Technical Note PR510
Printer Driver Q&As

CONTENTS

This Technical Note contains a collection of archived Q&As relating to a specific topic--questions sent the Developer Support Center (DSC) along with answers from the DSC engineers. Current Q&A's can be found on the Macintosh Technical Q&A's web site.

[Oct 01 1990]

Defining default LaserWriter Pro paper tray

Date Written: 5/17/93

Last reviewed: 5/19/93

How can I set the default paper tray on my LaserWriter Pro to something other than the 250-sheet cassette?

The paper tray selection in the printing job dialog is not "sticky"; it's not preserved from job to job. In LaserWriter 7.2, the choice of one of the two paper trays is stored in the print record. You may be able to hack out where this is stored and define a new "default" print record by changing the PREC(0) resource, but this is completely unsupported and you do so at your own risk.

In LaserWriter driver 8.0, the default paper trays are defined in the PostScript Printer Definition (PPD) file. If you're sufficiently adventurous, you can edit the PPD file for your printer to change these relationships. LaserWriter 8.0 requires files that meet the PPD 4.0 specification, available from Adobe's Developer Support program.

Back to top

Code for getting selected printer name

Date Written: 3/10/93

Last reviewed: 6/24/93

How do I get the name of the currently selected printer, for Systems 6 and 7?

The printer driver has a resource of type PAPA, ID -8192, which contains the name of the selected printer and the network object type. This resource is amazingly easy to get; you don't even need to know where the printer driver resource file is located. Just call `PrOpen`. The Printing Manager will open the currently selected printer driver for you, and place it at the top of the resource chain. Then see whether you can `GetResource` for `'PAPA'` ID -8192, and viola--you have a handle to the printer name.

This will work under both System 6 and System 7. However, under QuickDraw GX, this method is subject to change. Also note that this is a read-only field; changing it doesn't necessarily change the selected printer.

The printer name is the first string, and the object type is packed right up against it. If all you want is the name, then here's a C routine that will provide it:

```
OSErr GetPrinterName(Str255 prName)
{
  OSErr   err = noErr;
  char  **papaDataHandle;

  PrOpen();

  papaDataHandle = Get1Resource('PAPA',-8192);

  if(!papaDataHandle) {
    err = ResError();
  } else {
    BlockMove(*papaDataHandle, prName, (**papaDataHandle)+1);
    ReleaseResource((Handle) papaDataHandle);
  }

  PrClose();

  return err;
```

Back to top

StyleWriter II driver differences

Date Written: 3/17/93

Last reviewed: 6/24/93

My application won't print to a StyleWriter II although it works fine on all other printers including the original StyleWriter. Can you give me any suggestions? What's different about the StyleWriter II?

The StyleWriter II driver is a member of the "GrayShare" driver family, with new features such as support for grayscale printing (if Color QuickDraw is available) and printer sharing over the network. Its internal architecture is very different from previous printer drivers. In spite of thorough compatibility testing, some problems have shown up since the first release. In many cases, the driver revealed weaknesses in the applications themselves; for some other problems, a solution will be incorporated in the next release of the driver. Here are currently identified problem areas with the StyleWriter II driver:

- The StyleWriter II driver handles memory differently. For example, if an application has a dereferenced handle to an unlocked block while `CopyBits`'ing into the printing port, this may work fine on all other printer drivers, but the block is likely to move with the StyleWriter II driver, and the results are unpredictable.
- The StyleWriter II driver maintains its own A5 world internally. If an application installs a `GrowZone` proc and forgets to set up its own A5 in the `GrowZone` proc (ignoring the Technical Note "Register A5 Within `GrowZone` Functions"), the `GrowZone` proc may get called with the StyleWriter II driver's A5, which obviously is bad for the survival of both the application and the printer driver.
- The StyleWriter II driver calls the `pIdleProc` (in the job subrecord of the print record) more often than other printer drivers; in particular, it may be called during execution of `PrOpenDoc`. If an application reloads a print record previously used, containing an old (now invalid) pointer to an `idleProc`, and doesn't update the `pIdleProc` field before calling `PrOpenDoc`, disaster is very likely. Note that the Technote "pIdle Proc (or how to let users know what's going on during print time)" recommends installing the `idleProc` before `PrOpenDoc`.
- Like most other QuickDraw printer drivers, the StyleWriter II driver uses algorithms built into QuickDraw for rasterizing picture elements like ovals and arcs, but not necessarily with the same sequence of coordinate transformations. Because of the 360 dpi resolution, internal computations with the transformed coordinates may hit the 16-bit integer limitation of QuickDraw, and reveal bugs in the old QuickDraw routines that have never been discovered so far.
- The StyleWriter II driver (version 1.0) contains STR# resources with positive ID numbers that may conflict with STR# resources in an application. This will be fixed in the next release of the driver.
- Under certain circumstances, the StyleWriter II driver seems to have trouble with a PmForeColor call. This is under investigation and will be fixed.

Back to top

Determining printer font

Date Written: 1/26/93

Last reviewed: 4/26/93

Our application must print with Helvetica if available, and Geneva if not. How can I determine which fonts are available in a printer? I've been using the `wDev` field in the print record and assuming that a value of 3 means Helvetica will print correctly, but according to "Print Hints: Top 10 Printing Misdemeanors" in *develop* issue 12, this isn't dependable.

There's no procedural way to ask a printer if it even has built-in fonts, let alone what they are. By using `wDev == 3`, you've been assuming that all PostScript printers have built-in Helvetica fonts, and while most do, some do not. There are several things you can do, though.

First of all, you should know about font substitution. If this option is checked in the LaserWriter driver (it's checked by default), you'll get Helvetica instead of Geneva when drawing in Geneva on PostScript printers that have Helvetica. The driver adjusts all the font metrics so the widths match what you see on the screen. In some versions of the LaserWriter driver, you'll get Geneva instead--but only if the Geneva TrueType font is available, so you get excellent quality either way. This might be enough to solve your entire problem.

If not, there's the new `PSWriter` driver on AppleLink and on the *Developer CD Series* disc. `PSWriter` uses PostScript Printer Definition (PPD) files and supports a new `PrGeneral` call that lets you obtain the PPD file so that you can know things about your target printer, including what fonts are installed. While this might give misleading results if the user has never set up the printer (and is using the Generic PPD), it should be just fine in all other cases.

There's no way to tell whether the user has downloaded a font to the PostScript printer, making it available, because it won't be in the PPD file and there's no way to ask the printer.

Back to top

Where to get PostScript Printer Definition (PPD) files

Date Written: 2/9/93

Last reviewed: 4/1/93

Where can I get PostScript Printer Definition (PPD) files for Apple's PostScript printers?

Apple recently made PPD files for its printers--past and present--available. You can find them on AppleLink in the Software Sampler: Apple SW Updates: Macintosh: Printing Software: PPDs folder.

PPD files for non-Apple PostScript printers with ROMs by Adobe Systems are available from a clearinghouse maintained by Adobe. To request a PPD file from Adobe, send an internet message to "ps-file-server@adobe.com" with the subject "help". Adobe's list server will respond with mail about items available on the list server and how you can get them.

Back to top

LaserWriter Driver 7.2 isn't ColorSync aware

Date Written: 1/11/93

Last reviewed: 4/1/93

Is LaserWriter Driver 7.2 ColorSync aware?

No, LaserWriter Driver 7.2 isn't ColorSync aware. Apple plans to add ColorSync capability to future versions of the LaserWriter driver, but we don't have any more information or details than that.

Back to top

Printing nonstandard page dimensions

Date Written: 1/14/93

Last reviewed: 6/14/93

How do we override the standard page dimensions for the StyleWriter or, for that matter, any other printer? We want to print 5.5" x 8.5" (half letter-sized) pages.

The current printing architecture isn't designed for elegant (or sometimes *any)* communication from your program to the printer driver. All the calls you normally make (such as `PrOpenDoc`, `PrOpenPage`, and `PrPicFile`) are handled by the printer driver and not by the Printing Manager. The drivers manage the print records, put up the dialog boxes, decide how to render graphics and everything else in between. That's why writing a printer driver is so hard.

If you want to tell a printer driver "I want to use this size paper," that driver has to implement a driver-specific way to do it or it can't be done. There's no PrSetPageSize call in the current printing architecture. In fact, trying to print outside the rPage rectangle can cause physical damage on some printers, so customizing page sizes is not something you want to enter into lightly.

You can't change the paper size in the print record, because most drivers don't look there to see what size you *want;* instead they tell you what size you *get* in that field. The same is true for most of the print record: Unless *Inside Macintosh* tells you that you can write to that field, you can't (or doing so won't do any good).

The StyleWriter in particular has very limited paper options; it handles letter, legal, A4, and B5 paper sizes--and that's it. If you try to print other sizes, the hardware reports an error. This is pretty common in these days of very low-cost high-quality printers; features that you as a programmer would want are jettisoned to keep the cost down. Most low-cost laser printers have limited paper-handling capabilities as well. If you try to print continuous feed paper on a StyleWriter, you'll get a paper jam error after about 15 inches of paper pass through (long enough for the printer to realize it's not legal-sized paper). Your half-letter size pages are too short for the printer to use.

While QuickDraw GX has a much more robust printing architecture, that doesn't help right now for the current world. Something that you may wish to consider is making forms available on regular letter-sized paper, with two per sheet. That size paper works in *every* printer sold in the United States and could be perforated and detached by the customers.

All of Apple's LaserWriter printers have excellent paper handling but cost relatively more. The best way to handle custom page printing across *all* printers is to place your custom-sized image on an 8.5" x 11" page and provide special paper.

Many, but not all, of Apple's printers support a PREC 3 resource that contains extra page sizes in a custom format. The format of this resource is as follows:

```
PageSizeRecord = RECORD
 numItems: INTEGER;
 botRights: ARRAY [0..NumPageOptionsZB] OF Point;
 titles: PACKED ARRAY [0..0] OF CHAR;
```

numItems is the number of page sizes displayed in the dialog. The maximum number is six (since there is only room for six radio buttons in the Page Setup dialog). botRights is an array of points specifying the bottom right corner for each page. This corner is specified in 120ths of an inch, and it should define the physical size of the paper, not the imageable area.

titles is a packed array of page size names. Each name begins with a length byte, followed by the number of characters specified by the length byte. The strings are packed, so string1[LENGTH(string1)+1] would reference the length byte of string2.

To use this mechanism, your application should create a PREC resource with an ID = 4. You can do this very easily by copying PREC(3). Once you have a copy, you can modify the page sizes as you wish. When the printer driver is called to display the Page Setup dialog, it will first look for PREC(4). If found, it will use it to define the page sizes. If not found, the driver will use the standard sizes in PREC(3).

This method only works with some of Apple's drivers; the LaserWriter driver ignores these resources. If you decide to implement this feature in your application, be sure to warn your users that custom page sizes may not be available on all devices.

Use this feature at your own risk; it's very likely not to work under QuickDraw GX and later system software.

Back to top

LaserWriter driver 'feed' resource troubleshooting

Date Written: 11/24/92

Last reviewed: 3/1/93

The userdict definitions in my LaserWriter driver 'feed' resource (written per the Macintosh Technical Note "Feeder Fodder") get wiped out after each page. At least, every time my redefined showpage is executed the variables that should have been defined the first time in userdict are unbound. What could be happening here? How should I define something in 'feed' resource code that acts like a global variable for a redefined showpage?

Your operators shouldn't be wiped out with each page. You're being loaded before the PREC 103 is loaded and before the series of save and restore operators are called, which could otherwise wipe you out. Please read the Macintosh Technical Note "The Lo Down on Dictionary Downloading" for a robust description of this process. Thus, we must look to other reasons why your operators are being wiped out.

The natural culprit is your PostScript. Are you wiping yourself out? Second, you should heed the warning on page 4 of the "Feeder Fodder" Technical Note: "As always, calling or redefining operators defined by the LaserPrep (md) dictionary isn't supported." You could see problems as you describe if you're redefining LaserPrep dictionary commands. Other variables include the system software and LaserWriter driver versions that you're using. It could be an issue with the type of LaserWriter itself.

Back to top

LaserWriter driver doesn't `SwapMMUMode` for 32-bit addresses

Date Written: 11/17/92

Last reviewed: 6/14/93

When I use `CopyBits` to move a `cGrafPort`'s portPixMap to another `cGrafPort` (my printing port), it works like a charm when background printing is turned on, but when `CopyBits` gets called with background printing turned off, the image that prints isn't the image at all. Why is this happening?

You should be aware that since you're copying the pixels directly from the screen, the baseAddr pointer for the screen's pixMap may be 32-bit addressed. In fact, with 32-Bit QuickDraw, this is the case. This in itself isn't a problem, since `CopyBits` knows enough to access the port's pixMap's baseAddr in 32-bit mode, as follows:

```
mode = true32b;
SwapMMUMode(&mode);  // Make sure we're in 32-bit addressing mode.
// Access pixels directly; make no other system calls.
```

That's how you'd normally handle things if you were accessing the pixels directly yourself. Unfortunately, the LaserWriter driver doesn't know enough to do the `SwapMMUMode` and instead ends up copying garbage (from a 32-bit pointer stripped to a 24-bit pointer).

So why does background printing work? Because when you print in the background, everything is rolled into a PICT, which the driver saves off for PrintMonitor. Since the driver is using the standard QuickDraw picture bottlenecks to do this, and `CopyBits` knows to swap the MMU mode before copying the data into the picture, everything works great. Later, at PrintMonitor time, the picture is played back. Since the data is no longer 32-bit addressed, the LaserWriter driver doesn't have to call `SwapMMUMode` to do the right thing; it can just play the picture back.

The solution we propose for you is something similar. At print time (before your `PrOpenPage` call), call OpenPicture, copy the data from the screen with `CopyBits`, call ClosePicture, and then call DrawPicture within your `PrOpenPage`/PrClosePage loop. That should do the trick.

Note that copying bits directly from the screen is not something we recommend. Unless you have no alternative, you should *always* copy from the original source of the data instead.

Back to top

System 7.1 and printer driver compatibility

Date Written: 11/18/92

Last reviewed: 6/14/93

If a user upgrades from System 7.0.x to 7.1, a new version of the PrintMonitor is installed. Will existing printer drivers behave correctly with the new PrintMonitor?

When a user installs System 7.1, Easy Install will update any existing printer software. If the user picks Customize, he or she must choose the printers to be updated. Only the versions of the printer drivers that come with System 7.1 should be used with 7.1. Mixing and matching is bad.

Since only Apple's printer drivers use PrintMonitor, as long as you upgrade to System 7.1 by using the Installer there's no issue at all.

Back to top

System 6 and System 7 PrintMonitor

Date Written: 11/18/92

Last reviewed: 3/1/93

Is the version of PrintMonitor that ships with System 7.1 compatible with System 6? If so, which version of the Backgrounder file would be compatible for installation with the System 7.1 PrintMonitor on System 6.x systems?

Only Apple's printer drivers should be using PrintMonitor--the interface is undocumented and unsupported for other printer drivers. PrintMonitor-like things are a different story, but the specific Apple product PrintMonitor works only with Apple's printer drivers.

While the drivers may be used under System 6, users should use the versions of Backgrounder and PrintMonitor present on the System 6.0.8 disks. Since Backgrounder and PrintMonitor are tied together, we have to recommend against using the 7.1 PrintMonitor under System 6.

Note that in all these cases, nothing was specifically added to make things *not* work under System 6--they simply weren't tested and therefore can't be supported.

Summary:

System 6: Backgrounder and PrintMonitor from 6.0.8, current driver

System 7: Use whatever the Installer installs.

Back to top

Which printers to test printing

Date Written: 10/1/92

Last reviewed: 11/2/92

We need to know what printers on which to test our application. It prints only pixmaps. It's been tested on some LaserWriters. The question is, how many brands and models of printers do we need to test it on? Are there any definitive printing tests (for example, does success on a LaserWriter indicate correctness of the application)?

Unfortunately, there is no such thing as a universal printing test suite. Moreover, successful printing not only depends on the specific printer driver (and its version), but also on the available memory at print time, and on things like the font configuration in the system. (Given that you print pixmaps only, the last comment is not applicable. Most certainly, all potential problems will rather boil down to out of memory conditions). Also, we have seen applications with severe defects in their printing code print quite correctly to the LaserWriter, and fail only on a few other specific printers. Most other printing problems have to do with assumptions about the inner working of a specific printer driver, such as hacking the print record behind the back of the Printing Manager. In this case, trouble with at least one of the 150 plus different printer drivers out there is guaranteed!

Pragmatically speaking, you may want to test printing in priority on the printers most likely being used by your customers. Currently (as of November 1992), this is the PostScript LaserWriter (driver version 7.1.2 or later), the StyleWriter (driver version 7.2.2), the Personal LaserWriter LS (driver version 7.2), and one or another of the most popular third-party printers. In addition, we recommend that you test both on an older LaserWriter (up to the LaserWriter IINT), and the newer ones (LaserWriter IIf, IIg, NTR), because of the differences in the PostScript interpreters and gray-level support.

Back to top

Where to get documentation on writing a Macintosh printer driver

Date Written: 4/9/91

Last reviewed: 8/1/92

Where can I find documentation on how to write a Macintosh printer driver equivalent to the ImageWriter or LaserWriter driver? In particular, how are Printing Manager and QuickDraw commands translated into calls to the printer driver?

DTS's "Learning to Drive" document and "StdFileSaver" source code, available in AppleLink's Developer Support folder and on the latest *Developer CD Series* disc, are helpful references.

Back to top

Writing a Macintosh printer driver

Date Written: 5/3/89

Last reviewed: 8/1/92

I have a printer I would like to connect to the Macintosh. Where can I find information on writing a printer driver?

A Macintosh printer driver is more than a standard device driver. A printer driver contains code to implement all of the standard Macintosh Printing Manager routines. The code includes routines like PrOpen/PrClose, and PrOpenDoc/PrCloseDoc. A Printing Resource File contains all of the resources (including code) to implement the Macintosh Printing Manager for a particular device. A driver works best with all Macintosh applications if it supports both the high- and low-level Printing Manager interfaces, as well as the PrGeneral (IM V:410) routine and its associated opcodes.

Apple is not currently supporting the development of Chooser-selectable device drivers, at least not those that support printers. There are many reasons for this, and here are at least a few of them:

First, there is no documentation or examples available. Each of the Apple printer drivers is unique. They are all written almost entirely in 68000 assembler, and consequently, are not very easy to read. Since each driver is different, the only real documentation for how the drivers work is the source code to the driver. There is some general interface information available, but important information, like how the driver manages its print record, is described only in the source code. This information could be extracted and distilled into some kind of document, but this brings up the next problem:

The Macintosh Printing Manager is currently being revised and enhanced. These revisions will require major changes to the architecture of the Printing Manager. It's not clear what effect these changes will have on existing printer drivers, but it is very possible that the drivers that exist today will have to be revised significantly to run under the new Printing Manager.

Apple is designing and implementing the next generation Macintosh Printing Manager. Developers' suggestions for features are being taken into account. One of the goals of the new architecture is to make writing drivers much simpler, and to allow the sharing of code between drivers. When this architecture becomes available, Apple will reconsider its position of driver development, and will probably end up with some kind of licensing agreement for developers that want to write drivers. Until then, the preceding reasons should be adequate justification for NOT attempting a Macintosh printer driver at this time.

If you still aren't convinced, and want to write a driver despite the challenge, see the article in the December '88 issue of *MacTutor* magazine. There is an example printer driver written in C. It is just a skeleton driver, and does not handle any of the more difficult problems such as line layout, font substitution, graphics, or banding. But it is a start, and gives you a general idea of the structure of a driver. Since some developers began writing their drivers before Apple discontinued support, DTS is still answering questions concerning driver development. However, these questions are limited to those that can be answered by the DTS engineers. You should also be aware that some of the techniques used by Apple printer drivers are considered proprietary. Information on methods used for line layout as well as certain aspects of font handling will not be disclosed.

X-Ref:

Printing Manager

Back to top

Changing printer driver settings without using the dialog

Date Written: 11/6/91

Last reviewed: 6/14/93

To change default printer driver settings without using the dialog, keep a copy of the print record with the various settings you want, and when you want to choose your settings, just use your special print record. The current Macintosh print architecture doesn't allow procedural access to the print record, so you must call the dialogs once for yourself, choose your settings, and then save a copy of the print record that is produced. Some settings are job-dependent, but values that are retrieved from the print record can be changed using this technique. You can only save the settings in the Style dialog. Call PrValidate when retrieving a print record from the resource fork to make sure it is "good."

See *develop* , issue #1, pages 58-65, for a complete discussion of changing default printer driver settings and sample code.

Back to top

Macintosh print record wDev field

Date Written: 5/3/89

Last reviewed: 8/1/92

What is the Macintosh print record wDev value for the ImageWriter LQ, LaserWriter IISC, and AppleFax Modem?

Apple strongly discourages the use of the wDev field in the print record for several reasons. First, this field contains a unique ID for each printer driver on the Macintosh. Coding your application to use this ID makes it device dependent, and device dependence will cause many problems in the future as the Macintosh Printing Manager continues to evolve. Many applications currently check for wDev = 3 to determine whether or not to send PostScript. Although the Apple LaserWriter driver has a wDev of 3, third-party printer drivers for PostScript devices do not, so if your application determines whether or not to send PostScript based on wDev alone, your application may incorrectly print with QuickDraw on third-party PostScript devices. A second problem concerns spoolers and spool files. If a spooler is installed between an application and the printer driver that is going to do the printing, the application receives the wDev ID of the spooler instead of the target driver. If the application makes assumptions based on this ID, it probably gets unexpected results. In the future, it may also be possible to redirect spool files. This means that a file that was originally spooled for a PostScript printer may be redirected to a QuickDraw device. If the spool file contains only the PostScript representation of the document, it will be useless to the QuickDraw device. Despite these strong warnings, some developers are convinced they need the wDev values, and there may be some vertical applications where they're needed. For those cases, here is the current list of wDev IDs for Apple printer drivers:

```
    Device              wDev (Hi Byte)
    ImageWriter I/II:    1
    LaserWriter,
    LaserWriter Plus,
    LaserWriter IINT,
    LaserWriter IINTX:   3
    LaserWriter IISC:    4
    ImageWriter LQ:      5
```

DTS does not support the use of these constants. They are definitely subject to change.

Back to top

Determining if a printer driver accepts Color QuickDraw calls

Date Written: 11/21/90

Last reviewed: 8/1/92

How do you find out if a printer driver accepts Macintosh Color QuickDraw calls?

Check to see if the printer driver has returned a color grafPort to your application after the call to PrOpenDoc (that is, the port that PrOpenDoc returns).

To determine if the grafPort is color, you need to check to see if rowBytes from the grafPort are less than 0. The following code fragment demonstrates this idea:

```
(* This function determines if the port passed to it is a color port. If *)
(* so, it returns TRUE. *)


FUNCTION ColorPort(portInQuestion: GrafPtr): BOOLEAN;
BEGIN
    IF portInQuestion^.portBits.rowBytes < 0 THEN
        ColorPort := TRUE
    ELSE
        ColorPort := FALSE;
```

Back to top

Printing in mixed Macintosh System 6/7 network

Date Written: 12/11/90

Last reviewed: 8/1/92

What is the recommended printer driver for a network environment with mixed operating systems (such as System 6.0.5 on a Macintosh Plus and 7.0 on two Macintosh IIfx systems)?

In a mixed 6.0.x/7.0 network we recommend upgrading all systems to the 7.0 print drivers--even systems running 6.0.x. The 7.0 LaserWriter drivers work fine under system 6.0.x and will avoid any printer reinitialization problems. Use the 7.0 printing disk (or the printing install folder), and launch the Installer from that disk. It has scripts needed to install 7.0 print drivers on a 6.0.x system. Do *not* select the Installer options for printers from the main 7.0 Installer, because these scripts currently are only for systems running 7.0.

If a Personal LaserWriter NTR is used on the network, you must upgrade all workstations to the 7.1.1 LaserWriter driver or later. (The NTR requires at least LaserWriter 7.1.1.)

Back to top

System 7.0 LaserWriter driver & choosing nonstandard page size

Date Written: 4/8/91

Last reviewed: 6/14/93

With the System 7.0 version of the LaserWriter driver, when the user selects Envelope from the Page Setup dialog, the page size returned by the driver is still a standard page: 8.5 x 11. How do you recommend that applications display the page size when the user has chosen a nonstandard page size?

We recommend that you have the page preview show a full page instead of an envelope-sized page. The LaserWriter driver supports a large number of PostScript devices, and it can't be sure whether the envelope will be fed on the right, left, or center of the paper tray. If you show the full page, a user can print on any device by putting the text in the correct location for that device.

Manufacturers of PostScript printers can add custom page sizes to the LaserWriter driver. If they do, the representation on the screen will be whatever they decide to define. Applications should not try to interpret custom page sizes. If your application ignores the results returned by the driver, you risk incompatibility down the road.

Back to top

Asynchronous LaserWriter driver no longer supported

Date Written: 9/24/91

Last reviewed: 8/1/92

What is causing incorrect characters to be printed on the asynchronous LaserWriter driver we licensed from Apple for use with a non-Apple PostScript-equipped printer?

The printing of incorrect characters is probably due to an incompatibility between the async driver and the version of PostScript being used in the printer.

The reason for the incompatibility is that the driver is no longer supported and hasn't been revised for quite some time. The driver was originally developed outside of Apple, and Apple licensed the driver for its use. The company subsequently went out of business and the driver hasn't been revved since.

PostScript and Apple's AppleTalk LaserWriter drivers have been updated and changed quite a bit since then, which is why the characters print correctly when using AppleTalk, but not when printing asynchronously. The old async driver just doesn't know how to handle the new PostScript versions and therefore prints "garbage" characters.

The SL Laser II driver is no longer supported by Apple, and at this time Apple has no plans to update this driver or write a new one. Since things work correctly with AppleTalk, you should confine usage to AppleTalk when printing PostScript to the LaserWriter. If this is unacceptable, you might check around with different clearinghouses to see if a third-party developer has a compatible, up-to-date asynchronous LaserWriter driver that you can use.

Back to top

Use LaserWriter driver srcCopy instead of `srcOr` transfer mode

Date Written: 5/1/91

Last reviewed: 6/14/93

I'm creating PICTs that are comprised of many lines drawn in `srcOr` mode. When using the LaserWriter 6.x or 7.x driver with the Color/Grayscale radio button selected, some lines fail to print. Why is this happening?

The problem is a bug in the LaserWriter driver. Sometimes, when using a `cGrafPort`, the driver doesn't reproduce lines drawn in `srcOr` mode. (A `cGrafPort` is used when the Color/Grayscale print option is selected; in Black & White print mode, a regular `grafPort` is used.) A workaround is to use srcCopy instead of `srcOr` when drawing QuickDraw objects within your PICTs.

Back to top

Save and restore long word if using $948 under System 7

Date Written: 6/11/91

Last reviewed: 8/1/92

Unless my Macintosh application restores the contents of the long word at $948 after using that space for globals, the Finder draws icons incorrectly and my third-party LaserWriter driver crashes. Is somebody now using $948 for other purposes?

This is a known System 7 icon drawing bug, and also a bug with the printer driver that you are using.

The icon drawing utilities are trying to determine if a print page is currently open, by looking at that variable ($948 is part of printvars). This turns out not to be such a good idea, and it will be fixed in the next release of the system software.

The printer driver bug that you are experiencing is that every printer driver must return the variable at $948 to -1 when they are done printing. Also, while printing, the driver should set it to <> -1.

Back to top

Update Backgrounder if using LaserWriter 7.0 under System 6

Date Written: 7/10/91

Last reviewed: 8/1/92

When we use the LaserWriter driver 7.0 with System 6.0, the Chooser's Background Printing On button is always dimmed. Is there any way to enable background printing?

To get the LaserWriter 7.0 driver to work with System 6.0.x, update your Backgrounder file with the version on the Printing Tools disk that's used to install System 7.0.

In short, to use the 7.0 LaserWriter Driver, 7.0 PrintMonitor, and System 6.0.x, place these files from the System 7.0 Printing Tools installation disks into your System Folder. All three must be used together for printing to work correctly. After you've placed these files in the System Folder, you should find the Background Printing option enabled for your use.

There are no known incompatibilities with the 7.0 drivers and System 6.0.x, so everyone should use the latest drivers, even with a mixed environment. This will also get rid of "printer wars" that occur when users use more than one version of the LaserWriter driver to print to the same printer. (The printer must be reinitialized if the current user uses a different version of the driver than that used by the previous user.)

Back to top

LaserPrep 7.0 file & AppleShare Print Server

Date Written: 7/15/91

Last reviewed: 8/1/92

The LaserPrep 7.0 file is included on the System 7 Printing disk and the System 7 Group Update CD for upgrading the AppleShare Print Server to support the LaserWriter 7.0 driver. The AppleShare Print Server has its own LaserWriter driver built in and all it needs to print is a LaserPrep file. In fact, the AppleShare Print Server completely ignores any LaserWriter driver installed in the System Folder of the server.

Page 49 of the *System 7 Group Upgrade Guide* states the following procedure for upgrading an AppleShare Print Server to support the System 7 LaserWriter drivers:

1. Shut down the print server software.
2. Install the printer driver update.
3. Drag the LaserPrep icon from the Printer Update folder to the Server Folder of the print server Macintosh.
4. Restart the print server Macintosh and the AppleShare Print Server software.

Actually, step #2 in the *System 7 Group Upgrade Guide* is unnecessary. Installing the printer driver update has no effect on the AppleShare Print Server software.

Back to top

LaserWriter 7.0 driver and LaserPrep dictionary

Date Written: 8/29/91

Last reviewed: 6/14/93

In the old LaserWriter drivers it was possible to create a PostScript file with or without the LaserPrep dictionary ("F" or "K" key). Is it possible to generate a file without the dictionary with the LaserWriter 7.0 driver?

In 7.0 printing, a LaserPrep dictionary is always sent with a print job. This is done to prevent constant reinitialization of the LaserWriter by conflicting printer drivers. You cannot prevent it from being sent. Fortunately the 7.0 version of the LaserPrep dictionary is much smaller (~40K total) than its 6.x predecessors.

Back to top

LaserWriter ignores ForeColor while filling smoothed polygon

Date Written: 10/8/91

Last reviewed: 8/1/92

When doing the FillRgn for drawing the fill of a smoothed polygon (as described in Macintosh Technote #91) the foreColor isn't used on the LaserWriter. Any way to make it work?

You've discovered a design limitation of the LaserWriter driver. Things that have patterns associated with them are rendered by using the LaserWriter screen operators. This results in an assumption of the foreground color being black and the background color being white, which is what's causing the problem you noticed. In short, it makes the driver ignore your foreground and background colors.

You can work around the problem by working in an off-screen `GWorld` first and then CopyBitsing everything to the printer port using srcCopy. There are a couple of problems with this approach: First, don't do it with text or your text will be turned into a bitmap and you'll get the "jaggies." Second, you'll probably want to increase the printer port's resolution from its default of 72 dpi for better results. A value of 288 dpi works nicely since it's an even multiple of QuickDraw's native 72-dpi resolution. Also, make sure that the `GWorld` you create has the same bounds as the printer port's rPage rectangle to avoid unnecessary scaling and clipping. After you've done all that, draw into it and `CopyBits` the result to the printer port. The nice thing about doing everything off-screen first is that then you can use some of the non-printer-friendly transfer modes like blend or dithered. Also, you can use ForeColor/BackColor and get the right thing this way.

If you don't want to use this method for all printers, (since it can be quite a memory hog), you can check for the LaserWriter driver and use this method in just that case. For other drivers, you should just print as normal.

So, what do you need to do all this? Well, to set the resolution of the printer port, use `PrGeneral` as described in *Inside Macintosh* Volume V and *develop* #3.

To determine whether you have the LaserWriter driver or not, check the high byte of the `wDev` field in your print handle. This is described in the Technote "Optimizing For The LaserWriter--Techniques." While this method might break some day, it's currently the best way to determine which driver you're using, and Apple will have to let developers know before we break it.

If the high byte of the `wDev` is 3, then you either have the LaserWriter driver or a third-party driver impersonating the LaserWriter. Some PostScript drivers do that because many apps assume a `wDev` of 3 means a PostScript printer, and anything else doesn't. By acting like Apple's LaserWriter driver, they get preferential treatment from apps that "special case" for PostScript. That's not really a problem in this situation.

Back to top

LaserWriter driver PostScript error strings & document names

Date Written: 12/11/91

Last reviewed: 8/1/92

I discovered an interesting bug in the Macintosh LaserWriter driver. If the word "timeout" is in the name of a document, the LaserWriter driver will give a timeout error -8132. Are there similar magic words?

PostScript error messages are sent from the LaserWriter to the driver as text streams. The driver must check these strings to see if they contain an error message. If a document is named something that contains the same string as a PostScript error message, the driver may think there's an error when the printer sends the "status: printing document XXXXX" message. Other strings cause similar problems; one of them is "printer out of paper." If you want to see the rest of the strings, take a look at the LaserWriter printer driver resource type 'PREC' ID = 109.

Back to top

Personal LaserWriter NTR has longer product string

Date Written: 3/17/92

Last reviewed: 6/14/93

The product name stored for this printer is (LaserWriter Personal NTR) or (Personal LaserWriter NTR), depending on whether you're using statusdict or systemdict, respectively. In either case, the product's length is 24 characters. Since you're only allocating 20 characters for the 2nd string you use with cvs, you're getting a rangecheck error. You may want to make the string even larger than 24 characters to accommodate longer product names in the future.

Assumptions about the length of PostScript product strings are a common problem with new printers (having more verbose names). In fact, you should have the same trouble if you run your original PostScript on the Personal LaserWriter NT, which has a 23 character name.

Back to top

LaserWriter II SC fonts and 4x bitmap size

Date Written: 11/17/89

Last reviewed: 6/14/93

Why do I need to have four times the bitmap size of the font I want to print in on the LaserWriter II SC?

LaserWriter II SC fonts are four times the point size of the associated Macintosh screen font. The pixels on the screen are four times as far apart (center to center) as the dots printed by the LaserWriter SC. When the bitmaps for one of the printer fonts is printed, the result is a font with a resolution four times greater than that of the font displayed, but at a size identical to the font displayed on the screen.

Back to top

Font families shipped with LaserWriters

Date Written: 11/17/89

Last reviewed: 8/1/92

What fonts and sizes are shipped with the LaserWriter Plus, LaserWriter IINT, LaserWriter IINTX, and LaserWriter SC?

The LaserWriter Plus, LaserWriter IINT, and LaserWriter IINTX, shipped with a total of 11 font families with the sizes indicated below:

Times, Helvetica, Courier, Symbol: 9, 10, 12, 14, 18, 24

Palatino, Helvetica Narrow, ITC Bookman,

ITC Avant Garde Gothic, ITC Zapf Chancery,

ITC Zapf Dingbats, New Century Schoolbook 10, 12, 14, 18, 24

A total of four font families are shipped with the SC: Courier, Symbol, Times, and Helvetica in the following sizes: 9, 10, 12, 14, 18, 24, 36, 48, 56, 72, 96.

Back to top

Disable Graphics Smoothing for printing large bitmap images

Date Written: 5/3/89

Last reviewed: 6/14/93

When printing large (possibly scanned) Macintosh bitmap images, the page is printed with small lines running horizontally through the image. Why?

In System 6.0 the QuickDraw DrawPicture call was revised to fix some problems. One of these problems concerned large bitmaps. To help solve the problem of bitmaps that were too large to be printed, DrawPicture was modified to "band" the `CopyBits` request if it was too large. Banding is the process of converting a large bitmap into several smaller bitmaps. This banding usually occurs vertically down the page. When DrawPicture bands a large bitmap into pieces, the smoothing algorithm of the LaserWriter is applied to each piece separately, instead of being applied to the entire bitmap at once. Since the process of smoothing involves removing some pixels, hairlines will be created between the bitmap bands. There is no way to tell the LaserWriter driver that the smaller bitmaps are all part of one large bitmap, so the only solution to this problem is to disable Graphics Smoothing when printing large bitmap images.

Back to top

Difference between LaserWriter 7.0 and 7.1 Namer

Date Written: 3/3/92

Last reviewed: 8/1/92

Apple Software Licensing's "Exhibit C" lists localized Namer 7.0 versions in several languages, and mentions that these localized versions of the Namer are available in version 7.1. What is the difference in the Namer between 7.0 and 7.1?

The main difference between the LaserWriter utility 7.0 and 7.1 is that part of the Namer has been integrated into the 7.1 version of the software. Until the LaserWriter utility 7.1 was made available, the Namer was a stand-alone application. However, the LaserWriter utility does not do everything the Namer does. For example, the Namer renames AppleTalked ImageWriters, but the LaserWriter utility doesn't. For all your printer configuration needs, you should use the 7.1 version of the software.

Back to top

ImageWriter and printing multiple copies in draft mode

Date Written: 1/6/92

Last reviewed: 6/14/93

Our application can't print multiple copies on the ImageWriter in Draft mode. If we type in 3 copies in the ImageWriter driver dialog box in Draft mode, we only get one copy. Everything prints fine in Best or Faster modes. Is this a bug in the ImageWriter driver?

Although it seems like this must be a bug in the ImageWriter driver, it's not. In Draft mode, the application that's printing must make sure the required number of pages are printed. This involves cycling through the app's print loop for each copy to be printed.

In Draft mode on an ImageWriter, nothing is spooled to disk; the data is immediately sent to the printer. Therefore, the data is no longer available once it's sent to the printer (and there's no way for the print driver to resend it for multiple copies). When the ImageWriter spool-prints, the file that's created is printed the required number of times for you. Therefore, your app only needs to handle multiple copies when printing in Draft mode.

Here's what your app should do:

1. Validate the print record and present the job dialog. This allows the user to choose how many copies to print and allows the printer driver to adjust the print record to reflect how many copies your program has to print.
2. Get the number of copies that you're expected to handle from the print record. In the case of spool printing, this number will be set to 1 (since the multiple copies are handled for you by the printer driver). In any case, the following Pascal code will give you the correct info.
3. For each copy in `numCopies`, loop through the `PrOpenDoc/PrCloseDoc` section of your code.

This method is demonstrated in the code for the Macintosh Technote "A Printing Loop that Cares...," which describes a model print loop.

Back to top

Dogcow logo is trademarked

Date Written: 9/10/92

Last reviewed: 6/14/93

We would like to use the "dogcow" icon in our Page Setup dialog. Is the dogcow trademarked, and are there any restrictions on using this icon in our software?

Yes, the dogcow logo (along with its call, "Moof!") is a trademark of Apple and is proprietary. The dogcow appears on Apple's *Developer CD Series* disc and in other material. Apple has a pending U.S. registration on it. Accordingly, it's not available to third-party developers as an icon or file symbol.

Back to top

"Printer driver is MultiFinder compatible" bit

Date Written: 7/29/92

Last reviewed: 6/14/93

Could you tell me what the "printer driver is MultiFinder compatible" bit is used for?

The "printer driver is MultiFinder compatible" bit provides two features. First, it allows the printer driver resource file to be opened by multiple clients. This was obviously needed to support multiple applications printing simultaneously under MultiFinder. The other feature provided by the flag is the loading of PDEFs into the system heap rather than the application heap (which is where they go under the Finder).

The MultiFinder-compatible bit has a major limitation: If your driver has this flag set, you aren't allowed to add or resize resources, or do anything else that would cause the RAM-resident resource map to change. Although MultiFinder lets multiple applications open the printer resource file at the same time, it has no control over the resource map that gets loaded by the Resource Manager when the file is opened. Because of this, each client gets its own personal copy of the resource map. When these clients get done with the file, they write the resource map back to the file (via `UpdateResFile`). Obviously, if the resources have changed in any way, the last client to call `UpdateResFile` is the only one whose changes will be recorded. This is a "thrill seeker" method of handling the printer driver resource files, but since none of the Apple printer drivers currently add or resize resources, it made sense.

So the bottom line here is that if you want your drivers to be compatible under MultiFinder, you'll have to implement a scheme that doesn't require adding or resizing resources. It's OK to change the data in a resource, as long as you don't change its size. Changing the data won't cause changes to the resource map, so each client will still have accurate copies of the map.

Here's what would happen to your printer driver's resources under the Finder and MultiFinder when the MultiFinder-compatible bit is set:

* Under the Finder in system software version 6.0.x: All resources are loaded into the application heap--regardless of the resource attribute's bit setting. If the resource has the "load into the system heap" bit set, it will still be loaded into the application heap under the Finder. This makes sense in the Finder world because the application heap will usually have more room than the system heap.

* Under MultiFinder in System 6 or System 7: All the printer driver's resources will be loaded into the system heap. This is true whether the "load into the system heap" bit is set or not.

Why does the resource loading occur this way, even when the resource's "load into the system heap" bit is set? Patches to the `GetResource` trap load all your printer driver's resources into the system heap when the MultiFinder-compatible bit is set under MultiFinder, and into the application heap under the Finder (as described above), which is why you can't override this behavior.

By the way, you should be aware of the SetPDiMC MPW tool, which is available on the *Developer CD Series* disc. It will automatically set the MultiFinder-compatible bit for you when you build your printer driver.

Back to top

Use FillCRect off-screen, not directly to printer port

Date Written: 8/25/92

Last reviewed: 11/30/92

We're having problems with color patterns using the LaserWriter driver version 7.1.2. If we create a 'ppat' resource in ResEdit (32 x 32 bits, in this case) and then do a `FillCRect` to the port returned by `PrOpenDoc` (with color set so that it's a `cGrafPort`) with the pattern loaded by `GetPixPat`, we get a weird pattern. Doing the same to an off-screen

GWorld and using CopyBits to copy to the printer port works fine, if a little slowly. Are we missing something here?

You need to use the FillCRect call off-screen rather than directly into the printer port, for at least two reasons. First, the LaserWriter driver doesn't support filling objects with anything but black-and-white patterns because it uses the PostScript halftone screen functions to draw patterns. Second, the LaserWriter driver doesn't understand (or handle) pixPats. Therefore, your only recourse is the one you discovered--to copy to and from GWorlds. Unfortunately, FillCRect doesn't work with the LaserWriter drivers through version 7.2. After version 7.2 this probably won't be a problem.

Back to top

Color/Grayscale button on a non-Color QuickDraw Macintosh

Date Written: 8/31/92

Last reviewed: 6/14/93

The LaserWriter driver displays the Color/Grayscale option even on a non-Color QuickDraw Macintosh, which seems odd because the option normally causes the driver to return a cGrafPort to the calling application, and CGrafPorts are only available under Color QuickDraw. What happens if you're printing original QuickDraw colors using the LaserWriter driver on a non-Color QuickDraw system? Does the driver just use the Black & White code regardless of whether the Color/Grayscale button is selected?

Regardless of whether Color/Grayscale is selected or not, the driver returns a grafPort because it's not running on a Color QuickDraw machine. But, if the Color/Grayscale option is chosen on a Macintosh without Color QuickDraw, the driver doesn't just use the Black & White code to print. Instead, it sends the colors to the printer as best it can without Color QuickDraw. This can result in three different sets of output when printing classic QuickDraw colors:

- Black & White button selected (identical output whether the system has Color QuickDraw or not)
- Color/Grayscale button selected on Color QuickDraw Macintosh
- Color/Grayscale button selected on non-Color QuickDraw Macintosh

Back to top

Downloadables

Acrobat version of this Note (K). Download
