readstring
{0 get /char exch def char mask -8 bitshift or char mask -8 bitshift and not and
/echar exch def echar printashex flush 15 {37 sin pop} repeat mask echar add
dup mult1 mul trunc16  exch mult2 mul trunc16 add trunc16 adder add
trunc16 /mask exch def}{pop exit} ifelse} loop} def

% //// demo - remove before use. ////

1500 {37 sin pop} repeat

% Here is the expected result ...

% F983EF00C334F148421509DC30FA053D6DD4273E416E6A2EA64F917B5D20E111
% 9F220AF8FC50D545AB51A0D18B6DD7543D27A21CD55887C1C7D51608F6A316EE
% 8891D92A6E0D09D1D039159DA3A0781E1380B1228C

makeeexecfile
0 0 moveto 1000 0 rlineto 0 1000 rlineto -1000 0 rlineto closepath fill
showpage quit
```

Fig. 3 – Creating your own PostScript eexec file.

Another trick that should work is Option-F that will dump your PostScript output to the disk. That disk textfile can then be sent serially.

On the *IBM* and clones, the two usual fatal mistakes people make is trying to use their parallel printer port, or failing to properly set up the two-way *XON/XOFF* serial communications. We've seen details on this last month, but here's a quick review…

Select a comm program such as CrossTalk, picking up *9600* baud, *8* data bits, *1* stop bit, no parity, no output filtering and the *XON/XOFF* handshaking activated. Check your status with a *[T]*. Note that you must be Idle when sending a new file. If necessary, force resets with a *[C] [D] [C]* from your keyboard. Never send a new file unless your LaserWriter is idle or solid green.

On the Apple IIe and IIgs, I quite strongly recommend using AppleWriter and a Super Serial card. To preset your baud rate, use *[O]-J 1 9600,7,N,2* and then activate your *XON/XOFF* comm handshaking by starting all your files with this magic incantation…

```
% [I] X E
% [I] Z
```

Select some "wide open" print constants by setting *LM=0, PM=0, RM=240, TM=0, BM=0, PN=0, PL=66, PI=66, LI=1, SP=0, PD=1, CR=1* no top line, no bottom line, and a *LJ* justify. Next, set up a glossary key that combines a printing with a switch into your modem receive mode. This fake full duplex should give you most of your return error messages most of the time.

When in the [Q]-I modem receive mode, a [T] from your keyboard will inquire your status, while a *[C][D][C]* will force a reset.

I personally use an honest *57600* baud out the game paddle port of my Apple IIe. This is around seven times faster than earlier versions of AppleTalk. The overhead time is zero since the file management is done as part of the interbit delay times. Let me know if you need more on this.

On other host computers, your rules are the same. Create and save your PostScript program code as standard ASCII textfiles by using a suitable editor or word processor. Then send these files over two-way serial comm channels that have an XON/XOFF handshaking properly activated.

Do not ever attempt anything in PostScript on any system that does not give you full two-way comm and on-screen error messages.

Serial communication with your LaserWriter can later run as fast as 57600 baud. You do, of course, have to keep both ends at the same speed. You also have to be certain your de-facto baud rate is somewhere near your selected one. Rather often, overhead and handshaking will dramatically slow down your comm channels far under the selected baud rate.

If at all possible, end your files with a [D] end-of-file marker. Or else use the PostScript *quit* command. But note that *quit* can't reset a previous error, while [D] can.

Some word processors, especially those on a Mac, make it inordinately difficult to embed viewable control characters into your PostScript files. Avoid these programs at all costs. If you must, note that PostScript will accept an escape character as "\033" and so on. A reverse slash followed by an octal value enters that value as printable characters. More details in the red book.

If your word processor tries to force carriage returns every now and then, shorten all text lines as needed with the "\ [return]". An an example, AppleWriter will try to force a return every 240 characters in a long paragraph. Just throw in that "\[return]" every 200 characters or so. Good old WPL can do this automatically for you.

Once you have your two way comm up and reliably working, you probably would want to install a printing error trapper. The big advantage here is that your file gets printed out up to the time the error was made. Again, the error trappers are available free or cheaply from *Adobe*, off many BBS systems, or from me or PSRT.

Under *no* circumstances should a printing error trapper be used as a replacement for two-way comm and on-screen error messages.

If you need any more help getting started with PostScript, just give me a call. I must go through this routine a dozen times a day.

How Does EEXEC Work?

This certainly is a popular and ongoing helpline topic. Seems that bunches of you want to create and use your own files that use PostScript's enigmatic *eexec* operator.

Actually, *eexec* is one failed early attempt at making textfiles slightly harder to read at the triple penalties of longer textfiles, slower comm times, and the red flag waving "I've Got A Secret!" attention-drawing to its own intended use.

We've seen in previous columns (and in the *Ask The Guru II* reprints) how you can "sight read" any *eexec* listing simply by installing any old stack dumping error trapper and then selectively inserting or else removing the characters from the *eexec* data string. From the return error messages, the plaintext file can be readily rebuilt. All the needed tools are shown in the red, blue, green and black books.

But what is *eexec* and how does it really work? Figure two shows you some key details.

The centermost secret of *eexec* is a 16-bit *pseudorandom sequence*. The pseudorandom sequence is a magic string of numbers. Any few digits taken together will largely behave and appear as any "real" random number, yet all of the numbers will end up geting used and will be equally probable by the time your sequence repeats. Each repeat of the sequence is exactly the same as the previous one.

One good method to generate a pseudorandom sequence is to use a long shift register and then take several carefully selected bits and *exclusive-OR* them together to form the next input bit. This generates a long chain of apparently random bits that can be grouped together into pseudorandom words. We've looked at full details on this back in my *Apple Assembly Cookbook* and my *CMOS Cookbook*.

You might also use software to generate the longer pseudorandom sequences. One way to do this is to select your current pseudorandom value, add one magic constant to it, multiply it with a second magic number, and then add yet another magic number to it. To make things more interesting, you can replace your first magic adder value with your current *data* value. This forces the generated sequence to be highly data dependent, and unique for each and every use.

A second secret to *eexec* is the little known *reversibility* property of the logical *Exclusive-OR*, or EOR operator. If you EOR two bytes *x* and *y* together, you will get a new byte *z*, whose individual bits can represent the bit-by-bit EOR of the similar bits positioned in *x* and *y*.

Now for the neat part. If you take the answer and *EOR* it back against either input byte, you will get the the *other* input byte back as a result! For instance, if you EOR *z* against *y*, you'll get *x* back. EOR *z* against *x*, and you should get *y* back. Try it.

```
% This tests the sample file created by the makeeexecfile demo of figure three.
% It should eject one black page if correct…

currentfile eexec
F983EF00C334F148421509DC30FA053D6DD4273E416E6A2EA64F917B5D20E111
9F220AF8FC50D545AB51A0D18B6DD7543D27A21CD55887C1C7D51608F6A316EE
8891D92A6E0D09D1D039159DA3A0781E1380B1228C
```

Fig. 4 – Testing your eexec file of figure three.

Thus, your EOR logic operator is completely reversible. Your very *same* pseudorandom mask value can move you from plaintext to the encrypted form, and vice versa.

As figure two can show us, to encrypt an *eexec* file, you take one byte at a time from your plaintext input. That byte gets EORed against your upper eight bits of a 16-bit pseudorandom mask value. That encrypted byte is then converted to a pair of hex ASCII characters so it can be sent over a non-transparent data channel.

Note that your encrypted file is thus twice as long as your original in this form. It also might take longer to transmit over your comm channel. Note also that each input byte by itself generates another *pair* of hex ASCII output bytes.

After generating the encrypted hex pair, a new pseudorandom mask value gets calculated. This is done by adding your current encryped byte to your current mask value, aligned right as usual. Then a magic number gets multiplied to this result, followed by the addition of a second magic number. The final result is the new mask value in the pseudo-random sequence to get used for your next byte.

Note that the encrypted string is highly data sensitive, because each sequential data byte plays a role in determining the next mask value.

The magic multiplier and adder numbers are easily obtained by inputing a long string of double zeros and then doing an *eexec*. Your "unshifted" sequence generated by the return error message then lets you build up your entire pseudo-random generator. I will leave this as a student exercise.

The first four characters in an *eexec* sequence are usually ignored. For many uses, you can simply throw them away. Presumably, these let you insert a customer or user key into the *eexec* process.

Additional *eexec* insider secrets now appear in the new *black book*, otherwise known as the *Type I Font Format* from *Adobe Systems*.

What are This Month's PostScript Utilities?

I thought we'd have three of them that will let you further explore the PostScript *eexec* operator on your own. Figure three shows you how to encrypt your own *eexec* file, while figure four shows you how to test your *eexec* encrypted file. Finally, figure five shows you how you can undo your *eexec* file and convert it back into plaintext.

If you're wondering just why the algorithm that was used here seems so convoluted, it is because of a property of PostScript that you have to find a way around. PostScript normally allows signed integers whose max length is only 31 bits. Should you get a 32 bit result, those integers are converted to floating point reals and lose accuracy. Thus, if you desire to do accurate integer arithmetic using PostScript, you have to be certain you'll never get more than a 31 bit result.

Unfortunately, if you multiply two 16-bit integers by each other, sometimes you will get a 32 bit result. To beat this, please note that $(x + y)z = xz + yz$.

If $(x + y)$ is a 16-bit number, you can always find an *x* and a y whose max length is only 15 bits. Hint: You make x and y nearly the same size.

So, to keep your calculations as integers, you can first multiply $x * z$, and then truncate to 16 bits with an AND $FFFF. Then you multiply $y * z$, truncate again to 16-bits and then add the two intermediate results.

Hairy, eh what? All this seeming nonsense really does is allow you to multiply your two 16 bit numbers together without ever overflowing above an integer 31 bits.

The short example code I have picked for these files certainly does not need encrypted, but it is very interesting in its own light. This is called a *blackflasher* and ejects a solid black page.

Blackflashing can very much improve the quality of your solid blacks and uniform grays when doing any camera-ready copy or anywhere else you need one or two sheets of first rate output.

It turns out that typical *Canon* drums do have a slight memory or history of the previous images. By blackflashing once or twice, you "erase" any old ghost images and precondition the drum to a solid black. Naturally, this stunt eats toner for lunch and wastes paper, so you'll only want to do it when you really need first rate quality. ❄

```
/mask 16#D971 def /mult 16#CE6D def /mult1 16#6000 def
/mult2 16#6E6D def /adder 16#58BF def /strrx (X) def
/skip4 -4 def /trunc16 {16#FFFF and} def

/readeexecfile {{currentfile strrx readhexstring{0 get /echar exch def echar
mask -8 bitshift or echar mask -8 bitshift and not and /char exch def skip4
0 ge {strrx 0 char put strrx print flush 15 {37 sin pop} repeat /skip4 skip4 1
add def}{/skip4 skip4 1 add def} ifelse mask echar add dup mult1 mul
trunc16  exch mult2 mul trunc16 add trunc16 adder add trunc16 /mask
exch def}{pop exit} ifelse} loop} def

% //// demo - remove before use. ////

1500 {37 sin pop} repeat

% Here is the expected host-returned result for this demo . . .

% 0 0 moveto 1000 0 rlineto 0 1000 rlineto -1000 0
% rlineto closepath fill showpage quit

readeexecfile
F983EF00C334F148421509DC30FA053D6DD4273E416E6A2EA64F917B5D20E111
9F220AF8FC50D545AB51A0D18B6DD7543D27A21CD55887C1C7D51608F6A316EE
8891D92A6E0D09D1D039159DA3A0781E1380B1228C
```

Fig. 5 – Reading your eexec file of figure three.

Don Lancaster's

ASK THE GURU

April, 1990

Adobe's newest Black Book
The Pelsaer Binding Process
Type I Charstring Interpreter
Use of PostScript Action Tables
Finding An Effective Baud Rate

Adobe has released their new *black book*, and otherwise known as their *Type I Font Format*. It is everything they promised and bunches more. Yes, full details on *eexec*, *BlueValues*, all the hinting mechanisms, *flexproc*, and *Charstring* interpretion. Even the holiest of the most sacred of Adobe mantras (1183615869 internaldict begin) has been revealed.

You can get the black book for $14.95 directly from *Adobe*. I also do stock these as a service for all of you *Computer Shopper* readers.

One surprise: While Type I fonts are extremely well suited for the fast and compact storage of high quality Greek and Roman alphabets, they are pretty much useless for logos, alphabets of symbols, or anywhere else you want to use a rich variety of PostScript commands.

They definitely cannot be used for all the proc cacheing we looked at a column or two back. The Type I characters are pretty much limited to integer numeric defined curves and fixed straight lines to be stroked or filled in their entirety, and that's about all.

There are all sorts of non-obvious new opportunities for "open" Type I font architecture. I have listed a few unique possibilities in figure one to get you started.

Apple has at long last extended their warranty to a full year. This welcome move is long overdue.

The new *Hewlett-Packard* LaserJet III with its genuine Adobe PostScript cartridge does appear to be running away with all of the marbles. The III completely blows the *LaserWriter NT* away for entry level users. Tricks are pulled to eliminate many 300 DPI jaggies and to give the illusion of higher resolution. It still uses the older 8 PPM *Canon* SX engine.

No SCSI hard disk and no duplex option to date, and its cartridge PostScript remains rather slow. I will not be getting one just yet. And still no %$*@# carrying handles or outside toner adjustment.

After running nearly one million pages through a pile of *LaserWriter* printers, I have now pretty much concluded that the newer SX engine is simply not rugged enough for continuous and sustained use on business cards, book covers, heavy parchments, or index stock mailers. I've returned to doing the heavies on my older CX machines.

A reminder that there should be a major new PostScript board up on *Genie* whose main goal is to have one thousand downloads as soon as possible. Use PSRT or M835.

And I now have a new *PostScript insider secrets* brochure waiting for you when you call or write.

Onward and upward...

How About Yet Another Binding Systems Update?

I've been having lots of trouble lately with all those *Unibind* transparent covers cracking up when I attempt to shear them. Even with a clamped and ultra-sharp blade.

I think it is mostly caused by stale product combined with problems unique to my dry desert Southwest. Preconditioning the covers at a 100 percent relative humidity (use a hot tub) and padding the covers with scrap pages during shearing seems to help some. Make sure your dealer will guarantee you fresh stock.

On the other hand, *Unibind* does now produce an unbelievably great new product. This is called a *Pelsaer Binding*. It appears in figure two.

A Pelsaer binding is simply one preformed hot glue channel that has two flyleaf sheets attached to it via a quick release adhesive. You drop your text inside, and then can wrap your own custom cover round the outside. Pop it into your Unibind toaster, whomp it onto the cooling plate, and then optionally remove the flyleaves.

You could now use nearly any cover material or process, and can

1. A "shorty font" of only two or three characters can be carried along inside of a logo or other art cut, eliminating the need for any fancy fonts on the end-user's printer.
2. Seldom-used characters can be deleted off a stock font, releasing vm and making more room in older and smaller PostScript printers.
3. Characters can be intercepted to rapidly handle all of those non-linear transformations needed for perspective and "star wars" lettering.
4. Letters can now be paired for unusual foreign symbols or for such things as a logical compliment notation.
5. Since *eexec* isn't really needed, a true 8-bit comm setup can cut font file lengths and comm times in half by eliminating hex-ASCII pairs.
6. Hard-to-enter high ASCII characters can be redefined as low-ASCII keystrokes. Especially the emdash and bullet.
7. A new dictionary can be opened after *eexec* to allow redefinition and alteration any way you like.
8. Actual font paths can get diverted for true outlines, 3-D, chokes, spreads, multi-stroking, and other special effects.

Fig. 1 – Some new opportunities for "open" Type I fonts.

even custom letter the spine.

One unique trick is to print your cover with legal sized cover stock. Allowing for 1/8 inch of trim all the way around on a 1/2 inch thick book, you can handle a text that is as large as 6-5/8 inch wide by 8-1/4 inches high. You can letter the spine and make use of any and all cover enhancements, such as Bakerizing, Kroy Kolor, varnish, u-v overcoat, or lamination.

The cost for the Pelsaer covers is around fifty cents. This can be your choice of (A) less than any other perfect binding system, or (B) an obscenely overpriced use of less than a penny in raw materials.

Contrary to popular belief, a cold glue bookbinding is much better than using hot glue. Cold glues wet the papers better, are stronger and more flexible, longer lasting, let the book open flatter, and are even less tasty to roaches. About the only real limitations are that cold glue used to take longer to set up and had a limited pot life.

Those importers at *Planax North America* have come up with an interesting variation on cold glue. When pressure is applied, water is driven out of their glue, causing a rapid setup. It is possible to cycle cold glue perfect binding in under a minute on their machines. Sadly, their binding machine prices are unreal. You can buy the magic glop from them separately, though.

A second source of cold glues and related supplies is *Gane Bros.* Prices start around $16 per quart.

What is Meant by The "Click to Clunk" time?

The *click to clunk time* forms your ultimate measure of the speed of a PostScript printer setup, especially for Book-on-demand printing. The more you know about exactly what goes into your click-to-clunk times, the faster and better your results.

More often than not, several very simple hardware, software, or even attitude mods can very much reduce your PostScript print times.

The *click* is the carriage return or mouse action from your host that starts printing. The *clunk* is the start of the paper feeding for your page image. I routinely have my click-to-clunk page makeup times down in the zero to four seconds range when Book-on-demand printing a gonzo three column, multi-figure pages. You can do the same.

What normally happens during a printing? First, chosen PostScript code gets sent over a serial, parallel, SCSI or Appletalk channel. Very often, the initial code sent will be a *prolog*, a dictionary, or some set of definitions that must be in place before any useful output starts.

Eventually, enough code reaches your printer such that it can begin placing actual marks on its bitmap page. At that point, a race starts. The PostScript interpreter tries to empty its print buffer at a speed set by its hardware speed and by your programming style. The comm channel tries to fill the print buffer at a speed set by the comm rate and the number of characters in the print file. Obviously, your page cannot be completed until the entire file is sent and your print buffer is completely emptied.

Now, should your comm buffer be mostly full, you will be *PostScript intrerpreter limited*, and you could very significantly speed things up by a pseudocompiling or simply by writing more efficient code. Should your comm buffer remain mostly empty, you are *Baud Rate limited*, and you could speed things up by shortening your input files, and perhaps going to more computation intensive code.

One very gross test to find out if you are now baud rate limited is to notice whether your host recovers before the clunk. If you are baud rate limited, your host computer remains tied up until the clunk. If you are PostScript limited, the host *may* (depending on the comm setup) go on back to letting you do other host-based tasks.

Note that your PostScript interpreter is twiddling its thumbs all the time you are baud rate limited. And similarly, your serial comm channel gets put on hold and does nothing useful while you are PostScript interpreter limited. Either way can waste time and money. Ideally, you will normally want to end up just barely baud rate limited. By only a millisecond or so.

Regardless of how you communicate, you can define an *effective baud rate*. The effective baud rate is the measure of just how fast the bits go through the pipe that includes all of the overhead and *Hi – How are the wife and kids?* handshaking.

Which lets you directly compare

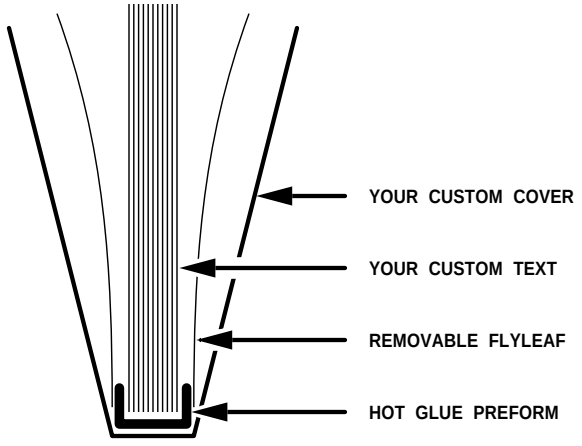


Fig. 2 – Unibind's new Pelsaer binding process.

parallel, serial, SCSI or *AppleTalk* rates. Usually your effective baud rate will end up much lower than you'd suspect. Very often, the *faster* CPU's will end up giving you much *slower* effective baud rates.

Figure three shows you a simple PostScript routine that will measure your effective baud rate for you. This rate measurement also could depend on your file sizes, since SCSI and AppleTalk are sometimes faster on longer files.

My full page, two figure, three column gonzo files typically range from 8K (uncompiled) to 14K (when pseudocompiled), so I've selected 10K as a meaningful file length.

All the program does is send out 10,000 characters of comments and then reports the total time it took to do so back to your host.

You might find the effective baud rate measurement results of figure four to be somewhat astonishing. As expected, the SCSI drive on the NTX is far and away your fastest, checking in at nearly 400 kilobaud. This would be even faster on longer files or combined jobs.

In second place is an old Apple IIe with a hand-crafted honest 57600 baud rate routed out the game paddle port. The effective and the actual baud rates are identical here, since that character-getting procedure is now included within the interbit time delays of the custom machine language coding. We have seen full details on this code in previous columns and in my *Ask The Guru* reprints.

The Macintosh II is in third place, and is about one-half the speed of using an old Apple IIe. AppleTalk communicates much slower than you might expect, because of its extensive handshaking and the "onion effect" of layer upon layer of controlling code.

The Mac SE clocks in at 17400 using AppleTalk, or slower than an honest 19200 baud would be. Again, this is caused by the insane AppleTalk software overhead.

The PC clones will typically top out around 10000 baud, no matter how fast you do set your requested comm rate. You should be able to beat this through custom bare metal machine language code. It should be at least theoretically possible to make a 386 communicate almost as fast as an Apple IIe does.

So what can you do to shorten your click-to-clunk time after you know your effective baud rate? If you are baud rate limited, either speed up your effective baud rate or shorten your files. As long as the baud rate limit remains, you can let PostScript do more work that takes longer and still end up ahead so long as it shortens your files.

Naturally, all of those usual file shortening stunts such as using persistent downloads and eliminating the comments can be used.

If you instead end up PostScript interpreter limited, then either clean up your code or else pseudocompile it. It also may pay to arrange your files so that PostScript has something important and time intensive to do right away, instead of waiting for a complete prolog.

There's lots of ways to increase your effective baud rate. On an IBM or clone, try using several comm programs to find the one whose effective baud rate is closest to its nameplate baud rate. Do not even think of running slower than 19200 baud. *CrossTalk* doesn't turn out all that bad, but even it could definitely be improved. Or write your own custom 57600 baud driver.

Simply trading your klunky old Mac II up to a sleek new Apple IIe will double your effective baud rate and could cut your PostScript print times in half.

But more to the point, replacing AppleTalk on a Mac SE with some direct SCSI comm scheme could raise all your effective baud rates, perhaps by as much as 30:1. Apple definitely does not want you to do this, since it makes AppleTalk look so abysmally putrid.

One way to handle fast comm is to host create what appears like a fake hard disk on the SCSI port on your NTX. Even writing each file to disk and then reading it back immediately after would often be 15 times faster than AppleTalk.

If you are any kind of a SCSI guru, let us talk some more on this. Apple appears to be purposely slowing their PostScript laser printers down by as much as 30:1 at present.

What is the Distillery?

As we have seen in the past, it is a fairly simple matter to do a *pseudocompile* on your PostScript code to speed it up for a later reuse. What pseudocompiling does is intercept all those PostScript operators, and then returns a file of *only essential final results* either to the hard disk or back to your host for recording.

The result is a somewhat longer file that most often should run significantly faster. In fact, for typical

```
%  This test program measures your effective PostScript baud rate and
%  returns it to the host for display. For most users most of the time,
%  the effective baud rate is far lower than you think it is.

/numchars 10000 def  usertime /strtt exch def

% Empty comment line that's fifty characters long.
% Empty comment line that's fifty characters long.

      (provide precisely 200 comment lines)

% Empty comment line that's fifty characters long.
% Empty comment line that's fifty characters long.

usertime strtt sub numchars 10000 mul exch div cvi
1000 {37 sin pop} repeat

(Effective baud rate: ) print (XXXXXXX) cvs print flush
1000 {37 sin pop} repeat (nnn) print flush
```

Fig. 3 – How to find your effective baud rate.

Book-on-demand pages, the final pseudocompiled files will often print at your full printer page feeding speed, *including all the page makeup* on a page after page basis.

But, before we go on, do note one thing: *Pseudocompiling won't work at all if you are baud rate limited!* In fact, it can make things even worse, since you may now have longer files. It is extremely important to optimize your effective baud rate before you do a pseudocompile.

At any rate, *Adobe Systems* has introduced a great new package called the *Distillery* and available for a nominal charge as part of their *Adobe Developer's Disk* or off our great *GEnie* PSRT bulletin board. This code is an amazingly versatile package which attempts to handle just about any asled fpr PostScript programming style. And more often than not, it returns a file that runs much faster and is only somewhat longer.

Besides all the obvious uses of showing problem areas in your programming and speeding up your run times, the Distillery gets quite handy any time you want to release a new demo disk without providing your source code. Simply run your code through the Distillery, and all of your fast run demos should not reveal in any way the code that originally created them.

There are some problems with the Distillery, but it does do all its work amazingly well. One limit is that a path is redefined every time it is used. I often like to triple stroke a line or a border, first to break the background, second to create the black outline, and finally the white or a grey fill. The Distillery will redefine the same path three times in a row, rather than using a *gsave-grestore* construct.

Also, the returned "print-a-string" files do have more characters in them than is really necessary, which causes problems if you're anywhere near an effective baud rate limit. The Distillery clipping procs simply do not work properly at the present time. And there are some other bugs in the present release.

Note that a manual intervention after using your Distillery can minimize many of these problems.

How does the Distillery compare against my *Gonzo Compile* that we looked at in previous columns? The Distillery is far more flexible, more adaptable, and considerably more powerful. But the Gonzo Compile generates shorter and more efficient *awidthshow* files that rerun much faster. Especially since it sorts out all of those string messages such that no font ever need be changed more than once per page.

Since both programs are open and unlocked, the chances are you might want to take the best of both worlds and combine them into one custom solution.

SCSI Hard Disk

HD-20 file read to the LaserWriter NTX
Effective Baud Rate: **398700** Effective Character Rate: **39870**

APPLE IIe

AppleWriter with custom driver out the game paddle port.
Effective Baud Rate: **57600** Effective Character Rate: **5760**

MACINTOSH II

Adobe Font Downloader via AppleTalk to NTX.
Effective Baud Rate: **32038** Effective Character Rate: **3204**

MACINTOSH SE

Adobe Font Downloader via AppleTalk to NTX.
Effective Baud Rate: **17496** Effective Character Rate: **1747**

APPLE IIgs

AppleWriter via Super Serial Card at 19200 baud.
Effective Baud Rate: **15827** Effective Character Rate: **1583**

IBM PS2 Model 50

Crosstalk at 19200 baud.
Effective Baud Rate: **10760** Effective Character Rate: **1076**

Fig. 4 – A few effective baud rate results.

What is This Month's PostScript Utility?

Let's look at an ultrafast and powerful old PostScript programming technique, that also can get you started on interpreting the *Charstrings* in your Type I fonts.

Most any computer language needs a way to jump six ways from Sunday at some point in a program. In machine language, we might call this an *option picker*. Detailed examples of option picking appear in my *Apple Assembly Cookbook*. In Pascal, there is a *case* command where, depending on a number, a chosen subroutine gets run.

In any language, you might use *ifelse* statements. But nestle these more than six deep and they end up certain to be wrong. Hoffschnagle's rule and all. You also can end up with monster spagetti code, variable execution time, and hard changes.

Instead, PostScript offers you a different and an elegant approach to jumping six ways from Sunday that I prefer to call an *action table*. Action tables are very fast and flexible.

An action table is nothing but an array of procs. You can create your action table array just once at the beginning of your code and then later select any proc for execution.

We might start out with a simple example. Let's first define a three entry action table…

/tbl [{proc0}{proc1}{proc2}] def

Elsewhere, you will define *proc0* as task #0, *proc1* as task #1, and *proc2* as task #2. Later on in your program, say you have a verified number on your stack from 0 to 2. Just do a

tbl exch get exec

and your newly selected procedure will cleanly, quickly, and automatically be activated and run.

There is no limit to the size of an action table, and one table can call itself or another of a different size. And your program structure does not change as you change entries.

One useful action table size is 256 entries. For instance, in my gonzo justify routines, an action table lets you substitute or redefine any of the *ASCII* characters, while you isolate such things as formfeeds, escape commands, carriage returns, newline commands, and the reverse slashes for any special treatment. Think about this for a bit, and you'll realize this gives you a universal emulator that can do darn near anything. With zero decisions or nesting being involved.

Let's look at a fancier example of a 256 entry action table that sometimes might call a second small action table. Once you do get well into the Adobe black book on Type I fonts, you'll be needing a means to interpret a binary *Charstring* back into its open-format commands. Figure five shows you an action table based program for this.

```
% Copyright c 1990 by Don Lancaster and Synergetics, 3860 West First Street, Box 809,
% Thatcher AZ 85552. (602) 428-4073. All rights reserved. Personal use is permitted so
% long as this header remains intact. Show and Tell disks for Apple, Mac, or IBM cost $39.50.

% Replaces a decrypted and clipped PostScript binary charstring on stack top by an array
% of plaintext commands also on top of stack. Limited to 500 commands per string.

% build charstring action table
256 array /cstringtable exch def 0 1 255 {cstringtable exch {notdefx} put} for 32 1 246 {cstringtable
exch dup 139 sub put } for cstringtable dup 01 {/hstem} put dup 03 {/vstem} put dup 04 {/vmoveto}
put dup 05 {/rlineto} put dup 06 {/hlineto} put dup 07 {/vlineto} put dup 08 {/rrcurveto} put dup 09
{/closepath} put dup 10 {/callsubr} put dup 11 {/return} put dup 12 {doescape} put dup 13 {/hsbw} put
dup 14 {/endchar} put dup 21 {/rmoveto} put dup 22 {/hmoveto} put dup 30 {/vhcurveto} put dup 31
{/hvcurveto} put dup 247 {0 longnum+} put dup 248 {1 longnum+} put dup 249 {2 longnum+} put dup
250 {3 longnum+} put dup 251 {0 longnum-} put dup 252 {1 longnum-} put dup 253 {2 longnum-} put
dup 254 {3 longnum-} put dup 255 {getlonginteger} put pop

% build an escape action table
34 array /csesctable exch def 0 1 33 {csesctable exch {notdefesc} put} for csesctable dup 0
{/dotsection} put dup 1 {/vstem3} put dup 2 {/hstem3} put dup 6 {/seac} put dup 7 {/sbw} put dup 12
{/div} put dup 16 {/callothersubr} put dup 17 {/pop} put dup 33 {/setcurrentpoint} put pop

% These are the actual routines that use the action tables…
/nextcspoint {cspoint csstring length 1 sub lt {/cspoint cspoint 1 add def}{exit}ifelse} def

/charstringtocommands {/csstring exch def /cspoint 0 def mark {cstringtable csstring cspoint get get
cvx exec cvx nextcspoint} loop] } def

/longnum+ {256 mul nextcspoint csstring cspoint get add 108 add} def

/longnum- {256 mul nextcspoint csstring cspoint get add neg 108 sub} def

/getlonginteger {nextcspoint csstring cspoint get 0 ne nextcspoint csstring cspoint get 8 bitshift
nextcspoint csstring cspoint get add 8 bitshift nextcspoint csstring cspoint get add exch {dup
16#FFFFFF or exch 16#FFFFFF and not and 1 add neg} if } def

/doescape {/escape cvx nextcspoint csstring cspoint get csesctable exch get exec cvx} def

% //// demo - remove before use

1000 {37 sin pop} repeat % optional host delay

% Adobe's black book page 59 "C" example - faked into binary string
/fakestring [189 249 180 13 139 239 3 139 239 1 248 236 239 1 139 22 249 80 6 239 7 252 236 6
248 136 7 248 236 6 239 7 253 80 6 9 14] def fakestring length string /tempstring exch def 0 1
fakestring length 1 sub {/ptr exch def fakestring ptr get tempstring exch ptr exch put} for

tempstring charstringtocommands % this does it

/magicstringarray exch def magicstringarray {cvlit (XXXXXXXXXXXXXX) cvs ( ) print flush 40 {37 sin
pop} repeat print flush} forall

magicstringarray 1000 {37 sin pop} repeat ( n n n) print flush quit
```

Fig. 5 – Type I Charstring interpreter using action tables.

After decryption and removal of four leading characters, a *Charstring* consists of a binary coded string, each "character" of which assumes one of 256 values. Those first few values represent Type I procs such as *hstem*, *vlineto*, or *callsubr*. The decimal 12 value is an *escape* that lets you go to the second table for the lesser used commands such as *dotsection* or *seac*.

Characters 32 through 246 should represent the integers from -107 up to +107 on a substitution basis. Note that this gives you as much as 5:1 compaction since a single binary 32 can represent those five characters needed for the ASCII -107 (don't forget the space!). Because of this compaction, Type I fonts often end up much shorter than most others.

The characters 247 through 254 should represent values from -108 to -1131 or 108 to 1131. These use the next byte in the *Charstring* to complete the value. Thus, the lesser-used numbers are less compacted.

Finally, character 255 goes ahead and gets a full 32 bit binary signed integer using the next four string characters. This one is rare.

The action table can dramatically speed up and simplify the code. You simply grab one binary character at a time out of your Charstring. If it is a proc, you post it. If it is an escaped proc, you find the proc and then post it. If it is a small number, you will substitute it. If a larger number, you calculate it. I have shown the result as an array that can be made executable or else printed to a hard disk or back to your host.

Since it is tricky to send a binary string over a typical comm channel, I have built up this example string using that "C" code example in the black book. Then we have both saved it to disk and reported it back to your host. You also have the option of switching between binary and hex ASCII pairs; I've left this extra hassle off for now.

I will have much more on Type I fonts in future columns. For now, I just wanted to show you just how an action table works and give you some bare bones code to get you started. Do read the black book. ❄

Don Lancaster's

ASK THE GURU

May, 1990

PostScript font path grabber
Rubber stamp opportunities
Perspective lettering routines
A hacker interchange standard
Non-linear transformation ideas

Your no-charge *Ask the Guru* helpline now fields between 40 and 120 calls per day. Of these, around 60 percent are Mac people, and 35 percent are whatever those IBM users are calling themselves these days. The remaining 5 percent of the calls are often in the "Boy, a whole flock of them flew over that time!" category.

Which is why *Ask the Guru* is in the Mac section. Another reason is that Apple's LaserWriter is wrongly perceived to be a Mac peripheral. As we have recently seen, all of the PostScript-speaking LaserWriters are totally device independent, and work beautifully with Apple, Atari, VAX, Commodore, clone, or IBM hosts. Yes, it is even theoretically possible to use a LaserWriter with a Mac, although you often may find cheaper and faster printing hosts elsewhere.

Apple has released their third CD-ROM title, called *A Disk Named Wanda*. Among other goodies are complete Mac operating systems for just about every major European language. The 400 Megabyte storage capacity of this single disk is somewhere around five to ten percent of that of the human brain. A scary figure if ever there was one.

Needless to say, CD ROM is going like gangbusters. Hundreds of titles are now available. Expect virtually all future Apple software and tech information to use this CD format nearly exclusively. As a reminder, Apple does have a special one-only price on their drives if you are a developer, dealer, or a user group. Apple also will line you up with mastering services for all your own custom CD ROM platters at $1500 for fifty copies. Including cases and cover art.

Separately, Apple has a new and handy *Developer Handbook* out that lists nearly all of their major people, programs and services. Including all names and addresses.

I recently presented one of my PostScript opportunity seminars for the people at *QMS* and their *Laser Connection* marketing arm. Outside of being a tad too far to commute in one day, I was very impressed with these folks and their offerings.

They do seem strongly end-user oriented and now have outstanding customer relations and PostScript help departments. Their offerings cover the gamut from PostScript printers roughly comparable to the LaserWriter NT and NTX on up to really durable and sophisticated high-end industry stuff, both in black and white and in incredibly good (but costly) full color. And they have even better goodies about to emerge from their skunk works. And all speaking genuine Adobe PostScript, of course.

QMS does have a thick and free new *Infobook* for you that details all of the low end products, especially the PS-810 through the PS-820 turbo PostScript printers. I soon hope to run several long-term performance comparisons of their turbo PS-820 against the NTX. My prediction is that the 820 should blow the NTX away, especially when it comes to a fast SCSI host comm. Stay tuned.

As we have seen in past columns, there is a new *Recharger* toner cartridge refilling newsletter. There's also a new trade group known as the *International Computer Products Remanufacturing Association*, or the

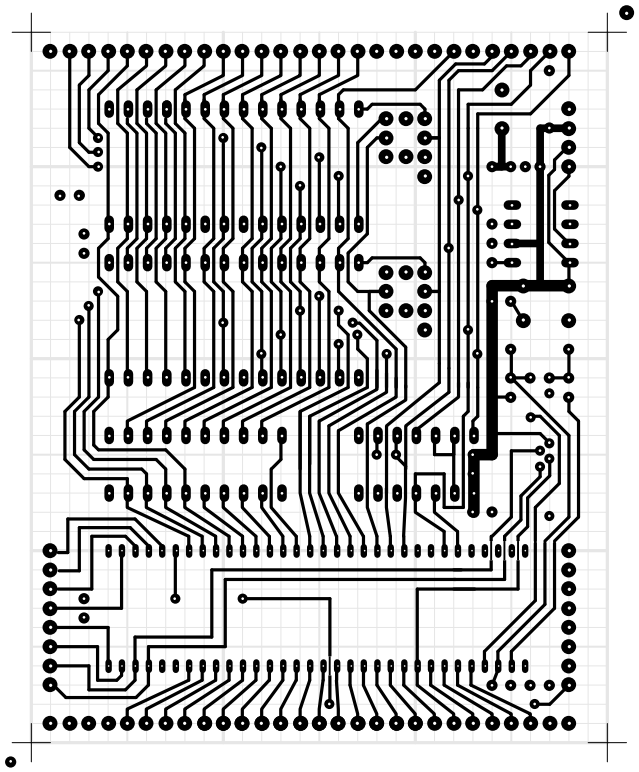


Fig. 1 – A typical hacker printed circuit download.

ICPRA for short. Publishing lead times being what they are, you just missed their first major Chicago conference on drum recoating.

I have a *Lazer Products* recoated SX drum here that is now on its *fifteenth* reload. The remanufactured and recoated drums now seem to have arrived with a vengeance.

It certainly was very thoughtful and considerate of *Canon* to single handedly create this great multi-megabuck industry which now employs thousands of independents nationwide. And, thanks to them, the 0.3 cents per page toner costs are now a reality. Again, without *Canon* and all their pricing and reloading policies, it never would have happened at all.

Thanks loads, guys.

Needless to say, I am profoundly and deeply disturbed over the impending demise of the Apple IIe and the IIgs. Which is sort of like watching an old and dear friend die of AIDS. Or perhaps it was the far more insidious and infinitely more infectious APW that did it. The one thought that keeps going through my head – How could they have been so monumentally stupid?

At any rate, fire sales are being held around the country to flush remaining IIe and IIgs inventories. Some of these are great bargains.

For instance, *A2 Central* is now flushing the entire *Addison-Wesley* IIgs library at less than 1/3 the usual price. An outright steal. The *Tech Alliance* folks (who used to be *Call A.P.P.L.E.)* are also having a major clearance of both their Apple and Mac software and books. Again at greatly reduced prices.

And some back issues and software stock still remains at *Apple Assembly Line*, especially several of the best low cost cross-assembly modules available anywhere.

I guess I put an awful lot of time and effort into the IIe and IIgs. The IIe was (and definitely remains) my kind of computer. And I really felt I tried to share most of it with you. But, I'd also guess someday in the dark, distant future I will have to actually start seriously using a Mac. Maybe I'll begin doing so just as soon as somebody offers something almost as good as AppleWriter. Today, of course, nothing comes even remotely close.

In keeping with these sad times, I decided to offer a "four-for-three" deal with my *Enhance I*, *Enhance II*, *Apple Assembly Cookbook* and *AppleWriter Cookbook* classics.

As another reminder, I am also now stocking Adobe's black *Type I Font Format* book and have recently expanded my *LaserWriter Secrets* book and disk combo.

On to a currently hot topic…

How Can I Solve Those Hacker Interchange Hassles?

The editors at *Computer Shopper*, *Popular Electronics*, *Probe*, *Midnight Engineering*, *Nuts and Volts*, *Audio Amateur, Byte*, *Radio-Electronics,* and *Circuit Cellar Ink* have all been separately grappling with a common problem: How can accurate printed circuit layouts get cheaply and easily put in the end user's hands?

Or dialplates, drilling templates, detailed test and debug info, software listings, or most any other tech writing or tech illustrations?

Putting any of these in the magazine takes up valuable space, and the third-generation photocopies or litho negatives you sometimes end up with are often of low accuracy and quality. And it sure gets hard to make any correction three months after the fact on 360,000 already printed pages, some of which are buried in Mule Hoof, Montana.

On the other hand, the BBS downloading will only work if the sender and sendee both have fully compatible hardware and software.

I'd like to suggest an obvious solution. It is called an *EPS* file, and is short for *Encapsulated PostScript*.

An EPS file can easily handle any printed circuit layout, a schematic, isometric or a perspective drawing, any dialplate, or text or graphics *to arbitrarily high resolution*.

The EPS files are extremely device independent. Since an *EPS* file is an

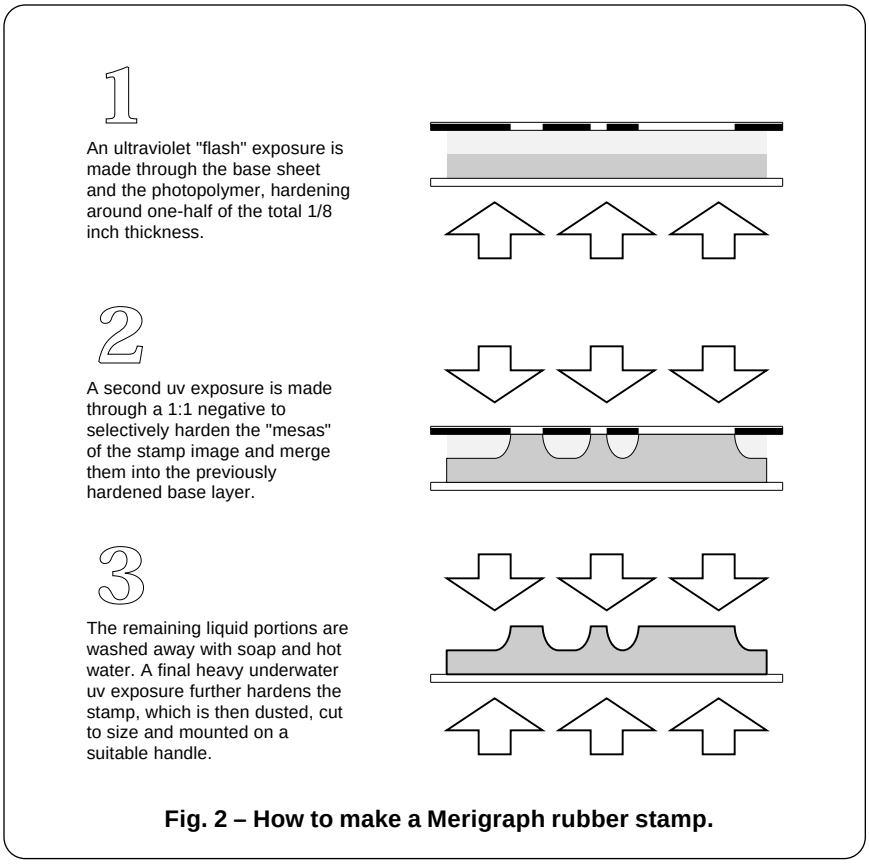


Fig. 2 – How to make a Merigraph rubber stamp.

ordinary ASCII textfile, it can easily be used with *any* word processor or *any* editor using *any* personal computer of *any* age and CPU.

At the output end, an EPS file can be used directly on a laser printer, a typesetter, or with a photoplotter. With simple and crude emulators as *Freedom of the Press*, *GoScript*, and *UltraScript*, you could also print your results on ordinary dot matrix or ink jet printers, albeit to a much lower resolution.

Better yet, practically all of those CAD/CAM and illustration packages either now use PostScript internally or at least are capable of generating EPS output files.

Figure one shows you a typical high quality printed circuit layout that is ideal for instant distribution via an EPS textfile to any editor or any word processor on any personal computer anywhere in the world.

Your EPS files can also be very compact. For instance, a fancy circuit schematic should need no more than a 10K textfile maximum. So, EPS stores cheap and loads fast.

But what if you are using an expensive or proprietary CAD/CAM layout package? Surely you could not provide a free copy of these to every reader in the country, could you? Especially if you are on a Mac and they are on a PC.

```
%  Pathgrab code (Version 47 or earlier, 68XXX Only)
%  written by LDO, 1990.  This line must remain on any reuse.

userdict begin /e {cexec}def /$cexec {<11a2b000 40> eexec} def

<6EA0339904E8000104020010000000002F172F6F000800044EFA0414000000000205F43FA
 FFF8229F4ED0207AFFF0D0FC00A020504E902040226F000422D822D84E7520572EAF0004
 2F6F000800042F480008207AFFC8D0FC005020504E90508F4E75207AFFB8D0FC00802050
 4ED020572EAF00042F480004207AFFA2D0FC00F820504E90588F4E7520572EAF00042F6F
 000800042F480008207AFF82D0FC011C20504E90508F4E7520572EAF00042F480004207A
 FF68D0FC00B420504E90588F2040226F000422D822D84E75207AFF4ED0FC009C20504E90
 2040226F000422D822D84E7520572EAF00042F6F000800042F6F000C00082F6F0010000C
 2F480010207AFF1AD0FC007420504E90DEFC00102040226F000422D822D84E752F6F0004
 000C201F225F205F2F0070001018600212D851C8FFFC42194E75221F201F225F205F2F01
 600212D851C8FFFC484051C800046004484060EE4E754E56FFFC48E7010849FA03582854
 2E140687000000432D47FFFC286EFFFC42144CDF10804E5E4E754E56FEE848E70108486E
 FEE8598F487A008C286E0008486CFF004EBAFF7E4EBAFF06588F2D6EFEE8FFF82D6EFEEC
 FFFC486EFEE82F2EFFFC2F2EFFF8286E00082F2CFEFC2F2CFEF84EBAFF18588F2D6EFEE8
 FFF02D6EFEECFFF43E2EFFF21D47FEF02F2EFFF4486EFEF142471E2EFEF048C72F074EBA
 FF42486EFEF0487A00184EBA02BC57C744071D47000C4CDF10804E5E2E9F4E750432332E
 300007766572736 96E4E56FFFA48E70118286E0008286C00082E1406870000011C2847
 2D54FFFC3D7C0180FFFA55AEFFFC286EFFFC0C944EF10E006600001859AEFFFC286E0008
 197C0001FEF060000044600000222E2EFFFC598728470C94E780D0B966000010286E0008
 197C0001FEF060000020536EFFFA4A6EFFFA6600000E286E0008422CFEF0600000081E3C
 0000679A286E00084A2CFEF067000010286E0008266EFFFC26532953FEF44CDF18804E5E
 2E9F4E754E56FFDE48E70118486EFFDE598F487A00DE286E0008486CFF004EBAFE2C4EBA
 FDB4588F2D6EFFDEFFF82D6EFFE2FFFC486EFFDE2F2EFFFC2F2EFFF8286E00082F2CFEFC
 2F2CFEF84EBAFDC6588F2D6EFFDEFFF02D6EFFE2FFF43E2EFFF2CFECFEF248C7DEACFEF4
 28472D54FFEC3D7C007FFFE6286EFFEC0C542079660000262E2EFFEC5C8728470C944A28
 00436600001454AEFFEC286E0008197C0001FEF060000002454AEFFEC536EFFE64A6EFFE6
 6600000E286E0008422CFEF0600000081E3C000067AE286E00084A2CFEF06700001449FA
 01142D4CFFE8286EFFE8266EFFEC28934CDF18804E5E2E9F4E750A70617468666F72616C
 6C004E56FFF048E70108486EFFF04EBAFC4E588F2D6EFFF0FFF82D6EFFF4FFFC2F2EFFFC
 2F2EFFF84EBAFC4C487AFD60598F487A0020286E0008486CFF004EBAFD104EBAFC784EBA
 FC4E4CDF10804E5E2E9F4E750D756E70726F74656374706174684E56FEE848E70108286E
 00082F144EBAFBE21D7C0001FEF0486EFEE84EBAFC80588F2D6EFEE8FEF82D6EFEECFEFC
 558F2F0E4EBAFD201E1F670000143D7C0004FEF22D7C00497CCCFEF46000000E3D7C0008
 FEF22F0E4EBAFDA04A2EFEF0670000082F0E4EBAFE404A2EFEF0670000082F0E4EBAFF2C
 1E2EFEF0488748C72F074EBAFBBE4CDF10804E5E2E9F4E7500000000241F225F205F2F02
 4240101842411219340 0B44163023401B4026002B10956CAFFFC6602B0414E75>

$cexec

not {(Couldn't define unprotectpath r) = quit} if end

/dumpchar {false charpath /TempStr 255 string def unprotectpath {exch TempStr cvs print
( ) print TempStr cvs print ( moveto) =} {exch TempStr cvs print ( ) print TempStr cvs print
(lineto) =}{6 5 roll TempStr cvs print ( ) print 5 4 roll TempStr cvs print ( ) print 4 3 roll
TempStr cvs print ( ) print 3 2 roll TempStr cvs print ( ) print exch TempStr cvs print ( )
print TempStr cvs print ( curveto) =}{(closepath) =} pathforall} def

%  //// demo - remove or alter before reuse  ////

userdict begin newpath 0 0 moveto /Times-Roman findfont 72 scalefont setfont
(T) dumpchar quit
```

Fig. 3 – A PostScript font path grabber.

There is a simple little *pseudo-compiling* trick you can pull. With pseudo-compiling, you can convert any source of PostScript code into a *Just the facts, Ma'am* form that uses nothing except the simplest of pick and position PostScript commands. Anyone can run a pseudo-compiled EPS file, and absolutely zilch will remain of the original code or the way it was generated.

Better yet, that pseudo-compiled PostScript often will print ridiculously faster than the original. And pseudo-compiling can be easily and quickly done by using Adobe's new and cheap *Distillery* program, or by one of my *gonzo* routines.

The *Distillery* is now available as DISTILL.PS #186 on *GEnie* PSRT.

We have seen a few examples of pseudo-compiling in the past issues and in the *Ask the Guru* reprints.

What about screen images? In any broad based user interchange standard, you certainly wouldn't want to use any screen images if they hurt device independence in any way.

Fortunately, new systems using display PostScript or those display PostScript emulators are now becoming readily available, as is the software which returns printer bitmaps for a screen display. So are programs and applications that internally capture and display EPS.

How does an EPS file differ from some ordinary PostScript textfile code? Actually by very little. There are a few required remarks at the start of your file plus a few optional ones. And a few obscure PostScript commands (such as *initgraphics*) are not allowed if they could somehow corrupt the program or system that the EPS file is being imported into. Several other unpopular commands (such as *settransfer*) must be carefully saved and restored if they are altered. EPS files are limited to a single page each, but you can use as many of them as you like.

To get started on all of this, get yourself copies of the *Encapsulated PostScript Files Specification*, v 2.0, and that *Document Structuring Conventions Specification*, v 2.1. Both of these are available free on request

through Cynthia Johnson at *Adobe Systems*. You should not create or use any EPS files without having further EPS details on hand.

The fundamentals on PostScript are covered in Adobe's blue *PostScript Tutorial* and the red *PostScript Reference Manual II*. The latter now includes full EPS details.

Lots of printed circuit layout, schematic, isometric, and perspective drawing routines can be found in my *PostScript Show and Tell*. Let me know your thoughts on a standardized hacker interchange format based upon EPS files.

What is PostScript Super Fax?

I am not at all impressed with today's fax machines. In fact, I think they are an outright con. You have grossly overpriced machines which produce an abysmally putrid print quality on ludicrous papers.

Yet, none of this need be. A slew of new PostScript superfax printers and software are about to hit the market that completely blow these traditional stand-alone machines out of the water.

Among other obvious benefits, PostScript superfax offers far lower costs, a much higher print quality, and shorter comm times.

It only costs around $40 or so in parts to add a full superfax capability to a PostScript laser printer. A stand-alone machine is not needed for many applications.

Here's how superfax works: A PostScript superfax machine makes the phone call and then issues an inquiry string to determine whether an ordinary fax or a superfax is at the other end. If an ordinary fax machine, a 216 DPI bitmap is created and compressed, then sent and received. Even using a dumb fax machine at the far end, you'll get a much higher final print quality, as no scanner is involved.

Instead, if there is a PostScript superfax at the far end, then the image gets sent as a PostScript program instead. When saved to disk at the far end, PostScript's device independence can give you instant *National Geographic* quality. *Yes, you can easily fax camera-ready art!* In addition, the far end machine prints you a hard copy to the best of its resoloution abilities. Even at plain vanilla 300 DPI, this is vastly better than traditional fax. Especially since you can print onto your choice of good papers.

What about transmission times? Sometimes the PostScript superfax could be slower, but generally it can be much faster. For instance, an ordinary fax business letter might consist of four million dots.

Let us assume a 10:1 compression, leaving us with 400 K dots, or 50K bytes. The same letter done in PostScript can be coded in less than 5K bytes, perhaps less if the letterhead or signature or background form are in dictionaries. Thus, much of your ongoing typical business correspondence could be up to ten times faster when using PostScript superfax.

```
% Copyright c 1990 by Don Lancaster and Synergetics, 3860 West First Street, Box 809,
% Thatcher AZ 85552. (602) 428-4073. All rights reserved. Personal use is permitted so
% long as this header remains intact. Show and Tell disks for Apple, Mac, or IBM are $39.50.

% This routine is similar to the PostScript -ashow- operator, except that you can non-linearly
% remap your text message onto any surface. For perspective, star wars, spheres, etc...

% Here are some typical path results from figure three...

/T {611 {14 729 mt 14 604 li 235 604 li 235 0 li 385 0 li 385 604 li 598 604 li 598 729 li cp}} def

/N {722 {68 729 mt 68 0 li 218 0 li 218 508 li 513 0 li 661 0 li 661 729 li 511 729 li 511 228 li 216
729 li cp}} def

/O {778 {742 359 mt 742 469 702 579 634 650 ct 573 713 484 741 391 741 ct 298
741 209 713 148 650 ct 80 579 40 469 40 359 ct 40 249 80 139 148 68 ct 209 5 298 -23 391 -23
ct 484 -23 573 5 634 68 ct 702 139 742 249 742 359 ct cp 391 108 mt 343 108 295 122 257 157
ct 209 201 185 280 184 359 ct 185 438 209 517 257 561 ct 295 596 343 610 391 610 ct 439 610
487 596 525 561 ct 573 517 597 438 598 359 ct 597 280 573 201 525 157 ct 487 122 439 108
391 108 ct cp}} def

/F {611 {74 729 mt 74 0 li 224 0 li 224 313 li 543 313 li 543 438 li 224 438 li 224 604 li 586 604 li
586 729 li cp}} def

/E {667 {79 729 mt 79 0 li 624 0 li 624 125 li 229 125 li 229 313 li 578 313 li 578 438 li 229 438 li
229 604 li 607 604 li 607 729 li cp}} def

/R {722 {80 0 mt 230 0 li 230 288 li 406 288 li 458 288 494 264 496 215 ct 494 162 494 104 496
68 ct 498 44 504 22 516 0 ct 677 0 li 677 27 li 661 37 649 46 649 87 ct 648 124 646 220 642 250
ct 634 316 586 334 561 351 ct 610 380 642 403 660 466 ct 678 529 663 581 653 613 ct 632 692
558 726 485 729 ct 80 729 li cp 230 604 mt 433 604 li 500 604 524 574 524 522 ct 533 450 470
413 435 413 ct 230 413 li cp}} def

/space {250 {}} def

% Your non-linear transform goes here. It accepts x and y on the stack and later returns
% x' and y' to the stack, following your remapping rule or rules...

/nlt {} def    % substitute your own nonlinear operator (see fig 5 for example)

% Service routines...

/mt {exch fontsize 0 get mul .001 mul xoffset add nlt exch fontsize 3 get mul .001 mul yoffset add
moveto} def  % nonlinear moveto

/li {exch fontsize 0 get mul .001 mul xoffset add nlt exch fontsize 3 get mul .001 mul yoffset add
lineto} def  % nonlinear lineto

/ct {3 {6 2 roll exch fontsize 0 get mul .001 mul xoffset add nlt exch fontsize 3 get mul .001 mul
yoffset add} repeat curveto} def % nonlinear curveto

/cp {closepath} def   /strrx ( ) def

/nlashow {/msg exch def msg {strrx exch 0 exch put strrx dup ( ) eq {pop (space)} if cvn load
/curchar exch def curchar 0 get /cwide exch def curchar 1 get /nlashow {/msg exch def msg {strrx
exch 0 exch put strrx dup ( ) eq {pop (space) } if cvx exec exec charproc fontsize 0 get mul 1000
div extrakern add xoffset add /xoffset exch def} forall} def

/charproc {stroke} def  % does what you want to your path

%  //// demo - remove before use. ////

/fontsize [72 0 0 72 0 0] def /xoffset 100 def /yoffset 200 def /extrakern 3 def

(FREE  FONT) nlashow showpage quit
```

Fig. 4 – A PostScript nonlinear text string remapper.

I've said about all I am allowed to on this for now. But expect some PostScript based major changes in what fax is and what it can do.

How About a Rubber Stamp Update?

The *Grantham Polly-Stamp* folks recently sent me one of their new rubber stamp kits. My students and the hired help sure have been getting off on it.

The first secret to custom rubber stamps, naturally, lies in PostScript. With PostScript, you can freely mix and match any combination of text, graphics, and line art you want for virtually any custom stamp of any size. You can also work 1:1 and do reverses, so there's no need for a darkroom or a stat camera.

The second secret to all of your custom rubber stamps is the magic liquid photopolymer. *Merigraph* is one brand name. While intended for flexographic carton printing, Merigraph performs just fine for rubber stamps of virtually any size.

You can get this stuff retail from *Grantham Polly-Stamp* or *IMEC*, in larger quantities from *R.A. Stewart*, or in railroad tank car lots directly from *Hercules Merigraph*.

Even retail, you are only looking at fifteen cents per square inch of stamp. Say a quarter after you allow for near misses and unused space between individual stamps.

Figure two shows you how the Merigraph process works. Usually, you do enough stamps side by each to make up a 4-1/2 by 9 sheet.

Merigraph is a chemical liquid photopolymer that hardens in the presence of ultraviolet light. You first "flash" expose an 1/8 inch layer of the liquid glop through an underlying backing sheet. This will harden the *entire* bottom of your photopolymer, to a depth of about *one-half* the liquid thickness.

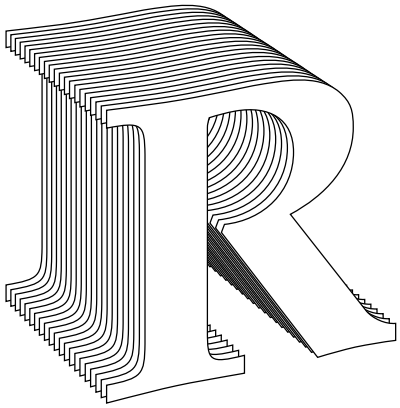
A second flash exposure is then made *from the other side*, through a negative, through a 1:1 PostScript transparency, or else a *Kroy Color* reversal. Those portions of your negative that transmit light will selectively harden areas within the Merigraph liquid, forming the hardened "mesas" where the letters and line art are to appear.

The second exposure is once again set to harden to a depth of about *one-half* the liquid thickness. Your solid mesas now merge with the solid backing, forming your typical rubber stamp shape.

You next peel off the negative or whatever and then wash away any remaining liquid photopolymer by using soap and hot water. The result at this point is a "raw" rubber stamp. You then expose yet a third time to thoroughly harden all of the remaining photopolymer. This gets done under water.

Lastly, you cut the stamps to size, dust them with powder, and mount them on individual stamp handles. The final polymer is around 50 durometer, and comparable to a good grade of stamp rubber.

You do have to use a negative material that separates easily from the partially cured photopolymer. I've found *Kroy Color* to be a good way to get a totally black and totally dense negative from the PostScript paper positive original. Just Kroy Color in the usual way, but then



```
/R { 722 { 124 115 mt 123 43 119 35 39 37 ct 39 -5 li 105 -2 156 0 195 0 ct 234 0 284 -2 350 -5 ct
350 38 li 269 35 265 43 265 115 ct 265 601 li 265 608 267 616 269 624 ct 288 627 309 629 329 629
ct 407 629 464 587 464 505 ct 464 411 396 354 304 362 ct 297 344 li 521 -5 li 551 -2 581 0 611 0 ct
643 0 676 -2 708 -5 ct 708 37 li 683 40 653 61 638 82 ct 457 356 li 540 386 606 445 606 539 ct 606
580 601 602 572 632 ct 520 686 440 683 370 683 ct 299 680 247 678 205 678 ct 163 678 110 680
39 683 ct 39 641 li 119 643 123 635 124 563 ct closepath}} def

/mt {exch fontsize 0 get mul .001 mul xoffset add exch fontsize 3 get mul .001 mul yoffset add nlt
moveto} def

/li {exch fontsize 0 get mul .001 mul xoffset add exch fontsize 3 get mul .001 mul yoffset add nlt
lineto} def

/ct {3 {6 2 roll exch fontsize 0 get mul .001 mul xoffset add exch fontsize 3 get mul .001 mul yoffset
add nlt} repeat curveto} def  /cp {closepath} def /strrx ( ) def

/charproc {gsave 1 setgray fill grestore 0.5 300 mul 72 div floor 72 mul 300 div setlinewidth stroke}
def

/nlashow { /msg exch def msg {strrx exch 0 exch put strrx dup ( ) eq {pop (space) } if cvn load exec
exec charproc fontsize 0 get mul 1000 div extrakern add xoffset add /xoffset exch def} forall} def

/prd1 {/zi exch def /yi exch def /xi exch def objrot cos xi mul objrot sin yi mul sub objrot sin xi mul
objrot cos yi mul add zi /zz exch def /yy exch def /xx exch def yo dup yy add div dup xx xo sub mul
exch zz zo sub mul} def      /nlt {currenty exch prd1} def

% //// demo - remove before use. ////

/fontsize [200 0 0 200 0 0] def /extrakern 10 def /xo -300 def  /yo 700 def /zo 500 def
/objrot 15 def 0 800 translate

60 -3 0 {/currenty exch def /xoffset 0 def /yoffset 0 def (R) nlashow} for showpage quit
```

Fig. 5 – Perspective letter using nonlinear transformations.

throw away the baby and keep the washwater. That's right. You throw away your original, and instead use the remaining "scrap" Kroy Color as your totally opaque negative.

You have to watch out for any fill-in on centers of small "e" letters and such, but Kroy Color does the job with few hassles. Their blue metallic works especially well.

Extra kerning is also a good idea.

The *Grantham* stuff is aimed at a three-tier retail craft market, so the individual pricing is higher than you'd like to see. At least for the serious desktop finishing uses. In particular, $1000 seems a tad much for an exposure unit that you can build yourself from a fluorescent fixture, two bulbs, and a timer. Or eliminated entirely by going solar and using the sun instead.

But all of Grantham's products and supplies are top quality. They are solidly designed, properly built, optimally packaged, and are lots of fun to use.

IMEC, a Grantham competitor, offers a refundable $15 video which gives you more details on rubber stamps in general. Sadly, IMEC's prices border on the absurd.

Because time and hassle equals dollars, you can decide for yourself whether to steal the plans or else go with what's on the shelf.

What is This Month's PostScript Utility?

We'll have our PostScript utility earlier than usual this month, since we need the results for what is to follow. This one has been numero uno on the request line.

As we found out last month, it is now a simple matter to grab any PostScript type I and type III font path. This new opportunity very much enriches what PostScript is and what you can do with it.

Figure three shows you another way for you to record plaintext font paths. It will work on any type I or type III font, be it internal, downloaded, custom, or created on the fly. It works by bypassing the font lockout on *pathforall* and directly accessing the underlying operator.

You simply select your font and the character in that font, and run this utility. A recordable proc gets returned to the host that includes both the width to the next character and a plaintext description of those involved *moveto*, *lineto*, *curveto* and *closepath* operators.

The imaging is initially done at 1000 points, which bypasses most hint machinery. You can alter this to investigate the hinting used at any particular point size.

I have only tried this utility on a late *LaserWriter Plus*, an NT, and an NTX, so it may or may not work on other machines. Be sure to let me know. I have included some short time delays to simplify your host recording. You can eliminate these if your comm setup lets you. The simplest host recorder I know of is AppleWriter's [Q]-I [esc]-R.

To use your returned font just do a *T exec* or whatever. Then stroke, fill, or clip your path. The width will remain on the stack as a real number, easing moving on to the next character. If you do not need this width info, be sure to *pop* it off the stack each time.

To scale your final font, divide it by 1000 and then multiply by your desired point size.

You may also want to replace the returned *moveto* with a *mt* and so on, to simplify the redefinitions we are about to look at. Which also will shorten your files. Do note that both *lt* and *ln* are reserved PostScript words, and must *not* be used as a *lineto* substitution. The *li* seems to work just fine here.

PostScript level II has eliminated the font lockout completely.

How Can PostScript Use Non-linear Transformations?

PostScript achieves its device independence by keeping a separate *user space* and a *device space*. To get between these two, *linear transformations* are continuously made on the fly to perform all your requested translates, rotates, and scales. These linear transformations are also referred to as the *currentmatrix*.

Specifically, the user space and the device space are related by…

$$x' = Ax + By + C$$
$$y' = Dx + Ey + F$$

where A is the horizontal scaling, B is the amount of lean, C sets the horizontal offset, D is the amount of climb, E is the vertical scaling, and F is the vertical offset.

While this can let you do great things, the stock PostScript linear transformations only allow you to translate, rotate, or scale an *entire* path at once. It would be infinitely more powerful to be able to *individually* change the translation, the rotation, and scaling for each and every point pair involved in every individual *moveto*, *lineto*, or *curveto* operator in the path.

Such a *nonlinear transformation* would let you "map" any PostScript path or routine onto any surface. Obvious examples would include perspective and "star wars" lettering, writing on a scroll, isometric cylindrical and spherical lettering, twisted surfaces, and intentional distortions. The new opportunities here totally boggle the mind.

Now that we can grab the font paths, it becomes a simple matter to redefine each *moveto*, *lineto*, and *curveto* operator so that each x and y data pair gets translated to some new x' and y' according to some set of suitable rules.

For instance, figure four shows you a new *nlashow* or a "nonlinear ashow" operator that will transform each and every data point in your text by a rule or set of rules you can define as *nlt*, short for a *non-linear transformation*.

Finally, figure five can show you how to create a large 3-D two point perspective letter.

Nonlinear transformation gives you an infinitely large new world to explore, of which perspective is only the tiniest of obscure niches.

What new can you think of here?

We can wrap things up with our usual reminders that most of those sources I've mentioned appear in the *Names and Numbers* appendix and that you can get tech help by calling (602) 428-4073. ❄

Don Lancaster's

ASK THE GURU

June, 1990

PostScript fractal ferns
The secret commons ploy
Toners for T-Shirt printing
Shared SCSI comm speedup
Duplex printing fundamentals

Apple has released two big binders for all their Apple and Mac technical notes, with free copies going to user group ambassadors. You can get your own copies from *APDA*, or else become an ambassador yourself.

Ambassadors receive all sorts of freebies, including tech notes, CD ROM disks, other software, videos, AppleLink access, special purchase offers, handouts, and bunches of other propaganda. As we've seen before, you can get a complete list of all your local Apple user groups at (800) 538-9696, extension 500.

Apple's new high speed SCSI card for the Apple IIe or IIgs sure looks interesting, and I've picked one up for testing. Most users will want this card to add a ProDOS compatible hard disk onto their system. So far, their documentation is poor, and several major unfixed bugs are present. I will be mostly using mine to directly read and write to my LaserWriter NTX hard disk from inside AppleWriter. More on this below.

Apple has also published the preliminary developer versions of their system 7.0 software for the Mac. Cost is $20 for CD ROM or $90 for ordinary disks. There seems to be a not-too-subtle hint here on Apple's preferred software media.

Yes, the new Royal outline font stuff is here in 7.0, but is renamed TrueType. To me, the vibes feel just plain wrong on this one. Really dumb even.

Their TrueType has absolutely no technical advantage, no marketing advantage, and zero performance advantage over the way PostScript used to be several years ago. I guess the real bottom line is that PostScript flat out ain't broke.

Speaking of which, Adobe is up to all sorts of interesting things. As we've seen, they've released their black book on the *Type I Font Format* which reveals all. I've got these in stock, or you can get a copy directly from *Adobe Systems*.

Adobe newly introduced a high end PostScript *emerald* controller. Emerald should dramatically speed up future PostScript phototypesetting and fancier laser printing. It is RISC based, and apparently does special hardware speedup tricks for Bezier cubics and BitBlts.

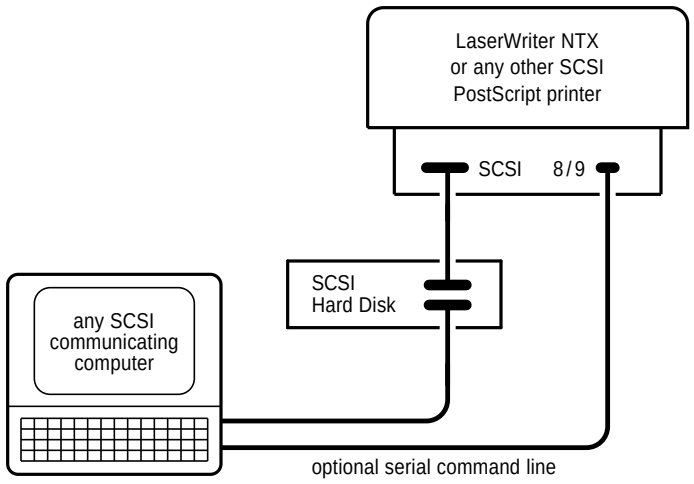
Adobe has also just announced PostScript 2.0. This is mainly a way of picking up and consolidating all of the little changes and improvements that got added over the past several years. Things like compressed file formats, composite fonts, bug repairs, improved color halftones, and immediately executed literal names.

Big new features include open font paths (!), automatic forms, and dramatic speedups for user proc-cached routines.

The other new stuff now includes dynamically allocated memory (no more *dictfull* errors) and a more uniform linewidth on lower resolution printers.

PostScript 2.0 is also considerably faster, owing to tricks learned in the process of developing the Display PostScript. Apparently their interpreter doesn't have to go as deep as often as it did originally, leading to a significant speedup.

Full details on Level II appear in the newly revised *red book II* and otherwise known as the Adobe *PostScript Reference Manual*. I do try to keep copies of this 800 page



With shared SCSI comm, both the host and the printer can share one or more hard disks, and talk to each other at a 400 kilobaud or higher rate.

Since most files will already be on the disk, the usual load-send-print cycles can often be eliminated, for another 2:1 speedup. Direct host access to font caches and bitmaps can give even faster results.

Command or control bytes can be sent via the older serial or AppleTalk comm lines. This is still fast, since only a few bytes are usually needed.

Fig. 1 – How High Speed Shared SCSI Comm Works.

handbook in stock for you here at *Synergetics*.

As a reminder, our major new PostScript BBS system should be going great guns on *GEnie* by the time you read this.

Let's see. You can now get my Book-on-demand reprints of most everything you see here in *Ask the Guru* or over in my *LaserWriter Corner* column. Or for that matter, over in my *Hardware Hacker* columns in *Radio-Electronics* or all the *Blatant Opportunist* stories in *Midnight Engineering*. Call for more info.

We have some heavy duty stuff for this month…

What is Shared SCSI Comm?

How would you like to instantly speed up all your PostScript makeready times by 30:1? This is trivially easy to do. If only we can get the printer designers to listen to what their customers want. Or if you care to get sneaky on your own.

As we've seen in past columns, most PostScript output is baud rate limited because of those horribly primitive comm channels in use today. Being baud rate limited is a serious problem when book-on-demand publishing, and whenever you are handling color seps or any other large images.

And this is bound to get much worse as we see faster chips, faster PostScript firmware implementations, and more and more people routinely pseudo-compiling all of their PostScript files.

To refresh your memory, an IBM typically talks to a LaserWriter at a 10 kilobaud effective baud rate. An Apple IIe could run at 57 kilobaud effective when bare metal driving its game paddle port.

But AppleTalk sending PostScript typically runs around 17 kilobaud effective on an older Mac or 34 kilobaud effective on a Mac II.

Because of its expense, its host limitations, and its unacceptably slow speeds, AppleTalk is clearly unsuitable for serious PostScript laser printing.

But SCSI communication blazes along at a minimum of 400 kilobaud, and potentially can run at five or more times that rate.

Obviously, SCSI comm is the only route to pick with PostScript. Also obviously, few PostScript printers now offer SCSI and some of those that do have made the SCSI access unbearably difficult. Not to mention totally undocumented.

So, I would like to propose a previously unthunk of communications setup which offers you as much as a 30:1 speedup in all the PostScript makeready times, will work with any host, allows simple and cheap networking, and offers lots of other advantages. I call this scheme *Shared SCSI Comm*, and it appears in figure one.

Few people realize that SCSI was designed from the ground up to allow multiple hosts. That's right. Just as one computer can host multiple SCSI hard drives, several computers can share the same SCSI hard disk, a CD ROM, PostScript laser printer, or whatever. Presto. Instant and fast networks.

One reason this has not been done too much in the past is that many software programmers have been keeping their actual SCSI operating systems secret. Another problem is known as *adjudication*. The funniest things happen should Purchasing grab a file, Production snatch the same old file, and then Purchasing rewrite its new file to disk. It's even more hilarious when the file that's involved is a CEO's list of mergers and acquisitions.

In a laser printing environment, adjudication should be rarely, if ever, needed. So, for all of our uses, adjudication is a simply avoidable non-problem.

Another SCSI network restriction has been its limited range. But, as we have seen a number of times before, if you cannot reach out and touch your PostScript printer from your work station, its utility to you will drop dramatically. Contrary to popular belief, PostScript printers are definitely *not* suitable for long distance networking.

Here is how shared SCSI comm

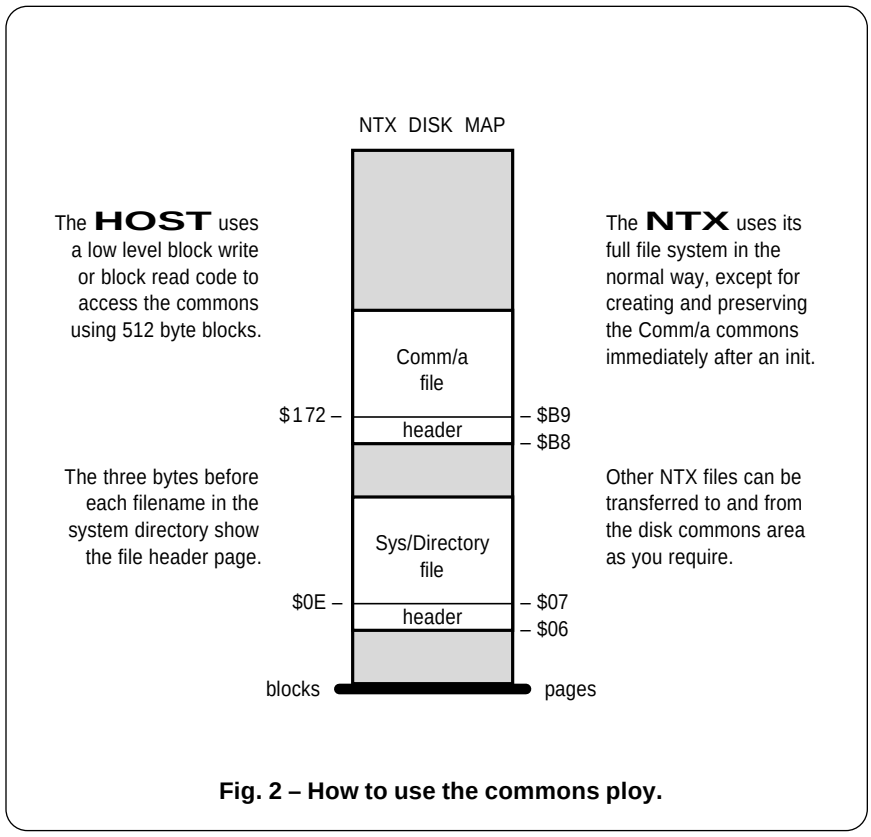


Fig. 2 – How to use the commons ploy.

works: A host computer with any old SCSI driver is connected to one or more SCSI hard disks. Your last hard disk in your chain also gets connected to a SCSI PostScript laser printer such as a *LaserWriter NTX*. Priorities are established so that the laser printer is the boss.

Your host computer places its PostScript files onto the disk. The printer takes them off. All at a 400 kilobaud or higher effective rate.

Wait. There's even more. If your word processor is routinely writing and saving to the laser printer hard disk, the file is *already there* at print time, and you can eliminate your usual load-transmit-print cycle.

Which could further *double* your effective comm speeds. And some word processors insist on throwing in extra carriage returns. The need to patch around these extra returns with the usual "\r" could get eliminated with shared SCSI comm.

If you are an advanced PostScript programmer, direct host access to the PostScript fonts and font cache on the hard disk can dramatically improve what you do and how you do it. Better yet, it might be possible to save partial or entire page bitmaps to your hard disk. We have looked at this *proc cacheing* stunt in previous columns and in the *Ask the Guru* reprints. Ultimately, you can instant print an entire book simply by using page bitmaps, regardless of how fancy your pages are.

Adobe is working up some new standards for SCSI comm, shared and otherwise. Do send in for their new ap notes, and give them some useful user input.

As I see it, there are three main routes to shared SCSI comm. One is for those printer manufacturers to provide us with firmware patches that give us direct SCSI communications on a par with the existing serial, parallel, and the AppleTalk comm setups. Including a two way hard disk "pass through" mode.

A second route is to publish and properly document the *entire* SCSI operating systems and hard disk formats in use for each printer, and make runnable copies of it available for independent use by any host.

The third route is something you can do here and now, instantly, and without any help from "them." You do have to be willing to hand bash a few bytes to accomplish this. For all of the secret details…

(A) HD.READBLK.300 reads a 512 byte block to host start location $1000 …

```
0300: RDBLK  20 10 C4  JSR $C410  % Dispatch address of SCSI sub
0303:        41        DFB $41    % Command for extended block read
0304:        0A 03     DW  $030A  % Location of command list
0306:        20 A8 FC  JSR $FDDA  % Hex error to screen (00 = none)
0309:        60        RTS        % Return to calling routine

030A: CMDLST 03        DFB $03    % Three parameters
030B:        01        DFB $01    % Unit number
030C:        00 10     DW $1000   % Data buffer low bytes
030E:        00 00     DW $0000   % Data buffer high bytes
0310:        72 01     DW $0172   % Disk 512 block low bytes
0312:        00 00     DW $0000   % Disk 512 block high bytes
```

(B) HD.WRTBLK.300 writes a 512 byte block from host start location $1000 …

```
0300: WRTBLK 20 10 C4  JSR $C410  % Dispatch address of SCSI sub
0303:        42        DFB $42    % Command for extended block write
0304:        0A 03     DW  $030A  % Location of command list
0306:        20 A8 FC  JSR $FDDA  % Hex error to screen (00 = none)
0309:        60        RTS        % Return to calling routine

030A: CMDLST 03        DFB $03    % Three parameters
030B:        01        DFB $01    % Unit number
030C:        00 10     DW $1000   % Data buffer low bytes
030E:        00 00     DW $0000   % Data buffer high bytes
0310:        72 01     DW $0172   % Disk 512 block low bytes
0312:        00 00     DW $0000   % Disk 512 block high bytes
```

(C) A simple BASIC routine to read 64 disk blocks to host start location $1000 …

```
100 PRINT CHR$(04); BLOAD HD.RDBLK.300
110 POKE 784,0: POKE 785,0: POKE 786,0 : REM LO-MID-HI START
120 FOR K = 1 TO 64
130 CALL 768 : REM READ DISK
140 POKE 781, PEEK (781) + 2: POKE 784, PEEK (784) + 1
150 PRINT: NEXT K
160 END
```

(D) Some more common error messages…

```
$00- None             $11- Bad Unit Num   $2B- Write Protect
$01- Bad Command      $27- I/O Error      $2D- Bad Block
$04- Bad Param Count  $28- No Drive       $2F- Off Line
```

Fig. 3 – Some bare metal SmartPort commons code.

Tell Me About The Commons Ploy.

Here is a sneaky way you can instantly add a direct shared SCSI comm to your existing LaserWriter NTX. You can also make backup copies of protected SCSI disks on any system, and read and write any data to or from them. Even with a totally locked, inaccessible, and a top secret operating system.

But you do have to carefully bash some bytes at the machine language level, and you can easily trash out all the files on your hard disk if you are not ultra careful. The method is also a tad on the cumbersome side and requires extreme care.

I call this the *commons ploy*, and the general scheme appears for your viewing pleasure in figure two.

Here is how the commons ploy works: The LaserWriter accesses its hard disk with its own internal and proprietary operating system. The host computer will access the same hard disk directly at the *block read* and *block write* level. Several keystrokes under the *SmartPort* on an Apple IIe is ideal for this.

When you initialize your LaserWriter hard disk, you immediately and very carefully create one or more dummy files of longer than

needed length and having names of something like *Comm/a, Comm/b*, and so on. You then find out *exactly* on which blocks all these files are sitting.

You next use your host to either read or overwrite these *fixed position* files using any plain old *block write* routine. Your LaserWriter doesn't know its textfiles have been corrupted and still uses the files in the normal manner. All the directory entries remain the same.

Yes, it works. And, yes, a simple PostScript ending *quit* command can ignore what is left of the file that you just wrote over.

Specifically, the LaserWriter uses a simple operating system alike but different somehow from plain old Apple ProDOS. This operating system uses pairs of 512 byte blocks formed into 1024 byte *pages*, and initially builds up from page zero. Except when it hides something sneaky in "high core".

Each file consists of one header page, followed by one or more data or text pages. The system directory starts on page $06 or block $0C, and all the actual filename entries start on page $07 or block $0E.

Those three bytes immediately before each filename are the page starting address for each directory entry. Note that these bytes are in a high byte first order.

Once again, double the hex page number to get the block number. Data or text actually begins on the *second* page of any individual file.

On a hard disk init, the following files get written, starting at page zero and working up…

%disk%Sys/Root0
%disk%Sys/Root1
%disk%Sys/AllocMap
%disk%Sys/Directory
%disk%FC/NameToNID
%disk%FC/MIDToFile

You then add one or more fixed "commons" files of names…

%disk%Comm/a
%disk%Comm/b

You must do this *immediately* on your init, before any font caches (or anything else) gets written to disk. Be certain to check, but you should find your first commons file having its header placed on page $B8, equal to block $170, The actual text starts on page $B9 or block $172.

Those text pages are plain old 8-bit sequential low ASCII textfiles without any headers, trailers, or checksums.

As you know, I just picked up a new Apple II High Speed SCSI card. Since it is insanely faster, cheaper, and easier to go bare metal on a IIe compared to a Mac, I picked this route for my initial experiments. Figure three shows you the simple procs which let you read SmartPort blocks and write SmartPort blocks, along with a trivial BASIC listing that lets you back up your LaserWriter hard disk 32K at a whack. Figure three also summarizes the SmartPort error messages you may get on your screen.

Note that I put my SCSI card in slot number 4. If you use a different slot, you will have to find a new SmartPort *dispatch address*. For slot #n, your dispatch address will be $Cn00 + PEEK ($CnFF) +$03. The slot #4 dispatch address is $C410.

Your commons files can now be directly used by writing onto them

Beauty of Fractals, by Peitgen and Richter

Earlier text that is both a coffee table book and a source of useful algorithms. From the same publisher that did the *Science of Fractal Images*. (Springer-Verlag, 1986)

BYTE Magazine

Occasional fractal articles and reviews. See *A Better Way to Compress Images*, January 1988, pp. 215-223 in particular.

Chaos – Making of a new Science, by James Gleick

Interesting and easily read survey of the entire field of fractals and chaos science. Frustratingly shallow at times. (Viking, 1987)

Fractal Geometry of Nature, by B.B. Mandlebrot

The original "horse's whatever" book that started it all. Mind blowing artwork in an otherwise unreadable and excessively egocentric text. (W.H. Freeman Press, 1982)

Science of Fractal Images, by M.F. Barnsley et. al.

Outstanding collection of useful hands-on fractal information. A good starting place to extend your own research. Includes algorithms and a good bibliography. (Springer-Verlag, 1988)

SCIENCE Magazine

Occasional fractal and chaos science articles and reviews.

SCIENTIFIC AMERICAN magazine

Occasional fractal articles and reviews. See their *Computer Recreations* column for ongoing info and comments.

SIGGRAPH Proceedings

Far and away the best annual computer graphics show. Includes the latest and best of fractal and chaos science developments, on all three levels – theoretical, tutorial, and artsy-craftsy. (Siggraph-ACM Press, annually)

Fig. 4 – Some fractal and chaos science resources.

with your host and then running them with a simple command string of a few bytes sent over your usual serial or AppleTalk channel.

To permanently keep any file, load the commons file into the NTX and then save it under a new name in a different place on the disk.

To return an existing file to the host, load the file into the NTX, pad the file up or down to match the old comm file length, delete your comm file, and then overwrite it with the new file. Your host then can read the blocks at the expected place.

All of which gets a tad awkward. But the commons ploy still seems much faster and far more open than any other route that is immediately available for your use.

I am working on AppleWriter patches that directly let you write to the LaserWriter disk. Let me know if you have any interest in these.

Let's see. The Apple *white* book, otherwise known as the *LaserWriter Technical Reference* has all of the usual LaserWriter SCSI commands in it. I've expanded and added to these in past issues and the *Ask The Guru* reprints. You also may find the *IIgs Firmware Manual* and the not yet updated *Apple II SCSI Card Technical Reference Manual* from *APDA* to be most helpful.

Needless to say, if you miss the commons area and overwrite the wrong block, you trash your hard disk contents and have to start over.

Naturally, it would be best to work directly with Adobe's existing directories. But so far, they have not published their formats. But maybe we can fake it. Stay tuned.

Let's make a contest out of this. Just contribute something to this shared SCSI or the commons ploy dialog, or else show me equivalent stunts that can make a Mac or a PC almost as good as an Apple IIe.

We'll have all our usual *Incredible Secret Money Machine* book prizes on this, along with an all expense paid (FOB Thatcher, AZ) *tinaja quest* for two going to the best of all.

```
/problistcreate {mark /counter 0 def probabilities {128 mul round cvi {transforms counter get}
repeat /counter counter 1 add def} forall counttomark 128 sub neg dup 0 gt { [1 0 0 1 0 0]
repeat} {pop} ifelse] /problist exch def} bind def

/doit {problistcreate 1 1 20 {problist rand -24 bitshift get transform 2 copy moveto 0.001 10
rlineto} repeat newpath numdots {problist rand -24 bitshift get transform 2 copy moveto 0.001 0
rlineto stroke} repeat} bind def

% /// demo - remove before use. ///

/numdots 6000 def   % increase for denser image; decrease to print faster

/transforms [ [0 0 0 .16 0 0] [.2 .23 -.26 .22 0 1.6] [-.15 .26 .28 .24 0 .44]
[.85 -.04 .04 .85 0 1.6] ] def

/probabilities [ .01 .07 .07 .85 ] def

1 setlinecap 0 setlinewidth 200 300 translate 30 dup scale doit showpage quit
```

Fig. 5 – Fractal fern done in PostScript.

What is Duplex Printing?

Duplex printing is simply printing on both sides of any sheet of paper. You do so in a single printer pass, without user intervention.

The main advantage of duplex printing is that it is self-collating, meaning that your documents can arrive ready for a binding. Which eliminates bunches of labor and lots of potential error sources.

In addition, a duplex printing minimizes paper jams and their seriousness. Done the "old way", a jammed page in the middle of the backside pass of a book-on-demand job can trash most of the book. This happens since the rest of the pages could end up with the wrong backs on them. With duplex printing, a jam usually trashes only one single page that is easily replaced.

Duplex printing also saves you on paper and filing costs. First, because only half as many sheets are needed when printing on both sides, and second, because your scrap will often be considerably less.

At this writing, there is only one game in town when it comes to duplex printing. That one is the *Hewlett-Packard IID*. I've recently ordered a used one from *Thompson & Thompson* so I can explore some rather badly needed productivity upgrades to my Book-on-demand publishing.

This machine has several major and painfully obvious flaws, so I'll certainly flush it just as soon as anything better shows up.

There is no provision for a SCSI comm essential for high speed and download-free access to all of your book-on-demand pseudocompiled pages or for the permanent storage of high quality fonts and caches. The maximum baud rate is one rather unfunny joke at 19200. At 75 pounds, the IID would also make a fair to middlin boat anchor.

While I haven't tested the brand new Adobe PostScript 55 cartridge for this beast at this writing (stay

tuned), I do feel that any cartridge PostScript solution will always be significantly slower than using a printer designed from the ground up to inherently speak PostScript as its native language.

Duplex printing requires double memory, since one page makes up as the previous one prints. Four Megabytes is essential.

What's in the IID? The top half is more or less a plain old *Canon* SX engine. At the rear is a new *switchback* mechanism. And underneath is the return path.

On the first pass, the front side gets printed and the page goes out to the switchback assembly. This reverses your paper direction and sends the sheet back *underneath* the main printer. The paper then makes a 180 degree turn back up into the original registration assembly, and positioned blank side up. From that point, the back side gets printed just as if a new sheet was fed. Software selects a "book" or a "calender" binding registration.

A second input paper tray is also provided, since the duplex paper path is there anyway. While this second lower tray is handy, note that heavy cover and index stock can only be fed from the upper tray. You are also only allowed to duplex print normal weight papers.

Amazingly, the per-side printing time is only modestly longer than the 8 PPM of a straight SX engine.

As with their other machines, HP has an outstanding service manual available to anyone. Part number is #33447-90904 and costs around $99. As we have seen in the past, their previous CX manual (#02686-90904) and SX manual (#33440-90904) make excellent Apple LaserWriter service and repair references.

Just ignore all of their arrogant nonsense found in these manuals concerning toner cartridge refilling. Since refilled cartridges reduce your per-page toner costs by as much as 20:1, only an absolute idiot would not refill their cartridges for multiple reuse.

I'll have lots more to say on this beast once I've done some more testing. Because of the lack of SCSI comm and its inherently slow way of implementing PostScript, the IID is clearly unsuited for its intended purpose. But duplex printing must be explored here and now.

Please Update Me On T-Shirt Printing

As you may have noticed, it gets really difficult to hand feed a T-Shirt through your *LaserWriter*.

Especially if someone happens to be wearing it at the time.

It seems we do have a brand new breakthrough for this month in a form of *heat refusable* fabric toners. These let you print your images to paper and then iron your transferred toner directly onto a fabric. But first, let's quickly review some of our earlier options on this.

Your PostScript-speaking LaserWriter is ideal for generating initial artwork, especially any involving circular or arc-justified text. Done the "old way", you print up reverse images of your artwork and text on paper, and then photograph the image to in turn cut a silk screen master. This route gets handy when lots of identical shirts are needed.

We have looked at the franchised *Sublicolor* process in previous issues. It does seem to work for T-shirts, but helpline lore seems to feel that this is both a very overpriced and limited quality system using ancient *Gestetner* duplicator technology.

One product we looked at was *Transfer Magic*. You can buy this at your local cloth or notions store, or get it direct in quantity through *EZ-International*. This is basically some stickyback glop on a transfer sheet.

You first create a reverse paper copy of your toner image and then stick the Transfer Magic to it. Then you soak your Transfer Magic in warm water, ultimately dissolving all the paper out from underneath your still-stuck toner. Finally, you iron the result in place onto your T-shirt or whatever. Since the toner ends up between the cloth and the transfer glop, it will get protected from most scuffing and wear.

Yeah, it is completely washable, and looks no worse than most other iron-ons. Several colors are possible with multiple toners, so long as the registration is not critical.

On to the new goodies. You can now buy *heat refusable transfer* toner as intended specifically for iron-on fabric use. You use it as you would any other toner cartridge in your CX or SX based laser printer. A paper or polyester intermediate image gets created which in turn is ironed onto the cloth. While the cartridge costs are still rather expensive, your final per-image costs are well under a quarter. "Short" cartridges of much fewer copies are also offered.

This new toner should eventually be available from just about any refilling supply wholesaler. For now, three stocking sources include *Black Lightning*, *Lazer Products*, and from *Thompson and Thompson*.

Compared to those iron-on fabric ribbons for dot matrix printers, you should end up with higher quality; a faster and quieter printing; and lower costs.

Black Lightning's newly released colors sure look good, especially the blue and the new true black. Their original black was more of a gray. Free samples are available.

Outside of a few quick T-shirts, I have not yet had a chance to fully check this stuff out, but its obviously something that is sorely needed. This refusable transfer toner should also be useful for low end hacker prototype printed circuits and dialplates as well.

Other artsy-craftsy stuff should also eventually happen, possibly including glass etching, sandblasting, Batik, or wooden plaques. The printed circuit stuff is a big topic with me, so please keep in touch on this.

What is This Month's PostScript Utility?

I thought we could have some fun with PostScript fractals this month.

There's a new endeavor known as *Chaos Science*. Actually, it's just a loosely knit collection of all sorts of unusual old and new stuff, much of it only weakly related if at all. As a

general rule, if it keeps a math freak off the street at night and if it looks pretty in an alien sort of way, then it can be called chaos science.

Since chaos science is futuristic, utterly and totally bizarre, and more than eminently hackable, it should be of foremost interest to most of you *Computer Shopper* readers.

Figure four will give you a quick summary of a few main resources in Chaos Science. You will probably want to start with James Gleick's *Chaos – Making a New Science*. This one is a thorough and easy-to-read but occasionally shallow intro.

The horse's whatever document, of course, is Mandlebrot's *Fractal Geometry of Nature*. While more than pretty to look at, this text is pretty nigh unreadable and is laughingly and pompously egocentric.

The book I am currently the most impressed with is the new *Science of Fractal Images*, especially in chapter five. A good summary did appear in the January 1988 *Byte* under the title *A Better Way to Compress Images*.

One of the things that struck me about this great book and the *Byte* summary article was that all the sample BASIC and C programs were forever making those complicated translate-rotate-scale transformations. All the kind of stuff which *PostScript* handles free, invisibly, routinely, and automatically all of the time. Contrary to those dire negative comments early in the red book, PostScript should be *nearly ideal* for fractal geometry uses, along with most anything else involving chaos science.

To test out this theory, I have translated the original code into the PostScript proc of figure five. This gives you *the fern*, the second most popular fractal image of all time. The *Mandlebrot Set* is numero uno. I am saving it for later.

Even though there is an awful lot of slop left in the code, I was quite pleased with both the final results and its speed under PostScript. I elected to create a custom probability table once ahead of time, rather than making lots of slower individual probability selections on the fly. Providing for 128 entries in this table, rather than the expected 100, does yield us some additional speedup and simplification. A lot more cries to be done.

When you are experimenting with the fern on your own, be sure to change the *numdots* variable to something in the high thousands for your original fast printing samples that will image in a few seconds to a minute or two. Later on, you could increase *numdots* as high as you need to for top quality final images that might take hours to print in their larger sizes.

One big gotcha: The six element matrices used by the original fractal people and PostScript differ somewhat. To get from one to the other, simply *interchange the second and third values in the array*.

There are some amazingly bizarre things about the figure five fern image. First and foremost, larger sizes of this fern original end up hauntingly beautiful. In fact, I've never seen *any* PostScript image *anywhere* that even comes close.

Second, the smaller pieces of the fern are definitely *not* a smaller, or *self-similar* copy of your whole picture. Unlike ordinary pictures, as you magnify, additional unique detail appears. But that is what fractal stuff is all about.

Third, although an apparently random process is used that picks four tasks having random probabilities in random orders, you'll always end up with the *same* final image! This will happen regardless of which random numbers get used in which order.

This is a stunning example of a *strange attractor*, otherwise known as the holy grail of fractaldom. As you just might expect, others are seeking out the possible strange attractors thought to underly the stock market prices and weather.

Fourth, believe it or not, the *entire* fern is coded as only 28 numbers in a small data array! There is no difference whatsoever between your fern and untold billions of other wildly different images except for these 28 numbers.

We thus have one absolutely mind-blowing picture compaction scheme going on here. For a full page PostScript fern, your data compaction can approach 300,000:1! Unfortunately, though, the image regeneration times do get out of hand with compaction levels that are this high. You can forget about real time. In theory, by going to just a few more numbers, we can create *any* image at all.

Fifth, and finally, this routine really does appear exactly like an authentic Black Spleenwort Fern. Now, Black Speenwort Ferns have been around for a rather long time, and they sure are beautiful, but nobody ever accused them of being very bright. Could the exact same process be used by the real fern to teach itself how to grow?

To me, the odds are overwhelming that this is the case. We thus appear to be tampering with some heavy duty stuff here.

Tellyawhat. For the second of this month's contests, just send me 28 numbers that create an image others would consider interesting. Or else, show me the absolute fastest routine you know of to set a single pixel in a final PostScript bitmap. This gets trickier than it sounds.

This fern code is an excellent clone buster. While it runs great on any real PostScript printer, it can and will give most, if not all, clones fits. So much for compatibility.

If you have any problems with this code (it has been verified many thousands of times by now), rerun the fern on a genuine Adobe PostScript printer.

Oh yes. If you do find the strange attractor that underlies stock market prices and send it to me, I'll be most happy to send you *five* of my money machine books instead of just one. In a fit of generosity, I might even consider a second *tinaja quest*.

We can wrap things up with our usual reminders that most of those sources I've mentioned appear in my *Names and Numbers* appendix and that you can get tech help and off-the-wall networking by calling or writing me. ❄

Don Lancaster's

ASK THE GURU

July, 1990

PostScript point rule
Apple to HP interface
Low cost color options
IID duplex printer test
Converting Adobe files

It sure has been a zoo around here lately. At any rate, I've managed to buy, beg or steal a whole pile of PostScript printers, and will be running bunches of long term real-world tests on them. Look for the *Hewlett-Packard* IID results this month, and details on the fast and exciting *QMS* Turbo 820 in my next column.

Adobe has announced PostScript level II. This combines faster speed, enhanced color processing, and an "internalizing" of most of the earlier improvements picked up along the way. Expect a new red book sometime this October. Meanwhile, you can contact *Adobe Systems* directly for more details.

We now do stock the revised *Red Book II*. Call (602) 428-4073 for any additional info. Besides full details on Level II, this 800 page update includes plenty of EPS and display PostScript info.

Apple is now offering a ROM upgrade for their LaserWriter NTX. Cost is in the $120 range. These new chips cure several of the grevious bugs in their flaky hard disk operating system, and improve both AppleTalk and emulations. More details after I have had a chance to test them out.

Expect some really fancy features on the newly introduced PostScript printers. Such as an auto three way serial network, and "fuzzy logic" that is swift enough to analyze the input data stream and select raw PostScript or switch in a suitable emulator. Thus, PCL data streams can automatically print as PCL, HPGL as HPGL, TEX as TEX, *Diablo* as *Diablo*, and, of course, *PostScript* as *PostScript*.

All this on the new PS-410 from *QMS*, that utterly and completely blows away the new Apple Personal Laserwriter NT. Besides being a much faster, considerably lighter, and far cheaper implementation of the same *Canon* engine, the PS-810 includes eight more fonts, a parallel port, a more powerful 68020 CPU and HPGL emulation.

The non-PostScript *Personal LaserWriter SC*, of course, is an outright joke that is far beneath any further comment here.

Several do-it-yourself *brand new* replacement hard coated drums are now widely distributed to the end user. These let you refill your SX cartridges dozens or more times at better than new performance. One of the leading suppliers is *CopyMate Products*, while others advertise in the *Recharger* trade journal.

Let's see. We are now shipping my *LaserWriter Secrets* book & disk combo, and we do have many hundreds of new downloads available through *Genie*. Write or call per the appendix for more info.

IID or not IID?

That is the question. Whether tis nobler to suffer all the slings and arrows of outrageously wimpy performance just to explore future duplex printing opportunities.

I've been thoroughly end-user and real-world testing the *Hewlett-Packard* Laserjet IID duplex printer lately. We'll start with the bottom line. This beast is disappointingly slow. The PostScript *copypage* operator does not work properly when duplexing. And the machine is far more frustrating to use than any of my other PostScript printers. The summary of the IID problems appears as figure one.

On the other hand, the duplex printing sure is a lot of fun to play with and dramatically lowers your scrap rate and increases employee productivity and morale. And no question that duplex will eventually be the *only* way to go for *Book-on-Demand* publishing.

And no question that I'll continue to actively use my IID as a production machine, warts and all.

Any duplex printer starts off with

1. Any duplex printer *seems* slow because each page counts double, since the backside has to print first, and because efficient pipelining only takes place on multi-page jobs.
2. When doing real-world duplex printing tasks, the IID really is as much as ten times slower than a NTX or a PS Turbo 820. See figure two.
3. Duplex copypage is fatally flawed, eliminating as much as 30 percent of the market for this machine and causing inexcusable slowdowns.
4. A SCSI hard disk is absolutely essential for serious duplex printing. This is conspicuously absent on the IID. So is shared SCSI comm.
5. The machine is extremely frustrating to use, caused by poor comm code, slow speed, excessive error trapping, and a flakey remote reset.
6. The font cache can be blown up and hung if you throw too fancy a series of jobs at it. Pixel line remapping is a no-no.
7. Cartridges are not conveniently refillable without major modification.
8. Bitmaps are difficult to access for reuse.
9 Minor problems: Illegible LCD display; no tray chaining; no handles; toner density not adjustable during a job; max baud rate too slow; excessive jams on parchment cover stock.

Fig. 1 – My critique of the duplex LaserJet IID.

a serious marketing disadvantage – it psychologically *seems* slower that a non-duplex printer.

This happens because: (1) Each output page takes twice as long to arrive since each counts double; (2) Page makeup times appear extra long since the back side prints before the front side does; (3) The *pipelining* you need for the maximum mechanical speed only will run full blast when printing three or more sequential pages; and (4) Extra time and effort can be needed to clear a page remaining in the switchback path.

In addition, a duplex printer also seems to eat toner cartridges twice as fast. Again, this is because each page counts double.

My IID speed testing results are shown in figure two. When running routine real-world code that does *not* use the duplex feature, the IID is slower than a *LaserWriter Plus* and annoyingly slower than the *LaserWriter NTX* or a *QMS PS-820.*

Which is about what you would expect when comparing a 12 Mhz 68000 against a 20 Mhz 68020. Especially when you allow for all of the inherent limitations of any cartridge based PostScript solution.

Unfortunately, when you try to run a real-world duplex job, this beast slows down even further. One reason for this is that the PostScript *copypage* operator doesn't work in a side-sensitive manner. Thus, if you are custom addressing brochures, you cannot simply erase and rewrite each new name. Instead, you must erase the *entire* back, the *entire* front, and redo each side of each page from scratch.

On my brochures, this defective *copypage* feature creates a 10:1 slowdown. And, yes, that is *after* you carefully pseudocompile and then optimize fast code specifically for any IID use.

Another reason for the slowdown is that there is no shared SCSI comm. Most users will buy their duplex printer for high volume and high productivity.

Preferably largely unattended by either a paid operator or a host computer. For these needs, a hard disk and a high speed shared SCSI comm are not luxuries. They are instead *absolutely essential.*

There's also some more minor IID problems. The liquid crystal display on the IID is totally illegible under all known lighting conditions and all viewing angles. The remote reset only works when it is not needed and blows up otherwise.

The resident error trapper seems patently excessive. And, *Canon* has attitude problems on $5 toner cartridge refilling. Returnable bitmaps are not obviously available. There's no handles. Tray chaining is conspicuously absent for no apparent reason. Parchment cover stock jams far too often.

In testing, I've found the IID to be incredibly frustrating, compared to the NTX or the Turbo 820. The poorly done serial comm, improper remote control, slow speed, error hassles, and lack of SCSI all gang up to make their IID a maddingly infuriating printer.

You can also blow up and hang the IID font cache by throwing too many fonts at it too fast. This with Book-on-Demand files which print just fine on the NTX or 820.

The *Processing Data* message on the LCD display goes on long after the machine has hung itself and is out playing in the tulips. Most IID blowups are usually obvious only after several infuriating hours going by with no new output.

At this writing, though, the IID seems the only mainstream duplex PostScript game in town. And the benefits of double sided printing are *very* impressive. Exasperating or not, my own IID does run nearly 24 hours a day. Sigh.

1. *Meowwrrr*, a graphic intensive cartoon character from my *PostScript Show and Tell* series…

LaserJet IID	70 seconds
LaserWriter Plus	64 seconds
LaserWriter NTX	31 seconds
Turbo 820 PS	27 seconds

2. A three column page makeup including one figure, a header, and a footer, using my gonzo justify…

LaserJet IID	116 seconds
LaserWriter Plus	97 seconds
LaserWriter NTX	56 seconds
Turbo 820 PS	48 seconds

3. A fancy double-sided brochure including art cuts and one custom self-mailer address…

LaserJet IID	75 seconds
LaserWriter Plus	62 seconds
LaserWriter NTX	29 seconds
Turbo 820 PS	23 seconds

4. One hundred custom autoaddressed copies of the previous brochure…

LaserJet IID	7500 seconds
LaserWriter Plus	832 seconds
LaserWriter NTX	779 seconds
Turbo 820 PS	773 seconds

5. Book-on-Demand publish one 200 page *Ask the Guru* volume…

LaserJet IID	102 minutes
Turbo 820 PS	24 minutes

Fig. 2 – Some PostScript IID Click-to-Clunk times.

The new *LaserJet* IIID should be arriving just around the time you read this. It should be very much better than the IID, especially if I succeeded in convincing Adobe that their *copypage* is fatally flawed for duplex use and have shown HP that shared SCSI comm is absolutely essential for any duplex printing at all. Stay tuned.

At this revision, the IID has been replaced by the IIID, which in turn has been replaced by the IIISI. And we *still* are stuck with the original wimpy PostScript version, and *still* have duplex *copypage* hassles, and *still* have no hard disk or shared SCSI comm.

How Do I Use PostScript Mac Fonts on Another Machine?

Funny you should ask that. This is one very popular topic on the helpline. Genuine Adobe PostScript printer fonts are usually stored on the Mac in a special format that is downloaded over AppleTalk by using the *Adobe Font Downloader*. On the older Apple machines or on an IBM, the same fonts are downloaded in the form of a standard ASCII textfile, the start of which is in plaintext, and the remainder of which is PostScript *eexec* encoded.

Fortunately, there are several fairly simple methods that get between the two formats, once you know an insider secret or two.

Some of the early Adobe font downloaders flat out asked you whether you wanted the Mac or the textfile format. To use these, you just select textfile and have at it. This feature was dropped on later versions of this downloader.

The fastest and easiest method requires a NTX or a PS-820 and a hard disk. When downloading the Mac font, just select *disk* and let the font be written to the hard disk.

Note that all fonts written to hard disk are placed in a subdirectory called *fonts*. Thus, an *Optima-Bold* becomes *fonts/Optima-Bold*. A hard disk catalog and disk read routine is shown in figure three. These let you return the actual font in its textfile form to any host for recording.

The second method works with any PostScript printer, but requires a Mac application swift enough to look around the Mac for fonts not already downloaded. Just create a short text file, and select your host based font. Then, while printing, do an *option-F* or an *option-K* to capture the file to a disk. Finally, erase the portion of the short text file that is *not* the downloaded font.

What happens here is that the needed font first gets sent to the printer, followed up by your text message. When saved to disk, you will see a combination of a font textfile and your message to be printed by that font. Erase a few characters here, and all that remains is the actual font, ready to transfer to any other machine as a standard textfile, using the AFT *Apple File Transfer* or any comm scheme.

A Mac-based font conversion utility called #207 UNADOBE.SIT is available on *GEnie* PSRT. The downloading charge is around twenty cents, and you can get your voice connect info simply by dialing (800) 638-9636.

Note that all the full top secret details of Adobe's font files appear in their *black book*, that is otherwise known as their *Type I Font Format* book. By one of those astoundingly insidious coincidences that seem to infest this column, I just happen to have a few of these on hand if you need one.

How Can I Interface Apple to H-P?

There's a lot of misinformation kicking around over interfacing a

```
% Copyright c 1990 by Don Lancaster and Synergetics, 3860 West First
% Street Thatcher, AZ. 85552. (602) 428-4073. All rights reserved.
% Personal use permitted so long as this header remains present.
% Write or call for our free PostScript insider secrets brochure.
% Free PostScript helpline 8-5 weekdays, Mountain Standard Time.

% UTILITY #1 -- A HARD DISK FONT LISTER

% Lists fonts downloaded to a NTX or 820 hard disk back to host.

/scratchstring 30 string def
/str (X) def

800 {37 sin pop} repeat  % optional short delay

/filenameheader (%disk%fonts*)  def % for Apple NTX old (1.0) ROM
/filenameheader (%disk5%fonts*) def % for QMS PS-810 Device #5

filenameheader {scratchstring print flush 80 {37 sin pop} repeat
(r) print flush 100 {37 sin pop} repeat

0 1 scratchstring length 1 sub {scratchstring exch 32 put} for
pop} scratchstring filenameforall

quit

% UTILITY #2 -- A HARD DISK FONT READER

% Reads any file on a NTX or 820 hard disk and returns it to host.

2000 {37 sin pop} repeat % optional time delay

/strrx (X) def

/rdfilename (%disk5%fonts/Benguiat-Book) def  % change this line

rdfilename (r) file /myworkfile exch def

{myworkfile read { strrx exch 0 exch put strrx print
flush 5 {37 sin pop} repeat }{myworkfile closefile quit} ifelse} loop
quit
```

Fig. 3 – Two hard disk font reading utilities.

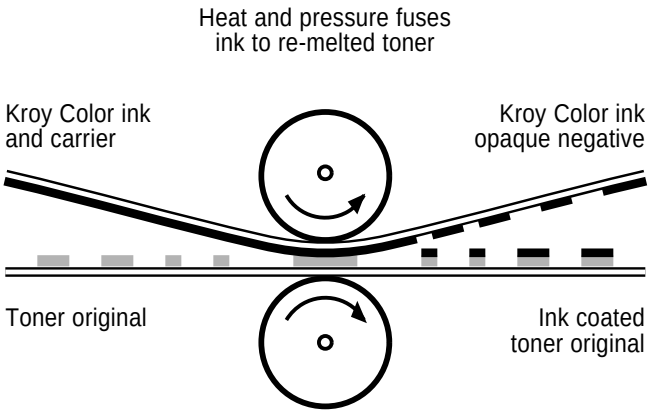


Fig. 4 – How the Kroy Color process works.

traditional Apple IIc, IIe, or IIgs computer to those *Hewlett-Packard* Laserjets, duplex or otherwise.

Contrary to popular belief, any LaserJet works just fine with any old Apple, once you resolve a fundamental detail or two.

Because of some ancient history involving a non-ASCII screen code, many Apple applications and most early Apple software output *high* ASCII text, consisting of seven data bits and a digital one.

The HP machines, instead insist on a full eight bit true ASCII word. Should you simply connect the two together, you'll get some Swedish-looking results when using PCL and nothing at all under PostScript. This happens because you are outputting high ASCII characters and trying to interpret them as low ASCII.

The cure is simple. Just use seven data bits, that extremely obscure *space* (or zero) parity, and one stop bit at the Apple end.

On your Super Serial Card, use the [I] 7 P software command. From *AppleWriter*, use [O]-J 1 9600, 7, S, 1. Or otherwise guarantee that your output text stream consists of low ASCII characters.

Unlike Mac and IBM systems, a printer driver is the responsibility of each individual application on the older Apple machines. Thus, you have to access each software package you have and customize it for the HP machines. This is easily handled in *AppleWorks* by using the *Custom Printer* feature.

To output any occasional high ASCII character in PostScript, just use the *reverse slash* technique as was detailed in the red book. For instance, the \261 gives you an em dash, while the \267 produces a bullet or large dot.

The simplest way to pick up a space parity on a IIgs is to use a Super Serial Card. This is also essential for proper *AppleWriter* use, especially when pseudocompiling.

Note also that there is an excellent PostScript *Imagewriter* emulator on the IIgs system master disk under the filename SYSTEM.MASTER/APPLETALK/IWEM. Which is a standard textfile that can be easily moved over to Mac, IBM, or any other platform. To invoke IWEM, use a _WBJ_ command string. Or else rearrange all the scenery to suit yourself.

Give me a call on the helpline if you need further assistance on any of this.

Tell Me All About Low Cost Color

Yes, you can now go out and buy full color PostScript printers. And, PostScript level II now offers far more in the way of special color halftone screens and powerful new color operators. But these remain high end solutions with high per-page copy costs that may not apply to you here and now.

What can you instead do simply and cheaply to add color to your routine PostScript output?

You just might like to check out the color copier abilities of your local quick print shop. Since color technology is moving so fast, you may be plesantly suprised by both the cost and the quality.

The simplest and most obvious first step on your own is to use colored papers. While the *Paper Plus* folks are my overwhelming favorite here, an outfit called *Paper Direct* has an interesting and refundable $15 kit of sample papers.

Yes, colored toners are available. Unfortunately, most users end up disappointed with both the price and the performance of color toner cartridges. Three sources of color laser printer toner do include *Don Thompson*, *Lazer Products*, and *Black Lightning*. Black Lightning also has special sublimination color toners for T-shirt printing.

There's a monumentally mismarketed and obscenely overpriced process kicking around which is known as *Kroy Color* or *Omnicolor*. This is simply a hot stamping foil converted so it sticks to a carrier. Toner is nothing but a mixture of black stuff and hot glue. Reheat it in contact with a hot stamp material, and the glue melts and grabs the real ink of the foil. Presto. Instant and impressive color.

This stuff works like a champ. Dozens of colors are available, with those metallic foils giving the best results. There's even a laminating film for menus and book covers.

Figure four shows you this process in detail.

Several sources for hot stamp foils do include *Transfer Print Foils*, *Lamart*, *USE Foils*, *Hoechst*, and *Maple Roll Leaf*. Yet others advertise in *Paper, Film, and Foil Converter* and in *Converting* magazines.

The *Foiled Again* newsletter from Transfer Print is a must have.

Some of these materials do seem suitable for direct toner use while some are not. The typical pricing in reasonable quantities is under one nickel a sheet, *provided you do your own conversion from bulk rolls.*

One source for the ready-to-use *Kroy Color*-like materials and fusion machines is *Lazer Products.* A hands-on demo on *Kroy Color* is included in my *Intro to PostScript* video.

You can also use tiny pieces of *Kroy Color* and either an iron or a model airplane wing fixer. This is a low cost but labor intensive way of sprucing up a letterhead.

Finally, I have run into a new process that can let you combine stunning full color with your own laser printing at sane prices. The *Photolabel* folks will take any color photograph and convert it into lots of self-stick labels of various sizes, with pricing as low as twelve cents each. These labels are *thin*, unlike the mess you get gluing ordinary photos onto a sheet of paper.

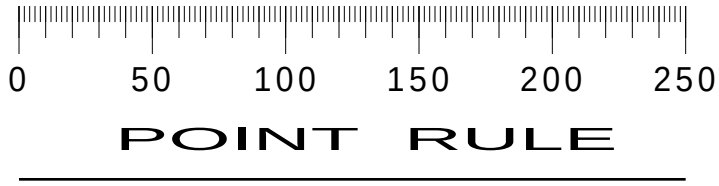
To use these photolabels, just laser print your whatever onto a colored paper or cover stock, *Kroy Color* the toner, and slap on a photolabel or two, and you have instant and custom full color. Beat out on a brick on your kitchen table.

Uses? Obviously house-for-sale or car-for-sale brochures. Custom or personalized calendars. The *Photolabels* would appear ideal for any fundraising projects that you want to keep both under local control and fully customizable.

But, why don't you just tell me instead? For our contest this month, just show me any tricks you are using towards any low cost color. There will be the usual *Incredible Secret Money Machine* book prizes, with an all-expense-paid (FOB Thatcher, AZ) *tinaja quest* for two going to the very best of all. Let's hear from you.

What is This Month's PostScript Utility?

Believe it or not, this is *Ask the Guru* column #65. I thought we'd rerun an "oldie but goodie" as our PostScript utility this month. Just in case you don't have column #27 or *Ask the Guru I* handy, figure five shows you a convenient PostScript point rule.




```
%  Copyright c 1990 by Don Lancaster and Synergetics, 3860 West First Street
%  Thatcher, AZ. 85552. (602) 428-4073. All commercial rights reserved.
%  Personal use permitted so long as this header remains present and intact.
%  Write or call for our free PostScript insider secrets brochure and our
%  product list Free PostScript helpline 8-5 weekdays, Mountain Standard Time.

%  A point ruler handy for layout and design. It prints diagonal on the page so that
%  it fits on one sheet yet is full length…

-30 212 translate -125 rotate

/xstrt -776 def /vstrt  200 def /tick1 -6 def /pos xstrt def 436 {pos vstrt moveto 0 tick1
rlineto /pos pos 2 add def 0.20 setlinewidth stroke} repeat  % 2 tick

newpath /pos xstrt def /tick2 -12 def 88 {pos vstrt moveto 0 tick2 rlineto /pos pos 10
add def 0.20 setlinewidth stroke} repeat  % 10 tick

/pos xstrt def /tick3 -18 def 18 {pos vstrt moveto 0 tick3 rlineto /pos pos 50 add def
0.20 setlinewidth stroke} repeat %  50 tick

% point numbers (5 spaces between each)
newpath /Helvetica findfont [ 12 0 0 13 0 0 ] makefont setfont /vstrt
vstrt 32 sub def /xstrt xstrt 12 sub def xstrt vstrt moveto -688 168 moveto
1.66 0 (100     150     200     250     300     350     400     450     500     550
600     650     700     750     800     850) ashow
-734 168 moveto 1.66 0 (50) ashow -780 168 moveto (0) show -465 146 moveto
/Helvetica findfont [40 0 0 15 0 0 ] makefont setfont (POINT    RULE) show

% bottom line
newpath -776 135 moveto 870 0 rlineto 1 setlinewidth stroke

showpage quit
```

Fig. 5 – A PostScript Point Rule.

While PostScript can use any unit of measurement you care to, most people most of the time use an approximation to a printer's *point* of 72 units per inch.

A point rule, of course, is your foremost tool for layout or changes. The one I've shown you here prints diagonally on the page. When you print this on cover stock, cut it and laminate it in plastic, this gives you the full page length ruler without needing to use extra long paper. Printing on a diagonal also averages out the paper stretch, so your rule should end up fairly accurate when used vertically or horizontally.

If you get a chance, run this point rule on a *Lino* for your maximum sharpness. But the results are certainly useful at 300 DPI.

Remember that most papers will stretch a percent or even more with humidity and use. So don't expect super accuracy over long distances, unless you print on a more stable material.

PostScript, of course, is ideal for creating rules for just about any measurement task, particularly for oddball scales. The weirder the scale, the more likely you can create some sort of specialty or custom product out of it at low cost.

Weaver's reed gauges would be one obvious example of this sort of thing. There are lots more.

Most of the PostScript routines you see here and in my *LaserWriter Corner* column are now available on *GEnie.* Just use M835 as your top secret access password. ❄

Don Lancaster's

ASK THE GURU

August, 1990

Papers and humidity
Generating histograms
More on duplex printing
HP PostScript Cartridges
QMS Turbo PS820 review

A month has passed, and I have been continuing my in-house testing of that *Hewlett Packard IID* duplex printer.

As we saw in the last column, the IID is inexcusably frustrating and annoyingly slow. It provides lousy serial comm, totally lacks absolutely essential SCSI hard disk support, and has a monumentally stupendous flaw in that its PostScript *copypage* operator is not side sensitive.

But none of this matters in the least. I run this machine virtually 24 hours a day, eagerly logging more time on it than most of my other printers combined.

Why? Because their new duplex printing process itself is that fantasmagorical.

Self-collating *duplex* or double sided printing very dramatically increases your throughput, your product quality, and your overall productivity. All the while sharply reducing paper jams and slashing your scrap rate. And the very few jams and mixups that do remain trash only a single quickly repaired page, rather than destroying the back sides of an your Book-on-Demand published volume.

Employee morale on a duplex printer is also ridiculously better. They actually fight to be able to use the machine.

I did find some more ongoing IID problems, though, and sorely wish this wasn't the only game in town. If anything, that *copypage* problem is turning out much worse than I first expected. To do something simple and mainstream like a pile of auto-addressed brochures, you have to erase the *entire* back, the *entire* front, and then rebuild the *entire* back and finally the *entire* front. At least a 10:1 slowdown, even *after* compiling. Just to change to the next name and its address. Really dumb.

Your choice of toner density and papers gets crucial for successful duplex work. Especially if large black areas exist on the backsides. Running a few blank pages through your machine the first thing in the morning is also a good idea.

It is super important to inspect the bottom (orange) roller daily, by flipping down that seldom-used green hatch at the extreme rear. This bottom roller *must* be kept spotlessly clean. Use a toner pickup rag or Q-tips after the fuser has cooled off. I also do suspect the entire fuser assembly should be replaced or rebuilt every 35,000 copies or so.

What do I really want? I want a duplex PostScript printer which will *totally unattended* by either a person or a host computer reliably Book-on-Demand publish a large pile of volumes overnight, fully stacked, collated and ready for final binding. And I strongly suspect that meeting my needs will meet those of a lot of others as well.

About all of those new low end PostScript printers. I was hoping to be able to report on the *Apple* Personal LaserWriter NT this month, but mine was DOA and I haven't seen it since. Two conspicuous features of this printer do include PostScript internal handling of IBM screen fonts and greatly improved envelope options.

From here, it does look like the *Hewlett Packard* LaserJet IIP with a PostScript cartridge produces the best text image quality (owing to a dot modulation technique), and that the *QMS* PS410 offers far and away the most features with the fastest operation.

1. - Any PostScript cartridge solution is inherently wimpy and slow, when compared against a RIP built from the ground up to speak interpreted PostScript as its native internal language.

2. - This simple code sequence causes a fatal blowup…

   100 200 translate 5 dup scale
   10 {53 45 {0.12 0 360 arc stroke 0} setscreen} repeat
   showpage quit

3. - The *copypage* operator is not side sensitive when duplex printing, causing severe problems on the IID.

4. - The IID page makeready times are three to four times longer than those of the NTX or the PS820. Other machines are correspondingly slow. The lack of a duplex copypage can cause another 10:1 or higher slowdown for certain files on the IID, especially when used with autoaddressed brochures.

5. - The serial comm is flakey and blows up when you need it most. The external soft resets only seem to work when they are not needed.

6. - This cartridge can choke on more complex jobs that run just fine on the NTX or the 820, especially when the *autojamrecovery* is active. Undocumented error #24 is a fatal blowup. You'll see lots of these.

7. - The *apathforall* operator is not implemented in the standard manner, making capture and recording of font paths extremely difficult.

8. - There is no apparent hard disk support, nor any provision for either shared or direct SCSI comm.

Fig. 1 – My critique of the HP PostScript cartridge.

I presently would rank the Apple Personal LaserWriter NT being *not acceptable*, owing to Apple's ongoing refusal to provide end user repair manuals or overnight VISA/800 replacement parts like HP does.

It is not at all clear to me why Apple continues to force all of their end users to buy HP manuals and HP parts to keep all of their Apple printers alive. It must be some sort of a secret rebate policy.

Let's see. Our great PostScript RoundTable on GEnie has really taken off. As of this writing, we've had nearly 2000 library downloads, and Adobe has promised us strong future support.

GEnie recently reduced all their rates and now have offered a greatly improved new *Star* service as well. There are no longer any penalties for 2400 baud operation.

That new *Red Book* should be out by this time, including details on the new Level II PostScript. New stuff includes the FAX compaction, forms, filtering, better color, and more precise halftones. I am now stocking these for you. Write or call for more info.

A reminder here that I do have a free desktop publishing insider's secrets brochure waiting for you if you call per the appendix.

How Good are Those PostScript LaserJet Cartridges?

There are presently two genuine Adobe PostScript cartridges newly available for the LaserJet series of printers. The older one is sold by *HP* as part #33439P and is intended for use on the IID, IIP, and LaserJet III. The newer one is sold directly by *Adobe* and is intended specifically for the older LaserJet II models. The street prices are approaching $300.

Together, these two will largely eliminate any need for host based PostScript solutions or for a third party imitation PostScript clone.

Figure one summarizes some of the features of that HP cartridge. There are several problems, all of which are more or less livable.

For openers, *any* cartridge based PostScript solution has to be an inherently wimpy one, compared to designing a RIP from the ground up to speak interpreted PostScript as its native language.

I've only found one fatal flaw to date, and this involves a curious interaction between their *setscreen* and *arc* operators. This is way out of the mainstream for most users. Full details appear in our GEnie PSRT downloads #144 and #148.

There does seem to be other times when the undocumented PS ERROR #24 *fatal blowup* happens. Certain complex files that print properly on the NTX or PS820 can blow up the cartridge, especially when the memory hogging but extremely useful *jamrecovery* feature is active.

As we saw last month, there is a really bad flaw in *copypage* in that it is not side sensitive. This is a real IID killer, but is of no consequence on the IIP or the III.

I also found that the serial comm blows up whenever you need it the most. Thus, your ability to do a remote reset is only available when you do not really need it. Sometimes the serial comm will hang so bad that a total power down is the only possible recovery.

We did see some speed trials last month. The HP cartridge on the IID runs at 1/3rd to 1/4th the speed of the NTX or the PS820 when running normal jobs; for a duplex printing where a side sensitive *copypage* is needed, the speed will drop by a further factor of at least ten.

Naturally, a HP printer without PostScript is an outright joke, so one of these two genuine Adobe solutions is essential.

While many sources for these cartridges do appear right here in *Computer Shopper*, you'll find other good ones in the *Computer Reseller* trade journal. A stocking source is *Don Thompson*.

What do You Think of the QMS Turbo PS820?

I have also been testing the *Turbo PS820* from *QMS*. This is by far my overwhelming favorite PostScript laser printer, and some of its better and worse points are shown you in figure two.

In general, anything that the unexpanded Apple NTX does, the 820 can do better and friendlier. There's more resident fonts in some extra *Helvetica* variants. There are

The GOOD stuff–

The PS820 is far and away my favorite machine. Anything an unexpanded NTX can do, the PS820 can do faster and better. The current PostScript is version 51.7.

There are switchable and chainable dual paper trays; extra resident fonts; better mode switching; improved hard disk operation with simpler cables and eight device addressability; Diablo (Daisywheel), PCL (LaserJet), and HPGL (Plotter) emulations; upgraded jam recovery; and more comm options.

QMS has absolutely outstanding direct end user service and support. They are a smaller company that obviously cares about their customers.

The BAD stuff–

The PS820 speed is only marginally faster than the NTX, typically only five percent or so. The memory supplied is only 2 Megabytes. More would certainly be useful. The hard disk system can be further improved.

The QMS "list" and "street" prices traditionally have been much closer than with Apple. You have to shop around to get a great price.

The PS820 could be dramatically improved by providing a duplex printing option, additional memory, and direct or shared SCSI communications.

Fig. 2 – A summary of the QMS Turbo PS820 laser printer.

dual trays which can get used for letterhead-envelope, firstpage-copy, or simply chained together for more paper capacity. The jam recovery is better. The mode switches are much easier to change. The hard disk is improved and allows separate addressing of eight devices.

Disk cables are less cumbersome, and the hard disk itself is optionally self-terminating.

Besides those usual *Diablo* and *LaserJet* emulations, you also have a *HPGL* emulator for plotters.

Their genuine Adobe PostScript used is version 51.7, compared to 51.8 on the latest NTX upgrade.

Above all, the QMS product support and their PostScript help is infinitely better than Apple's. QMS is a smaller company that cares about their end users and works directly with them. They are also very good at attention to detail and listening carefully to what the final buyer really wants.

The negatives here are minor. While the PS-820 is probably the "best" PostScript laser printer of its generation, it's not a giant killer. It has only modest advantages over the NTX. When you use both hard disks and the new 3.0 ROM upgrade on the NTX, the PS820 is typically only five percent faster.

Traditionally, the differences between QMS "street" and "list" prices have been a bunch closer than with Apple, so it takes some shopping to pick up the PS820 at a really great bargain price.

Now, if only we could bolt some duplexing stuff on the PS820 frame (the holes are already there!), use some high speed SCSI comm and add extra memory and HP's dot modulation, we would really have something.

The bottom line: The PS-820 is my main machine for just about everything, including an overnight Book-on-Demand production of all my *LaserWriter Secrets* volumes.

A far more detailed review of the PS-820 appears as PSRT Library #150 on GEnie.

Tell me all About Paper and Humidity

Things are usually rather dry out here in the Arizona desert, except for our late summer monsoons. At that time, our paper jamming and binding problems skyrocket. Particularly on the back side feeds on non-duplex printers.

I've tried all sorts of schemes for uncurling paper, involving everything from refrigerators, microwave ovens, heavy long term clamping, back-rolling, and even a hot tub. Finally, an old line printer told me the truth. The *only* way to uncurl paper is to prevent it from curling in the first place.

You'll find the best laser printing will take place between 25 and 45 percent relative humidity. Note that these values are considerably *lower* than what a printshop will usually recommend.

Humidity is normally measured using a thermometer-style device known as a *hygrometer*. You can buy high quality hygrometers through *Abbeon-Cal* or from *Heathkit*, while surprisingly low cost (Would you believe $3?) units are available from the *Klockit* folks. If you are at all serious about desktop publishing, you *must* have a hygrometer on the wall above your laser printer.

A second ploy is to use a fake milk crate for your scrap paper. A glance at this box will immediately tell you how much trouble to expect. At 30 percent humidity, the pages will all lie flat and neatly collated. At 40 percent, there'll be some disarray. At 50 percent, you'll see noticeable curl starting.

Finally, at 60 percent or higher, you will start getting the *Milk Crate from Hell* effect, with the papers all desperately clawing all over each other to see who can get out first.

Your stock papers are best stored unopened and flat on steel shelving. Open papers that are to be stored for a long time should be well wrapped, preferably in a *plastic* or film shrink wrap.

Simple methods of controlling humidity are to use fans, and to make sure such things as clothes dryers, dishwashers, humidifiers, evaporative coolers, and any other moisture *sources* are not adding to the problem.

Most air conditioning also tends to dehumidify; that is what the condensate pump is all about.

In extreme cases, you may want to add a real dehumidifier to your work area. These are available from such sources as *Grainger* or *Sears*.

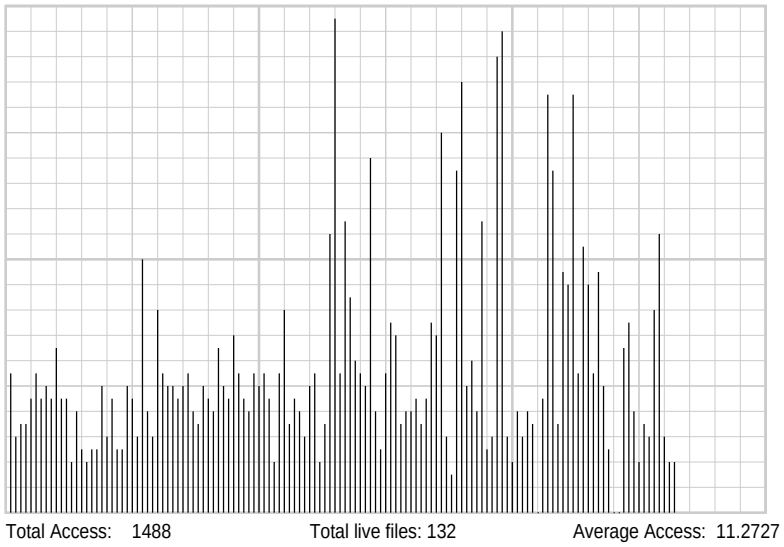


Fig. 3 – A PostScript histogram or "popularity poll".

Costs start at $200, and the smaller units need ordinary 110 volt ac line power. Access to a floor drain or a garden hose heading outside is often required.

What is This Month's PostScript Utility?

Please do not call PostScript a *page description language*. To do so is a gross and demeaning insult, and roughly comparable to referring to UNIX as a *checkbook balancer*.

PostScript is one totally *general purpose* computer language that can easily hold its own against any modern contender. In fact, dirtying otherwise clean sheets of paper is a hopelessly restrictive and a very minor use of PostScript's truly astounding capabilities.

For most people, your PostScript computer is by far the most powerful and the most flexible one you own. Having the most available useful memory and far and away the cleanest architecture.

Unfortunately, some epsilon minus has inadvertently and stupidly slapped a "printer" label onto the outside of your best computer.

Think of your Mac or your 386 as a second rate and rinky-dink dumb terminal that gets attached to your "real" computer. And then do as much of your work as you possibly can directly in PostScript.

Our utility this month gives you an example of using PostScript as one general purpose computer language. What this utility does is accept *any* ASCII word processor, data base, or spreadsheet report in *any* format from *any* host source, carefully extracts only the essential information from that report, and then presents the interpreted results in a new and powerful form.

Er, let's get more specific.

I've gotten into some *histograms* lately. They are a fascinating subject in themselves, and do have a wide variety of uses. As figure three shows us, a histogram is a bargraph style popularity poll for some members of some group.

For instance, you just might be running a video rental store, and want to know how each particular title is doing, and or whether that cross-genre *Godzilla versus the Night Nurses* classic has peaked in its popularity.

Or you might be publishing some PostScript halftone images and want to make sure each available gray shade does the best possible job, similar to the dodging and the burning of photo darkroom work.

But this particular histogram takes a standard listing of our brand new GEnie PSRT library and then analyzes its performance. This is an especially powerful predictor of the future. This code also downright addictively can fulfill the innermost secret dream of most authors – to be able to watch over the shoulder of their readers as they peruse your material.

Once you have a histogram, you may want to *filter* it to extract other information. Obviously, you could count up the total usage, the total entries, and then divide the two to get the average performance.

Figure four shows a filter I've used that "smooths out" the results into a new histogram. The sudden jump in user access around download #65 happened because we experimentally added user tutorials to the existing mix of free fonts and PostScript utilities. Obviously, the filtered histogram shows that this was a very good idea that produced almost a 60 percent increase.

Which is what histograms are all about – spotting new trends, then analyzing the past, and finally predicting the future.

Yes, you could simply average out your histogram results, much as some stock market people run a 10 day and a 30 day moving average. But it seems to me that any near neighbors should "count more" towards a smoothing than any distant relatives. I chose to use a fancy filter here that I call a *seven file Gaussian weighted average*. A digital signal processing person will immediately recognize this as a low pass filter.

Say you have a string of values in the middle of your original noisy histogram. Call them *p, q, r, s, t, u,* and *v* in increasing left to right order.

The filter I am now using is…

$$s' = (0.2p + 0.5q + 0.8r + 1.0s + 0.8t + 0.5u + 0.2v) / 4$$

Thus, your new value s' is one fourth the sum of the old s, plus eight tenths of its immediate neighbors, half of its adjacent pair, and

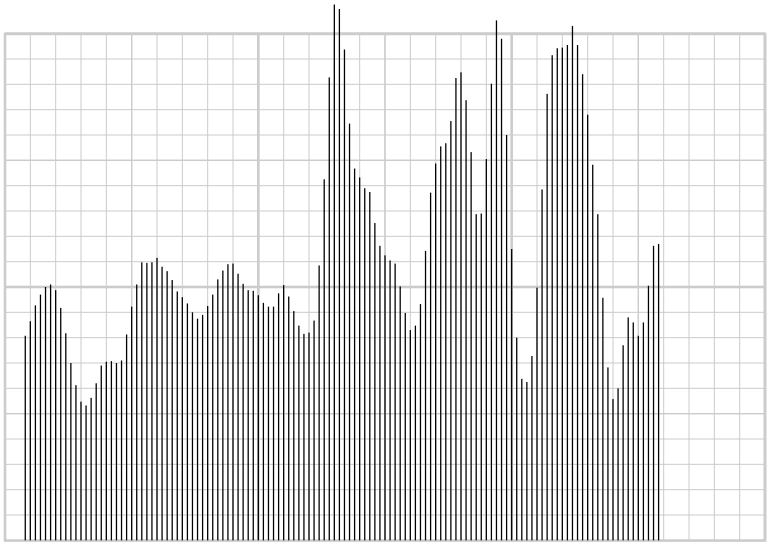


Fig. 4 – Filtered (smoothed) histogram shows trends.

two tenths of the next adjacent pair. The quarter comes about since .2 + .5 + .8 + 1 + .8 + .5 + .2 = 4.

This is just one of many possible filter functions. To filter, you decide what you want and then go for it.

At any rate, much of the needed histogram code appears in figure five, and you can get the whole thing ready to run as GEnie PSRT library download #146.

To use, you drop your raw report into the middle of the code. AppleWriter's WPL is absolutely ideal for this sort of thing and does it all in three keystrokes.

Then your original report gets scanned line by line. Each line next gets *qualified*, making sure that it contains needed data and otherwise rejects titles or other extraneous stuff. The qualification used in this example makes sure you have a 71 character line, verifies spaces in locations 5, 60, and 67; plus 0-9 numerals in 4, 59, and 66.

This validation is good enough that you can simply throw the log of your *entire* GEnie session at it, and it will automatically fish out all of the good stuff for you.

But watch that your validation does not do more than expected. An early version threw out authors of middle European descent because their names were too long!

Once validated, the data for each download is extracted and placed into one long array. This is done in such a way that up to 65536 array values are allowed, compared to the 510 element limit you might hit by building up an array directly on the PostScript stack.

In this example, the data we want is the number of download accesses made for each particular file. All of the rest gets ignored.

The noisy histogram then gets plotted using the *rubbergrid* graphing techniques that we've looked at in past columns and in my *Ask the Guru II* reprints. The total response, the number of files, and an average response then gets calculated and displayed.

A second plot then gets drawn, applying the Gaussian filter to the array, and generating the smoothed results. Rather than worry about filter *windowing* problems at each end, I just chomped them off.

PostScript variables let you adjust those horizontal and vertical scale factors, the histogram density, the number of blocks, and so on. Just rearrange all of the scenery to suit yourself. Any way you like.

Finally, your array and the key calculated values are returned to your host for a possible recording. Alternately, they could go on hard disk. With fancier routines, this lets you later see how your histogram changes through time.

Do note that a two-way comm channel is required for this final log feature. Naturally, you should *NEVER* use PostScript if you do not have your two-way comm channel alive and active.

Most of those other PostScript routines you see here and in my *LaserWriter Corner* column are now available on *GEnie*. Just use M835 (roundtable) or M836 (the download library) for your top secret access passwords. For email messages, you can reach me via SYNERGETICS on the M200 electronic mail page.❄

```
%  Copyright c 1990 by Don Lancaster and Synergetics, 3860 West First Street
%  Thatcher, AZ. 85552. (602) 428-4073. All commercial rights reserved.
%  A fully expanded and documented version appears as GEnie PSRT #147.

/quadpixel {transform 4 div round 4 mul itransform} def /setgrid {save /rubbersnap exch def
quadpixel /size exch def quadpixel exch quadpixel exch translate size dup scale} def /drawlines {72
300 div lw mul size div setlinewidth /hposs 0 def #hlines gs div 1 add cvi {hposs 0 moveto 0 #vlines
rlineto stroke /hposs hposs gs add def} repeat /vposs 0 def #vlines gs div 1 add cvi {0 vposs
moveto #hlines 0 rlineto stroke /vposs vposs gs add def} repeat} def

/showgrid{ seegrid {gsave /#vlines exch def /#hlines exch def 106 45 {pop pop 0} setscreen 0.9
setgray /gs 1 def /lw 1 def drawlines fat5 {/gs 5 def /lw 3 def drawlines} if fatter10 {/gs 10 def /lw 5
def drawlines} if grestore}if} def /fat5 true def /fatter10 true def /seegrid true def /line1 {0.06
setlinewidth} def /mt {moveto} def

/vertblocks 25 def /horizblocks 30 def /blocksize 12 def /vertscale 0.5 def /horscale 0.2 def /xxpos
125 def /yypos 100 def /yspacing 50 def /totalnums 1000 def /reporttohost true def /reportdelay
2000 def 300 string /buffer exch def

/gottoend? {dup (%!!%) search {pop pop pop exit }{/str exch def validate grabvalue} ifelse } def
/grabvalue {{str 60 7 getinterval cvi hsa exch aindex exch put /aindex aindex 1 add def} if} def

/validate {pop str length 71 eq {true}{false} ifelse {str 5 get 32 eq str 60 get 32 eq and str 67 get 32
eq and str 4 get dup 0 48 add ge exch 9 48 add le and and str 59 get dup 0 48 add ge exch 9 48
add le and and str 66 get dup 0 48 add ge exch 9 48 add le and and}{false} ifelse} def

/generatehisto {save /snap exch def totalnums array /hsa exch def /aindex 0 def currentfile {dup
buffer readline {gottoend?} {exit} ifelse } loop pop pop hsa 0 aindex getinterval /hsa exch def
processhsa snap restore } def

/processhsa {/sumtotal 0 hsa {add} forall def /avg sumtotal hsa length div def save /snap1 exch def
xxpos yypos blocksize setgrid horizblocks vertblocks showgrid line1 0 1 hsa length 1 sub { /posn
exch def posn 1 add horscale mul 0 moveto 0 hsa dup length 1 sub posn sub get vertscale mul
rlineto stroke} for snap1 restore save /snap1 exch def xxpos yypos vertblocks blocksize mul add
yspacing add blocksize setgrid horizblocks vertblocks showgrid line1 3 1 hsa length 4 sub {/posn
exch def posn 1 add horscale mul 0 moveto 0 hsa dup length -2 sub posn sub get .2 mul hsa dup
length -1 sub posn sub get .5 mul add hsa dup length 0 sub posn sub get .8 mul add hsa dup
length 1 sub posn sub get 1 mul add hsa dup length 2 sub posn sub get .8 mul add hsa dup length 3
sub posn sub get .5 mul add hsa dup length 4 sub posn sub get .2 mul add 4 div vertscale mul
rlineto stroke} for snap1 restore} def

%  //// demo - remove or alter before reuse. ////

generatehisto

150     PS820REV.TXT                X SYNERGETICS       900830      26460     4     1
        Desc: QMS Turbo PS820 Review
148     HPCARTXX.PS                 X G.REISEL          900827       6300     4     4
        Desc: Fatal HP Cartridge Blowup Demo

                (continues for rest of listing; MUST exactly match download format)

  2     LANDSKPE.PS                 X DANTECH           900630       1260     6     6
        Desc: Landscape selector LS1
  1     BACKWARD.FNT                X SYNERGETICS       900628       1260    11     4
        Desc: Reverse ffont LS1

%!!%  % required marker
showpage quit
```

Fig. 5 – PostScript utility to generate histograms.

Don Lancaster's

ASK THE GURU

September, 1990

PS Perspective lettering
IId duplex printing bug
Apple's new CD releases
LaserWriter repair manuals
Some telecomm adventures

Apple Computer should be strongly congratulated for starting up their toll free *Customer Assistance* number at (800) 776-2333, and usable during normal business hours. Here is where you are supposed to go *after* you get the usual runaround or misinformation from your local dealer.

User group ambassadors have received a new CD ROM #110 titled *October 1990 Product Training*. Dealers have gotten their new *References and Presentations 5.0* CD. And all the developers have received their new *Night of the Living Disk* CD. These are all crammed full of info on all the recent product intros, primarily as *Hypercard* stacks. Contact your local user group to borrow these disks. As usual, you can find your local user group simply by calling (800) 538-9696.

Apple has newly announced a rather strange Apple IIe plug-in card for the Mac LC. Curiously, this wondrous beast accepts only the IIe (65C02 based) software and *not* IIgs software and runs *only* on the LC. The ad copy for the data sheet on this one is ludicrously out of date. There's a few hundred thousand IIe programs available, not the mere 10,000 claimed. GEnie alone stocks 16,000 or so of them. But calling this their "world's largest collection of personal computer software" quite obviously has not been true for a rather long time.

I was both rudely surprised and utterly apalled that *Apple Computer* seems to be aggressively marketing a $9000 product which is inherently defective. I am referring to their old *LaserWriter Mailing List*. The sample I tested included nearly 20 percent outright duplicates, combined with an unacceptably high level of typos and otherwise undeliverables.

Worse yet, the nixie rate on this list now exceeds an intolerable 25 percent. Most of these came back with the dreaded *Forwarding Order Expired* stamp, meaning that all these names are ancient.

Oh, yes. The response rate to a well tested and targeted flyer was also totally negligible.

These dupe and nixie levels exceed allowable industry standards by a factor of ten or more, making this offering not only less than useless, but a downright dangerous time and dollar sink. Obviously, this list has never been cleaned and never tested. Certainly, the usual industry standard six month test remailings are not being done on an ongoing basis. Not at all.

Avoid this turkey unless you can get written guarantees that Apple has cleaned up their act.

As of this Guru revision, very serious problems remain with the Apple lists. It seems that Apple refuses to believe how badly their list brokers are lying to them.

There are lots of new PostScript books out. Adobe has issued their *Red Book II* that includes full details on PostScript Level II. Glen Reid has released his brand new "gray" book *Thinking in PostScript*. And I have recently expanded my *LaserWriter Secrets* book/disk combo, which is an upgrade of the stuff you've been reading in the *LaserWriter Corner*. Yes, I try to keep all three of these in stock for you.

Where can I get a LaserWriter Repair Manual?

Why, through *Hewlett Packard*, of course. You certainly would not expect Apple to actually sell you the repair and maintenance manuals for all their own products, would you? How silly can you get?

As we've seen a number of times in the past, the *Canon* laser engines used by HP, Apple, QMS and others are pretty much the same animal for comparable machines. All that differs is the case arrangement and the scope and quality of all the support electronics provided. Thus, most mechanical parts and their repair procedures are identical between Apple, QMS, and HP.

Figure one shows you the HP service manual part numbers, their target HP printers, and their nearest equivalent Apple and QMS printers. Manual costs are typically around $30 with overnight delivery by way of VISA and an 800 number. Parts

HP MANUAL	HP PRINTER	APPLE PRINTER	QMS PRINTER
02686-90920 (CX Engine)	LaserJet I	LaserWriter LaserWriter Plus	PS800
33449-90906 (SX Engine)	LaserJet II LaserJet III	LaserWriter NT LaserWriter NTX	PS810 & Turbo PS820 & Turbo
33459-90906 (SX Engine)	LaserJet IID LaserJet IIID	– – – –	– – – –
33471-90904 (LX Engine)	LaserJet IIP	Personal LW NT	PS410

Fig. 1 – Use these HP Manuals for Apple LaserWriter repairs.

are also readily available, either individually or as modules.

Uh, whoops. H-P just recently increased all the prices of their manuals to the $100 range. You know how it is when you suddenly start selling lots of something. You have to raise prices to pay for the extra people in the shipping department, all of the new order takers, the extra annual reports, and so on.

Laser printers are actually fairly simple devices. If you can change the blade on a power lawn mower, you can easily perform 90 percent of your own LaserWriter maintenance and repair work. Thus, it is almost essential that you have these manuals on hand for any intelligent long term use of your LaserWriter.

Other sources for LaserWriter repair parts include *Don Thompson* and the dozens of advertisers who appear in *Recharger* magazine.

Does Telecommunicating Work?

A good question. I have been wondering that for a number of years. I live in a remote desert area where intrastate phone rates are outrageously high and no local BBS access nodes are yet available. But *GEnie* recently hired me as a sysop to try and put together the best possible PostScript BBS anywhere ever. So, I have only very recently gotten into telecommunicating in a rather big way.

The potential of telecomm is awesome. Instant email communications with anybody anywhere. Bypassing the post office and the message delivery services. Right now solutions to those real world problems that manufacturers and suppliers just as soon would not provide you with. Direct contact with end users worldwide with the same interests as yours. Untold zillions of free or low cost software programs. Prompt answers to any question on anything, asked at any time from any place.

But the reality of telecomm today is considerably less.

First, telecommunicating can be expensive. The only bigger surprise than your first $768.43 BBS phone bill is the $7,684.30 one your eighth grade son or daughter just racked up. Obvious cures are faster comm rates, more intelligent software, a thorough knowledge of who you are calling, and doing as much as you can off line.

Many BBS services offer some scheme for automated late night delivery that uses a script to check for all your mail, pick up selected downloads, read chosen messages, and so on. This can get done fully unattended and can be much faster and far cheaper than "being there." The *Aladdin* feature of *GEnie* is a typical example.

Second, telecomm can be klutzy. Many systems are still in the dark ages with limited text only formats. Some remain on klunky mainframes which slow down infuriatingly with added customers. Others provide maddingly slow delays when they change modes. There's too many busy signals far too often. Sorely needed are totally global searches and "You are here" functions.

Third, telecomm is very diffuse. You sure have to wade through a lot of garbage to get at those goodies. Such things as bulletin board messages are inherently inefficient for many uses much of the time.

Fourth, telecomm can lack device independence. Far too many files are available only in one particular format for one specific host.

Fifth, compaction is now a zoo. A compaction obviously will let you upload and download faster and need less BBS stroage. But there are dozens of incompatible formats, such as .ZIP, .SIT etc. Instead, your actual compaction used inside the BBS should be invisible to you, and *all* files should be available in a totally uncompacted form, along with your choice of any popular download compaction scheme, preselected on a global basis. No compaction should *ever* get uniquely linked to any particular download or any specific host.

This becomes an especially severe problem with PostScript fonts that are themselves device independent.

Sixth, and finally, there's some really strange denizens inhabiting bulletin boards. Especially anytime after 3 AM. A lot of these seem to

```
% Copyright 1990 by Don Lancaster and Synergetics. 3860 West First Street,
% Thatcher, AZ. (602) 428-4073. All commercial rights reserved. Personal use
% permitted so long as this header remains present and intact. This file is
% available on GEnie PSRT. PostScript secrets book+disk $29.50 VISA/MC.

% DUPLEX PRINTER PAPER AND TONER CONDITIONER AND TESTER

% Certain combinations of paper and toner may not print properly on a duplex
% printer, even if the combinations work fine when straight-through printing.
% This gets MUCH worse if the primary corona wire is the least bit dirty.

% The usual symptom is a blotchy, less dense, and occasionally "splattered"
% second side. Note that the "REAR" prints first and "FRONT" second.

% If your second side sheet "gets better" by sitting out in the open with a
% five minute half life, the problem is almost certainly a dirty corona wire.

% The apparent cause is the heating and calandering done to the paper during
% the first side fusion. When combined with a dirty primary corona wire, the
% electrostatics will change adversely.

% This tester simply ejects two totally black double-sided pages. It is normal
% for the first printed side of the first page to not be uniform.

% On the SECOND page, IF the second side print quality is worse than the
% first side print quality, try cleaning the corona wire with a Q-tip.

% turn on the duplexing if it is available

statusdict begin statusdict /setduplexmode known {true setduplexmode} if end

% draw a big black box

0 0 moveto 1000 0 rlineto 0 1000 rlineto -1000 0 rlineto closepath fill

% and show two double pages

copypage copypage copypage showpage quit
```

Fig. 2 – A PostScript duplex print quality tester.

prefer smoke to light, and many of them do have extremely short fuses and do seem to take umbrage over everything. I have yet to figure out how to deal with this.

There are many tens of thousands of BBS systems alive today. Many are reviewed right here in *Computer Shopper*. The biggies have most of the marbles though, since they often provide toll free local service nodes, and easily can command the highest quality uploads.

In fact, a BBS that charges but has free nationwide local access lines can end up considerably cheaper than a "free" BBS that does not.

Let's look at four wildly different examples for on-line service telecomm you might find of interest.

The first of these is *The Well*, the electronic arm of the *Whole Earth Review*. This one is heavy into new age stuff, alternate energy, home power, artisans working alone or on through that loosely knit "briarpatch" network, the environmental issues, and lots of those "small is beautiful" counterculture topics.

USENET is an international network that's usually available only through universities. Another name for it is *Anarchy* 101. This is far and away the greatest pirate cove anywhere in the known universe.

The time from when someone decides to keep something a secret to when a decrypted (and vastly improved!) explanation appears on USENET normally gets measured in nanoseconds. In several cases, the response time clearly has exceeded the speed of light.

My own favorite BBS is the *Dialog Information Service*. It costs $130 per hour, and is the *only* service that I gladly and willingly pay for. With care, Dialog can be your most outstanding telecomm bargain.

Naturally, it is not what you pay, but what you get. Dialog gives you instant world-wide research on any scholarly topic. One tiny and almost negligible corner of Dialog is called INSPEC, and provides only a paltry twenty million on-line references involving computers, electronics, and physical sciences.

Ferinstance, those of you who are following all my *Hardware Hacker* rantings over in *Radio Electronics* do know that we are heavily looking at *magnetic refrigeration*, a new cooling method that can be fifty times more efficient than freon.

From my just suspecting that something was up and the words "magnetic refrigeration" to having abstracts of the very latest eighteen worldwide key papers in the field cost $27 and took around twelve minutes. Done from a sand dune in the middle of the Upper Sonoran desert. Convenient, even.

Finally, *GEnie* is one large information utility widely regarded as having the best downloads and the best sysops (with one or two obvious exceptions) in the industry. Over 100,000 downloadable files are available within their hundreds of software libraries.

There are two sides to GEnie. The "free" or *Star Services* side costs you $4.95 per month and can give you unlimited and untimed access for email, stock quotes, movie reviews, hobby roundtables, and such.

The "computing" side gives you access to the software libraries and other higher value information at a baud rate independent $6 per hour.

There are bunches of computer and special interest boards. Any typical roundtable will feature an extensive public domain or shareware download library (The MAC board has around 25,000 files), a question-and-answer BBS having hundreds of replies in dozens of separately accessible topics, and real time conferences with name brand biggies in the field.

My own tiny corner of GEnie is called PSRT, short for the *PostScript Roundtable* found on page 835. Come visit sometime. Besides PostScript and desktop stuff, we're now also heavily into the hardware hacking, midnight engineering, meowwrrr pfftting, and tinaja questing.

So, What is Better?

Well, since you asked, how about this heresy… Contrary to popular belief, the US post office is far faster and ridiculously cheaper than telecommunicating by email.

That's right. A CD ROM mailed once a month has an effective baud

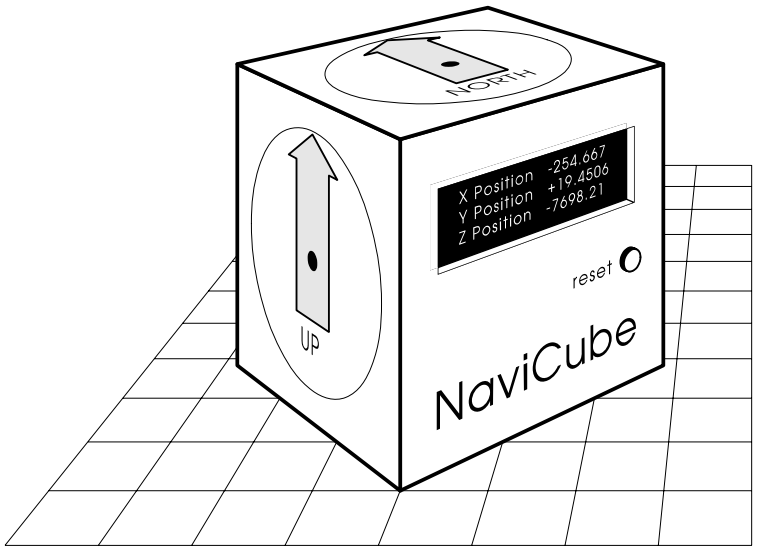


Fig. 3 – A demo of my fast perspective lettering routines.

rate of around 1600 or so. Thus, your typical modem would have to get run *continuously* for an entire month just to try and keep up. At $6 per hour, it would cost you $4320.00 for what the post office will gladly do for under a buck. Delivered 27 or so days faster.

So why not bundle up *everything* that goes on for the month on a BBS and then offer it as a single CD ROM disk? Anyone who wants anything gets everything. In a form that is far easier to search and work with.

Which leads us to yet another heresy. Traditionally, information has always been sold at the full manufacturer's list price, or more often, vastly above retail. Especially whenever you factor in such older things as ad costs and door-to-door encyclopedia salespeople.

Also in the past, information has been fairly expensive to duplicate and copy. Especially to an identical appearance and quality.

With CD-ROM, you now have information that has separate "retail," "street," "wholesale," and "bulk" prices. The bulk pricing even has a name – it's now called *shovelware*. In short, information is becoming a commodity. The traditional pricing structure is certain to dramatically change. Radically.

On one hand, original authors should now theoretically be able to receive a much larger share of the pie. On the other, the pie itself will be much smaller and will sit on a rack with a zillion others. Do you have any further thoughts on this?

One of my longer term goals around here is to put all my early books and articles onto CD ROM and combining it into an attractive Book-on-Demand end user kit.

Any Duplex Quality Bugs?

A really big one, that turned out very difficult to pin down but ultimately easy to cure.

As we have seen, duplex laser printers are absurdly better than equivalent non-duplex ones because of their tremendous improvements in productivity, scrap rates, and the outstanding employee morale that they can create.

I started getting some severe print quality problems with my *Hewlett Packard IID* duplex PostScript laser printer recently. The more I looked into it, the worse the problem got.

The symptom is that the first side prints fine, while the second side is gray, splotchy, and even mottled. Sometimes it even looks like you randomly sprinkled some water on the second side.

Here's where we seemed to be at: It appeared that certain combinations of cartridges, papers, density settings, and toner sometimes may not perform properly when duplex

```
% Copyright 1990 by Don Lancaster and Synergetics. 3860 West First Street,
% Thatcher, AZ. (602) 428-4073. All commercial rights reserved. Personal use
% permitted so long as this header remains both present and intact.
% PostScript secrets book+disk combo available for $29.50 VISA/MC.

% The complete and commented set of perspective drawing utilities appears as
% GEnie PSRT #172 PERSPEC.PS. GEnie connect voice info at (800) 638-9636.
% Single printed copies available free from Synergetics at (602) 428-4073.

200 dict /perspectivedict exch def perspectivedict begin

/psave {/zh zz def /yh yy def /xh xx def} def /pm {px psave moveto} def /px {/zz exch def /yy
exch def /xx exch def yo dup yy add div dup xx xo sub mul exch zz zo sub mul} def /prm {/zi
exch def /yi exch def /xi exch def objrot cos xi mul objrot sin yi mul sub xh add objrot sin xi
mul objrot cos yi mul add yh add zi zh add px moveto psave} def /prmtostack {/zi exch def /yi
exch def /xi exch def objrot cos xi mul objrot sin yi mul sub xh add objrot sin xi mul objrot cos
yi mul add yh add zi zh add px} def

/xzshow {/msg exch def /zoffset exch def /xoffset exch def /mt  {exch fontsize 0 get mul .001
mul xoffset add exch fontsize 3 get mul .001 mul zoffset add 0 exch prmtostack moveto} def
/li {exch fontsize 0 get mul .001 mul xoffset add exch fontsize 3 get mul .001 mul zoffset add
0 exch prmtostack lineto} def /ct {3 {6 2 roll exch fontsize 0 get mul .001 mul xoffset add exch
fontsize 3 get mul .001 mul zoffset add 0 exch prmtostack} repeat curveto} def /cp
{closepath} def /strrx ( ) def msg {strrx exch 0 exch put strrx dup ( ) eq {pop (space) } if cvn
mychardict exch get exec exec charproc fontsize 0 get mul .001 mul extrakern add xoffset
add /xoffset exch def} forall} def

/mychardict 100 dict def mychardict dup

/N {740 {76 0 mt 150 0 li 150 650 li 590 0 li 664 0 li 664 739 li 590 739 li 590 126 li 176 739 li
76 739 li cp}} put dup

/a {683 {620 547 mt 549 547 li 549 443 li 508 521 421 559 335 559 ct 259 559 182 532 99
443 ct 79 425 42 350 42 277 ct 42 194 76 125 96 103 ct 166 23 255 -13 337 -13 ct 423 -13
507 26 549 103 ct 549 0 li 620 0 li cp 335 488 mt 457 488 549 391 549 270 ct 549 156 451
58 337 58 ct 220 58 116 151 116 271 ct 116 392 215 488 335 488 ct cp}} put dup

/v{554{238 0 mt 315 0 li 546 547 li 466 547 li 277 94 li 88 547 li 8 547 li cp}}put dup

/i {200 16 add {63 8 add 0 mt 137 16 add 0 li 137 16 add 547 10 sub li 63 547 10 sub li cp 63
613 mt 137 16 add 613 li 137 16 add 739 li 63 739 li cp}} put dup

/C {813 {767 549 mt 747 603 686 659 639 689 ct 576 731 501 751 426 751 ct 216 751 44
582 44 372 ct 44 154 215 -13 432 -13 ct 571 -13 706 70 770 193 ct 688 193 li 630 112 532
58 432 58 ct 264 58 118 199 118 372 ct 118 534 259 680 427 680 ct 526 680 630 633 685
549 ct cp}} put dup

/u {608 {545 547 mt 471 547 li 471 258 li 471 211 466 164 438 125 ct 406 82 357 58 304 58
ct 244 58 191 85 161 137 ct 139 174 137 216 137 258 ct 137 547 li 63 547 li 63 258 li 63 186
69 120 102 85 ct 164 12 224 -13 294 -13 ct 370 -13 439 15 475 85 ct 475 0 li 545 0 li cp}} put
dup

/b {682 {63 0 mt 133 0 li 133 94 li 176 23 272 -13 351 -13 ct 431 -13 499 8 575 94 ct 605 127
640 197 640 276 ct 640 438 503 559 344 559 ct 263 559 173 521 137 445 ct 137 739 li 63
739 li cp 346 488 mt 468 488 566 398 566 274 ct 566 155 470 58 351 58 ct 231 58 132 147
132 269 ct 132 329 147 379 190 422 ct 231 464 287 488 346 488 ct cp}} put dup

/e {650 {611 247 mt 609 326 598 390 547 453 ct 492 520 413 559 327 559 ct 168 559 42
432 42 273 ct 42 113 169 -13 329 -13 ct 445 -13 557 68 595 178 ct 518 178 li 485 103 403
58 322 58 ct 221 58 120 143 116 247 ct cp 116 314 mt 134 415 225 488 328 488 ct 429 488
518 411 537 314 ct cp}} put pop end

%  //// demo - remove or alter before reuse. ////

perspectivedict begin 200 dict /mynewdict exch def mynewdict begin -220 100 translate /xo
200 def /yo 800 def /zo 500 def /objrot 45 def /extrakern 1 def
/charproc {fill} def /fontsize [120 0 0 120 0 0] def

530 600 translate 0 50 0 pm 22 25 (NaviCube) xzshow showpage end quit
```

Fig. 4 – PostScript fast perspective lettering sampler.

printing. *Even though these very same combinations will work just fine when double pass printing!*

Something about the paper temporarily changes as the first side is printed. Specifically, at your fusion rollers, the paper can get squashed, heated, calandered, dehumidified, and electrically discharged. Which apparently causes enough surface changes that the second side will get printed under totally different local conditions than the first one.

Curiously, letting your paper sit out in the open for a few minutes will "reset" the problem and allow normal printing. But just waiting in their covered switchback channel does not seem to matter.

Yes, I did try changing the fusion assemblies, toners, settings, and papers. But the solution turned out to be absurdly simple: *A dirty primary corona wire in a duplex printer will foul up your second side printing ridiculously earlier than it will foul up the first side!*

Apparently the reason for this has to do with the electrostatics that are involved in charging up the newly modified paper surface.

The cure is obvious: Clean your primary corona wire in the machine itself with a Q-tip every time you reload your cartridges. Be careful when you do this, since the wire is rather fragile.

Figure two is a short PostScript routine that lets you condition and test your duplex printer. All it does is print up two double sided black sheets. As with any printer, the first sheet conditions the drum by doing a *blackflashing*. A partial dark gray on this is normal and expected.

But your second sheet should be equally black on both sides. If it is not, you have a problem. In general, this test will appear much worse than your needed printouts, since larger black areas are the ones most affected by this problem.

What is This Month's PostScript Utility?

I thought we'd look at some new higher speed perspective lettering and drawing utilities.

In the past, high speed PostScript perspective lettering was rather tricky, owing to those locked and encrypted font paths and the fact that a non-linear transformation is needed to handle all the resulting trapezodial letter shapes.

In recent *Ask the Guru* columns, we've looked at directly capturing font paths and then using nonlinear transforms to rapidly distort your letters into virtually any shape. This new technique is much faster than the glacial and klunky *pixel line remapping* routine we originally were forced to use.

I have recently upgraded my old perspective utilities so they now include high speed and easy-to-use lettering in any font, new layout grids, directional repeats, and flat-to-surface perspective mapping. A typical example is shown in figure three, while I have excerpted some of the key lettering routines for you in figure four.

The really neat thing about these utilities is that you can use a plain old word processor on any old computer and a 2K textfile to completely blow away most fancy illustration or CAD/CAM programs. Especially when it comes to fancy lettering, zillions of fully detailed bricks or shingles all shown correctly in true perspective, or for ultra-compact files that print fast.

There are two steps involved in doing perspective lettering. First, the paths for all the letters in your message in the chosen font must be captured 1000 points high into a dictionary known as *-mychardict-* or else linked to it. Each font character consists of a width value followed by a proc that describes your font shape as a collection of *mt* (moveto), *li* (lineto), *ct* (curveto), and the *cp* (closepath) operators. Note that *only* these operators are permitted, as "caught" by a font path grabber.

Secondly, the message string you want to print is sent to one of three procs called *xzshow*, *yzshow*, and *xyshow*, depending on which side of the arbitrarily rotated current object you are working on.

These operators do behave somewhat like the *show* operator. Except that they do the needed nonlinear transformations, now have optional predefined kerning, and let you do anything you want with your final font paths. The key to this entire operation is that each *x* and *y* point involved in each and every *moveto*, *lineto*, or *curveto* gets transformed to a new *x'* or *y'* to appear correctly into your final perspective space.

Thus each and every end of each and every subpath in the character is correctly positioned in your final perspective space.

The routines also now support a *printproc* operator which lets you stroke, fill, or outline your font. Or do any of those multiple stroking tricks you'll find over in the *LaserWriter Corner* free fonts. A *fontsize* operator lets you change the height and width independently and still use a single path grab. While this uses a [X 0 0 Y 0 0] font array, you're only allowed to change X or Y.

These routines can be extended for center or right justification. And we'll be seeing lots more on PostScript nonlinear transformations in future columns.

A complete set of the perspective utilities appears ready to run on GEnie PSRT library as download #169 PERSPEC.PS, while several font path grabbers are currently being developed. You can also write or call me for a free printed listing of either of these.

Fortunately, PostScript Level II makes font grabbing a simple and automatic process. All font paths are now unlocked and unprotected! We'll be having bunches more on this as soon as we can.

A reminder here that we also now have a brand new set of *LaserWriter Corner* reprints that I do call *Don Lancaster's LaserWriter Secrets* and seperately available. Here you'll find lots of free fonts, insider tricks, unique publications and unusual sources of supplies. Most of the contents can get used with any PostScript laser printer, not just a LaserWriter. And a ready-to-run companion disk is included in your choice of formats.❄

Don Lancaster's

ASK THE GURU

October, 1990

A new PostScript wish list
Using the Adobe Distillery
Double distilled compiling
Graphics on a Bezier surface
Meals-in-minutes packaging

Rumor has it that the highly touted Apple IIe card for the Mac LC is incapable of running *AppleWriter*! Especially in its modem record mode essential for useful PostScript work.

I'll try and work up some sort of a fix on this, so stay tuned.

Let's see. Apple has now quietly dropped the *Apple IIc Plus* and its related printers. The end of an era, for sure.

1. **PLATFORM INDEPENDENT DISPLAY POSTSCRIPT.**

   Reason: To eliminate the ludicrous transformations needed for on-screen displays. To speed up editing and debugging.

2. **EXPLICIT SUPPORT OF SHARED SCSI COMM.**

   Reason: Most of today's PostScript printing is severly baud rate limited. Shared SCSI comm provides as much as a 50:1 speedup.

3. **FONT LOCK PERMANENTLY REMOVED FROM PATHFORALL.**

   Reason: Instantly available font paths greatly simplify and ridiculously speed up the nonlinear transformations needed for perspective, starwars, banner, flag, and similar creative display typography.

4. **A READABLE AND RECORDABLE FRAMEDEVICE.**

   Reason: Dramatic speedups of most step-and-repeat routines. Greatly simplified book-on-demand publishing, especially when combined with CD-ROM. Incredibly powerful editing and post-processing possibilities.

5. **CEXEC OPERATOR DOCUMENTED IN DETAIL.**

   Reason: To level the playing field and give all end users the same power tools that only a favored few have today.

6. **A VIDEO FRAMEDEVICE OUTPUT**

   Reason: Fast and useful debug tool, especially when speeding up code. Also opens up alternate applications such as printed circuit production, CAD/CAM, Santa Claus machines, sign routers, and vinyl cutters.

7. **SYSTEM MONITOR AVAILABLE TO ALL**

   Reason: To allow rapid end user correction of problems such as the fatal copypage flaw in the duplex mode.

8. **A FULLY OPEN FONT CACHE ARCHITECTURE**

   Reason: By allowing the cacheing of anything of any size, instead of just small typography, such things as multiple logos and other calculation intensive routines could end up running much faster.

9. **FULL SCSI HARD DISK DOCUMENTATION**

   Reason: Host interaction with the actual on-disk font cache allows many speedup and post-processing opportunities.

Fig. 1 – My PostScript wish list.

Apple does have some interesting new publications out. They have a new *Apple II Guide* and the revised *Macintosh Development Tools and Languages Guidebook*. And for those of you interested in disabilities, the handicapped, or special education, there's a revised *Connections Guide: Computer Resources for the Disabled*. There is also a new *Apple Computer Resources for Special Education and Rehabilitation*.

Contact your dealer or local user group for info on the first three. The final book is available through *DLM Teaching Resources* for $19.95.

The 3-M folks have now come up with *Post-It* notes in a precut and laser printable sheet form. Product #7709. Availability does still seem limited at this writing. The obvious uses include custom notepads, both personal and for resale.

Two great PostScript books now include that *gray* book from Glen Reid, otherwise known as *Thinking in PostScript*, and the *beige* book authored by Ross Smith, and titled *PostScript – A Visual Approach*. I do try and stock *all* of the very best PostScript books for you. Write or call for a complete list.

A reminder here that we now do have this great PSRT RoundTable up on *GEnie* with lots of PostScript and all the other *Ask the Guru* and *LaserWriter Corner* stuff on it. And including lots of fully debugged and ready-for-use downloads of most of the printed routines you see here and in our earlier reprints.

You can voice call (800) 638-9636 for connect info. I have also now put together a PSRT sampler disk for you. Call or write me per the end trailer for details.

What Do You Really Want From PostScript?

Did you know that all of those LaserWriter printers have a built-in system monitor that lets you view

memory, add or alter any portions of the internal code, and change or repair any or all routines? All you do is jumper two secret connections in the serial cable to activate your monitor. A set of predetermined bytes in your ROM chips will then preselect the "privilege" level of access you are granted.

This is one example of the many secret features of the PostScript language which you end users are prohibited from using. At one time long ago and far away, some of the restrictions may have made slight sense to somebody, somehow. But all they do today is grievously slow down, incapacitate, and very much infuriate the end user.

Figure one is a list of some of the more crucial intentionally crippling restrictions forced upon all of you PostScript end users. I like to call this my *wish list*.

Note that most insiders already have access to some or all of these features. Many could be found by nosing around the larger university related BBS systems. The others by rounding up all the usual suspects. Thus, there appears to be a rather uneven playing field. And a very foggy one as well.

Note also that no new technology is needed here. A few simple words on an interoffice memo or two can instantly provide everything the entire wish list asks for.

And one of the big side effects of fulfilling my wish list would be to make PostScript so powerful to all the end users that it would become totally unassailable by any clone or competing technology.

Word has it that the upcoming PostScript Level II should have unlocked font paths. Which would be one major step in the right direction. One down, nine to go.

Sigh.

Compiled PostScript?

Sort of. And certainly usefully.

You'll find two main methods of handling most any computer language. If the language is *interpreted*, each instruction gets used when and as it occurs at the highest level. If your language gets *compiled*, some special process is gone through to make the final run time code be as compact, fast, and as efficient as is reasonably possible.

Compiling also divorces the run time code far away from the original applications package that generated that code. Which can get handy for such things as providing fully device independent printed circuit art on BBS systems without the need for any applications support.

This can solve a crucial problem of both the hardware hackers and all their technical editors. Using pseudo-compiled PostScript in its EPS format to directly provide the end-users accurate and first quality printed circuit layouts, dialplates, templates, whatever.

Repeated use of compiled code often could run much faster than interpreted code at the triple costs of all your time required for your compiling process, deferred error messages, and moderately longer program files.

Compiled code is also quite hard to edit or change. To the point that you'll nearly always edit and then compile and rarely vice versa. This, of course, is true of just about *any* compiling. Not just PostScript.

PostScript is an interpreted language. While a true compiling of your PostScript output down to the machine language level is not yet readily available, there are several tricks you can attempt to pseudo-compile your PostScript code.

The results can be an incredible speedup. For instance, we routinely make up a three column and 6000 character page with two figures, a header and footer in three seconds

```
/bdef { bind def } bind def
/ldef { load def } bind def
/selectfont { exch findfont exch scalefont setfont } bdef
/DF { selectfont currentfont def } bdef
/BEGINPAGE { pop /pagesave save def } bdef
/ENDPAGE { pop pagesave restore showpage } def
/AW { moveto awidthshow } bdef
/F /setfont ldef

1 BEGINPAGE
/F1 /Palatino-Bold 10 DF -0.6 0 32 0.1 0 (Compiled  PostScript?) 128.73 720 AW
/F2 /Palatino-Roman 9.5 DF
0.65 0 32 0.1 0 (Sort of. And certainly usefully.) 110 695 AW
F2 F 1.9 0 32 0.108918 0 (You'll find two main methods of ) 110 682.5 AW
F2 F 0.00576103 0 32 0.1 0 (dealing with any computer language.) 100 670 AW
F2 F 1.9 0 32 0.282986 0 (If the language is ) 100 657.5 AW
/F3 /Palatino-Italic 9.5 DF 1.9 0 32 0.282986 0 (interpreted) 187.447 657.5 AW
F2 F 1.9 0 32 0.282986 0 (, each ) 232.788 657.5 AW
F2 F 0.855241 0 32 0.1 0 (instruction gets used when and as it ) 100 645 AW
F2 F 0.634014 0 32 0.1 0 (occurs at the highest level. If the lan- ) 100 632.5 AW
F2 F 0.222581 0 32 0.1 0 (guage is ) 100 620 AW
F3 F 0.222581 0 32 0.1 0 (compiled) 138.481 620 AW
F2 F 0.222581 0 32 0.1 0 (, a special process is ) 173.234 620 AW
F2 F 1.54727 0 32 0.1 0 (gone through to make the final run ) 100 607.5 AW
F2 F 0.33065 0 32 0.1 0 (time code be as compact, as fast, and ) 100 595 AW
F2 F 0.65 0 32 0.1 0 (as efficient as possible.) 100 582.5 AW
F2 F 1.10939 0 32 0.1 0 (Compiling also "divorces" the run ) 110 570 AW
F2 F 0.459408 0 32 0.1 0 (time code far away from the original ) 100 557.5 AW
F2 F 1.74072 0 32 0.1 0 (applications package that generated ) 100 545 AW
F2 F 1.62958 0 32 0.1 0 (that code. Which can get handy for ) 100 532.5 AW
F2 F -0.158258 0 32 0.1 0 (such things as providing fully device-) 100 520 AW
F2 F 1.48734 0 32 0.1 0 (independent printed circuit artwork ) 100 507.5 AW
F2 F 0.397416 0 32 0.1 0 (on ) 100 495.0 AW
/F4 /Palatino-Roman 9 DF 0.397416 0 32 0.1 0 (BBS) 113.788 495.0 AW
F2 F 0.397416 0 32 0.1 0 ( systems without needing any ) 129.811 495.0 AW
F2 F 0.65 0 32 0.1 0 (applications package support.) 100 482.5 AW
F2 F 0.65 0 32 0.1 0 (  ) 110 470 AW
1 ENDPAGE

% Total non-header length:              1808 bytes
% Typical baud rate limited run time:   1064 milliseconds on QMS Turbo PS820
% Raw PostScript run time:              150 milliseconds on QMS Turbo PS820
```

Fig. 2 – Example of Distillery Compiled PostScript.

or so. From AppleWriter on a IIe.

Most of the time, there is no point in compiling PostScript unless you are (1) completely happy with your uncompiled version in its final form except for its speed; (2) are going to use your file at least several times in the future, and (3) are using a comm channel that is not at all baud rate limited, especially with the possibly longer compiled code.

Or, separately (4) if you do not want the end user to require or use the original applications package.

Or, finally (5) if you are using a horrendously slow phototypesetter for all of your final high resolution camera-ready art. Even with only a single use, you can sometimes use a Pseudo-compiling to save bunches of time and money.

Thus, while compiled PostScript code is outstanding for book-on-demand publishing from hard disk based files or for a wide distribution of a printed circuit board pattern, you might not want it for everyday routine use.

There are several approaches to compiling. Rather than the strict definition of "make it all machine language", we'll define a compiling as any one-time stunt you can pull to reduce to an absolute minimum the work PostScript has to perform during your future printings. For instance, there is no point in making justification calculations each time. Instead, you save *only the results* of those calculations and use these results instead.

One simple and rather obvious compiling trick is called *binding*. If you simply use a *bind def* instead of *def*, PostScript objects get linked directly to other objects, eliminating the name lookups. Those *//name* immediately executed names also do the same thing. Binding sometimes gives you a ten to fifteen percent speedup. But it can also make any later code modifications difficult to do. Full details in the red book.

By far the most general way of compiling PostScript code is with an *Adobe* product called the *Distillery*. Adobe has graciously uploaded a freeware copy of the Distillery to our *GEnie* PSRT RoundTable as our file #186 DISTILL.PS.

At any rate, what the Distillery does is intercept any operator that would make any marks on the page, such as *lineto*, *awidthshow*, and all the other markers.

It then asks "What is the absolute minimum amount of info needed to use this operator?" And then either returns that info to your host for recording, or else can write it to a selected hard disk file. The new file becomes your run time code.

Their Distillery works great for some routines and only so-so for others. Often you have to try it and see. You just might want to modify your programming style or your original applications packages to make better use of the Distillery.

While the Distillery often makes code much shorter, at times it can make a simple and short routine into an unbelievably complex data string. Obviously, in these cases, you may want to mix the original code with the compiled code to get the tightest final package.

While it is amazing what it does and how well it generically handles pretty near any input, there are some Distillery bugs. For instance, superscript and subscript aren't supported in the font matrix unless you make the character height and width at least slightly different. And original code that insists on using individually spaced words, rather than calling PostScript's powerful *awidthshow* operator, will produce inherently longer code. As much as three times longer on the average.

The Distillery does not seem to know how to save a path for reuse. This means that your *entire* path gets repeated for such things as a fill and stroke. Obviously, you could

```
/F {exch findfont exch scalefont setfont} bind def
/A {moveto 0 exch 0 32 5 2 roll awidthshow} bind def

 /Palatino-Bold 10 F
 -0.6 .1(Compiled  PostScript?)128.7 720 A

 /Palatino-Roman 9.5 F
 .65 .1(Sort of. And certainly usefully.)110 695 A
 1.9 .1089(You'll find two main methods of)110 682.5 A
 .005761 .1(dealing with any computer language.)100 670 A
 1.9 .2830(If the language is)100 657.5 A
 1.9 .2830(, each)232.8 657.5 A
 .8552 .1(instruction gets used when and as it)100 645 A
 .6340 .1(occurs at the highest level. If the lan-)100 632.5 A
 .2226 .1(guage is)100 620 A
 .2226 .1(, a special process is)173.2 620 A
 1.547 .1(gone through to make the final run)100 607.5 A
 .3307 .1(time code be as compact, as fast, and)100 595 A
 .65 .1(as efficient as possible.)100 582.5 A
 1.109 .1(Compiling also "divorces" the run)110 570 A
 .4594 .1(time code far away from the original)100 557.5 A
 1.74072 .1(applications package that generated)100 545 A
 1.630 .1(that code. Which can get handy for)100 532.5 A
 -.1583 .1(such things as providing fully device-)100 520 A
 1.487 .1(independent printed circuit artwork)100 507.5 A
 .3974 .1(on)100 495.0 A
 .3974 .1( systems without needing any)129.8 495.0 A
 .65 .1(applications package support.)100 482.5 A

 /Palatino-Italic 9.5 F
 1.9 0.2830 (interpreted)187.4 657.5 A
 0.2226 .1 (compiled)138.5 620 A

 /Palatino-Roman 9 F
  .3974 .1 (BBS)113.9 495.0 A

 showpage


% Total non-header length:          1282 bytes
% Typical baud rate limited run time:  754 milliseconds on QMS turbo PS820
% Raw PostScript run time:            133 milliseconds on QMS turbo PS820
```

Fig. 3 – Example of Double Distilled Compiled PostScript.

hand edit your intermediate code to get around this limitation.

Figure 2 shows you some typical stock Distillery output.

I have just released the latest version #13 of my new *Guru's Gonzo Justify Power Tools*. This code now includes an optional automatic and Distillery-compatible but text-only compiling feature. You can easily download a shareware copy and its documentation from PSRT on *GEnie*, or else write me directly for a free printed listing.

It is also possible to use a sneaky new technique which I call *double distilling*. Most PostScript end users most of the time become baud rate limited, especially if they are using AppleTalk. If you can pick up find any tricks that *shorten* your distillery files while adding only a minor computation speed penalty, you might further speed up your run times by as much as an extra 35 percent or so.

For instance, you could drop any leading or trailing zeros. For most users most of the time, you could round off to four place accuracy.

You can eliminate spaces before or after self-delimiting parentheses. And trailing spaces within strings. And empty strings. At a tiny speed penalty, you can eliminate the "0", "0", and "32" and their intervening spaces which are used with a horizontally set *awidthshow*.

You can also sort your files so that each font only gets changed *once* on each page. The Distillery already predefines "made" or "scaled" font dictionaries such that only the very fast and VM free *setfont* operator is used for an actual change. So your gain by sorting is not spectacular. Still, as the figures show us, you do gain a fair amount of speed and save enough characters this way to end up more than worthwhile.

Figure 3 shows you how your Distillery file can be further shortened by *double distilling*. At this time, this process is custom and labor intensive. Let me know if you have any further interest in seeing improved versions of this double distilling process.

When you are baud rate limited, your processing time is directly proportional to the total number of characters in your file. The "Baud Rate Limited" times shown here are based on the AppleTalk on a typical Mac that's running at a 17 kilobaud effective rate. Your own baud rate limiting could easily become much worse than this.

This is why we *must* have shared SCSI comm *now*.

Any New Product Packaging Ideas?

All of your laser printed output, book-on-demand published stuff, or similar software goodies should be delivered to the end customer in an attractive, "see-thru", and protected package of some sort.

The traditional method that the printshops use is the same *shrink wrapping* used in your grocery store. You can get further information on shrink wrap products in *Printer's Shopper*, or *Quick Printing*, *Printing Impressions*, or the *Instant & Small Commercial Printer* mags.

But I think I've found a better method. A good quality look, better touch, and best durability.

A number of years ago, several firms introduced *Meals in Minutes* pouch packaging systems. These were intended for sealing leftovers in plastic baggies that could be frozen, boiled, or nuked. The best of these included a powered vacuum pump that actually sucked the extra air out of the package.

I think these are absolutely great, and these are one of the few things around here that I actually use for its intended purpose. Well, some of the time anyway.

Apparently the product bombed, or at least the market got flooded, for all the discount houses are now offering these at bargain prices.

In particular, do check out that *DecoSonic* from *The Lighter Side*, the *Seal-A-Meal* from *COMB*, *PaknSave* from *Heartland*, or *Vac-U-Pac* from *Damark*. The prices are in the $19-$39 range. Do make sure your model includes a vacuum pump.

Now for the secret part. These work beautifully with the plain old *Zip-lock* bags, especially those heavy duty "freezer" versions.

One tip: You usually throw away the zip-lock part and just make the rest of the plastic tightly fit your product. To do this, simply peel away your scrap side while your plastic is still hot and you are still holding the lever down. With some practice, you'll get a tight and uniform edge seal every time.

We use them here at *Synergetics* to seal all our book-on-demand and

Fig. 4 – Accurate Graphics and Text on a Bezier Surface.

```
% Copyright c 1991 by Don Lancaster and Synergetics, Box 809, Thatcher AZ, 85552.All
% commercial rights reserved. Personal use permitted if this header remains preset.
% LaserWriter Secrets book+disk $29.50 VISA/MC. Free voice helpline (602) 428-473.
% Available on GEnie PSRT library as #202 TWIXTBEZ.PS.

/mychardict 10 dict def mychardict dup

/F {611 {258 326 mt 373 320 391 304 423 177 ct 444 177 li 444 510 li 423 510 li 39 383 373 367
258 361 ct 258 590 li 258 641 265 650 301 649 ct 387 647 li 544 643 552 508 562 47 ct 582 474
li 582 681 li 17 681 li 17 660 li 69 660 99 627 99 575 ct 99 106 li 99 54 69 21 1721 ct 17 0 li 340
0 li 340 21 li 288 21 258 54 258 106 ct cp }} put dup /R {722 {498 0 mt 695 0 li 65 21 li 683 21
673 16 645 57 ct 455 335 li 516 350 561 374 594 425 ct 634 488 610 569 569 611 ct 15 666
416 681 338 681 ct 26 681 li 26 660 li 78 660 108 627 108 575 ct 108 106 li 108 5478 21 26 21
ct 26 0 li 349 0 li 349 21 li 297 21 267 54 267 106 ct 267 317 li 292 317 li cp 267 606 mt 267 633
288 645 308 647 ct 383 655 417 622 437 586 ct 451 560 458 476 442 427 ct 412 338 36 350
267 349 ct cp }} put dup /E {667 {262 326 mt 377 320 395 304 427 177 ct 448 177 li448 510 li
427 510 li 395 383 377 367 262 361 ct 262 590 li 262 641 269 650 305 649 ct 391 64 li 548 643
556 508 566 474 ct 586 474 li 586 681 li 21 681 li 21 660 li 73 660 103 627 103 57 ct 103 106 li
103 54 73 21 21 21 ct 21 0 li 593 0 li 637 206 li 610 206 li 539 62 462 18 319 30 t 268 34 262 48
262 99 ct cp }} put dup /space {250 {250 0 mt }} put dup /O {778 {733 336 mt 733 44 681 569
591 627 ct 525 669 455 690 388 690 ct 321 690 251 669 185 627 ct 95 569 43 454 43 36 ct 43
218 95 103 185 45 ct 251 3 321 -18 388 -18 ct 455 -18 525 3 591 45 ct 681 103 733 18 733 336
ct cp 388 8 mt 370 8 349 12 327 23 ct 287 42 259 80 236 147 ct 221 190 211 261 211336 ct 211
411 221 482 236 525 ct 259 592 287 630 327 649 ct 349 660 370 664 388 664 ct 406 64 427
660 449 649 ct 489 630 517 592 540 525 ct 555 482 565 411 565 336 ct 565 261 555 10 540
147 ct 517 80 489 42 449 23 ct 427 12 406 8 388 8 ct cp }} put dup /N {722 {617 -1 mt 617 575 li
617 627 645 660 697 660 ct 697 681 li 498 681 li 498 660 li 550 660 578 627 578 57 ct 578 249
li 230 681 li 20 681 li 20 660 li 47 660 57 639 71 622 ct 108 577 li 108 106 li 10 54 80 21 28 21
ct 28 0 li 227 0 li 227 21 li 175 21 147 54 147 106 ct 147 529 li 588 -10 li cp }}put dup /T {667
{409 649 mt 433 648 498 661 554 603 ct 603 553 602 505 607 479 ct 629 479 li 629 61 li 30
681 li 30 479 li 52 479 li 57 505 56 553 105 603 ct 161 661 226 648 250 649 ct 250106 li 250 54
220 21 168 21 ct 168 0 li 491 0 li 491 21 li 439 21 409 54 409 106 ct cp }} put po

/pcurveto {8 copy /y3 exch def /x3 exch def /y2 exch def /x2 exch def /y1 exch def/x1 exch def /y0
exch def /x0 exch def} def /xtt {x3 x2 3 mul sub x1 3 mul add x0 sub tt 3 exp mul 2 3 mul x1 6
mul neg add x0 3 mul add tt dup mul mul add x1 3 mul x0 3 mul neg add tt mul add x add} def
/ytt {y3 y2 3 mul sub y1 3 mul add y0 sub tt 3 exp mul y2 3 mul y1 6 mul neg add y 3 mul add tt
dup mul mul add y1 3 mul y0 3 mul neg add tt mul add y0 add} def
```

Fig. 5a – Excerpts From my Twixt Bezier Routines…

book+disk products, as well as for protecting business cards and out-the-door orders. The sealers are also great for protecting stuff at trade shows from wear and tear.

You could get "real" polyethelene tubing through the *Associated Bag Company* or from *Harwil*, but that is no fun at all.

As usual, all of our *Names and Numbers* have now been gathered together for you in the ending appendix to this volume.

For this month's contest, just tell me about any other gross abuse or obscene misuse that you've been making in applying something well away from its intended purpose. For fun or profit.

This Month's PostScript Utility?

I thought I'd throw a real heavy at you this month. A new method that I've recently developed that lets you quickly and accurately place text and graphics on any surface defined by a pair of cubic spline curves. One typical "flag waving" example is shown you in figure four, while enough excerpts of my PostScript code to get you going appear in figure five.

Other obvious uses for my new TWIXTBEZ.PS routines are for banner fonts, text on a cylinder, twisted film effects, pennants, any curved perspective surfaces, fancy logos, lettering on a scroll, a funhouse mirror reversal, and just about any other distortion you can think of. The processing speed is ridiculously faster than the earlier *pixel line remapping* we looked at earlier.

The method uses bits and pieces of stuff from previous columns and from the *Ask The Guru III* reprints. Especially the nonlinear transformations, the length of Bezier curves, and stopping at the selected point along a single Bezier curve.

The new black magic added here, though, will involve *automatically converting straight lines into smooth flowing curves, given only the end points of those lines!* Thus, the tops and bottoms of all your letters will properly curve themselves to conform to your new surface.

In the routines shown here, all vertical lines are kept that way. This is what you usually will want in any text-oriented logo or clip art.

Where do you start? Your text or graphics must first get *completely* converted into combinations of one of four allowed routines. These are *mt* movetos, *li* linetos, *ct* curvetos, and *cp* closepaths. Characters in any font must be predefined into a 1000 point high proc definition of form *{ -charwidth- {char defs using only mt, li, ct, and cp}}*.

Now for the tricky part. You can optionally convert any of those *lt* linetos into a curve that smoothly flows over your new surface! To do this, you first set *showlinesascurves* to *true*. Your code then takes any linetos and neatly breaks them up into individual *numcurvesperlineto* segments. Each individual segment is then changed into a cubic spline, both of whose influence points lie *halfway* along their original line path, thus "bending" it.

What this does is convert your original line into an end aligned group of fairly weak curved splines. This guarantees you are at the right point on the surface at each end of each spline, rather than the "short cut" your single straight line would attempt. Since each spline itself can end up curved, this further helps "adhere" to your new surface.

In figure four, four curves per lineto are used to bend the tops and bottoms of all the letters, while sixteen curves per lineto are used for your upper and bottom lines which have to sweep clear across the entire surface. Naturally, the more curves you use, the better looking your result, but the slower the calcs.

While not immediately obvious, the *closepath* operator also needs modified so it does not take any shortcuts. One rather easy way to handle this is to close up the path yourself, and then only use *closepath*

as a zero-length formality.

You select your upper and lower Bezier surfaces using an *[x0 y0 ang0 x1 y1 ang1]* matrix. Regular *Ask the Guru* readers will recognize this as the same matrix employed for a two point gonzo curvetracing. In this matrix, *x0* and *y0* are the starting coordinates of your curve, while *ang0* is the angle you want to head out in as a first step.

Similarly, *x1* and *y1* form your ending coordinates of your curve, and *an*g1 is the final direction the curve is heading in just as it reaches the end point. As usual, 0 degrees is dead east, while 90 degrees points due north.

With a single curve, you can be "bent", have an "S" shaped *inflection point*, could have a single cusp, or might even loop. For fancier paths, you can repeat the entire process as often as you need to.

The upper curved path will only be approximate. This does happen because we want to keep verticals straight up and down, and because things occur faster in the "t" space along the "more bent" portions of any spline. Any exact solution here might involve repeatedly solving ugly cubic equations.

Thus, if your upper path doesn't give you quite what you want on the first cut, just bend it a little more or a little less, and it should fall in place the way you intend it to.

Drawing and graphics gets done by using your *sketchmode* feature. You should call the *sketchmode* proc immediately before any drawing is done. This scales things so you are working directly in points on your Bezier surface. The x direction is now *along* your curved surface, and your y direction stays vertical.

Characters are shown using my new nonlinear variant of the *ashow* operator called *twsshow*. The height and width of each individual letter is selected by a */fontsize [width 0 0 height 0 0] def* matrix.

What you do with your character path is set by *bezcharproc*.

Instead of the simple fill shown here, you could outline and stroke, shadow, choke, spread, flow, or do most any combination of the special effects. Note that this is insanely more flexible than what the original *ashow* operator permitted.

You can do individual character kerning by following the examples detailed in figure five. In this case, we have tightened the distance on either side of the "O" and made the final "T" somewhat narrower than is normal. Note that things happen faster on the "more bent" portions of your curved surface. You can also mix character heights and vertical offsets for special effects, especially for menus or product labels.

We will be seeing several more examples of these non-linear text transformations right here, on my *GEnie* PSRT, and in my *LaserWriter Secrets* reprint series.

Those rumored improvements in PostScript level II should make an on-the-fly font path grabbing and nonlinear transformation a quick and easy process. Sadly, we are not quite there yet.

Fully commented TWIXTBEZ.PS routines appear in the *GEnie* PSRT library as file #202. You can call me for a free printed copy.

As a second contest this month, just send me any old clip art, use example, or any extensions to my TWIXTBEZ.PS routines that you've found handy. ❄

```
/bezierlength {pcurveto /oldx x0 def /oldy y0 def /blength 0 def 0 1 numpoints 1 sub div
1.0001 {/tt exch def xtt ytt /newy exch def /newx exch def newx oldx sub dup mul newy oldy
sub dup mul add sqrt blength add /blength exch def /oldy newy def /oldx newx def} for
blength }def /numpoints 30 def

/buildpatharray {/mat exch def mat 0 get /xx0 exch def mat 1 get /yy0 exch def mat 3 get /xx3
exch def mat 4 get /yy3 exch def xx0 xx3 sub dup mul yy0 yy3 sub dup mul add sqrt 3 div
gunghofactor mul /zz0 exch def mat 2 get cos zz0 mul xx0 add /xx1 exch def mat 2 get sin
zz0 mul yy0 add /yy1 exch def xx3 mat 5 get cos zz0 mul sub /xx2 exch def yy3 mat 5 get sin
zz0 mul sub /yy2 exch def mark xx0 yy0 xx1 yy1 xx2 yy2 xx3 yy3 bezierlength]} def

/setlowerpath {buildpatharray /lowerpath exch def} def /setupperpath {buildpatharray
/upperpath exch def} def /botftt {lowerpath twixtftt} def /topftt {upperpath twixtftt} def

/twixtftt {/arry exch def /tt exch def arry 0 get /x0 exch def arry 1 get /y0 exch def arry 2 get
/x1 exch def arry 3 get /y1 exch def arry 4 get /x2 exch def arry 5 get /y2 exch def arry 6 get
/x3 exch def arry 7 get /y3 exch def x3 x2 3 mul sub x1 3 mul add x0 sub tt 3 exp mul x2 3
mul x1 6 mul neg add x0 3 mul add tt dup mul mul add x1 3 mul x0 3 mul neg add tt mul add
x0 add y3 y2 3 mul sub y1 3 mul add y0 sub tt 3 exp mul y2 3 mul y1 6 mul neg add y0 3 mul
add tt dup mul mul add y1 3 mul y0 3 mul neg add tt mul add y0 add} def

/mt {savecp exch fontsize 0 get mul .001 mul xoffset add exch fontsize 3 get mul .001 mul
yoffset add nlt moveto} def /origli {savecp exch fontsize 0 get mul .001 mul xoffset add exch
fontsize 3 get mul .001 mul yoffset add nlt lineto} def /ct {savecp 3 {6 2 roll exch fontsize 0
get mul .001 mul xoffset add exch fontsize 3 get mul .001 mul yoffset add nlt} repeat curveto}
def /cp {closepath} def /savecp {2 copy /ycp exch def /xcp exch def} def /lict {/yy exch def /xx
exch def yy ycp sub numcurvesperlineto dup add div /yd exch def xx xcp sub
numcurvesperlineto dup add div /xd exch def numcurvesperlineto {xcp xd add ycp yd add 2
copy xcp xd 2 mul add ycp yd 2 mul add ct} repeat} def /li {showlinesascurves {lict}{origli}
ifelse} def

/twsshow {/msg exch def /yoffset exch def /xoffset exch def /strrx (X) def msg {strrx exch 0
exch put strrx dup ( ) eq {pop (space)} if cvn mychardict exch get exec exec bezcharproc
fontsize 0 get mul 0.001 mul extrakern add xoffset add /xoffset exch def} forall} def
/sketchmode {/fontsize [1000 0 0 1000 0 0] def /xoffset 0 def /yoffset 0 def} def /nlt {/yyyy
exch def /xxxx exch def upperpath 1 get lowerpath 1 get sub /unitheight exch def xxxx
lowerpath 8 get div /tt exch def tt botftt /ybotpos exch def /xbotpos exch def tt topftt /ytoppos
exch def /xtoppos exch def xbotpos yyyy unitheight div ytoppos ybotpos sub mul ybotpos add
} def

%  //// demo - remove or alter before reuse. ////

100 200 moveto 10 dup scale /showlinesascurves true def /gunghofactor 2 def 106 45 {dup
mul exch dup mul add 1.0 exch sub} setscreen [2 1 80 32 14 55] setlowerpath [2 14 45 32 16
50] setupperpath

sketchmode 0.2 setlinewidth 1 setlinejoin 1 setlinecap /numcurvesperlineto 16 def 0 13 mt 34
13 li 34 0 li 0 0 li cp gsave 0.95 setgray fill grestore stroke

/numcurvesperlineto 4 def /extrakern 0.1 def /bezcharproc {fill} def /fontsize [5.5 0 0 10 0 0]
def 2 3 (FREE F) twsshow xoffset 0.3 sub yoffset (O) twsshow xoffset 0.3 sub yoffset (N)
twsshow /fontsize [4.5 0 0 10 0 0] def xoffset yoffset (T) twsshow showpage quit
```

Fig. 5b – ...Twixt Bezier Routines, continued.

Don Lancaster's

ASK THE GURU

November, 1990

Material Conversion Secrets
Emergency Password Repairs
Book-on-Demand Publishing
Lamination and Page Protection
Toner Cartridge Refilling Update

Apparently the system does work. At least for some of the people during some of the time. As you should know by now, *Apple Computer* has its new customer service line up and running at (800) 776-2333.

Their hotline is intended as an appeals court for any *non-technical* problems only. To use it, you *must* already have gone through your dealer or service center and given them every reasonable chance to correct your problem.

I inadvertently got to test this help line over one major mishap involving a fatally flawed Apple LaserWriter mailing list rental. The system works, and Apple is to be highly commended for it.

Apple has also released some new system software for their older machines. This now includes IIGS system software version 5.0.4 and the IIE system software version 3.2, otherwise known as *ProDOS* 1.9. At long last, a stab was made at some more or less useful QUIT code!

As we have seen a time or two before, the best place for ongoing technical info on any older Apple products is Tom Weishaar at *A2 Central*. Besides a great newsletter, he stocks all sorts of IIgs and IIe books, disk based magazines, and nice software products. Tom also is involved in those A2 and A2PRO boards on *GEnie*. You'll find many tens of thousands of downloads on these, including additional info on the new system software.

Hewlett Packard has combined the service and repair manuals for their IID and IIID duplex printers into a newer and compact manual. Their stock #33459-90906. Apparently the largest mechanical difference between these two is that the IIID has a lumpier case.

While that IIID should be faster and offer the higher print quality (owing to a new dot modulation), I have held off testing it since the current HP PostScript cartridge is such a fatally flawed wimp.

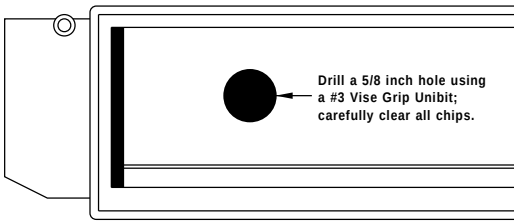
Hopefully, the new *PostScript II* cartridge should soon be available and should solve the duplex *copy-page* problem. Stay tuned.

And you saw it here first: Watch for a 50 megabyte SCSI hard disk that's built into a removable plug-in font cartridge! This could really solve some major hassles *if* it gets tightly linked to the *PostScript* font cacheing and *if* it also supports direct shared SCSI comm.

Two corrections to our previous columns: The *QMS* Turbo PS820 is in fact memory expandable to eight megabytes; my test machine was not. I would recommend a three megabyte minimum for this printer when used with a hard disk. And yet another obvious use for compiled PostScript is to dramatically shorten the long makeready times on fancy phototypesetters.

A reminder here that we now do have a great PSRT RoundTable up on *GEnie* with lots of PostScript and other *Ask the Guru* and *LaserWriter Corner* stuff on it. And including debugged and ready-for-use downloads of most of the printed routines you see here. You should voice call (800) 638-9636 for connect info. I've also put together a PSRT sampler disk for you. Call or write me per the appendix for details.

To refill an older CX cartridge with the punch-and-go method, you first snap off the cardboard label and then drill a toner filling hole…



A second CX hole is needed to let you empty the spent toner holding tank. This area is found underneath the cartridge…

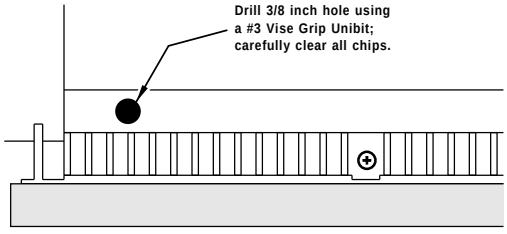


Fig. 1 – Refilling a Canon CX cartridge.

What's Happening in Toner Cartridge Refilling?

By now, you regular *Ask the Guru* readers should know that *Canon* toner cartridges as are used in the

LaserWriters, LaserJets, and QMS printers can be easily refilled by yourself for around $6 per shot. Because of up to a 15:1 reduction in all your per-page toner costs, it is insane not to personally do all your own cartridge recharging. Not to mention considerably better print quality, many newer toner options, and much longer cartridge life.

There's lots of new things coming down in the refilling industry. The biggest news involves all the new third-party retrofitted hard coated drums, which are good for a dozen or more low-hassle refills. *Copymate* is one importer, and the drums are available in smaller quantities from my two ongoing favorite refilling supply houses of *Don Thompson* and *Lazer Products.*

As usual, most of the *Names & Numbers* listed here have been gathered together into a common ending appendix to this volume.

Meanwhile, that *Recharger* trade journal has matured into a "must have" resource. Two of the latest hot products are that *Channelock* #357 end cutter for pulling the main LX cartridge pins, and a great cleaning and conditioning solvent known as *Bestine* from *Union Rubber.*

Figures one through three show you how to refill all your own CX (LaserWriter Plus), SX (LaserWriter NT and NTX), or the LX (Personal LaserWriter NT) cartridges.

I overwhelmingly prefer to use what I call the *punch and go method*, since this delivers far and away the lowest possible end user per-page toner costs. As an end user, your ultimate goal should be your final per-page costs, and *not* your total number of actual recharges per cartridge that you get.

My punch-and-go method very highly displeases certain rechargers who insist that arcane tools and techniques are essential to refill a cartridge. In reality, anyone can do a punch-and-go in two minutes flat. Once again, the object is the lowest possible per-page toner costs to the end user. Total teardown clearly fails to deliver on end-user bottom line costs.

We still charge around $18 for a refilling in our strictly limited and hand carried six mile service area. I can still get away with such an outrageously high charge since I live in a very remote rural area. But we very strongly urge our customers to do their own refilling. Travel of *any* kind is bad news for the cartridges. So is swapping your own cartridges for unknown outsiders.

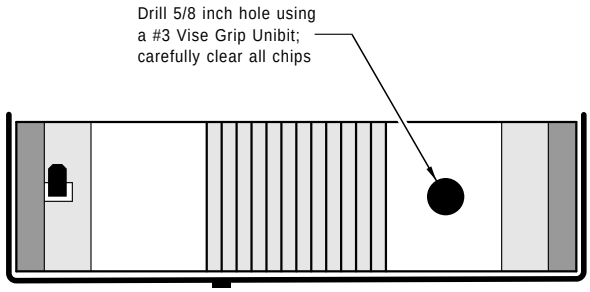
I strongly recommend you limit your recharging services to a six mile or smaller radius. Just as in delivering pizza, very ugly things happen to your reliability, your quality control, operating costs, your profits, and your customer relations should you dare exceed this absolute maximum customer service area.

Actually, I can not fathom why anyone anywhere would ever use a recharging service, when they can easily do the job themselves in two minutes flat for less than $6. That translates to paying someone else as much as $1200 per hour to not get their hands dirty.

At any rate, details for the CX cartridge appear in figure one, the SX in figure two, and on the LX in figure three. The general idea is fourfold: Firstoff, you remove any "used" toner from the spent toner holding tank. Then you refill the fresh toner hopper with a new bottle of toner, typically 250 grams or so. Third, you replace the fusion wiper felt pad in the printer itself. And finally, you update the toner history label for accurate records.

With my method, the CX and SX cartridges need two holes drilled in them on your first refill. The best way to handle this is with a special conical drill called a *#3Unibit* by *Vise-Grip*. *Jensen Tools* has them, if you can't find one at a larger hardware store. This "ice cream cone" shaped step drill cuts a perfectly smooth and round hole in thin brittle plastic, while producing a single and easily removed chip. The holes are easily sealed with Scotch Tape. Or, to get fancier, you can use

The SX cartridge punch-and-go refilling process is similar to the CX, except for the hole locations. The filler hole is shown here…



One or more spent toner drain holes must also get added to the SX cartridge. The plastic is thin, so use a conical step drill…

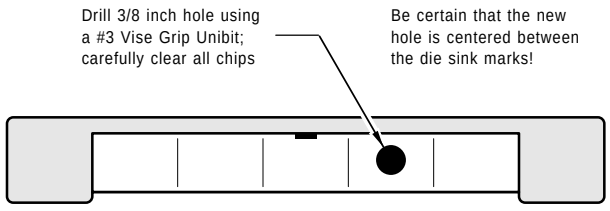


Fig. 2 – Refilling a Canon SX cartridge.

a nickel *Caplug*.

Naturally, you do have to be certain you get a very good seal if you use tape. After doing literally thousands of refills, I have yet to have any tape ever come loose. But do be extra careful.

Those newer LX cartridges are much simpler to refill. You first pull the two main pins and then pop out the fresh toner cartridge. Then you use the *Caplug* that's already there for your refill. You can often go two refills without draining the spent toner holding tank. To empty this tank, remove both the nylon drum bearings using four Phillips screws, and then *carefully* remove the drum, avoiding any and all fingerprints or any and all exposure to any strong light. Outdoors, your spent toner can simply be gently shaken out, or else vacuumed.

After all these years, *Canon* is to get attaboyed for producing a more or less maintainable and refillable cartridge.

A dab of *Locktite* on the main pins is a good idea. If your pins come loose, try replacing them with expandable "C" pins from *Small Parts*. Or use some epoxy.

Once again, you should also replace your fuser wiper wand and clean both corona charging wires every time you do a reload.

For all my own book-on-demand production, I will typically buy a used-one-time cartridge having a preinstalled longer life *Copymate* drum, and then punch and go from there. I've been using all of these techniques for many millions of copies to date, and they sure seem to work for me. At $30 per drum and nine refills at $6, you are looking at per-page toner costs of 0.21 cents. Which is well within the acceptable range for all of my book-on-demand publishing.

Speaking of which…

Update me on Your Book-on-Demand Publishing

The story so far: Several years ago it appeared to me that traditional book publishing was rapidly going to hell in a handbasket. The days of "mail in your manuscript and then laugh all the way to the bank" were clearly ending.

We've seen bunches of publisher mega-mergers that have left fewer and fewer places to submit books. The submission process itself has also gotten out of hand because of those infamous and notoriously incompetent *publisher's committees*, who will sit on a title for fourteen months and then return it because it was "not timely". Author's royalties are sharply reduced, and further frustrated by creative accounting practices.

We've had those chains replace the Mom and Pop bookstores, who now stock more individual copies of far fewer total titles. And who will automatically tear off and return covers for credit for virtually all titles after an intolerably brief 22 weeks on their shelves.

Worst of all, we've had the IRS decide to start paying publishers to shred books, just like those other government agencies pay farmers not to grow crops.

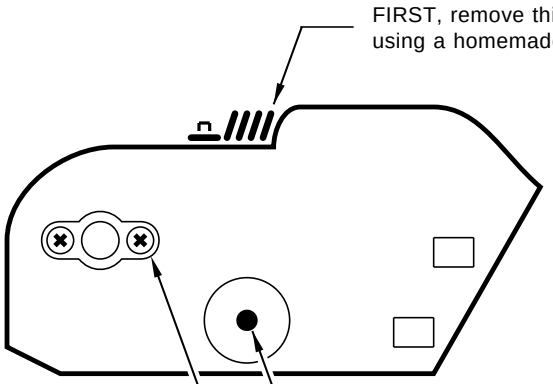
All this happened because of an accounting rule change. Backlist titles must now be carried at their full value, instead of their scrap or remainder value. The net result was the virtual elimination of backlists. Put in another light, the IRS has recently caused more books to get shredded than were destroyed in all of the dark ages. Many times over.

And people are simply buying fewer books than they otherwise might because of "alternate information formats" such as in video rentals, computer games, the trade journals, cable services, and telecomputing. Out here on my sand dune, I could rent a video from a dozen nearby places, but a 300 mile drive is involved for me to reach my nearest usable bookstore.

One revolutionary new method of creating and disseminating info is one which I call *Book-on-Demand* publishing.

With my Book-on-Demand publishing, all the individual titles are produced and delivered only *when and as they are ordered*. Which can eliminate most of the staggering

The LX cartridge does not need any modifications before any refill, hard drum upgrade, or any spent toner draining…



But be EXTREMELY careful not to touch the photosensitive drum or expose it to any strong light or ANY light of long duration.

Fig. 3 – Refilling a Canon LX cartridge.

front end expenses of traditional publishing. Production costs are ridiculously lower, especially if the total number of copies that actually end up geting sold are unknown.

Production times from author submission to your first reviewer copies can be measured in minutes rather than months. You can also now target a title much more tightly to your intended reader. Such value added items as companion disks, templates, or ready-to-use printed circuit mylars can now be easily included in your product.

When using Book-on-demand, the quaintly outdated concept of "getting a manuscript accepted" no longer has any meaning. Author's royalties can now be paid *hourly*, or at the serious risk of getting a "slow pay" reputation, only twice a day.

Because you have no inventory, there are no remainders, and your backlist can continue forever, free of any IRS penalties. Changes can be made at any time for any reason. Under certain conditions, author royalties can exceed 90 percent of the final reader list price. Your works can be customized for any user, up to and including putting their name in gold on the cover.

Above all, you get to make all of your own mistakes, rather than having to pay others to make them for you. Things get done your way, rather than the way others decided was good for you.

Key secrets to Book-on-Demand publishing include using genuine Adobe PostScript on a duplex (two sided) printing engine; working directly in raw PostScript (most especially all figures and artwork); personally doing all of your own maintenance and cartridge refilling to keep per-page printing costs in that 0.2 cents per page region; becoming very strongly SCSI comm oriented; and compiling all your PostScript code to virtually eliminate page makeready times.

Despite several really serious hardware limitations, the Book-on-Demand publishing already works well. I'm currently publishing eight titles by this method, and will certainly be doing lots more.

Eventually, I'd want to combine Book-on-Demand with CD-ROM, providing a one zillion title disk and an attractive printing materials kit directly to the end user.

Unlike some of those "books-on-disk" folks, I very strongly feel that an attractively bound paper hard copy sets the absolute essence of publishing – Omit the actual books themselves, and you will end up having thrown out the baby and drunken the washwater.

At present, any decent duplex printer does not exist, since both of the *HP* IID and IIID machines are hopelessly (and apparently intentionally) crippled. But the *Canon* duplex engine itself seems to work just fine, although it is quirky and a tad on the slow side.

What we need (and should very shortly see) is this engine combined with PostScript Level II and a fast RISC computer, a reliable shared SCSI hard disk and comm setup, large paper capacities at both ends, and some dot modulation for enhanced resolution.

My prediction is that this new machine's label will start with a Q, rather than an H or an A.

The goal here is the overnight production of a fully collated pile of books ready for binding. Totally unattended by either a person or a host computer.

At any rate, several of my titles are painfully and slowly produced overnight on a IID in its full duplex mode, while most of the other ones run on the ridiculously faster and infinitely easier to use (but sadly non-duplex) Turbo PS820.

Yes, the IIID would be faster. But, at this writing, you are still stuck with the same wimpy old PostScript cartridge having a fatally flawed duplex copypage.

While I'm still using the *Unibind* cover processing, their brand new *Pelsaer* system has a lot going for it in that you can now print on your spines and have an unlimited choice of protected and laminated cover materials. Figure four once again shows you this new binding.

My *LaserWriter Secrets* book now prints in around 28 minutes, and I should shortly be able to double this speed. Such a doubling translates to roughly a half of a million dollars of product per printer per year.

Besides a genuinely useful duplex printer, other stuff sorely needed next include an improved binding process beyond Pelsaer, perhaps a cold glue one, along with sharply reduced costs of desktop finishing machinery such as cutters, joggers, and laminators.

We also need effective ways of dramatically speeding up all the

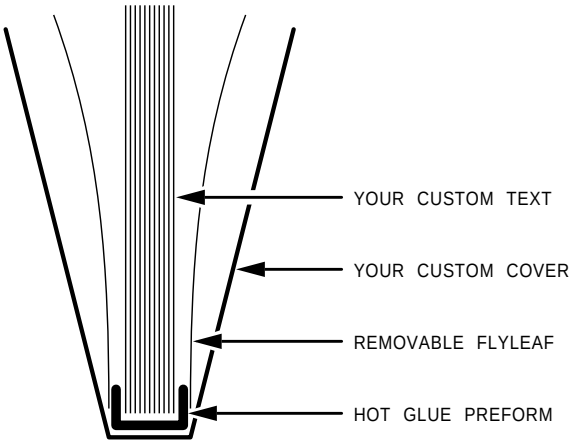


Fig. 4 – Pelsaer binding lets you print on spines.

PostScript applications packages so they can someday compete with the page makeup times and compact size of raw PostScript.

As of this writing, the next steps in Book-on-Demand are a Level II duplex printer, an improved binding system, simplified compiling, better lamination and covers, and a sanely priced trimmer. Plus, of course, wider CD ROM availability and improved distribution.

I'm actively working on solving these problems as they become economically feasible. It is all just a matter of time and priorities.

For more on Book-on-Demand publishing see GEnie PSRT Library #77 DEMAND.TXT, or join me and the others in several of our lively RoundTable discussions on this exciting new opportunity.

Tell Me About Laminating

One of the sad things about toner is that it isn't nearly as versatile as ink. At least not yet. Toner is not remotely as durable, doesn't come in lots of great colors, and does not let you do all sorts of really neat stuff like all the "fuzzy" hot split plastisol screen inks from *Gerber* or the raised thermography inks from *Sunray* or *Thermotype*.

We've already seen *Kroy Color* as one possible way to alter the color and visual effects from your laser printouts. But, what else can get done to make your toner products more durable? Especially for such touchy-feely things as book covers or restaurant menus?

One little known technique is called *Bakerizing*. This is a zero cost process to make toner far more durable while you simultaneously pick up a medium gloss.

To Bakerize, simply place your final toner image in contact with a clear piece of half mil mylar. Then apply heat and pressure. An empty or "fully used" Kroy Color carrier can work just fine for this.

You can get a quick sample of what Bakerizing is all about by opening the lid on your printer when a page is a third of the way out. After clearing the paper jam, the portion of the page that was under the fusion rollers will be Bakerized to a durable high gloss.

The *Kroy Color* folks do offer a handy and useful clear laminating material which looks great while offering reasonable protection. And I often use these for book covers, menus, business cards, and even bumperstickers. But the pricing is totally and outrageously out of line at 70 cents per sheet or more.

Lower cost alternatives to Kroy Color materials and machines are available through *Lazer Products*.

One tip with these materials: Either laminate both sides, or else "backroll" your product against a table edge to prevent curling.

Trade book covers often use one of four methods to improve their scuff resistance. The cheapest route is nothing at all, being careful to use as little ink as possible and avoiding both tiny print and large solid ink areas, especially bleeds.

The next cheapest options are a varnish overcoat, or the more costly and more durable uv-curing clear ink overcoat. The best of all is to actually use a polyester or other clear film overlay. Since the ink ends up under the plastic, you have to scuff your way through the film before abrading the ink.

Before I tell you where to go for these films, let's take a short side trip. Have you ever wondered precisely where your last potato chip bag came from?

No, not from a printer, but from a bizarre and a little-known outfit called a *converter*. These folks take large rolls of stuff, and then print

```
%  This utility attempts to read your existing password and reports it to you.
%  It is intended solely for Adobe 68000 machines, and may fail if your
%  EEPROM or non-volatile memory is in fact defective.

/#.{ userdict begin <
204F3399020E00010402001001FC0012206F000443F9000001E822C822D822D822
D84EF900000092D1F9000001EC20504ED0D1F9000001F020504ED02F08206F0004
3050D1F9000001EC2F500004205F4E7500004E56FFFC4EBA014E2D40FFFC4A806D
0A0CAE00000200FFFC6D0A487A00E84EBA0114584F2F2EFFFC2079000001F84E90
584F2F004EBA012E584F4E5E4E754E56FFF0487A00CB4EBA0104584F20402F2800
042F104EBA00EE20402F2800042F104EBA00CA4FEF00102D40FFF020402D50FFF8
2D680004FFFC08EE0007FFF82F2EFFFC2F2EFFF84EBA00B4504F4A8066564EBA00
CA4EBA00EE2D40FFF4202EFFF44EBA00DA41FA004E4EBA00CA661A23FC0026FEF4
000001F8487AFF46487A00594EBA00AA504F6028202EFFF44EBA00AE41FA002A4E
BA009E6D0C23FC000001E0000001F860D2487A00394EBA0048584F4E5E4E754037
0000000000000403D33333333333372616E6765636865636B0076657273696F6E00
726561646565726F6D00756E646566696E65640000307C00744EFAFEA2307C007C
4EFAFE9A307C008C4EFAFE92307C009C4EFAFE8A307C00B44EFAFE82307C00E04E
FAFE7A307C00E84EFAFE72307C01004EFAFE6A307C011C4EFAFE624EBAFE720004
00004EBAFE6A000C00004EBAFE6200280000307CFFD04EFAFE42000000000000000
0000000000000000000000000000160024002A00340046007E0108013001340000
> cexec currentfile closefile} def <000000000101> eexec

/dumpeprom {0 1 511 {dup 16 mod 0 eq{ (n) print flush} if readeerom dup 16 lt {(0) print
flush} if 16 (  ) cvrs print ( ) print flush} for (n) print flush ( current password = ) prin
readpassword 10 (           ) cvrs print (n) print flush} def

/printeprom {/Courier findfont 12 scalefont setfont 50 750 moveto (Eprom dump ) show 50
700 moveto 0 1 511 {dup 16 mod 0 eq {dup 0 ne {dup (  decimal address=  )show 10 (
) cvrs show}if currentpoint 12 sub exch pop 50 exch moveto} if readeerom dup 16 lt {(0)
show} if 16 (  ) cvrs show ( ) show} for 512 (  decimal address=  )show 10 (      ) cvrs show
currentpoint 24 sub exch pop 50 exch moveto (Current password = ) show
readpassword 10 (           ) cvrs show currentpoint 24 sub exch pop 50 exch moveto
(Password is in 177-180, SN is 509-512) show currentpoint 12 sub exch pop 50 exch
moveto (Normal access eescratch starts at 187) show currentpoint 12 sub exch pop 50
exch moveto (Horizontal margin value long at 169) show currentpoint 12 sub exch pop 50
exch moveto (Vertical margin value   long at 173) show currentpoint 12 sub exch pop 50
exch moveto (an empty long at 165) show currentpoint 12 sub exch pop 50 exch moveto
(printer name starts at 132-164) show currentpoint 12 sub exch pop 50 exch moveto
(Brought to you, by Woody Baker       )show currentpoint 12 sub exch pop 50 exch
moveto (Rt.1 Box I, Manor, Tx. 78653   512-272-4511) show showpage} def

/readpassword {177 readeerom 24 bitshift 178 readeerom 16 bitshift or 179 readeerom 8
bitshift or 180 readeerom or} def

printeprom dumpeprom readpassword == flush quit
```

Fig. 5 – PostScript trashed password emergency repair.

and chomp and glue and score and fold and laminate and diecut and whatever to make all of the packaging products you see today.

Now, here's the kicker – most of those big rolls of stuff are cheap, especially on a square foot (or industry standard MSI "thousand square inches") basis. Those actual conversion costs are not all that expensive either, provided the job run is long enough to pay for the setup and any die costs.

But the person hiring the converter makes you pay through the nose. The reasoning is that "value added" and the "distribution" costs have to get paid somehow. The materials cost of just about anything you'd find blister packed in a hobby or craft shop drives this home.

For instance, a laminating plastic bought by the big unconverted roll should cost you around three cents per book or less. Should you hire a converter to do some cutting for you, another half cent or possibly a full penny would be involved. The very same sheet from your friendly neighborhood *Kroy* dealer will cost you seventy cents or more.

The bottom line here is simple: *Do all of your own converting!* Or else hire a converter to do so for you on a per-job absolute minimum basis. This eliminates a bunch of middlemen and distributors.

I'm currently in the process of evaluating several polyester laminating film sources. These include *Catalina Plastics*, *Multi-Plastics*, *Madico/FSK* and *Intermax Trading*. I do suspect that precut self-adhesive sheets will be much easier to work with than bulk rolls.

Sadly, converters are *very* hard to communicate with. They do speak their own language and live in their own little world. And seem rarely interested in anything new.

There's also lots of traditional lamination machines available, as can be used for badges, passports, maps, or diplomas. The plastics used here are usually much heavier and often much more expensive. They also are far from subtle and can end up looking on the gross side of gruesome. One source of these machines and supplies is *Printer's Shopper*; a second reasonably priced source is *USI Image Creators*. Watch for their loss leader sales, where machines get as low as $140..

As usual, more information on most of these products is available by way of our *Names and Numbers* appendix. Important trade journals for the converting industry are *Converting*, the *Paper, Film, and Foil Converter*, and the *Packaging and Converting* magazines.

For this month's contest, just send me an essay in 50,000 words or less on "The ways I protect and improve upon my toner images". There'll be all of those usual *Incredible Secret Money Machine* book prizes, plus an all expense paid (FOB Thatcher, AZ) *tinaja quest* for two going to the very best of all. As usual, send all your written entries directly to me here at *Synergetics*, rather than to *Computer Shopper* editorial.

This Month's PostScript Utility?

This month's PostScript utility was contributed by Woody Baker. Woody, as you regular *Ask the Guru* readers already know, is the developer of the *Bakerizing* process of the preceeding section, besides having authored many unique PostScript procedures and routines.

There are times and places where your PostScript password may blow up. Perhaps you may have forgotten it if you were dumb enough to ever change it from zero. Or possibly a flakey piece of software has trashed it. Or maybe a virus on your network has maliciously altered it.

Depending upon the machine, the consequences of a blown password can range from an inconvenience to an unmitigated disaster.

For instance, on those original *LaserWriter* and *LaserWriter Plus* machines, to very this day Apple will require around $1200 and several weeks time for a simple password reset. And with no guarantee whatsoever that the very same problem will not happen all over again three minutes later.

On the newer LaserWriter NTX, the problem is not nearly as bad. To force a password reset, simply flip to the *AppleTalk* position under power, wait half a minute, flip back to *Serial*, and repeat the process. This will reset your password back to zero. And, of course, trash out all of your serial settings in the process.

Even with no blowups, it is nice every now and then to be able to make sure that nobody is messing around with your machine.

So, this month's PostScript utility is an emergency trashed password repair program. Figure five shows full details. All this does is read your existing password back to your host, in addition to giving you a complete printed dump of the password itself and all of the 256 nonvolatile memory locations.

The code is Adobe specific, but does seem to work on most "real" versions of PostScript.

Once you verify your present password, you can easily reset it back to 0 by doing a…

```
serverdict begin
currentpassword exitserver
statusdict begin
currentpassword 0
setpassword end quit
```

This is a lot safer than blindly and automatically forcing the machine language reset by directly writing to the nonvolatile storage area.

You will find this file all ready to download as *GEnie* PSRT Library #188 PASWRDFX.PS.

Please let me know what you want to see in the way of future PostScript utilities. There's lots of stuff that I'd dearly love to do that just might not have a wide enough interest base to be worthwhile.

I hope to have a real heavy for you as next month's PostScript Utility – a complete and a sophisticated stock history investment analysis program. Which does use PostScript as language, rather than PostScript as a plain old paper marker. Be sure to watch for it. Or if you do want to get in ahead of the hoarders, check out our *GEnie* PSRT library files #221 and #223. ❄

Don Lancaster's

ASK THE GURU

June, 1991

PS Level II features
Stock history analyzer
Unusual print materials
New PostScript products
Improving toner durability

Expect bunches of brand new PostScript Level II printers just about the time you are reading this. I'm in the process of begging, borrowing, and actually buying some of these for thorough testing and debugging.

Since I strongly feel that it is flat out not possible to properly review *any* laser printer without spending six months while shoving 60,000 copies through the machine in a real end-user environment, this may take a while. And several printer builders seem less than totally eager to have all of their wares honestly reviewed. Especially those not now using *Canon* engines. But do stay tuned here for ongoing info.

1. – ASSORTED GOODIES

Level II code is inherently much faster than earlier versions, and speeds up even further with new RISC processors. Dictionaries now dynamically alter their sizes, eliminating *dictfull* errors. Erratic linewidths on repeated lines can be made uniform with a new *strokeadjust* feature. A new class of resources is available. Garbage collection and VM reclamation is now possible. New pattern features greatly improve tiling.

2. – BINARY ENCODING

A new ultra-dense and token based coding scheme is available that lets you place PostScript code in a compact form that still executes rapidly.

3. – COLOR ENHANCEMENTS

A number of new color formats are now supported including RGB, CIE, and CMYK. Halftone screens have been expanded to allow dramatically better control of full color images, especially on the high end.

4. – FILTERS

A filter is a super-emulator that puts itself between your data source and the printer. Some of the early filters include ASCII85, which is much more efficient than Hex ASCII; TIFF personal computer compacation; CCITT, which gives you instant fax coding and formatting, JPEG, which is able to compress color pictures by as much as 100:1; and RUNLENGTH, which compacts black and white bitmaps.

5. – FONT PATHS

Most font paths are fully recoverable for non-linear transformation as the font lockout on *pathforall* has finally (!) been removed. Amen.

6. – FORMS

Forms can be created and cached ahead of time and then get rapidly dropped in place. This is especially useful in step-and-repeat projects such as business cards.

7. – USER PATHS

Automatic proc caching is now possible for logos or other complicated routines that are to be reused. Instead of rebuilding the entire routine on each reuse, a cached bitmap is used instead. This can be much faster.

Fig. 1 – Some new features of PostScript Level II.

There are lots of new PostScript products this month. For any of you unfortunates that got snookered into those obsolete and dead end TrueType fonts, you could instantly upgrade these into genuine Type I Adobe PostScript by using a new *Ares FontMonger* product.

Any set of characters from any source, be it scanned, live video, bitmapped, third party, or hand sketched can be rapidly converted into first rate PostScript fonts by an extremely powerful and superb quality tool known as the *ATF Type Designer*, and currently available from *Kingsley-ATF*.

For an interesting extension to Hypercard II, check out the *Hyper PS Tools* by *Lupin Software*. This lets you send your HyperCard fields as real PostScript while receiving and displaying full feedback and error messages. A name tag and a button printing demo stack is included.

Adobe Systems has a great new ap note – a complete listing for the *JPEG Technical Specification, Revision 8*. This one shows you all of the key insider details needed to compress color images. Especially with Level II. Single copies are available at $15. As usual, nearly all of the sources referenced here appear in the *Names & Numbers* appendix.

There's a new *Lasersetter Owners Association* intended to support high end uses of PostScript. Besides the user services, they have an on-line BBS found at (415) 841-6302. Special thanks to *TypeWorld*, a great freebie trade journal for this tip.

And the usual reminders here that there is a great PostScript board up on *GEnie* as PSRT. Power users can use M835 to access this round-table and M836 for the download library. A pair of excellent PSRT X-ref and planners are now up as files #245 and #246.

I have also got a free *PostScript*

Insider Secrets mailer waiting for you should you call or write per the end blurb. Also note our ongoing no-charge PostScript helpline.

Speaking of which, by far the hottest helpline topic these days involves those of you asking…

Just What is So Awesome About PostScript Level II?

PostScript Level II is the newest standard in desktop publishing, and offers several really exciting new features. I've summarized some of these in figure one.

The good news is that the new 800 page "Red Book II," and otherwise known as the *PostScript Language Reference Manual, second edition* has full details on all the new Level II stuff, as well as for old PostScript, Display PostScript, EPS files, and the party-line document structuring conventions. The bad news is that the book is so awesome that it scares away beginners who just want to know a little about a few of the simpler PostScript commands.

At any rate, the Red Book II costs around $29 and should be available in your local bookstore. I also stock these as a service for you *Computer Shopper* readers.

I still can not reveal any precise speed results, but do stay tuned. In theory Level II should end up *much* faster than earlier versions, owing mostly to new techniques learned in the process of developing Display PostScript. And Level II will often be associated with RISC chips or co-processor speedup devices for even more speed.

On the other hand, many users today needlessly insist on *severely* baud rate limiting themselves. Most especially those PC users. PostScript execution speed is totally meaningless if your comm with your printer should end up baud rate limited. Improved resolution can also cost you dearly on print speeds. Some operations at 1200 DPI take *sixteen* times longer than at 300DPI. Thus, many printer designs may opt to give you a little extra resolution instead of a lot more speed.

There are zillions of secondary refinements in Level II, that picked up all of those goodies added on a catch-as-catch-can basis along the way. A lot more memory is now dynamically allocatable, so *dictfull* errors are now a thing of the past. Repeated fine lines can now be done far more uniformly through a new *strokeadjust* feature. And garbage collection to reclaim VM is now far easier and more powerful.

Two major new features that I have yet to explore are the unique *binary encoding* formats, which let you both store and communicate programs in vastly smaller space; and high end color enhancements which very greatly improve color separation and printing.

In the past, if you wanted to emulate something, you wrote your own emulator and then persistently downloaded it. Naturally, since PostScript is so incredibly powerful, it can emulate virtually any older printing format. Level II now adds what Adobe calls *filters* to the input and output communications and disk channels. You can think of a filter as a universal emulator.

There are some really exciting new filters. But by far the most profound of these is a fax filter that *directly* lets your PostScript printer send and receive top quality fax.

In fact, PostScript Level II can completely and ludicrously blow fax out of the water. A test can be made to find out if the other end is a quaint and outdated fax machine or a sleek new Level II printer.

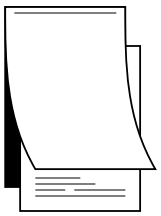
If a quaint oldie, the image gets sent at fax speed and resolution. Even here, the results will be *much* better than your usual scanned fax images. If a Level II printer, your image is instead transmitted as device-independent PostScript.

Which means that you can now send top quality *camera ready* art to arbitrarily high resolution and do so faster and cheaper than you could send a grubby old fax image. And do so anywhere in the world.

One more time: *With Level II, you can now transmit camera-ready color*

BAKERIZING

With BAKERIZING, your toner image is temporarily placed in close contact with a smooth film. Heat and pressure is then applied. The toner becomes much blacker and takes on a beautiful medium to high gloss. Some films can be reused.



LAMINATING

With LAMINATING, a thin clear plastic overlay is permanently attached to your copy by using heat and pressure. This process seems ideal for menus and book covers, or where extreme scuff resistance to "lock in" your toner is required.

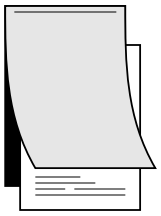


Fig. 2 – Two methods to improve toner durability.

```
%  Copyright c 1991 by Don Lancaster and Synergetics, Box 809, Thatcher AZ, 85552. Any and all
%  commercial rights reserved. Personal use permitted so long as this header remains present and
%  intact. LaserWriter Secrets book+disk $29.50 VISA/MC. Free voice helpline (602) 428-4073. Full
%  code and docs available on GEnie PSRT library #221 GENIEVST.PS and #223 GENIEVST.TXT.

%  This PostScript-as-language routine will take any GEnie log that has a stock history dump in
%  it and then return to your host a detailed investment analysis model of that stock history, simply
%  by following selected arbitrary and fully programmagle buy-sell rules of your choice.

%  WARNINGS: The only warranty, express or implied is APPROXIMATE QUANTITY ONE. Use
%  beyond a "fun with numbers" game is at your own risk. A two way RECORDABLE comm setup is
%  ABSOLUTELY ESSENTIAL for this PostScript program. Parallel ports need not apply.

%  There is purposely no graphic output here, although the graphical extensions that are possible
%  totally boggle the mind. Extensive comm delays have been included for host viewing and
%  recording. Eliminating these very much speeds things up. Everything is programmable.

% CURRENT RULES: sell on sellrise above closing; buy on inverse sellrise minus bidask below
% closing. Buy in hundred lots at or above buysize. Pay commissions, allow for OTC bid/ask,
% and dual rate interest. Interest calculations are approximate. Ignore taxes and capital
% gains cutoff. No short positions or dividends. Change agressiveness with market.

% USER VARIABLES

/percent {0.01 mul} def                 % allow comissions as percentage
/initialcash 10000 def                  % initial cash balance
/buysize 10000 def                      % approx buy purchase size (fixed for now)
/commission 1 percent def               % comission on a buy or sell (can be fancy proc)
/bidask 0.25 def                        % bid-ask spread if OTC
/sellrise 33 percent def                % sell on this rise; buy on inverse drop - bidask
/closeonly true def                     % use only closing prices?
/currentcash initialcash def            % initialize cash balance
/currentstock 0 def                     % initial stock balance
/interest+ 8 percent def                % interest rate for positive cash balance
/interest- 10 percent def               % interest rate for negative cash balance
/showinterest true def                  % show daily interest to host?
/showdailyreport true def               % show daily reports to host?
/totaldays 1000 def                     % more than number of days analyzed
/sharesheld 0 def                       % no shares to start
/maxshares sharesheld def               % start with zero
/worstcash initialcash def              % start with initial
/worstmarket initialcash def            % same
/bestmarket initialcash def             % same
/interest 0 def                         % running variable
/initialdelay true def                  % host turn around delay? (for IIe)
/guessmarket true def                   % try and second guess economy?
/lowtrip 23 def                         % minimim "normal" price range
/hightrip 35 def                        % maximum "normal" price range
/hiloadjust 25 percent def              % agressive/conservative factor

%  SERVICE ROUTINES

%  grabvalue grabs the data needed as strings in arrays. In this example, we want characters 0-5
%  read as a GEnie date, 6-12 read as the stock volume, 13-21 read as the daily high, 22-30
%  read as the stock low, and 31-40 read as the stock closing price
/grabvalues {{str dup 0 6 getinterval cvi datearray exch aindex exch put dup 6 7 getinterval cvi
volumearray exch aindex exch put dup 13 10 getinterval cvr higharray exch aindex exch put dup 22
10 getinterval cvr lowarray exch aindex exch put dup 31 10 getinterval cvr closearray exch aindex
exch put pop /aindex aindex 1 add def}if} def   300 string /buffer exch def

% report returns values to the host and provides an optional time delay
/report {print flush 50 {37 sin pop} repeat} def   /r1 {(      ) cvs report} def

% grabtoday gets the current array values for this date
/grabtoday {/datetoday datearray today get def /volumetoday volumearray today get def /hightoday
higharray today get def /lowtoday lowarray today get def /closetoday closearray today get def} def

% sumshares goes through the holdlist and totals the shares
/sumshares {/sharestoday 0 def 1 1 lastbuy { holdlist exch get /sharestoday exch sharestoday add
def} for /sharevaluetoday closetoday sharestoday mul def} def

% reporttoday reports the days activities back to the host ...
/reporttoday {sumshares showdailyreport {(date:      ) report datetoday r1 (r) report (shares held )
report sharestoday r1 (r) report sharestoday dup maxshares gt {/maxshares exch def}{pop} ifelse
(shares value: $) report  sharevaluetoday r1 (r) report (cash held:    $) report currentcash r1r) report
(market value: $) report sharevaluetoday currentcash add r1 (rr) report (next buyat ---> $) report
buyat r1 (r) report (next sellat --> $) report sellat r1  (rr) report} if currentcash dup worstcash lt
{/worstcash exch def}{pop} ifelse sharevaluetoday currentcash add dup dup worstmarket lt
{/worstmarket exch def}{ pop} ifelse dup bestmarket gt {/bestmarket exch def}{ pop} ifelse} def
```

listing continues as figure 3.b …

Fig. 3a – PostScript stock history investment analyzer…

separations and any final prepress art faster, cheaper, and more conveniently than using traditional fax!

The other filters are also quite impressive. There's a JPEG filter to compress high quality color images by as much as 100:1. A new ASCII85 filter can replace the old hex-ASCII transmission standard but is nearly twice as efficient. Several personal computer compaction filters are available, including TIFF. Plus plain old run length encoders similar to the previously secret one used for cacheing larger sized fonts.

Mercifully, the obscene font path lockout on *pathforall* has at long last been flushed. Most font paths are now instantly grabbable on Level II for such things as custom logos and non-linear transformations. Amen brudder. Hallelujah!

A unique new forms capability dramatically speeds up many jobs. It is sort of a super step-and-repeat. You can actually cache your entire form as a bitmap or as a compact runlength file and quickly reimage it as often as you like. The first time you image the form it images up in the usual manner. But at the same time a bitmap or else a runlength encoded copy goes into a cache for reuse. The reuse can be hundreds to thousands of times faster. Among other things, this should bypass the lack of a duplex copypage that is so horribly crippling the IId and IIId double-sided printers.

There's also a new feature called *User Paths*. These are sort of like forms in that your first use of any PostScript proc that you want to reuse also gets cached. This is very similar to the *proc cacheing* stunts we looked at in previous issues, but far more convenient.

Presumably, the Level II upgrade chips should be made available for most existing PostScript printers. Stay tuned for price, performance, and speed comparisons.

How Can I Make Toner More Durable?

I have long been saddened by toner being far less durable and not nearly as pretty as real ink. So much

so that I have finally decided to do something about it. I have been working very closely with two film conversion houses to come up with a pair of new products that very dramatically can improve both the appearance and durability of laser printed toner hard copy. And do so at an acceptably low end user cost.

The first of these is called *Golly Gee Mister Science Bakerizing Film* and is shown you in figure two. If you place a toner image in contact with an ultra smooth surface and then apply heat and pressure, an almost magic improvement takes place. What happens is the toner remelts and conforms to the ultra smooth surface. The result is a solid ultra-dense black image having a medium to high gloss.

Besides looking much better, the newly Bakerized image is far more durable. The process is related to the ferrotype drum pass that makes high gloss photo prints when using traditional darkroom techniques. The technical term for what is happening is *calandering*. You simply will not believe the quality upgrade the first time you see it. One stunned old-line printer recently said it looked almost like engraving.

To Bakerize, just put your toner hard copy in the GGMSBF carrier and then either (1) Route it back through your copier or laser printer while imaging a blank copy; (2) Shove it through an *Omnicrom* or *Kroy Color* machine; or else (3) Just iron it through a muslin pressing cloth or other thin fabric.

What's really amazing is that Bakerizing is essentially a zero cost process. Some films can be reused many times. I'm still searching for the best GGMSBF compromise between reuse and cost.

Bakerizing is absolutely ideal for business cards, for announcments, letterheads, and point-of-sale signs. But it can end up a tad excessive and overbearing if used routinely on any large blocks of ordinary finer print text.

The second new magic material is known as *Golly Gee Mister Science Laminating Film*, and is also shown

```
% buy first decides which price to use. That price plus bidask is rounded to the narest 100
% shares above buysize and goes into the holdlist. The holdlist pointer advances
% The cost of the shares plus the commission is subtracted from currentcash
/buy {closeonly {closetoday}{lowtoday} ifelse bidask add /carryprice exch def buysze carryprice div
100 div ceiling 100 mul /numshares exch def /currentcash currentcash numshares caryprice mul dup
commission mul add sub def /buyat carryprice sellrise 1 add div bidask sub def /selat carryprice
sellrise 1 add mul adjustsellat def /lastbuy lastbuy 1 add def holdlist lastbuy nushares put sellatlist
lastbuy sellat put reporttoday (r============r) report (BOUGHT ) report numshaes cvi r1 ( shares
at $) report carryprice r1 (rr) report grabtoday reporttoday /numbuys numbuys 1add def} def

% sell sells the last stock, pays a commission, credits currentcash, and reports ..
/sell {lastbuy 0 gt {holdlist lastbuy get /numshares exch def /lastbuy lastbuy 1 sb def /sellat sellatlist
lastbuy get def closeonly {closetoday}{hightoday} ifelse /saleprice exch def salepice numshares mul
1 commission sub mul /currentcash exch currentcash add def saleprice sellrise 1 ad div bidask sub
/buyat exch adjustbuyat def lastbuy 0 eq {sellatlist 0  sellat put } if (r============r) report (SOLD )
report numshares cvi r1 ( shares at $) report saleprice r1 (rr) report grabtody reporttoday /numsells
numsells 1 add def}if} def

% summaryreport summarizes the total trading sessions for the history
/summaryreport {(rr) report (maximum shares owned at one time ----> ) report mxshares r1 (r) report
(best market value at any time -------> $) report bestmarket r1 (r) report (wrst market value at any
time ------> $) report worstmarket r1 (r) report (worst cash position at any tme -----> $) report
worstcash r1 (r) report (total completed trades ----------> ) report numsels cvi r1 (r) report
(remaining open buy positions ---> ) report numbuys numsells sub cvi r1 (r) reort (rr) report} def

% gottoend? decides when you are done scanning the currentfile.
/gottoend? {dup (%!!%) search {pop pop pop exit }{/str exch def validate grabvales} ifelse} def

%  validate throws out incorrectly formatted lines. This example uses a 71 charactr line
%  with spaces in locations 5, 60, and 67, and a valid number in locations 4, 59, nd 66.
/validate {pop str length 47 eq {true}{false} ifelse {str 6 get 32 eq str 13 get 3 eq and str 22 get 32 eq
and str 5 get dup 0 48 add ge exch 9 48 add le and and str 12 get dup 0 48 add ge xch 9 48 add le
and and str 21 get dup 0 48 add ge exch 9 48 add le and and }{false} ifelse} def

%  guessmarket tries to decide whether today's stock is "high", "low" or "normal" nd makes
%  you more aggressive at low times, more conservative at high times.
/adjustbuyat { guessmarket {closetoday lowtrip lt {hiloadjust sellrise mul 1 add dv} if closetoday
hightrip gt {hiloadjust sellrise mul 1 add mul} if} if} def /adjustsellat {guessmaket {closetoday lowtrip
lt{hiloadjust sellrise mul 1 add mul}if closetoday hightrip gt{hiloadjust sellrisemul 1 add div}if}if} def

%  MAIN GRABBER LOOP

%  analyzestock is the main outer loop that scans the input textfile, strips out dta values,
%  creates numeric arrays of those values, and then analyzess the results.
/analyzestock {save /snap exch def totaldays dup array /datearray exch def dup arry /volumearray
exch def dup array /higharray exch def  dup array /lowarray exch def dup array /clsearray exch def
pop /aindex 0 def currentfile {dup buffer readline {gottoend?} {exit} ifelse } loo pop pop  datearray 0
aindex getinterval /datearray exch def volumearray 0 aindex getinterval /volumearry exch def
higharray 0 aindex getinterval /higharray exch def lowarray 0 aindex getinterval /owarray exch def
closearray 0 aindex getinterval /closearray exch def initialdelay {2000 {37 sin po} repeat} if
doanalysis snap restore } def

%  STOCK INVESTMENT ANALYSIS PROCESSING CODE

%  doanalysis takes the values in datearray, volumearray, higharray, lowarray, andclosearray
%  and models an investment strategy following your selected rules ...
/doanalysis {300 array /holdlist exch def 300 array /sellatlist exch def /lastbuy  def holdlist lastbuy 0
put sellatlist lastbuy -99999 put /today 0 def /buyat 999999 def /sellat -999999 df /numbuys 0 def
/numsells 0 def /oktoselltoday true def grabtoday buy 1 1 aindex 1 sub { /today exh def grabtoday
lastbuy 0 ge {closeonly {closetoday}{hightoday} ifelse sellrise 1 add div bidask sb adjustbuyat dup
buyat gt {/buyat exch def showdailyreport {(buyat ^ to $ ) report buyat 100 mul ound 100 div r1 (r)
report } if}{pop} ifelse} if closeonly {closetoday}{lowtoday} ifelse buyat le {buy /oktoselltoday false
def} if oktoselltoday {closeonly {closetoday}{hightoday} ifelse sellat ge {sell} i} if /oktoselltoday true
def currentcash 0 gt {interest+}{interest-} ifelse 250 div currentcash mul  /interst exch def
/currentcash currentcash interest add def showinterest {(Int: ) report interest 00 mul round 100 div r1
(r) report } if} for (r================) report (r) report (TRADING SUMMARY) report (rr) report
/showdailyreport true def reporttoday summaryreport} def

%  Your stock history file gets inserted below, immediately following the -analyzetock- command,
%  but before the required %!!% marker. Usually, your ENTIRE GEnie log can be used

analyzestock  % this one line does it all!

 DATE    SALES    HIGH       LOW      CLOSE     AMOUNT   TYPE
890102    1440    25.000    24.500    24.500      --
890103    1474    25.000    24.500    24.500      --
 --- stock history to be analyzed goes here ---
910301    7180    48.000    46.500    46.750      --

%!!%  % required marker
quit
```

Fig. 3b – …PostScript stock history analyzer, concluded….

you in figure two. This film places a thin low gloss plastic overcoat on top of your hard copy, locking the toner inside. The effect is vastly more subtle and better looking than traditional plastic laminating from the office supply store. Durability is high and the glare can be low.

Obvious uses for the laminating film are for menus, booklets, and book covers, especially any of those published on a Book-on-Demand basis. You can use the GGMSLF the same way you do the Bakerizing film, except that the entire film sheet gets permanently bonded to the surface of your hard copy. The feed sheet and edges are trimmed away as needed.

Yes, you can Bakerize and then laminate for the ultimate in gloss, blackness, and protection. It just takes longer to do, that's all.

At this writing, the costs have not yet been finalized, but I hope to keep things on the cheap. Mostly by eliminating several tiers of market distribution. You can write, call, or *GEnie* [SYNERGETICS] email me for some free evaluation samples and pricing. I'd also like your help with ongoing end user testing.

Whaddya Mean I Can't Have Blue?

We might call this one *Bizarre Laser Printer Experiences #349*. Let's start with the bottom line: There are a lot of materals you could shove through a laser printer. Some of them will always work. Others will never work. And yet others...

We've seen how *tyvek* sleeves or unsupported acetates are absolute no-nos, and how most laser printed envelope quality sucks and should continue to do so until such time as the envelopes are redesigned as a single thickness that is specifically intended for laser printing. And we have seen that plain old offset paper is cheap, opaque, and looks great for Book-on-Demand uses.

We have also seen how most of the parchments print beautifully for awards and certificates, although a few of them are just plain too greasy to accept toner. And how certain thicker materials will feed just fine, but others will jam badly.

I've long been enamored of those great *Die-O-Perf* die cut products, especially their PM response card auto-mailers. Well, recently, my ivory brochures would laser print just fine, but the blue ones would misfeed often enough to end up unacceptably costly and frustrating. While these materials measured identical in thickness and felt the same, you could actually *hear* the difference as they fed through the registration assembly. Swish versus skloosh. While a fresh leading edge trim cut sometimes helped, it was definitely not a cure.

Die-O-Perf was certainly more than helpful, but you couldn't really expect them to strongly support a product that was misapplied in an unintended use. Their hint to make sure you feed the non-perforated edge first would be a good one for most users. But I'm running on a duplex IID, and there is no leading edge, since a die cut form reverses itself in the switchback path.

The cure? Switch from the blue ones over to the gold ones. Nobody seems to know why, nor how long this problem will remain.

So, experiment all you like with printing materials. But recognize that those products which are not specifically laser rated might often cause unexpected problems months or even years into use.

Two additional sources of several interesting materials that may or may not end up laser printable are *Paper Plus* and *Paper Direct*.

What is This Month's PostScript Utility?

As we've seen a number of times before, PostScript is a totally general purpose language that can easily hold its own against any and all the modern contenders. To refer to PostScript as a "page description language" is a gross and demeaning insult roughly comparable to calling UNIX a "checkbook balancer."

To prove this, I set out to write a super heavy duty and incredibly powerful PostScript utility that has *nothing* whatsoever involved with putting any marks upon any pages. Yes, you could do the same thing in C or BASIC, but I found that writing in PostScript was far easier, a lot more fun, and considerably more flexible than using any of those traditional languages.

Not to mention being fully device independent. This code runs on *any* host computer without mods.

And, should you chose to, the graphical extensions you could add to this program can be positively mind-boggling. I purposely left any graphics off to prove a point.

Since use of this program can instantly make you filthy rich, we also do have to add the following disclaimer: The only warranty here, expressed or implied for what is to follow, is *approximate quantity one*. Any use of what follows beyond a "fun with numbers" game is solely at your own risk. Although I did personally profit immediately the very first time I ran the program, simply by correcting some stupid mistakes I'd been making.

What we have here is a stock trading analysis package that I call GENIEVST.PS. You throw any old stock history at it, and it applies whatever trading model you like to that history, reporting your profits and losses back to your host for recording and analysis. Key parts of the PostScript code listing appear in figure four, while excerpts from a sample trading session is shown you in figure five. You can also get the listings ready-to-run, better documented, and in a more "open" form as *GEnie* PSRT #221 and #223.

There are two portions to this code. The first portion takes a stock history from a data base or word processor format and converts it into internal data arrays. And the second portion applies the trading model to those arrays, following any set of trading rules per your choice. Naturally, the best way to test a trading model is to run it first on target stock histories.

PostScript is great at formatting text, so your input stock history can be in pretty near any database or

textfile form. In this example, we've used the normal *GEnie* stock history download format. The filtering in the code is so good that you can simply throw your entire *GEnie* log file session at it, and it will automatically fish out the stock history and throw everything else away.

In this specific grabbing example, my PostScript *validate* proc ignores any textfile lines that are not 71 characters long, do not have spaces in locations 5, 60, and 67, or hold valid numbers in locations 4, 59, and 66. This rejects unwanted lines.

The trading model can be nearly anything you want it to. In this example, we are buying in hundred lots, dollar cost averaging, selling on a *sellrise* percentage increase and buying on an inverse sellrise drop. Should a stock wander below its "normal" trading range, you get more aggressive over buying and less eager to sell. Similarly, get above the "normal" trading range, you sell more and buy less.

A variety of report formats is available, depending on the detail you need. You can also easily add fancy charts and graphs if you want to. At present, the host-reported results are intentionally slowed down to a good reading speed. You can dramatically speed things up if you really care to.

All of this is handled with simple pocket calculator arithmetic easily done by PostScript. A dual interest rate is used, a low one for surplus cash and a higher one for borrowed funds. Everything is fully programmable, so you can easily rearrange the scenery to suit yourself.

Oh, yes. You do need a two-way and recordable comm setup on your PostScript printer for this program to work. Use *Applewriter* [esc]-R on the IIe, *SendPS* on a Mac, or serial *Xtalk* or *ProComm* on a PC. Needless to say, you should *NEVER* use any parallel port with PostScript, since it blindfolds you and ties both of your feet together. Really dumb.

The first thing to try with your model is to decide to become more conservative and far less greedy while accepting lower percentage returns. Select a sellrise value of 4 percent and see what happens. Very simply, you will get taken to the cleaners. At one point you end up down by $140,682.00 and owe nearly $15,000.00 in comissions. Thus, by not being greedy enough, your risk factors and commissions skyrocket for only a modestly higher return.

My first use of this code drove home the fact that larger and longer term gains are the ones to shoot for, since they are vastly more profitable to you at far lower risk.

The astounding result: If you are not greedy enough, you'll lose and lose bad in a stock trading program. I suspect this has to do with there normally being more "power" or "energy" in those lower frequency components of any complex waveform, random or otherwise.

This is an extension to the *matched filter* techniques we looked at in my *Incredible Secret Money Machine.*

Oh yeah. The particular stock analyzed in figure five is good old *Adobe Systems.* With an optimum analysis strategy, I've found ADBE to consistently yield at least a 70 percent annual return, year in and year out. Plus dividends. Some of my other current favorites include SGAT, CHPS, AAPL, ASA, or, if you are interested in quite high risk bottom feeding, CKP. ❄

```
BOUGHT 500 shares at $24.75

date:          890102
shares held:      500
shares value:  $12250.00
cash held:     -$2498.75
market value:   $9751.25

next buyat --->   $18.36
next sellat -->   $32.92

Int: -1.0
buyat raised to   $21.93

BOUGHT 500 shares at $21.00
BOUGHT 700 shares at $15.75
SOLD   700 shares at $22.75
SOLD   500 shares at $30.25
SOLD   500 shares at $36.38
BOUGHT 300 shares at $36.75
SOLD   300 shares at $46.50
BOUGHT 300 shares at $35.50
BOUGHT 400 shares at $30.25
BOUGHT 500 shares at $22.25
SOLD   500 shares at $33.00
SOLD   400 shares at $40.25
SOLD   300 shares at $43.75
================
TRADING SUMMARY:

date:          910301
shares held:        0
shares value:      $0.00
cash held:     $37822.00
market value:  $37822.00

maximum shares owned at one time ---->    1700
best market value at any time ------->  $37822.00
worst market value at any time ------>   $1837.02
worst cash position at any time -----> -$24513.0
total completed trades -------------->       7
remaining open buy positions -------->       0
```

Fig. 4 – Excerpts from a typical trading session.

Don Lancaster's

ASK THE GURU

July, 1991

The latest printer intros
Mac LC monitor options
Working with limited vm
A PostScript menu justify
Networking laser printers

Let us shout this one from the rooftops – That Apple IIgs RGB color monitor A2M0014 absolutely, and positively can *NOT* be used with your Mac LC. Which could add as much as another $1000 into your changeover costs.

Those IIgs monitors operate at normal television horizontal scan frequencies of 15735 Hertz, while most newer color computer systems do require a substantially higher horizontal scan rate.

Yes, the connectors are the same size and shape. But the connector pinouts and the scan frequencies are totally different. And hardware modifying the horizontal scan rate of a color monitor is both a horrendous and dangerous job.

You can use a VGA color monitor with your LC by going to a special cable. But, since the quality and the resolution of these vary all over the lot, do be certain you see what the actual LC output looks like before you buy one. We may look at the VGA cable in a future column.

The LC does seem to be selling quite well. Already, there are new software patches that can let you run *Excel*, as well as plug-in cards that upgrade you to the 68040 and the math co-processor found in all the bigger Macs.

At this writing, though, Apple is still not shipping their Apple IIe card for the LC. And I still have a hollow feeling that this card will not run AppleWriter. Especially for any two-way LaserWriter comm at a 59,600 rate. Or in the modem record mode. Stay tuned on this.

There are also still rather serious problems with Apple mailing list rentals. Particularly their LaserWriter printer user list. Apparently Apple simply refuses to admit how bad they are being taken by their third-party list brokers. My testing of the "new" and "improved" list showed that it was *never* cleaned, is *not* being maintained, and is utterly ancient. The nixies were apalling. And the response to a well tested mailer was zero.

If you do rent an Apple list, be sure to run the smallest possible test into your tightest possible target market. Even then, expect a lousy response. And be certain to get a written guarantee that includes the date of the last list cleaning.

A big thanks to *Hewlett Packard* for all the technical data they have been sending me recently. H-P does have a *Peripherals ISV/IHV* program similar to being an Apple or Adobe developer. You might contact them per the *Names and Numbers* sidebar for more details.

Our usual reminders here that we have this great Guru and PostScript BBS up on *GEnie* as PSRT, and you

1. Always do a fresh boot when running fat code. Put nothing in the machine that you do not need.
2. Minimize the number of downloaded fonts. Do not download any font you are not actually using. Fake italics by leaning the normal font.
3. Limit persistent downloads to only those tools you actually need. Trim any excess fat off those tools.
4. Avoid using re-entrant code. Code that wrongly calls itself is a major cause of beginner vm blowups.
5. Use extensive save-restore bracketing. The usual method is…

   save /savesnap exch def
   your stuff
   savesnap restore

   The two main **invalidrestore** causes: (1) failing to have a save object on the top of the stack, and (2) leaving any new dictionaries open.

   You can also nest save and restores to break up something that needs lots of vm into several somethings that only need a fraction as much.
6. Use **vmstatus** command every now and then to measure your limits.
7. Arrays (8 bytes per element), Dictionaries (20+ bytes per key pair) and Strings (1 byte per element) gobble up vm. Minimize their sizes.
8. Whenever possible, reuse a dictionary, array, or string.
9. Note that the **put** command usually rewrites an old definition, while the **def** command creates a new one.
10. If a deferred proc has too many elements for the stack, the usual cure is to do a **(all the proc stuff) cvx exec**. The string can be up to 64K long.
11. The ADOBE DISTILLERY inherently uses lots of vm. Distill single pages rather than whole documents.
12. If at all possible, put extra memory into your printer. Most SCSI hard disk systems blow up at 2 Megs, but work fine at 3 Megs.
13. If only a few special font characters are needed, grab the specific font paths, rather than using an entire font definition.
14. Be willing to test and experiment to get at the problem.

Fig. 1 – Workarounds for limited VM PostScript printers.

can get your voice connect info via (800) 638-9636. You could also reach our no-charge *Ask the Guru* voice helpline per the appendix. And we now do have both the *Ask the Guru III* and *LaserWriter Secrets* reprints freshly updated.

And, of course, I still have a free *PostScript Insider Secrets* brochure waiting for you whenever you call or write.

What is new in PostScript Printers?

There's a bunch of brand new PostScript laser printers fresh out from *H-P, Apple, QMS*, and others. And I will certainly be testing some of them on a long term basis for all you *Ask the Guru* readers. But, sight unseen, I am sorely disappointed with the entire lot. In fact, I'll call this sorry batch the "lost generation" of PostScript printers.

PostScript level II is as stunning an improvement over PostScript as PostScript was over everything else. But not one of this current crop uses it! At least not yet. To me it makes far more sense to go with what you have or else pick up some older machine now, and then get a good one when Level II becomes widely available. For level II will surely obsolete anything earlier.

Besides much faster speed, level II includes a full fax capability. The programming details are in the new red book II, which I now do have in stock for you. Word has it that the first actual level II machine will be a *QMS* color job that should be out "real soon now".

One very positive feature of some of the new printers: Duplexing, or double sided printing, is apparently going to become a low cost ($600 list, $400 street) bolt-on option. As we have seen, a duplex printing is essential for any Book-on-Demand publishing, and extremely handy otherwise. Fun, too.

Making duplexing a low priced option to a high volume machine makes far more sense than using a separate model. As with earlier printers, you don't really print on both sides at the same time. You print one side, grab the sheet and route it back through a switchback assembly and then print the other side on the second pass.

That HP IIIsi probably has the highest profile of the new machines. Wide open, it runs at 17 pages per minute and uses new double life cartridges having a microfine and ultra black toner in them. It also has the new resolution enhancement technology, which is a simple dot modulation that can dramatically improve certain forms of 300 DPI output. Both a duplex bolt-on and a full internal Adobe PostScript are available as options. A fast RSIC chip is used.

But I can see several good reasons for not jumping out and buying a IIIsi just yet. It is definitely *not* level II and uses the same old wimpy PostScript 52.2 of those earlier cartridges. Complete with lousy serial comm and a *copypage* that is not side sensitive in duplex mode.

And no, you can *NOT* use any existing PostScript cartridge with this machine, since the CPU is no longer Dr. Moto.

Appallingly, there is no SCSI hard disk support for font cacheing, font storage, or even for any Book-on-Demand production. Which seems an utter and total absurdity in a high end 17 PPM PostScript printer.

To the possibly fatal flaws of no level II and no SCSI disk or comm, we can add several obvious minor ones. It will take the rechargers a while to get cheap reloads for this machine. In an ugly new surprise, an improperly formulated microfine refill toner can explode.

The earliest sources of low cost refills are likely to appear as ads in *Recharger* magazine.

The IIIsi street price is certain to

Fig. 2 – A precise menu justify example.

drop after all the dust settles. And bunches of other new machines are about to be released.

But these are all my first "sight unseen" impressions. We'll most definitely be seeing lots more on the IIIsi as it currently seems to be running away with all the marbles.

Can a PostScript Printer Be Used on a Network?

Well, I certainly don't think so. And I have no idea where such an outlandishly silly myth could ever have started.

As we have seen several times before, if you cannot reach out and physically touch your PostScript printer from your work station, its usefulness to you will drop dramatically. Several of the obvious reasons for this include paper jams, stock selection, quality monitoring, cartridge refilling, laminating, density adjustments, Bakerizing, hard disk or shared SCSI comm access, envelopes, control of any persistent downloads, error debugging, VM management, etc...

With the big exception of display PostScript, what you see on your screen is probably not what you will get on your page. So you do have to actually look at your final copy to be certain it is what you expected.

More important, PostScript is a totally general purpose computer language that *demands* instant and full two-way recordable host communication and interaction. Some networks (particularly those that include print buffers or servers) either outright eliminate or else severely restrict your host return channels. Many better PostScript programs simply will not run on some networks. Examples include the popular *Adobe Distillery* or my GENIEVST.PS stock analyzer.

Worst of all, a network is often used as a lame excuse to buy one expensive printer rather than two or more cheaper ones. There are a number of very good reasons why two cheap 8 PPM PostScript printers will blow the fusion rollers off one expensive 16 PPM machine in most network environments.

First, nearly all of the time, the overwhelming majority of network PostScript users will end up being severely baud rate limited. For instance, the 57600 baud game paddle port of an Apple IIe can send PostScript nearly *four times faster* than AppleTalk's 17200 kilobaud effective baud rate usually found on smaller Macs.

Baud rate limiting causes a slow printer or a fast printer to output in precisely the same click-to-clunk time. More often than not, two networked 8 PPM printers will end up at least *twice* as fast as one 16 PPM machine.

Second, your typical PostScript network jobs tend to involve lots of really short tasks mixed in with a few very long ones. If a long job is running, all of the short ones still have to wait. If you have two or more printers, the long jobs and the short jobs can all run at the very same time.

Third, the tasks of multiple cheap printers can be specialized. You can use one printer for letterhead and envelopes only. Or economically use your older or slightly defective cartridges for any rough drafts and in-house uses.

Fourth and obviously, if you have two printers and one goes down for any reason, you can still stay on the air. And still print now.

If you feel you absolutely must network some PostScript printers, be sure to use a "cluster" or a "star"

```
%  Copyright c 1991 by Don Lancaster and Synergetics, Box 809, Thatcher AZ, 85552. Any and all
%  commercial rights reserved. Personal use permitted so long as this header remains present and
%  intact. LaserWriter Secrets book+disk $29.50 VISA/MC. Free voice helpline (602) 428-4073. Full
%  code and docs available on GEnie PSRT library #276 MITZIYUP.GPS, #220 GUTIL13A.PTL,
%  #219 GONZO13A.PTL, and #220 GONZO13A.TXT. Free printed copies on request.

% Note: This code DEMANDS persistent downloads of GONZO13A.PTL and GUTIL13A.PTL

gonzo begin /endtheline {/curwide txtwide roomleft sub def justx cvx exec oktoprint {printline} if} bind
def /cw {save /snapc1 exch def /oktoadvance false def /oktoadvance false def /linestring linestring2
def /justx (justL) def 3 1 roll /ypos exch def /xpos exch def stringgonzo curwide snapc1 restore} def
end

/menudots (. ) def /menufont {font1} def /mdoteht 3 def /menudelim (   ) def /cropleadingspaces true
def

/drawmdots { gsave  menufont xxm yym moveto txtwide menudots stringwidth pop dup /mdot1 exch
def div floor cvi {menudots show} repeat grestore } def

/spchomp {cropleadingspaces {dup 0 exch {32 eq {1 add}{exit} ifelse } forall} if exch dup length 2
index sub 3 -1 roll exch getinterval} def

/mlineproc {mline length 0 gt {drawmdots mline menudelim search {/lmstr exch def pop spchomp
/rmstr exch def  } if gsave 1 setgray xxm yym moveto mdoteht setlinewidth 0 0 lmstr cw mdot1 div
ceiling mdot1 mul 0 rlineto stroke xxm txtwide mdot1 div floor mdot1 mul add yym moveto 0 0 rmstr
cw mdot1 div ceiling neg mdot1 mul 0 rlineto stroke grestore xxm yym lmstr cl xxm txtwide add yym
rmstr cr /yym yym yinc sub def} if} def

/cm {gsave  /msg exch def  /yym exch def /xxm exch def {msg  ( \ n)
 search {/mline exch def pop /msg exch def mlineproc} {/mline exch def mlineproc exit } ifelse} loop
grestore} def

%  //// demo - remove or alter before reuse. ////

gonzo begin gutility begin /txtwide 240 def /cstretch 0.1 def /sstretch 0.5 def /yinc 26 def /font1
/Palatino-Roman 12 gonzofont /font2 /Palatino-Italic 12 gonzofont font1

125 365
(Knockwurst Ala King                    $3.27
Hot Clam Sundae                         $1.15
Coconut Anchovy Pizza                   $8.35
Butterscotch Pudding |2au gratin|1      $1.25
Pastrami & Kiwi Blintz                  $4.50
Sopapillas, Enchilada Style             $2.46
Grape Chili |2au jus|1                  $4.45
A|kvocado & Liver Surprise                .15
Chocolate Sushi                        $25.50
) cm

showpage quit
```

Fig. 3 – Excerpts from my gonzo menu justify routines.

arrangement of your workstations, so that *every* user can reach out and physically touch the machine.

Measure your actual comm times to find out how often and badly you are being baud rate limited by your network. Then do something about it, such as switching to shared SCSI comm at 650,000 baud. Finally, be certain that your network setup does not in the least interfere with any PostScript computed results being instantly returned to a host for recording or display.

How Can I Save VM?

Our no-charge PostScript helpline sure has been getting a lot of calls on those $995 rebuilds and retrofits of the older Canon CX laser printers. Popular sources include *The Printer Works* and *Don Thompson*.

In general, these "bargains" are remarkably comparable to a $995 hot tub. They look good in the ad and on the dealer's showroom, but you definitely would not want to ever put water in one or have anyone actually sit in it.

Bluntly, you get between *ten* and *twenty* times the performance with a $1990 PostScript printer today than you can get for $995. The price may be there, but the bang-for-the-buck definitely is not.

The key problems include (1) the frustratingly slow speeds, (2) vm blowups, and (3) all of the inherent limitations of host computer based PostScript. On the other hand, the CX engine does do a better job on covers and business cards than most newer machines.

As of this writing, I strongly feel that by far the best bang-for-the-buck remains the *QMS* PS410.

At any rate, there are lots of times and places where you can run out of memory in your PostScript laser printer. Even though these older CX machines are the worst offender, if you are not careful, you can easily run out of memory in *any* PostScript printer of any size.

Figure one summarizes several of the tricks you can pull to conserve limited memory, especially on older or gutless machines.

The available RAM in most any PostScript printer is usually split up into four major pieces: A *font cache*, the *framedevice*, the *system reserve*, and *virtual memory*, or simply *vm*. Your virtual memory gets used up any time you persistently download a routine. Or a font. Or anytime you define a string, any array, or any dictionary entry.

In early versions of PostScript, there is no "garbage collection" as such. Except that you can bracket high memory usage with a *save* and a *restore* to recover nearly all the memory needed between the two. In extreme cases, you can even use multiple saves and restores to try and reuse the same portion of vm over and over again.

Key secrets to working with limited vm include doing a fresh reboot on any problem code, not downloading what you do not need, avoiding reentrant code, keeping strings and arrays and dictionaries as small as possible, and carefully using *save* and *restore* constructs to continually reuse what limited vm you do have available.

Naturally, if any extra RAM can be added to your machine, you might want to do so. Most SCSI hard disk systems tend to be flakey and blow up at 2 megs, but work just fine at 3. And extra RAM is not all expensive these days.

Good sources for RAM are those ads in *Computer Reseller* magazine.

What is This Month's PostScript Utility?

Every once in a while, some typesetting job comes up that is trivial to do on a plain old typewriter, but seemingly difficult to handle well in PostScript. A good example of this is a *menu justify*. In a menu justify, you have a left text message, a row of dots, and a right text message. Programs, catalogs, directories, and price lists also will often need a menu justification.

Usually you want your left and right messages to be shown in full proportional mixed and kerned fonts, but want your dots to remain fixed pitch. Ideally, you want the spacing and the alignment of the dots to be constant. And you want whole dots only.

While simply fill justifying a line with a bunch of dots in it can work, deciding how many dots to use gets tricky and may require more than one pass. And the decimal points on your prices may wobble.

Figure two shows you a typical high quality menu justify. Note that all the dots are precisely vertical aligned and precisely spaced. There are no partial dots and no crowding of any characters. And it all comes down in one easy pass.

Figure three shows you some sample code. While this is written to make full use of my *Gonzo* justify routines found on *GEnie* PSRT (800) 638-9636 and in all my *Ask the Guru* reprints (602) 428-4073, you might easily adapt the ideas to any raw PostScript code of your chosing.

The key secret is to put down any and all possible dots first in *all* dot positions. Then you erase all of the dots that you do not want.

When erasing, you carefully erase only up to a dot boundary, to prevent any partial dots. You do this by measuring the actual width of the kerned and mixed proportional message, and then rounding it up to the next whole dot increment.

If you care to, any character or dingbat of any size could get substituted for a plain old dot.

You only have to enter your left message and right message for each entry, separated by at least two spaces. Any extra spaces nicely get ignored. Forget about entering the dots. They are fully automatic. Very handily, the same routine can even auto-convert ordinary lists from any source into full menu justification.

Border? What border? Oh, you like that huh? We'll save this gem for some other time. Suffice it to say that you are looking at a mere 170 bytes of raw PostScript that completely blows away most of those illustration and drawing packages. Should you want an advance peek, check out our new ready-to-run MITZIYUP.PS on *GEnie* PSRT, or else write or call me for a free listing. ❄

Don Lancaster's

ASK THE GURU

August, 1991

Low volume book publishers
That Apple IIe emulation card
VGA monitors and your Mac LC
Incredible Secret Money Machine
A serendipitous PostScript border

The Mac LC does seem to be selling quite well. Several obvious LC shortcomings are now getting filled with third party products. For instance, low cost math co-processors that now let you run *Excel* and otherwise speed you up are available from *Quantum Leap Systems*, as the *Crunch-It* from *PSI*, *QuickMath/LC* from *Novy Systems* and the *Apex* by *Second Wave Inc.*

At the same time, Apple is now shipping extra regular and video RAM in its stock LC offerings. First for system 7 and secondly to extend the colors available to 256.

The really big LC question, of course, is...

How Good is the Apple IIe Card for the Mac LC?

Well, Apple is now shipping the IIe emulation card for the LC. I have had one for a month now, and have been thoroughly trying it out. The bottom line: At the present time, the LC/IIe emulation card is not even remotely in the same league as a real Apple IIe when it comes to quickly and unklutzedly running Apple IIe software, for accessing networked school lab software, for error free telecomm, or for use for any high speed PostScript Book-on-demand production. After my full month of testing, I've now loudly voted "nay" and have gone back to using my good old IIe.

Worse yet, any IIGS owner who wants to "upgrade" will be rudely surprised when they find out that their color monitor can not be used on the LC, and that they may have to buy an additional and highly oddball single sourced disk drive.

A summary of my Apple IIe card discoveries appears in figure one.

There are nine assignable "virtual cards" that are surprisingly well done. These include a Super Serial Card for a printer, a second Super Serial Card for a modem, and an Appletalk card. As is usual, only two of these are permitted to be active at any time.

The other virtual cards include a video card permanently set in slot three, a mouse card, a clock card, a serial RAM card, and the usual disk drive cards in slots five and six.

Yes, ProDos *Applewriter* does run, even in its modem receive mode. And even with its internal printer baud rate settings. Sort of. But not for me. First, I had been going out the game paddle port at 57600 baud for all my LaserWriter serial comm. This can be four times faster than AppleTalk on any older Mac. Sadly, there are not any LC annunciator outputs. Which severely slowed my PostScript printing down.

All of my PostScript return error messages and any return recordable data get garbled by my LC. Even at 19200 baud. The apparent cause is a

1. The IIgs color monitor and most television receiver/monitors can not be used with the LC because of the higher horizontal scan rates.
2. The internal hard drive is not accessible from the IIe side, nor is the external SCSI connector. This prevents LaserWriter shared SCSI comm.
3. Network file servers are not accessible from the IIe side, especially AppleShare. This is a fatal flaw for most educational users.
4. The first external 3.5 drive apparently must be the obscure, obsolete and expensive Unidisk 3.5. The regular 800K 3.5 Mac/IIgs drive does not seem to work as the first external drive. At least not for me.
5. Space parity is no longer available, making use of LaserJets and QMS printers tricky with older unmodified "high ASCII" software.
6. The annunciator outputs are missing from the game paddle connector. These are often used for LaserWriter serial comm at 57,600 Baud, or two to four times faster than AppleTalk. PostScript Book-on-demand printing thus takes much longer on an LC than on a real IIe.
7. Incoming characters are dropped from popular communication programs and PostScript error messages, especially when using color monitors or multiple gray levels. A slow screen scroll seems to be the culprit.
8. Operating speed in the "fast" mode is somewhat slower than an accellerated IIe for most uses. For screen intensive operations, the screen updates are intolerably slow and far more ungainly.
9. Crucial keys, including escape, closed apple, and ctrl are now in wildly different and super klutzy locations. You can, however, redefine some keys or use a IIgs keyboard. Except for the %$#& caps lock.
10. Printer cables are not generally swappable. The LC usually demands "direct" or "modem" cables, while the IIe usually requires "modem eliminator" or "printer" cables. Connector styles also differ.
11. There are no real slots, preventing such things as quick EPROM programming, home power control, speech generation, or any of hundreds of other creative and innovative IIe uses.
12. Minor gripes: No eject button on internal disk. No "reverse the lid" copy stand. Some older copy protection schemes are unreadable. Emulation card slot assignments seem unnecessarily restrictive.

Fig. 1 – My critique of the Apple IIe card for the Mac LC.

slow and ungainly scrolling routine that drops groupings of characters after each carriage return.

There is another AppleWriter problem that may impact other IIe software. Lots of the earlier Apple programs output "high ASCII" code with the eighth bit set. This gives HP LaserJets and QMS printers fits.

The quick and dirty high ASCII cure was to use 7 data bits, SPACE parity, and 1 stop bit on the Apple IIe end, while running HP or QMS at 8,N,1. Sadly, space parity seems to get ignored by the LC. To beat this, I patched ProDos AppleWriter 2.1 by changing out $4CAA: 09 80 to 4CAA: 29 7F.

Other serious LC complaints are that you cannot access the hard disk from the IIe side and do not have any network access whatsoever to the *AppleShare* server.

Those rearranged key locations, particularly *control* and *escape*, are so totally absurd as to be beneath any rational comment.

The very popular *ProTerm* comm program worked most of the time, but the final screen speeds end up unacceptably slower on the LC. The LC makes ProTerm so infuriating and klutzy as to be useless.

Apple is working on some of these card problems, especially the networkability, and an upgrade or two can be expected soon. Possibly by the time you read this.

What Monitor Options are Available for the Mac LC?

The Mac LC is intended for use with a horizontal scan rate near 35 kilohertz, or over double that of the IIgs or standard television monitor scan rates of 15.735 kilohertz. Thus, an Apple IIgs monitor or most tv color monitor / receivers won't work at all with a Mac LC. Don't even think of trying it!

Yes, the DB15 connectors are the same size on the older IIgs color monitor and on the newer Mac LC monitors. But neither the pinouts nor the scan rates are the same.

On the other hand, genuine *Apple* brand color monitors can be very expensive and seldom discounted. Fortunately, there is a provision in the LC to drive most popular VGA color monitors now available at bargain prices. Figure two shows you the simple adaptor cable required to connect these two.

Note particularly that jumper between pins 7 and 10 at the Mac LC end. This is needed to activate an internal VGA mode in the LC.

Usually it is best to make up a short adaptor and attach it to a regular cable, rather than building up an entire special cable. Since the quality of any VGA monitor could vary all over the lot, be certain to look at real LC output before you make any commitments.

Are There Other Alternatives In Short Run Publishing?

As you regular readers know, I am rather big on *Book-on-Demand* publishing where individual titles are produced on a when-ordered and as-ordered basis. The Book-on-demand works particularly well if you do not know how many copies you are going to sell. Or if you are dealing with custom products or continuous upgrades. Some of my latest Book-on-demand titles now now do include the brand new *Ask the Guru III*, *Hardware Hacker III*, and *Midnight Engineering* reprints.

But it seems I just ran completely out of stock on my *Incredible Secret Money Machine*. That big mountain in the storehouse is now history. Besides, a revised second edition has been long overdue. Since the ISMM had already used up three printings and was an established classic, there is no doubt that many more could be sold. The ISMM is in its photographable hard copy form only. No PostScript and some hard-to-recode illustrations.

What are the alternatives between your own low end Book-on-demand self-publishing, and high volume traditional printers? For use when you have a proven market, might need a quality product, and don't

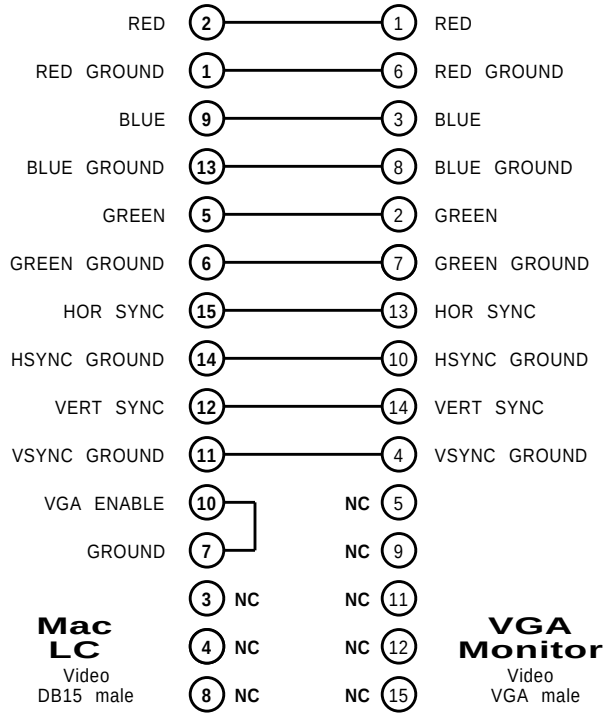


Fig. 2 – Mac LC to VGA color monitor cable adaptor.

want to take time out to rekey and redesign the entire text?

I shopped around and found three interesting types of publishers which seem able to economically produce small quantities (500-5000) of books.

The first of these is *Thompson-Shore* who is an old-line printer that specializes in short runs of custom hardbound and quality paperbacks. These folks have a free *Printers Ink* newsletter available on any request. They usually do a top quality job with a fast turnaround.

Have you ever wondered just how those paperback books on the newstand *already* have reviewer's quotes on them? What happened is that the publisher went ahead and slapdash threw together several hundred el-cheapo copies and sent them to the reviewers ahead of time. One firm that specializes in this sort of thing is *Crane Typesetting*. They do have a free reprint for you titled *Bound Galleys: Where, How, When and Why*. Crane is fast and cheap, but they are definitely not in the heir-loom business.

A final resource is *Omnipress*. These folks specialize in hundred- to thousand-copy lots of instruction manuals, user guidebooks, and such. Often in wire bindings.

The pricing depends upon the source, the quantity, the binding, and all your cover options. But, in general, these three sources are far lower than working with a local jiffy printer. Besides knowing how to better handle any specialized tasks. Please let me know if you have any experiences with similar short run publishers. Good or bad.

By the way, if you would like to preview the new intro and revisions to the new second edition of *The Incredible Secret Money Machine*, I've placed #286 MONEYMACH.TXT up on *GEnie* PSRT. Lots more on Book-on-demand publishing appears as file #77 DEMAND.TXT

```
% Copyright c 1991 by Don Lancaster and Synergetics, Box
% 809, Thatcher AZ, 85552. All commercial rights reserved.
% Personal use permitted so long as this header remains
% present. LaserWriter Secrets book+disk $29.50 VISA/MC.

% Free voice helpline at (602) 428-4073.

% Stock tool excerpts from my GONZO13A.PTL…

/roundpath {/rpdata exch def /rprad exch def rpdata length 1 sub
cvi /rppoints exch def rpdata 0 get rpdata 1 get moveto 2 2
rppoints 2 sub {/rpvalue exch def 0 1 3 {rpdata exch rpvalue add
get } for rprad arcto pop pop pop pop} for rpdata rppoints 1 sub
get rpdata rppoints get lineto} def

/superstroke { save /sssnap exch def /sscmd exch def mark 0 2
sscmd length 2 div cvi 1 sub 2 mul {/aposn exch def gsave
sscmd aposn get setlinewidth sscmd aposn 1 add get setgray
stroke grestore} for cleartomark sssnap restore newpath} def

% //// demo - remove or alter before reuse. ////

50 150 translate 10 dup scale

1 setlinejoin 1 setlinecap

1.99 [20 12 9 12 9 14 11 14 11 12 9 12 9 48 11 48 11 46 9 46 9
48 40 48 40 46 38 46 38 48 40 48 40 12 38 12 38 14 40 14 40
12 20 12] roundpath closepath
[0.8 0 0.5 1 0.28 0 0.1 1] superstroke

showpage quit
```

Fig. 3 – A 222 byte fancy PostScript border.

What is This Month's PostScript Utility?

As I promised last month, we'll now look at a quite compact and a stunningly serendipitous brand new border routine. Figure three shows you all the details.

The trick is a technique I call *arcto crowding*. If you do not give your PostScript *arcto* operator enough room to operate, it has to "inside itself out" and produces all sorts of wild effects.

What we have done is create a box having tight squares on each corner. Squares so tight that *arcto* has to inside itself out to execute. To get fancy, you take a single path and then repeatedly restroke it in thinner and thinner sequences of white and black lines. Or even high quality grays.

For other neat borders, try radius values of -6, -3, 0, 0.8, 5, and 7. Do not make the radius exactly equal the corner size or you will get a div0 blowup. Thus, if the corner boxes are 2.0 on a side, use 1.99 instead.

Your ready-to-run border code appears as file #277 MITZIYUP.PS available on *GEnie* PSRT.

I'm always on the lookout for examples of "raw" PostScript code which completely blow away the fancy illustration and applications package. This border can be done with about 222 bytes of code, not counting the stock *roundpath* and *superstroke* routines already in my PostScript utilities GUTIL13A.PTL.

Let's have us a new serendipity contest. Just come up with any old raw PostScript hack that can use stunningly few bytes to produce a mind-blowing result. We'll have all the newly revised *Incredible Secret Money Machine* book prizes, along with a big all-expense-paid (FOB Thatcher, AZ) *tinaja quest* for two going to the very best of all.

Either mail your entries to me per the blurb below, or else send them to me via [SYNERGETICS] through the *GEnie* email service. ❄

Don Lancaster's

ASK THE GURU

November, 1991

Special education software
Least square fuzzy data fits
Video compression schemes
PostScript-to-anything ideas
Embroidery on the Macintosh

Apple has a brand new 1991 *Mac Development Services Directory* available. It lists hundreds of developers, contract programmers, and service bureaus. You can pick up a copy at your local user group or dealer.

There's a new *Mac Research Users Group* which otherwise calls itself *MacSciTech*. You can modem them at (508) 755-5242 for more info, or else contact them through AppleLink under *cons.lab.mfg*.

Bunches of software for special education are now available by way of *Braintrain* at (800) 633-1221.

Apple is rumored to be dropping their APDA membership fee just about the time you're reading this.

What is New In Video Compression?

Mostly a totally wild three-way shootout, whose final results are going to be *very* interesting. You'll find details in Figure 1.

How many bytes of computer memory does it take to brute-force store a movie? For a 90 minute film to VCR quality, figure 262 x 350 x 60 x 8 x 3 x 60 x 90 = 713 gigabits or a tad under 100 gigabytes.

A result you get by multiplying the vertical resolution times your horizontal resolution, the number of fields per second, the bits per color, the number of colors, the number of seconds in a minute, and the total number of minutes per film. Not the sort of job you'd care to hand key.

At least not before lunch.

Now, say you decide to veg out on a rental video of *Godzilla Versus the Night Nurses*. Are you really going to ask your brain to process a full hundred gigs? Well, perhaps some extra attention during the tapioca pudding scene. But the rest?

Obviously, there has to be a lot of *redundancy* in those hundred gigabytes. Redundancy gets introduced in individual fields through edges, patterns, and large areas. Even more redundancy appears in sequential fields as only a small percentage of pixels normally change from one field to the next.

Video compression is any method of squashing your total number of bits on down into something more managable but still acceptable. The exact choice of a decent video compression scheme is the centermost problem today in multimedia, high quality computer graphics, and in HDTV standards.

A compression scheme is *lossless* if it lets you put everything back exactly the way it was. A lossless compression is important for such things as bank balances and most hard disk files. But compressing beyond 3:1 and staying lossless gets tricky with most data types. Even for typical images.

For video, considerably higher compression rates are possible by using lossy compression. The loss of data does degrade the final picture. So the trick is to pick a good tradeoff between the compression ratio and the quality you will accept. Today's lossy compression ratios are often around 30:1 for "good" quality down to 100:1 for "useful" images.

A compression scheme is *balanced*

(A) JPEG (Joint Photographic Experts Group) DCT Compression…

Takes 8 x 8 pixel arrays and performs a math transform on them, rearranging the picture energy and leaving a sparse data set. The data set has some values removed and others quantized to lower accuracy. The sparse array is then efficiently recoded. Often balanced and lossy.

While almost an industry standard, JPEG requires complex and unusual hardware and software. It also introduces objectionable tiling artifacts.

More info: Adobe Systems (415) 961-4400
LSI Systems (408) 433-8000

(B) Wavelet Compression…

Takes video rows and performs a simple math transform on them, followed by video columns, rearranging the picture energy and leaving a sparse data set. The data set has some values removed and others quantized to lower accuracy. The sparse array is then efficiently recoded. Usually balanced and lossy.

Uses standard or near standard digital signal processing hardware or software. Far simpler and faster than DCT. No tiling artifacts.

More info: Aware Incorporated (617) 577-1700

(C) Fractal Compression…

Breaks down images into underlying fractal components that in turn can be extremely compactly encoded. Lossy and unbalanced. Hundreds of high quality images can be stored on a single floppy.

Allows simple software-only playback, even full animation. Encoding is slow and costly, requiring special hardware. Real time compression is not possible at present. Ideal for CD ROM storage.

More info: Iterated Systems (404) 840-0310

Fig. 1 – Three popular video compression schemes.

if it takes about the same time and circuitry for the compression as the decompression. It is *unbalanced* if the decompression is much faster and simpler than the compression. Unbalanced compression has the advantage of needing little or no specialized hardware at the display end. At the penalty of requiring lengthy and expensive compression algorithms.

The first compression scheme of figure 1, and the one with the most momentum is known as JPEG (Joint Photographic Experts Group) compression. This uses the math trick called a *Discrete Cosine Transform*.

The picture is broken down into 8 x 8 arrays of 64 pixels each. The DCT is then run on each 64 pixel group, by exotic math and trig. The result for each 64 pixel block is a number. A number that takes into account inherent picture redundancies such as edges, patterns, and areas.

The neat part about the DCT is that it will redistribute the picture energy. Most of the final numbers are either zero or near zero, creating a sparse data set.

By recoding, you can do a lossless 3:1 or higher DCT compression. But, better yet, by ignoring some values, and quantizing all the others to less accuracy, you can go lossy and arbitrarily increase your compression ratio. All the while trading off the picture quality and accuracy. The DCT is balanced, requiring about the same time and effort to compress as decompress.

All sorts of DCT hardware chips have recently been announced, with *LSI Systems* being one leader in the DCT field. (408) 433-8000.

A complete 100 page JPEG spec is available from *Adobe Systems* (415) 961-4400. This reprint is free to all Adobe developers; $15 to others.

The main advantages of the JPEG method are that it is soon to become an international standard and that lots of time and effort have been put into it. It is here now.

The disadvantages are that the method is unnecessarily cumbersome for what it delivers, and that objectionable picture *tiling* artifacts get introduced. Unfortunately, JPEG looks like a standard which might end up obsolete before the ink on the final version is even dry.

Wavelets are a brand new way of doing mathematical transforms that is revolutionizing everything from oil exploration to cardiology on to animal vision. Wavelets are about to completely blow traditional *Fourier* analysis out of the saddle.

In fact, if you are now doing anything at all advanced in either electronics or computing, and are not now getting into wavelets in a big sort of way, then you have just volunteered for early retirement.

A useful (but difficult) book on wavelets is available from *Jones and Bartlett* (617) 482-3900. I also have some coverage in my new *Hardware Hacker III* reprints.

There are several good wavelet compression schemes. These take picture rows and transform them, followed by the picture columns. Unlike the DCT, the wavelet transformations are nothing but a bunch of simple shifts and adds. Shifts and adds which are easily and quickly done in standard or near-standard digital signal processing software or hardware.

There are no tiling artifacts, and the system can be balanced. It also appears that wavelet techniques are used in the internal processing of human and animal vision.

Similar to the DCT, the wavelet transformation can redistribute the picture energy into a sparse array for lossless compression. The sparse array can then be lossy compressed by ignoring some values and quantizing others to less accuracy.

The picture degradation caused by a higher compression value is normally in the form of broadband noise. Which most viewers find far less objectionable than the jaggies, Hershey-barring, or tiling.

Surprisingly large amounts of the broadband noise can often be added without too many complaints.

The leading proponents for the wavelet compression are *Aware* at (617) 577-1700. You can now contact them directly for hardware, for the emulation software, and for the free wavelet technical papers and ap notes. They also do have a great simulator and tutor known as the *UltraWave Explorer*.

The advantages of the wavelet compression are that it ends up far cleaner, far faster, and far simpler than JPEG, while at the same time creating better and more consistent final results. But an awful lot more time and effort has been dumped into JPEG and those folks will not either gracefully or willingly admit that they have been had.

Finally, we have a real maverick known as *Fractal Compression*. We have seen in previous columns and in the *Ask the Guru* reprints how to do a full page fractal fern using a mere 28 data values in a compact array. Which approaches a lossless compression ratio of 50,000:1. You can also find this fern as *GEnie* PSRT #84 FRACTFERN.PS.

With fractal compression, you

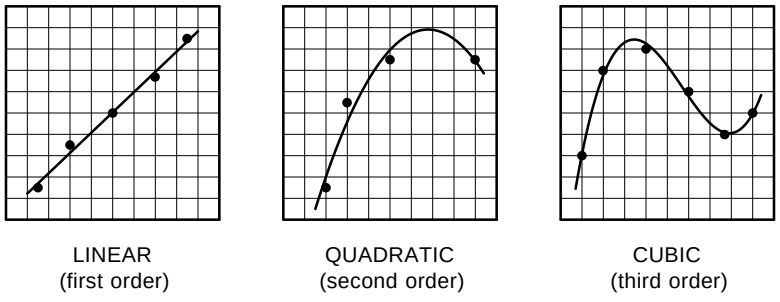


Fig. 2 – Some least squares fuzzy curve fits.

break your image down into fractal elements and then transmit only the codes for those elements. The system is unbalanced, requiring long times and lots of costly hardware and software on the compression end. But playback can be done with little or no special hardware at all. Thus, you can replay your fractal compressed VGA or similar video now on your *existing* computer.

Even stunning full animation.

The folks at *Iterated Systems* (404) 840-0310 are the leaders in fractal compression. A few of their newly introduced products do include the *Fractal Factory Video Player*, and the *Fractal Factory Slide Projector*, plus their *P.OEM* development systems and hardware. The slide projectors can store a hundred images on an ordinary floppy. And it can run at real animation speeds.

They also have lots of books, ap notes, and literature available.

The big advantages for fractal compression include the very high compression ratios and the ability to use nothing but software for full animation playback on an ordinary personal computer. Two major disadvantages are that a compression can not be done even remotely near real time and that horrendously expensive software and hardware is required on the compression end.

Most of the images I've seen so far are quite spectacular. They are particularly impressive when run on larger and faster machines.

Just who is going to be the big winner in video compression? Stay tuned as events unfold.

What's New in Embroidery?

This one should really have you in stitches.

I haven't fully tested it yet, and it is single-source expensive but not outlandishly so. The support software is also primitive and klutzy yet definitely useful. This is the *POEM 500* Mac embroidery system, manufactured by *Aisin* and stocked in depth by *Leonard's Distributors* (916) 967-6401.

What you've got here is a classic mechanical embroidery machine having a serial Mac interface and supporting software that lists for around $1200.

Your active area is about four inches on a side, making it ideal for such things as shoulder patches, gimmie caps, or nearly any other personalized textile item.

The number of different colors available are unlimited. You do, of course, have to change your thread cassettes for each color. And fancy jobs will take longer than simple ones. As with most sewing, your process is labor intensive and will demand someone into fabrics in one way or another.

Your max sewing speed is 300 stitches per minute. The software uses Hypercard and all of the usual tools. Resolution is the 72 DPI of the Mac screen. Serial comm is used.

Besides being one source for an unlimited supply of monogrammed gifts, any computer controlled embroidery machine offers all sorts of

(A) The **LINEAR** or **FIRST ORDER** least squares fit uses an equation of…

$$y = a_1x + a_0$$

Here are the two linear equations in two unknowns that you need to solve to get the best fit for your data…

$$(\text{sum}x_1)a_1 + (\#\text{pts})a_0 = (\text{sum}y)$$
$$(\text{sum}x_2)a_1 + (\text{sum}x_1)a_0 = (\text{sum}x_1y)$$

(B) The **QUADRATIC** or **SECOND ORDER** fit uses an equation of…

$$y = a_2x_2 + a_1x + a_0$$

Here are the three linear equations in three unknowns that you need to solve to get the best fit for your data…

$$(\text{sum}x_2)a_2 + (\text{sum}x_1)a_1 + (\#\text{pts})a_0 = (\text{sum}y)$$
$$(\text{sum}x_3)a_2 + (\text{sum}x_2)a_1 + (\text{sum}x_1)a_0 = (\text{sum}x_1y)$$
$$(\text{sum}x_4)a_2 + (\text{sum}x_3)a_1 + (\text{sum}x_2)a_0 = (\text{sum}x_2y)$$

(C) The **CUBIC** or **THIRD ORDER** least squares fit uses an equation of…

$$y = a_3x_3 + a_2x_2 + a_1x + a_0$$

Here are the four linear equations in four unknowns that you need to solve to get the best fit for your data…

$$(\text{sum}x_3)a_3 + (\text{sum}x_2)a_2 + (\text{sum}x_1)a_1 + (\#\text{pts})a_0 = (\text{sum}y)$$
$$(\text{sum}x_4)a_3 + (\text{sum}x_3)a_2 + (\text{sum}x_2)a_1 + (\text{sum}x_1)a_0 = (\text{sum}x_1y)$$
$$(\text{sum}x_5)a_3 + (\text{sum}x_4)a_2 + (\text{sum}x_3)a_1 + (\text{sum}x_2)a_0 = (\text{sum}x_2y)$$
$$(\text{sum}x_6)a_3 + (\text{sum}x_5)a_2 + (\text{sum}x_4)a_1 + (\text{sum}x_3)a_0 = (\text{sum}x_3y)$$

(D) For **QUARTIC** or **HIGHER ORDER** least square polynominal fits, just extend the math in the obvious direction for n linear equations in n unknowns. Solve for the coefficients. Extra precision may be required.

(E) For least square fits to **OTHER EQUATIONS**, write out the least squares error equation. Then take partial derivatives with respect to each coefficient and set them to zero. This again leads to n linear equations in n unknowns. Solve for the coefficients.

Fig. 3 – The math behind least squares fuzzy curve fitting.

opportunities at flea markets, fairs, or for specialty advertising. There should also be major possibilites in fund raising, in kiddie crafts, and special education as well.

More on this beast after we have a chance to check it. Their thick felt samples are really impressive. Bee and Kathy are a lot more into string bending than I am, so we'll see what they come up with.

My big gripe on this should be obvious: Any genuine *Adobe* PostScript software interface and driver would make the system infinitely more useful, more powerful, and more convenient. Especially for the circular shoulder patch text.

The first person to come up with a universal PostScript-to-anything interface will run away with a very large bag of nickels. Besides looms, embroidery, and sewing machines, other obvious goodies to bolt onto a PostScript-to-anything interface include signmakers, Santa Claus desktop prototyping systems, for engravers, animation stands, vinyl cutters, new virtual reality, printed circuit drilling, fully programmable routers, and CAD/CAM.

How Can I Connect The Dots?

Actually, it is a lot harder than it seems. So, I thought I'd throw some heavy stuff at you for this month's PostScript utility. Which is a quick and convenient way to draw engineering curves starting with a few noisy data points.

One of the big things everyone wants to do by using PostScript is literally connecting dots. To take a series of sparse, noisy, and low resolution data points, and then convert them to smooth and high resolution PostScript curves.

Compactly and quickly.

The obvious way to do this is with cubic splines and groups of PostScript *curveto* or *rcurveto* operators. I've already shown you an everyday workaround using my *Gonzo Guru Curvetracing Utilities.*

Our older classic puddy tat #79 MEOWWRRR.PS on *GEnie* PSRT is one example, while the full details appear in my *PostScript Beginner Stuff* package.

The secret to the curvetracing is to hand spec each joint with a triad of *xposition*, *yposition*, and *slopeangle* values. Then you use some fairly weak splines to create the smooth composite curve.

But curvetracing can be tedius and does not work very well with fuzzy or noisy data. Working with inaccurate data can get tricky fast. Especially when you try to decide just how many splines to use and exactly how to line them up. Or when to substitute straight lines or sharp corners. Any truly general solutions may turn out unbearably slow since they might take several hundred approximation passes to do the work.

The big, fat, ugly, slow, heavy, arcane, obtuse, and hairy math involved in doing the job right is shown to you in *Curve-Fitting with Piecewise Parametric Cubics*, found in the 1983 *Siggraph/ACM* #3 issue of *Computer Graphics*, pages 229 to 239 (212) 869-7440.

I have been chewing away at this beast for several months now, and hope to eventually have some real goodies for you. As a stop along the way, I decided to explore fitting fuzzy data in general, doing what is known as a *least squares curve fit*. And I found out that many engineering plots and responses (pump efficiencies, transistor responses, temperature nonlinearities, etc.) can be drawn quickly and simply just by dropping in a few sloppy points and then doing a plain old least squares fit to a single and simple power series.

Figure two shows you the three most popular least squares power series fits to fuzzy data. These are the *linear* fit (put a straight line through it), the *quadratic* (pick a parabola), or the *cubic* (a third order power curve). I found that fourth and fifth order curves can be used to quickly draw just about any curve found in an engineering data book, starting with very few and possibly noisy points. Only *one* set of data points and a *single piece* curve is normally needed.

Fitting one single curve to real world data works best when the curve does not cross itself and when there are no violent changes in the curve's direction.

Even the least squares math is hairy, but it sure beats using those successive approximations to fifth order *Newton-Raphson* iterations that the full spline solution seems to need. Figure three shows you the key math involved in doing least squares error fitting. Enjoy.

All we mean by least squares is that we try to divide the errors up so that all the points share equally in the compromise. The square is used so that all the negative values end up below the curve and positive ones above. To do a least squares fit, you write out an expression for the sum of the errors squared for your data points along the precise curve you are trying to fit.

Then you find the *slope* or *partial first derivative* of all the errors with respect to each variable involved. Set the slope or derivative to zero to find the minimum. This results in a new system of linear equations of n variables in n unknowns which you solve once. Solving for n gives you the everyday working formulas.

Naturally, I do feel PostScript is absolutely ideal for this sort of thing since it is a totally general purpose computer language that is super easy to use. The final results are somewhat on the longish side, so I have posted them to *GEnie* PSRT. Start with #289 LINEAREQ.PS and #290 FUZZYFIT.PS and then pick up the ongoing newer stuff that shows you some of the secret special tricks involved in improving accuracy on the fourth and fifth order curves. Included is code for solving linear equations, routines to fit fuzzy data, and handy utilities to draw your curves, the background graph, and all of the individual data points. Additional tutorial info appears in #302 HACK43.TXT.

You can get your voice connect info for *GEnie* at (800) 638-9636. The average cost of a PSRTdownload is around twenty-one cents. ❄

Don Lancaster's

ASK THE GURU

December, 1991

Laser printer hard disks
Mac LC IIe card upgrade
Flocks and flocking ideas
Avoiding baud rate limiting
Studying Click-to-Clunk times

Apple has released the new software 2.0 upgrade for their Apple IIe emulation card used on the Mac LC. This gives you IIe hard disk access and lets you use AppleShare. Printer access is also improved. It still does not allow use of 800K platinum drives on the Apple side. Part number for the $29 upgrade kit is M1222LL/A.

Only some software changes are involved; the firmware ROM chips remain the way they were.

You will have to trash out and reinitialize the hard disk whenever you do the upgrade, since the disk will get partitioned into a ProDos portion and a Mac portion. Yes, you can read and access your Apple files from the Mac side. But only when using AFT or other programs which know how to read ProDos.

Several Apple IIe & IIGS related products have also now been newly introduced. While well done and sorely needed, they seem way too little and far too late. These include a new GS/OS 6.0 operating system, support for the denser disk drive formats (including MS/DOS), and an Ethernet card. Since the latest LaserWriter offers really high speed *Ethernet* comm, I am especially interested in this gem.

Word has it that the new *Hewlett Packard* IIIsi laser printer will slow down ludicrously under Windows. Which has thoroughly hacked off dozens of my helpline callers. HP often responds to its obvious and major technical printer flaws using the "circle the wagons" and "shoot the messenger" approach. It will be interesting to see what gets done when on this one.

Adobe Systems (415) 961-4400 has a new freebie bulletin for you on *Type I Multiple Master Typefaces*. Read all about the next font technology improvement beyond Level II.

I've just got a sample of the new *Xante* Accel-a-Writer II tray for the NTX and hope to review it for you shortly. (800) 992-6839. At my first glance, the obvious advantages are RISC high speed true 600 x 600 DPI resolution out of your LaserWriter. With auto Mac/IBM interfacing and optional dual-page makeup. And the negative stuff seems to include fake PostScript and a non-standard disk operating system.

In addition to the Mac embrodery system we did look at last month, *Meistegram* (800) 321-0486 does offer several computer controlled monogramming options. Since old-line franchises are involved, check out your alternates thoroughly.

I've gathered all of my previous

1. Hard disks very dramatically improve the speed and convenience of PostScript printing. If you *can* use a local hard disk, be sure to *do* so.
2. Always turn your hard disk on several seconds *before* you apply printer power. Always turn your printer off first.
3. Always use an absolute minimum of three megabytes of printer memory if your printer is connected to a hard disk.
4. Always keep a complete, full, and separate backup of *all* your hard disk files readily available.
5. Never defeat your startup page. It tells you if your disk is working and warns you of impending doom.
6. Always *completely* re-initialize and rebuild your hard disk once a month. Or any time your test page starts taking much longer to print out.
7. On the LaserWriter NTX, be certain to use Revision 3.0 or higher firmware. Note that the disk filenames change when you do this upgrade, from *%disk%myfile* to *%disk5%myfile*, where the five (or whatever) is your preset device number.
8. There is only one difference between storing a font and storing any old file used for any old purpose. The font goes in a special subdirectory called *%disk5%fonts/Myfontname*.
9. Some SCSI hard disks use a 25 pin connector physically identical to a serial RS232 connector. Immediate and permanent damage is almost *certain* to result if you plug a SCSI device into RS232 or vice versa.
10. Some SCSI hard disks will lock themselves out for a half hour or so after power is even briefly removed. Thus, you should *never* briefly power down a hard disk.
11. Never send a new file to your printer if the disk is actively modifying its font cache. Always wait till the activity light goes off.
12. Avoid pixel line remapping and other "lots of fonts in a hurry" techniques while your hard disk is active. Creative use of the *setcacheparams* operator can help bunches here.
13. If at all possible, set up a *shared SCSI comm* environment where you can directly transfer files from your host to your hard disk. While highly custom, this technique is ridiculously faster than serial or AppleTalk.
14. Additional hard disk info appears in the Apple White book, in my *Ask the Guru* reprints, and on *GEnie* PSRT files 99, 100, 107, 108 and later.

Fig. 1 – Guidelines for laser printer hard disk use.

Book-on-demand publishing stuff in to a new *Book-on-demand resource kit*. Write or call me per the end blurb for more info.

As usual, most of the products mentioned appear in the *Names and Numbers* appendix. Further help is available via a no-charge voice helpline at (602) 428-4073 and by way of *GEnie* PSRT at (800) 638-9636.

Onward and upward...

Please Update me on Laser Printer Hard Disk Systems

High end PostScript laser printers *demand* a dedicated SCSI hard disk for exclusive printer use. In fact, it is extremely foolish to buy a LaserWriter NTX or a *QMS* turbo PS820 printer without picking up (and aggressively using) the hard disk at the same time. Note that this is a totally different hard disk than gets used by your host computer.

There are many major advantages to hard disk laser printing. Most of these involve speed and convenience. Your font cacheing becomes virtually permanent, instead of its going away each time you power down. You can now locally store hundreds or even thousands of now quickly available fonts. You can prestore any and all of your favorite persistent downloads. Other times, your hard disk can eliminate the need for most persistent downloads and the VM resources they waste in your printer.

You can store individual files up to entire books on your hard drive, and access them in seconds, rather than those long serial or AppleTalk comm times normally involved. By using a hard disk, your overnight Book-on-demand publishing can be automated and done unattended by an operator or host computer.

Many new Postscript-as-language routines require the ability to write their non-printed output files to a hard disk. Foremost among these is the *Adobe Distillery*, but there are hundreds more.

And, finally, if you are sneaky enough, you could set up a *Shared SCSI comm* which lets you directly communicate between the host and printer at SCSI speeds, rather than the excruciatingly slow AppleTalk or serial rates.

For most users, any old 20 meg SCSI disk is more than adequate. You usually do have to be sure it is an Apple-compatible drive that is capable of returning its size to the host. Watch this detail.

Figure one sums up some of the hard-to-find guidelines for laser printer hard disk use.

Early versions of the LaserWriter NTX had *severe* hard disk blowup problems. The worst of these can be greatly eased by going to the new ROM 3.0 upgrade, using at least 3 megs of RAM, and always turning your hard drive on several seconds *before* your printer.

Never send any new file to your printer while its activity light is on or it is reworking its internal font cache. Be careful!

As with any hard disk anywhere, you should keep a separate and a complete backup of all of your files somewhere else.

Laser printer hard disks are still so flakey that they will eventually blow up. Thus, you should *always* re-initialize and rebuild your disk once each month. One clue that does not bode well is when your startup page starts taking longer.

By the way, it is *never* a good idea to defeat the test page if you have a hard disk in your system. Firstoff because it tells you whether your disk is currently active. And second, because progressively longer test page times tell you that disaster is about to strike.

Much more on hard disk systems appears in the Apple White book, otherwise known as the *LaserWriter Reference*. By one of those astonishing coincidences that seem to infest this column, I do have copies of these in stock for you here at

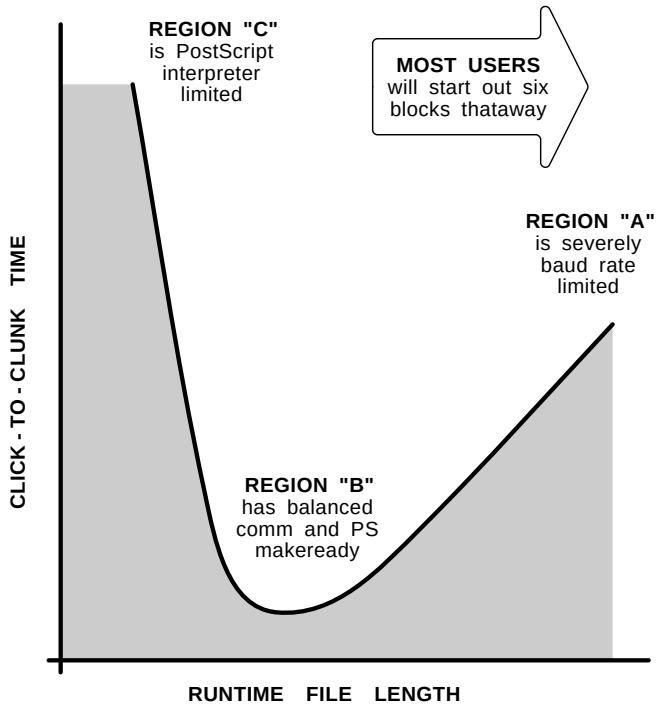


Fig. 2 – PostScript's secret Click-to-Clunk speed curve.

Synergetics.

Other good sources for hard disk info does include the *Ask the Guru* reprints, and *GEnie* PSRT, especially files #99, 100, 107, and 108.

How Can I Tend My Flock?

I do see you are thinking fuzzy again. Although flocks and flocking have traditionally been the most arcane of artsy-craftsy backwaters, they are about to hit our desktop publishing big time and offer some exciting new opportunities. So, I guess it's time for a rundown...

One old-line source of traditional "cabbage duster" flock is *Don Jer.* (800) 336-6537. They sell mostly to the industrial education market for the stamp box and award plaque crowd. You paint on colored glue glop and then dust the flock into place. Then you shake off the excess and let it dry. Instant fuzz.

A leader in flocks intended for heat transfer to fabrics and T-Shirts is *HIX Corporation* (800) 835-0606.

One of their more interesting products is their paper release sheet completely covered with a fuzzy flock. In your choice of two dozen colors. You then paint or silk screen some thin cold glue onto the flock and then sprinkle some heavier hot glue crystals into the wet areas.

Which in turn creates an iron-on transfer that lets you apply fuzzy letters, numbers, or artwork on any textile, T-shirt, or gimmie cap.

The cost is seventy cents a sheet.

Sadly, toner doesn't seem to want to grab flock the same way it does *Kroy Color* and other hot stamp materials. At least my first try failed.

Although it should be possible to PostScript laser print an image and do "something" to it that makes the flock permanently stick to it. Fuzzy and "cuddly" letterheads should be an unusual and possibly a winning new product.

The big boys do not use flock. Instead, they use a *hot split plastisol.* Which is a thick colored plastic that gets heat transfered to your fabric. Before your transfer can cool, the backing is violently stripped away, literally tearing the plastisol in half *along its thickness*, and leaving a permanent felt-like fuzzy surface on the fabric.

One major player in the hot split plastisol supplies is *Gerber Scientific.* (203) 644-1551. A freebie booklet on *The Art of Applying Transfers* is now available from *Stahls* (800) 521-9702. Two trade journals on this sort of stuff include *Screen Printing* (513) 421-2050 and *Signcraft* (813) 939-4644.

Let's make a contest out of this. Either send me some direct PostScript laser printed cuddly fuzzy stuff samples, or else add to our flock dialog in some way.

There will be all of those newly reissued *Incredible Secret Money Machine* book prizes for those best dozen or so entries, along with an all expense paid (FOB Thatcher, AZ) *tinaja quest* for two going to the best of all. As usual, send your written entries directly to me at *Synergetics* per the end blurb.

What is Meant by PostScript's "Click to Clunk" time?

How would you like to decrease the makeready time of your PostScript printer by a factor of ten or more? Actually, this is usually real easy. All you have to do is eliminate all of the common and really dumb mistakes others are usually making when they blindly try and apply the PostScript computer language.

As an example, I routinely do a three column 6000 character fully justified page makeup with headers, footers, and a pair of figures in two

(A) If you are BAUD RATE LIMITED...

1. Shorten your files! Eliminate comments. Use persistent downloads.
2. Give PostScript more to do in such a way that it needs fewer file characters. Examples: */m* for */moveto*; Reducing to minimum needed accuracy; not repeating or transmitting redundant data; defeating screen echo; or using more efficient codings.
3. Avoid the *image* operator, especially when raw PostScript gives better results faster. Recode essential image files in binary or in ASCII-85 formats, rather than as hex-ASCII pairs.
4. Eliminate your baud rate limit completely by using files on a hard disk. Or switch to Ethernet or shared SCSI comm.

(B) If you have BALANCED COMM...

1. Measure your *actual* and *exact* de-facto baud rates. These will almost always be *much* worse than you think.
2. Measure the execution times of *individual portions* of your PostScript code to find out what takes the longest. Typically, 20% of your code takes 80% of your time.
3. Improve your hardware, software, firmware, and programming style.

(C) If you are POSTSCRIPT INTERPRETER LIMITED...

1. Give PostScript less to do by using more but simpler commands in possibly longer files.
2. Compile your files using the ADOBE DISTILLERY and then, if needed, by doing a second double distilling.
3. Minimize use of known time-wasters such as irregular clipping intervals, or crude translations from the pre-PostScript world.
4. Use table lookup or predefined array methods to replace extensive calculations or complex logic.

Fig. 3 – Reducing your Click-to-Clunk times.

to four seconds of makeready time. From an Apple IIe, no less.

The insider secrets we are about to look at perform best when your PostScript output is to be used more than once in the future. Obvious places include Book-on-demand publishing and uploads to bulletin boards that many others will reuse. Or, when doing some rather-slow phototypesetting. Often, several simple and quick tricks done once can dramatically shorten all of your future print times.

Your *Click-to-Clunk* time is just that: The measure of how long it takes from the click of a mouse or return key until the printed page clunks into your exit tray. Your personal Click-to-Clunk time is the *only* real measure of printer speed.

There is a little-known optimum file length for any printed PostScript page. Get under this magic file length, and your PostScript interpreter time takes too long. Get above the magic length, and you baud rate limit. Most beginning PostScript users most of the time end up horribly baud rate limited. Especially with AppleTalk.

As figure two does show us, the Click-to-Clunk curve for a typical printer is not at all obvious. There are three definite regions to your Click-to-Clunk curve. By knowing which region you are in, you can take easy steps to improve your makeready time. On the other hand, if your steps are inappropriate for where you are on the curve, you can do far more harm than good.

Figure three summarizes some good strategies for each region.

In region "A", you are *baud rate limited*. There are so many characters in the file that the PostScript interpreter has to continually wait for new ones before it can continue. If you end up baud rate limited, the speed and performance of your printer does not matter in the least. Bytes cannot be used before they arrive. And most users most of the time are *severely* baud rate limited.

Baud rate limiting can set in with files as short as 8K. With typical comm setups, if control does not return to your host until printing starts, than you're likely to be baud rate limited.

If you are baud rate limited, the obvious thing to do is to shorten your file. Remove unneeded characters and excess documentation. Rearrange your file so that your PostScript interpreter has stuff to do right away, rather than having to wait forever before it starts. Use earlier and shared persistent downloads whenever possible.

Since your printer is obviously much faster than you need, think about having it do extra jobs that can shorten your files. For instance, instead of using dozens of *moveto* commands, use a *m* instead, thus shortening your file by hundreds or thousands of characters.

Instead of repeating a zillion "0, 32, 0" constants before each *awidthshow*, use some creative stack rolling and insertion to eliminate hundreds or thousands of characters.

What kind of accuracy do you need? For routine positioning under 600 DPI four decimal places will do just fine. Instead of a 456.789, use 456.7 instead. Better yet, use 4567 and work at one-tenth size. Which throws a zillion no longer needed decimal points out of your file. For *setgray* values, two place accuracy should be more than enough.

Decimal numbers are one really lousy way of storing any value. For instance, 4567 gobbles up five bytes (including the space) to represent itself. By other codings, those five bytes should be able to represent up to 256^5 or over ONE TRILLION different values!

What about images? These are real byte wasters. Very often, you can substitute raw PostScript code for your image, and pick up a 1000:1 or higher reduction in your byte count. While actually improving your appearance and print speed at the same time.

The image operatator often gets used as a sloppy crutch by lazy writers of pre-PostScript application packages who just don't know any better. The time to feel sorry for them has long since passed. Packages such as the *Adobe Illustrator* or *Aldus Freehand* can greatly help you in tracing over all your images with real PostScript.

My own personal preference, of course, is to just use raw PostScript instead.

If you absolutely must use an image, make the active area as small as is reasonably possible. Lots of sequential 00 values in your image code is a good sign that you can cut your size and lose nothing. Then substitute a better coding, such as binary or the new ASCII85 in the red book. Binary cuts your code by half; the easier to use and much more transportable ASCII85 by around forty percent.

A trick to region "A" speedup is to give your PostScript computer a little more work to do, while at the same time significantly shortening your file sizes.

Over in region "C", the PostScript interpreter is taking far too long to make up the pages. Use the *Adobe Distillery* and a possible double distilling pass to dramatically speed up your runtimes here. We have looked at this compiling in detail a few columns back. A shareware copy of #186 DISTILL.PS is now available on *GEnie* PSRT.

In region "B", your comm and interpreter times are at a balanced minimum. Characters arrive just in time, immediately before they are really needed. To improve on this apparent minimum, measure your de-facto baud rates. Chances are they are ridiculously slower than you expect. Then find out which portions of your program are taking up the most time. Finally, improve your hardware, software, firmware, algorithms and programming style.

Obviously, that big dip in region "B" is a rapidly moving target. But if you land "in the trough" for a NTX at an honest 19200 baud, you'll have a good solution for most other users and many other printers. With more practice, you'll do even better.

As a second contest this month, just tell me all about any sneaky, seldom-used, or off-the-wall PostScript speedup technique. ❄

Keyword Index