New Technical Notes

Macintosh

Developer Support

LaserWriter Optimization Techniques

Imaging

Revised by: Pete "Luke" Alexander October 1990
Written by: Ginger Jernigan February 1986

This Technical Note discusses techniques for optimizing code for printing on the LaserWriter.

Changes since March 1988: Updated the "Printable Paper Area" and "Memory Considerations" sections as well as the printer IDs, moved the error messages from the end of the Note to Technical Note #161, A Printing Loop That Cares…, and removed the "Spool-A-Page/Print-A-Page" section because Technical Note #125, Effect of Spool-A-Page/Print-A-Page on Shared Printers, already thoroughly covers this topic.

Introduction

Although the Printing Manager was originally designed to allow application code to be printer independent, there are some things about the LaserWriter that, in some cases, have to be addressed in a printer dependent way. This Note describes what the LaserWriter can and cannot do, memory considerations, speed considerations, as well as other things you need to watch out for if you want to make your printing more efficient on the LaserWriter.

How To Determine The Currently Selected Printer

With the addition of new picture comments and the `PrGeneral` procedure, an application should never need to know the type of device to which it is connected. However, some developers feel their application should be able to take advantage of all of the features provided by a particular device, not just those provided by the Printing Manager, and in doing so, these developers produce device-dependent applications, which can produce unpredictable results third-party and new Apple printing devices. For this reason, Apple strongly recommends that you use only the features provided by the Printing Manager, and do not try to use unsupported device features.

Even though there is no supported method for determining a device's type, there is one method described in the original *Inside Macintosh* that still works for ImageWriter and LaserWriter printer drivers. This method is **not** supported, meaning that at some point in the future it will no longer work. If you use this method in your application, it is up to you to

weigh the value of the feature against the compatibility risk. The following method works for all ImageWriter, ImageWriter II, and LaserWriter (original, Plus, IINT, IINTX) drivers. Since all new devices released from Apple and third-party developers have their own unique ID, it is up to you to decide what to do with an ID that your application does not recognize.

If you are using the high-level Printing Manager interface, first call `PrValidate` to make sure you have the correct print record. Look at the high byte of the `wdev` word in the `TPrStl` subrecord of the print record. Note that if you have your own driver and want to have your own number, please let DTS know, and DTS can register it.

Following is the current list of printer IDs:

Printer	wDev
ImageWriter I, ImageWriter II	1
LaserWriter, LaserWriter Plus, LaserWriter IINT, LaserWriter IINTX, and Personal LaserWriter NT	3
LaserWriter IISC, Personal LaserWriter SC	4
ImageWriter LQ	5

If you are using the low-level Printing Manager interface, there is no dependable way of getting the wDev information. You should **not** attempt to determine the device ID when using the low-level Printing Manager interface.

Using QuickDraw With the LaserWriter

When you print to the LaserWriter, all of the QuickDraw calls you make are translated (via QuickDraw bottlenecks) into PostScript®, which is in the LaserWriter ROM. Most of the operations available in QuickDraw are available in PostScript, with a few exceptions. The LaserWriter driver does not support the following:

- `XOR` and `NotXOR` transfer modes.
- The grafverb `invert`.
- `_SetOrigin` calls within `PrOpenPage` and `PrClosePage` calls. Use `_OffsetRect` instead. (This is fixed in version 3.0 and later of the driver.)
- Regions are ignored. You can simulate regions using polygons or bitmaps. Refer to Technical Note #41, Drawing Into An Off-Screen Bitmap, for how to create off-screen bitmaps.
- Clip regions should be limited to rectangles.
- There is a small difference in character widths between screen fonts and printer fonts. Only the end points of text strings are the same.

What You See Is Not Always What You Get

Unfortunately, what you see on the screen is not always what you get. If you are using standard graphic objects, like rectangles, circles, etc., the object is the same size on the LaserWriter as it is on the screen. There are, however, two types of objects where this is not the case: text and bitmaps.

The earlier noted difference between the widths of characters on the screen and the widths of characters on the printer is due to the difference in resolution. However, to maintain the integrity of line breaks, the driver changes the word and character spacing to maintain the end points of the lines as specified. What this all means is that you cannot count on the positions or

the widths of printed characters being exactly the same as they are on the screen. This is why in the original MacDraw®, for example, if one carefully places text and a rectangle and prints it, the text sometimes extends beyond the bounds of the rectangle on the printed page. If an application does its own line layout (i.e., positions the words on the line itself), then it may want to disable the LaserWriter's line layout routines. To disable these routines, use the `LineLayoutOff` picture comment described in the LaserWriter Reference Manual and Technical Note #91, Optimizing for the LaserWriter—Picture Comments.

The sole exception to this rule is if an application is running on 128K ROMs or later. The 128K ROM Font Manager supports the specification of fractional pixel widths for screen fonts, increasing the screen to printer accuracy. This fractional width feature is disabled by default. To enable it, an application can use `_SetFractEnable`, after calling `_InitFonts`.

Applications can use picture comments to left-, right-, or center-justify text. Only the left, right, or center end points are accurate. If the text is fully justified, both end points are accurate. Technical Note #91, Optimizing for the LaserWriter—Picture Comments, discusses these picture comments.

Memory Considerations

To print to the LaserWriter, you need to make sure that you have enough memory available to load the driver's code. The best way to do this is to have all the code you need for printing in a separate segment and unload everything else. When you print to the LaserWriter you are only able to print in Draft mode. You are not able to spool (as the ImageWriter does in the standard or high-quality settings), and your print code, data, and the driver code have to be resident in memory.

In terms of memory requirements, there is not any magic number that always works with all printer drivers (including third-party printer drivers) that are available for the Macintosh. To make sure there is enough memory available during print time, you should make your printing code a separate segment and swap out all unwanted code and data before you call `_PrOpen`.

Printable Paper Area

On the LaserWriter there is a 0.45-inch border that surrounds the printable area of the paper (this is assuming an 8.5" x 11" paper). If you select the "Larger Print Area" option in the Page Setup dialog box, the border changes to 0.25 of an inch. This printable area is different than the available print area of the ImageWriter. An application cannot print a larger area because of the memory PostScript needs to image a page. PostScript takes the amount of memory available in the printer and centers it on the paper, and there is not enough RAM in the LaserWriter to image an entire sheet of paper.

Page Sizes

Many developers have expressed a desire to support page sizes other than those provided by the Apple printer drivers. Even though some devices can physically support other page sizes, there is no way for an application to tell the driver to use this size. With the ImageWriter driver, it is possible to modify certain fields in the print record and expand the printable area of

the page. However, each of the Apple drivers implements the page sizes in a different way. No one method works for all drivers. Because of this difference, it is strongly recommended that applications do not attempt to change the page sizes provided in the "Style" dialog box. If your application currently supports page sizes other than those provided by the printer driver, you are taking a serious compatibility risk with future Apple and third-party printer drivers.

Speed Considerations

Although the LaserWriter is relatively fast, there are some techniques an application can use to ensure its maximum performance.

- Try to avoid using the QuickDraw Erase calls (e.g., `_EraseRect`, `_EraseOval`, etc.). It takes a lot of time to handle the erase function because every bit (90,000 bits per square inch) has to be cleared. Erasing is unnecessary because the paper does not need to be erased the way the screen does.

- Printing patterns takes time, since the bitmap for the pattern has to be built. The patterns `black`, `white`, and all the gray patterns have been optimized to use the PostScript gray scales. If you use a different pattern it works, but it just takes longer than usual. In addition, the patterns in driver version 3.0 are rotated; they are not rotated in version 1.0.

- Try to avoid frequently changing fonts. PostScript has to build each character it needs either by using the drawing commands for the built-in LaserWriter fonts or by resizing bitmaps downloaded from screen fonts on the Macintosh. As each character is built, it is cached (if there's room), so if that character is needed again PostScript gets if from the cache. When the font changes, the characters have to be built from scratch in the new font, which takes time. If the font is not in the LaserWriter, it takes time to download it from the Macintosh. If the user has the option of choosing fonts, you have no control over this variable; however, if you control which fonts to use, keep this in mind.

- Avoid using `_TextBox`. It makes calls to `_EraseRect`, which slows the printer, for every line of text it draws. You might want to use a different method of displaying text (e.g., `_DrawString` or `_DrawText`) or write your own version of `_TextBox`. If an application is currently calling `_TextBox`, changing to another method of displaying text can improve speed on the order of five to one.

- Because of the way rectangle intersections are determined, if your clip region falls outside of the `rPage` rectangle, you slow down the printer substantially. By making sure your clip region is entirely within the `rPage` rectangle, you can get a speed improvement of approximately four to one.

- Do not use spool-a-page/print-a-page as some applications do when printing on the ImageWriter. It slows things down considerably because of all of the preparation that has to be done when a job is initiated. Refer to Technical Note #125, Effect of Spool-A-Page/Print-A-Page on Shared Printers, for more information.

- Using `_DrawChar` to place every character to print can take a lot of time. One reason, of course, is because it has to go through the bottlenecks for every character that is drawn. The other is that the printer driver does its best to do line layout, making the character spacing just right. If you are trying to position characters and the driver is trying to position characters too, there is conflict, and printing takes much longer than necessary. In version 3.0 of the driver, there are picture comments that turn off the line layout optimization, alleviating some of the problem. Refer to Technical Note #91, Optimizing for the LaserWriter—Picture Comments, for more information.

Clipping Within Text Strings

When clipping characters out of a string, make sure that the clipping rectangle or region is greater than the bounding box of the text you want to clip. The reason is that if you clip part of a character (e.g., a descender), the clipped character has to be rebuilt, which takes time. In addition, because of the difference between screen fonts and printer fonts, chances are that you cannot accurately clip the right characters unless you are running on the 128K ROMs and have fractional pixel widths enabled.

When to Validate the Print Record

To validate the print record, call `PrValidate`. You need validation to check to see if all of the fields are accurate according to the current printer selected and the current version of the driver. You should call `PrValidate` when you have allocated a new print record or whenever you need to access information from the print record (i.e., when you get `rPage`). The routines `PrStlDialog` and `PrJobDialog` call `PrValidate` when they are called, so you do not have to worry about it if you use these calls.

Empty QuickDraw Objects

QuickDraw objects that are empty (i.e., they have no pixels in them) and are filled but not framed, do not print on the ImageWriter and do not show up on the screen; however, on the LaserWriter they are real objects and do print.

Further Reference:

- *Inside Macintosh*, Volume I, QuickDraw
- *Inside Macintosh*, Volume II, The Printing Manager
- *LaserWriter Reference Manual*
- Technical Note M.IM.OffscreenBitMap—
  Drawing Into An Off-Screen Bitmap

- Technical Note M.IM.PictComments —
  Optimizing for the LaserWriter—Picture Comments
- Technical Note M.IM.Spooler —
  Effect of Spool-A-Page/Print-A-Page on Shared Printers
- Technical Note M.IM.PrintLoop —
  A Printing Loop That Cares…

- PostScript Language Reference, Adobe Systems, Incorporated
- PostScript Language Tutorial and Cookbook, Adobe Systems, Incorporated

MacDraw is a registered trademark of Claris Corporation.
PostScript is a registered trademark of Adobe Systems, Incorporated.