Apple II Technical Notes

Developer Technical Support

Apple IIGS #93: Compatible Printing

Written by: Matt Deatherage
September 1990

This Technical Note discusses printing on the Apple IIGS and how you can make your printing code more compatible.

How Does Printing Work Anyway?

There are, in general, two types of printing done on the Apple IIGS. The first kind is "desktop" printing, which uses the Apple IIGS Print Manager to render images created by QuickDraw II onto an output device. The other kind of printing is "text" printing, which is similar to the way classic Apple II applications print—you send ASCII text somewhere and a printer prints it as ASCII text. This printing normally involves no graphics and is very quick.

This Note covers both types of printing, and by understanding the internals and the methods used to print, you can avoid compatibility headaches in the future.

Desktop Printing

Desktop printing uses the Apple IIGS Print Manager. The process is described in detail in the Print Manager chapter of the *Apple IIGS Toolbox Reference*, and usually consists of a simple print loop:

Open a document (`PrOpenDoc`)
    Open a page (`PrOpenPage`)
    Draw or Image the page in your favorite way
    Close the page (`PrClosePage`)
    Repeat for each page

Close the document (`PrCloseDoc`)
Print the document if it's spooled (`PrPicFile`)

Note that you should **always** call `PrPicFile` at the end of your print loop. It completes the printing process, even for immediate or draft printing.

There's one real secret about the Print Manager that can cloud your understanding of printing—the Print Manager doesn't actually do anything. It loads, unloads, and keeps track of printer drivers and port drivers and performs some necessary housekeeping, but that's about it. Many people believe that the Print Manager is responsible for all imaging, managing documents, managing a printing `grafPort` and such, but it's not. (The myth is perpetuated by the *Toolbox Reference* which refers to these functions as handled by the Print Manager.) In fact, these functions are handled by printer drivers.

You actually call the printer driver for all of the routines in the print loop; all the Print Manager does is make sure the driver is loaded and dispatch to it. Therefore, most of the compatibility issues you have with printing are not with the Print Manager, but with printer drivers.

Dealing With the Print Record

It's the printer driver's job to get information about a printing job from the user (it's the printer driver that handles the style and job dialog boxes, since the Print Manager cannot generically know what style and job options any printer can support), keep track of it, and print the document using those settings. Those settings are kept in a data structure associated with a document known as a print record.

Apple had only released two printer drivers at the time the first volume of the *Toolbox Reference* was published, and therefore the descriptions of the print record in that volume tend to be absolute. For example, the `iDev` field is documented as "one for an ImageWriter and three for a LaserWriter." In fact, the `iDev` field is the only method of print record interpretation available and there are several values for it:

$0001 = ImageWriter
$0002 = ImageWriter LQ
$0003 = LaserWriter
$0004 = Epson
$8001 = Generic dot-matrix (interprets the style subrecord like the ImageWriter driver)
$8003 = Generic laser printer (interprets the style subrecord like the LaserWriter driver)

If you have checks in your code like "If it's not $0001, it must be a LaserWriter," you have problems with most of the other printer types.

The $8000 and greater `iDev` values are defined for third-party printer drivers. The printer driver has no way other than the print record to keep track of values for a given print job, so it has to store all such information in the print record. If all third-party drivers use proprietary style subrecord formats, no applications can read or set any of those values. Those drivers which can use the compatible $8000 and greater `iDev` values indicate to applications that the definitions in *Toolbox Reference* for the ImageWriter and LaserWriter drivers apply to these drivers as well. `iDev` values of $0002 or $0004 also interpret the style subrecord as the ImageWriter driver does.

Print Record Rules

Remember: the print record is the only way the printer driver has to maintain information about a particular job. The print record belongs to the user, the document, and the printer driver—**not**

the application. Here are some rules for staying out of print record trouble.

- Always call `PrValidate` when changing fields in the print record. Even if a driver interprets the style subrecord like the ImageWriter driver, it may not support all the ImageWriter's style features (e.g., color printing). Calling `PrValidate` every time you change something in the print record gives the printer driver a chance to look at the havoc you've wreaked and correct it if necessary.

  You do not always get a feature you want. If a printer does not support color printing, you can set the "color" bit all day long and `PrValidate` clears it every time. You should be prepared for a new printer driver that does not support the features you want, and inform the user that the feature is not supported by this printer.

- Do not patch `PrValidate` to make it ignore bogus values in the print record unless instructed to do so by the printer driver author.

- **Never**, **never** tread on reserved fields in the print record. If you find a particular driver storing useful values some place, forget it. This is the only place a driver has to store information about a print job and some of it is not going to be supported.

  In particular, never try to interpret any values you may find in the `printX` subrecord of the print record. This subrecord is for the private use of printer drivers. Although `printX` is currently the worst compatibility risk, you must not tamper with other reserved fields.

- If you want to learn more about printing, learn how printer drivers work. The specifications are in Apple IIGS Technical Note #35, appropriately entitled "Printer Driver Specifications." An understanding of how printer drivers do their work is an understanding of how printing works.

Text Printing

Text printing generally uses the built-in ASCII mode of most dot-matrix printers to print text quickly and efficiently.

Desktop printer drivers often have a "draft" mode, where they print text immediately instead of imaging it in the appropriate font and style. This is accomplished by intercepting low-level QuickDraw II routines called bottleneck procedures. When QuickDraw is called to draw text, the printer driver gets control instead and sends the text to the printer.

Although this is useful to users of desktop printer drivers, it is not a required feature of any printer driver, and those that do implement it each do so in their individual way. For example, the LaserWriter driver doesn't support this model of "draft" printing because the LaserWriter is normally a PostScript® device—sending straight ASCII to it doesn't necessarily work.

To imitate the way classic Apple II applications print, your application prompts the user for some device through which to print, and ASCII characters are sent through that device. There are a few ways to do this.

Using the Print Manager

You can still use the Print Manager to print in ASCII mode by bypassing the printer driver. Simply use the Port Driver to send ASCII characters to the given target device with the `PrDevWrite` call. The specifications for Port Driver calls are in Apple IIGS Technical Note #36, also appropriately entitled "Port Driver Specifications." You make port driver calls as if they were Print Manager calls.

Although this method has been used, Apple does not recommend it. If the selected port driver is a network driver, this method is troublesome.

Using the Text Tools

By using the Apple IIGS Text Tools, you can ask the user what slot to print through and send ASCII characters to that slot or port. Although this is better than using the Port Driver, it still has problems. The Text Tools cannot be fully GS/OS Slot Arbiter compatible; therefore, there might be GS/OS devices accessible to the user to which your application does not let him print. Also, it's difficult to detect which slots really have Text Tools' devices without knowing about Apple II firmware, and prompting the user for a slot number invites trying to print to the disk firwmare, which usually justs reboot the machine (unceremoniously).

Using GS/OS

GS/OS supports character drivers, such as printer interfaces, and using them is the best way to handle ASCII printing. GS/OS supports loaded drivers for character devices if you have them, and generates drivers for character devices it can recognize. In addition, GS/OS drivers have identification words so you can prompt with real messages instead of cryptic slot numbers.

You can use the GS/OS call `DInfo` to loop through all drivers and prepare a list of character drivers. You can then change their device IDs into text phrases, place them in a list, and prompt the user to select one. This call usually results in a list such as "Printer port, Modem port, Remote Print Manager, Printer interface, Text screen [the Console driver]." You may wish to change the names of the devices slightly to make the choice easier (e.g., "network printer" instead of "Remote Print Manager").

Apple strongly recommends using GS/OS for ASCII printing from 16-bit applications.

Note: The Remote Print Manager (.RPM) device driver in System Software 5.0 to 5.0.2 has a bug which causes character loss. System Software 5.0.3 fixes this bug.

Further Reference

- *Apple IIGS Toolbox Reference*
- *GS/OS Reference*
- Apple IIGS Technical Note #34, Low-level QuickDraw II Routines
- Apple IIGS Technical Note #35, Printer Driver Specifications
- Apple IIGS Technical Note #36, Port Driver Specifications
- Apple IIGS Technical Note #69, The Ins and Outs of Slot Arbitration
- Apple IIGS Technical Note #75, BeginUpdate Anomaly