Open-Apple™

Releasing the power to everyone.

November 1985
Vol. 1, No. 10

ISSN 0885-4017
single copy price: $2.00
photocopy charge per page: $0.25

Special Issue: Solving Printer Problems

I am sure a survey of ***Open-Apple*** readers would show that the vilest piece of equipment that can be connected to a computer is a printer. While great advances have been made in taming these noisy cantankerous beasts during the last few years, it is quite clear from ***Open-Apple's*** mail that printers continue to plague many of you—programmers and non-programmers alike.

Printers are trouble because "standards" for printers are weak or nonexistent. The wide variety of equipment and software in use makes it impossible to give specific "do this" answers to most printer-related questions. The right answer depends on the exact eccentricities of the equipment, software, and connections involved.

The bad news is that the only person who is in a position to solve YOUR printer problems is YOU. You are the one who selected the particular combination of hardware and software you have—there may be fewer than a hundred other systems exactly like yours in the world.

To solve printer problems, you have to learn what your printer's special features are and how to turn them on and off. You have to learn about the features your software has for dealing with printers. And you have to learn about a mysterious creature that sits between your software and printer passing information along. This creature is responsible for many printer-related problems because it likes to snack on a character now and then or to burp a few extra characters into your manuscripts and reports.

The good news is that this month's ***Open-Apple*** is a special issue dedicated to a detailed study of the printing process, which should give you the background information you need to solve your own printer problems. In this article we'll study the Apple-to-printer connection with reference to several different types of printers, software packages, and other equipment. Even though you may not have any of the exact items I'll use as examples, you'll be able to learn a lot about your own system by looking for similarities and differences.

There are five points at which trouble can occur when you are using your computer's printer. Your software—for example, AppleWorks, *Apple Writer*, or an Applesoft program—must be capable of sending a stream of characters somewhere. This is the first place trouble can occur.

The next is that mysterious creature that eats letters for breakfast. If you have an Apple with slots, this creature is a plug-in interface card. If you have an Apple IIc, this creature is a built-in interface and connector. The IIc's built-in interface works exactly like a card in a slot (except that you can't remove it and throw it into the sea), so all further references to "interface cards" apply to it as well.

The third point of trouble is your printer itself. The fourth is the connection—the wire and electronic signals—that run between your computer and your printer. The fifth is the internal connection between your software and your interface card.

The software-to-interface card connection

Let's begin our investigation with the connection between your software and your interface card. This connection lies in the domain of the Apple's operating system—usually DOS 3.3 or ProDOS. Under ProDOS, the active "system" program—Basic.system, for example—actually handles this connection. If you are using Applesoft, you tell DOS to send all further output to the printer with commands like these:

```
PR#1                          (from the keyboard)

1000 PRINT CHR$(4);"PR#1"  (from inside a program)
```

This example assumes, of course, that the printer is connected to an interface card in slot 1. This is the customary slot for printers, but both DOS 3.3 and ProDOS allow you to put a printer in any slot. Other operating systems don't have this flexibility.

The downside of this flexibility, however, is that you must always *specify* which slot your printer is connected to. If you've used *Apple Writer*, you are familiar with that program's "Print Destination" parameter on the (control-P) "Print/Program Commands" menu. You use this parameter to tell the program which slot your printer is in. If you don't set it correctly, *Apple Writer* won't print. (If you set it to the nonexistent slot 8, incidentally, *Apple Writer* will send the printed output to a disk file.)

With AppleWorks, you set the slot by choosing to "specify information about your printer" from AppleWork's "other activities" menu. Every time you add a printer or change printer specifications, AppleWorks asks you to identify what type of printer you have, to type in a name for it, and to specify which slot it's connected to (you can choose to print-to-disk from this menu also).

Applesoft, as mentioned earlier, uses the DOS PR# command to route a program's output to a printer. Much confusion arises, however, because

"YES THEY ARE, DID YOU HAVE AN APPOINTMENT?"

Applesoft itself has a version of the PR# command. When you enter PR# on the keyboard, DOS sees it first and no harm occurs. Inside a program, however, Basic sees PR# first if you neglect to use CHR$(4) (and strange things begin to happen). PR# is an Applesoft command only because Applesoft needs a way to turn on a printer in a DOS-less environment (all there was when Applesoft was written). Avoid trouble and stick with the DOS CHR$(4) version of PR# when writing Applesoft programs.

Typically, getting a printer turned on is easy. Getting it turned off again sometimes causes problems, however. Early Apple manuals said to turn the printer off with a PR#0 command. Slot "zero" is where the Apple's own keyboard and 40-column screen were "connected" on the Apple II and II-Plus.

But then came 80-column cards and newer Apples. Eighty-column screens found a home in slot 3. But if a program turns the 80-column screen on with a PR#3, then prints with a PR#1, how should it get back to the screen?

If you use PR#0 for this, strange results usually occur. The problem is that your Apple's built-in software thinks you want to use 40-column mode but your hardware thinks you are still using 80-column mode (the only way to tell it you're not is to *type* "escape control-Q" on the keyboard or to *print* "control-U" to the screen.)

On the other hand, if you use PR#3 to return to the screen, the contents of the screen will be erased. This may be satisfactory in some situations, but in many others it is not at all the desired result.

The problem is that PR#3 tells the operating system to *initialize* and start using the 80-column card. It is this initialization that erases the screen. Since the card has already been initialized, this step can be skipped with no ill effects. To do it, you need to tell the operating system to connect into the 80-column card at address $C307, rather than the usual $C300. As mentioned in our January issue (page 6), with DOS 3.3 you do it like this:

```
POKE 54,7 : POKE 55,195 : CALL 1002 : REM  DOS 3.3 only
```

With ProDOS, on the other hand, you have two options. Basic.system allows an *a(ddress)* parameter to be used with the PR# and IN# commands. The parameter can be used in two different ways. If you give no slot number, Basic.system connects the output stream to the address you give. Alternatively, if you do give a slot number, Basic.system permanently changes the connection address for that slot, but does not actually change the connection. Thus, with ProDOS you can do it this way:

```
100 PRINT CHR$(4);"PR#3"        : REM first time 80-col sceen is turned on
110 PRINT CHR$(4);"PR#3,A$C307" : REM permanently change connection address

.....more program

500 PRINT CHR$(4);"PR#1"   : REM turn on printer

.....more program

600 PRINT CHR$(4);"PR#3"   : REM because of line 110, this connects at $C307
```

You could also make line 110 *PR#0,A$C307* and put a PR#0 in line 600. Then PR#3 would *coldstart* the 80-column card and PR#0 would *warmstart* it. The other alternative for returning to an unmodified 80-column screen with Basic.system is like this:

```
500 PRINT CHR$(4);"PR#1"   : REM turn on printer

.....more program

600 PRINT CHR$(4);"PR#A$C307 : REM no slot number given
```

The interface card-to-printer connection

Now let's look at the connection between your interface card and your printer. It consists of a cable and lots of black-box magic.

There are two very different electronic techniques commonly used in the Apple II world to connect printers to computers. These two types of "interfaces" are called serial and parallel. When you buy a printer, it will come with one type of interface or the other. You have to specify which type you want. (Some more expensive printers come with both.) Likewise, when you buy a printer interface card for an Apple slot, you have to specify whether you want parallel or serial. Obviously, your printer and interface card must match. You can't expect anything from a serial to parallel connection other than, perhaps, a few sparks.

To understand the difference between the two it's necessary to remember that the characters that appear on your printer exist inside your computer as 8-bit binary numbers. All the commonly-used alpha-numeric characters were long ago assigned binary values by an agreement called the American Standard Code for Information Interchange, or ASCII (say ask'-ee). The binary number 0100 0001, for example (decimal 65, hex $41), is the code for the capital letter A.

The parallel interface. Back when minicomputers were the rage, a company called Centronics was a big printer manufacturer. The engineers at Centronics designed a simple, inexpensive method to wire their printers to computers. The company was so successful that computer manufacturers began to build that kind of interface into their machines. After that other printer manufacturers began building it into their printers. And that's where the "Centronics parallel interface" came from. Perhaps because its details have never been blessed by an industry committee, it is the most standard "standard" in all of computerdom.

The Centronics interface uses eight wires to send all eight bits of a character at one time (in parallel). When a character is active on the eight wires, another wire is used to send a signal called a *strobe* to the printer. This tells the printer that the character is waiting on the eight data lines. After the printer has accepted the data, it sends a return signal, called *acknowledge*, on yet another wire. The computer then knows it can go on to the next character. This strobe/acknowledge protocol is an example of *hardware handshaking.*

The only problem with the Centronics interface is that it can get kind of expensive to use that many wires over long distances. Can you imagine the expense of having to use 10 phone lines to call a local bulletin board? Characters that must travel, especially over the phone, are typically broken into bits and sent one at a time over a single data line. This is how a serial interface works.

The serial interface. Just as we have a standard parallel interface, there is also a "standard" serial interface. This baby was developed back in the dinosaur age before electrons tickled the first microprocessor. At that time you used your dinosaur by calling it on the phone using a modem and a "data terminal" (usually a teletype machine). The RS-232-C standard specifies how the modems and data terminals were to be connected.

The standard shows as much forethought as a kid gives to sticking a frog in his back pocket when he hears mom coming. The engineers decided to put standard plugs into all modems and data terminals. Even though a serial connection really *requires* only two wires (a data line and a common ground), the RS-232 engineers added a few more wires so that printers and data terminals could get their acts together. They put in separate data lines in each direction. They added an "I'm OK" line in each direction. They added a line a modem uses to tell a data terminal the phone is ringing; a line a data terminal uses to tell the modem to answer it; a line a modem uses to tell a data terminal it has...in short, the official RS-232 connector ended up with 25 pins and a wire for every purpose.

But the worst part is that the two connectors *aren't the same.* In order to keep things "simple", the RS-232 engineers decided the wires inside their cable would run straight from pin to pin. Thus pin one runs to pin one, pin two to pin two, and so on. The implication of a scheme like this is that if one of the devices uses pin 3 to *send* data, the other device must use pin 3 to *receive* data. So, RS-232 modem connectors send data on pin 3; RS-232 data-terminal connectors send data on pin 2. And modems receive data on pin 2 (where data terminals send), while data terminals receive data on pin 3 (where modems send).

Now put your Apple between the modem and data terminal (printer). When talking to a modem, the Apple has to send data on pin 2. When talking to a printer, the Apple has to send data on pin 3. It would have been much simplier if RS-232 had said that all connectors should have the same pin format. The wires would not run from like-numbered pin to like-numbered pin, but *all cables and all connectors would be the same.* Now we have modem-type connectors and data-terminal-type connectors (neither of which logically belongs on a computer) and we have regular wired-straight-across cables for linking connectors of opposite types and cross-wired cables for linking connectors of the same type. The latter are called "null modem" cables, a name which doesn't help a bit in keeping this all straight.

But wait. Keeping the wires from getting crossed is only the beginning of making an RS-232 connection work. The "standard" does not specify the speed at which bits will be sent, the format the bits will appear in (how do you tell where one character ends and the next begins?), or a protocol for telling the sending device to stop and wait a minute. It's up to you to get your devices to agree on all this other stuff.

The parallel standard, on the other hand, handles all these things automatically. Consequently, it is a much easier interface to use. Easier, cheaper, and printers within a few feet of the computer meant that the parallel interface has been the dominant type in the Apple world for years,

particularly with the dot-matrix printers Centronics made famous and Japan made affordable.

Then came the Macintosh and the Apple IIc. Apple built stripped-down *serial* interfaces into these two products—apparently because parallel connectors were simply too big to fit on the machines. The Macintosh uses a nine-pin serial connector and the Apple IIc uses a five-pin connector. (Apple IIc cables usually have a special five-pin plug on the computer end and a standard 25-pin RS-232 plug on the other—with 20 pins left dangling in mid-air).

Since releasing these two computers, Apple has switched all of its peripheral products to the serial interface. Since you *can't* use a parallel device with an Apple IIc (except by adding a serial-to-parallel black box between the computer and the device), the serial interface is now being used much more widely in the Apple community.

The nitty-gritty of the RS-232 serial interface

If you don't have a serial interface, or if your printer seems to work just fine except for special features, I encourage you to skip this section. It explains what to consider when you are hooking up a serial device for the first time. Once you have a serial connection that works this information is just needless clutter for your mind.

No serial device will work correctly until you figure out whether the connectors you'll be plugging your cable into are modem-type connectors or data terminal-type (printer) connectors; until a common data transmission speed or *baud rate* is selected; until a common *data format* and *parity* are selected; and until a common *handshaking protocol* is selected.

The cable. You can usually assume a serial printer has a printer-type RS-232 connector. You can assume nearly nothing about the serial connector on your computer. The serial interface card Apple itself sells, which is known as the Super Serial Card (now you know what those cryptic references in Apple manuals to the "SSC" mean), has a thing called a jumper block on it so you can (manually) make its connector look like either type. When the little triangle on the jumper block points to the word "modem", the card's connector is configured like a printer's; when the triangle points to "terminal", the card's connector is configured like a modem's. This makes sense if you consider it backwards.

If you have an Apple IIc, of course, your connectors aren't exactly RS-232 anyhow. At least they are both wired the same way. If you are attaching a device with a printer-type connector, simply use what your dealer calls an "Apple IIc printer cable". If, on the other hand, your device has a modem-type connector, get an "Apple IIc modem cable". Note that the two cables are not interchangable, though the differences are all inside where you can't possibly see them.

If you have an Apple with slots, one good alternative is to buy a serial interface card specifically designed to work with printers. The correct cable is usually in the box with such cards and your problems are solved.

If you have a Super Serial Card, either put on your thinking cap or just start trying all the possible combinations of standard cables, null-modem cables, and jumper block positions. Eventually you will hit the right one. Unfortunately, however, unless you have correctly guessed the right baud rate, data format, and handshaking protocol, your printer still won't work and you may not even know you've gotten lucky.

The baud rate. The term *baud rate* refers to how fast the bits are coming down the line. Both the Super Serial Card and the Apple IIc can handle 15 different baud rates varying from a very slow 50 bits per second to a very fast 19,200 bits per second. Most printers also support several different baud rates. If you choose a very slow rate, your printer may not print as fast as possible because of the slow character flow. If you use a very high baud rate, on the other hand, some sort of handshaking protocol will be required to keep the computer from sending characters faster than the printer can print them.

The Apple IIc's printer port is automatically set to 9,600 baud when the computer is turned on. This is a nice fast rate and is a good one to set your equipment to if you have several choices and can't decide which to use. If you have a printer or other device that doesn't support 9,600 baud, then you have to tell the IIc to use another speed. You do this by sending special commands to the IIc's built-in "interface card". We'll look at this in more detail later.

Likewise, you can send commands to the Super Serial Card. But you can also communicate with this card and with printers that support several baud rates through an infernal contraption known as a dip switch. These little boxes of 4 or 8 switches are built into the most inaccessible area of the device you want to communicate with. (You have to "dip" your fingers into the electrical soup to use them, hence the name.) They are specially designed so that each individual switch is too small to see, yet flipping or sliding it requires the horsepower of a tow truck.

The Super Serial Card manual recommends using the tip of a ballpoint pen to flip dip switches; the Imagewriter manual absolutely forbids this ("they can leave foreign matter behind, which eventually gets into the switch") and recommends a toothpick instead. Take your pick.

Up, down, right, and left have little meaning when dealing with dip switches. Instead, a secret code is used to keep the enemy confused. Since we're all allies here I can give you the key: ON, 1, and CLOSED all mean the same thing. Likewise, OFF, 0, and OPEN have the same meaning. Usually the switches themselves are marked OFF-ON or 0-1 while the manuals talk about OPEN and CLOSE. Other combinations are also used to keep the enemy thoroughly confused. Some say the system works too well, but we have our priorities.

This next part is going to be the hardest for some of you to understand, so prepare yourselves. In order to figure out how to set the dip switches on your printer and on your interface card so that they agree on a certain baud rate, *you will have to refer to the manuals that came with your equipment.* Look in the index under "switches, setting" or "dip switch settings" or "setting dip switches". Chances are you won't find anything, but it's worth a try. Many manuals include a reference card that shows the possible settings.

In addition to baud rate, dip switches on printers also control other aspects of a printer's operation. More about this later.

The data format. Next you have to figure out which *data format* you're going to use. Imagine yourself sitting at the end of a serial connection watching the bits drop off the line. If your job was to convert these bits into characters, how would you tell where one character stopped and the next began? Unless you know the data format, friends, you can't.

When characters are sent serially, bits that are always ON (or 1) are inserted between each character. These bits are sometimes called *stop bits.* The rest of the time they are called *start bits.* Apple's Imagewriter manual says its data format includes one start bit and one stop bit. Since the "stop" bit of one character and the "start" bit of the next are always adjacent to each other, this is the same thing as two start bits and no stop bits or no start bits and two stop bits. I sure hope the enemy is as confused about this one as we are.

In between the start/stop bit or bits you will always find some data bits. There can be as few as five or as many as eight of these. You may also find something called a *parity* bit. The parity bit is an optional, extra bit, and no two descriptions I've read of how it works have been the same. Fortunately, most devices don't use it.

In Appendix F, "Interface Specifications", the Imagewriter manual says in small print that it expects a data format of 1 start bit, 7 or 8 data bits, no parity bits, and 1 stop bit. In Chapter 5, in a section called "Data Byte Length", the manual more specifically says that the printer always expects 8 data bits, but a dip switch controls whether the eighth one is recognized or ignored. These are the *only* two places in the Imagewriter's manual where the data format is referred to, and only the incomplete latter section is indexed.

Printers are notorious for poor documentation. Even though the Imagewriter manual carefully hides this rather essential data-format information, it is by far the best printer manual I've ever seen. (In the October issue I said (page 78) that Apple's printer documentation doesn't show you how to download character sets; in fact, the Imagewriter manual has a very nice section on this. Apple's earlier Dot Matrix Printer manual, which is an excellent example of pap, didn't even mention the feature.) My point is that if the best manual around hides the data format in an appendix and even then gives a sloppy, inexact description of it, you can understand why people have problems with printers.

The default data format of the Apple IIc's printer port is listed in that computer's reference manual as 8 data bits, no parity bits, and 2 stop bits. Though the description is slightly different, this is electonically the same as what the Imagewriter expects. You will often see such a data format written as "8N2."

The handshake. Now we're all done except for the handshaking. Early printers didn't handshake. You had to send characters to them at slow baud rates so that they could keep up. Usually the sending device had to build in a delay after a carriage return to give the printer time to move back to the left edge of the paper.

Nowadays, most printers have a small or large amount of RAM memory to store the last bunch of characters transmitted. This allows the printer to operate as fast as possible. At high baud rates, however, the RAM will soon fill up and characters will spill out on the floor unless the printer has some

protocol for telling the computer to STOP A MINUTE. The primary symptoms of data overflow are characters missing from your manuscript and other garbage.

There are two primary protocols used for handshaking in the Apple world. One, the *data terminal ready* protocol, is a hardware handshake. The other, the *XON-XOFF* protocol, is a software handshake.

Under the data terminal ready protocol, the printer's "I'm OK" (or *select*) line in the RS-232 interface is monitored. When the printer stops saying it's OK, the computer stops sending characters. Both the Apple IIc and the Super Serial Card support this protocol.

Under the XON-XOFF protocol, which is also known as the DC3-DC1 protocol (DC means "device control" in the arcane world of ASCII control codes), the printer sends a special XOFF character (control-S) when it wants the computer to stop sending, and an XON character (control-Q) when the computer can resume sending. If you've used a modem much, you've probably already discovered that you can usually type these two characters to keep incoming messages from scrolling off your screen before you read them. Showing no favoritism, Apple uses this protocol on the Macintosh. The Super Serial Card *can* use it, but has to be commanded to. The Apple IIc's built-in interface doesn't support it.

The Imagewriter supports both protocols; you tell it which to use by setting a dip switch. If your printer is set up for one protocol and the computer for another, one of two things will happen: nothing (the printer won't work at all), or the printer will work fine for awhile and then start skipping characters or printing garbage.

Apple's Laserwriter printer supports only the XON-XOFF protocol (and AppleTalk). Since the Apple IIc supports only the data terminal ready protocol, you shouldn't try to hook the two together unless you are prepared to try some special tricks such as using a slower-than-normal baud rate, longer-than-normal carriage return delays, or a special printer driver.

Printer peculiarities and problems

Now that we have our computer and printer connected, let's see how a printer works. The ASCII character set officially consists of 128 numerical codes. Of these, 33 are unprintable control codes and 95 are printable characters. The printable characters include 26 capital letters, 26 small letters, 10 digits, 32 symbols and punctuation marks (some of these are used for accented characters in international character sets), and the space.

All 128 of these codes can be represented with seven bits. Printers with a data format of eight bits (this would include all parallel printers) use the eighth bit in a variety of ways. We will discuss the problems this causes in a moment.

Each of the 33 ASCII control codes is supposed to have a standard use. For example, control-M is universally recognized as a carriage return. Some other commonly used control characters are control-G, bell (on printers that *have* bells or buzzers); control-H, backspace; control-I, horizontal tab; control-J, line feed; control-K, vertical tab; and control-L, form feed. Not all printers recognize all of these codes, however, and some control codes are used in unique ways by some printers.

A line feed tells the printer to advance the paper one line. A form feed tells it to advance to the beginning of the next sheet.

Teletype machines, the first kind of printer used with computers, expected both a carriage return and a line feed at the end of each line. If only a carriage return was received, the paper did not advance and the next line was printed over the first. If only a line feed was received, the paper advanced but printing continued at the same right-left position on the line.

Typewriters, on the other hand, advance the paper *and* return to the left margin in response to a carriage return. The net result of all this is that some printers advance the paper when they see a carriage return code, some don't, and some, like the Imagewriter, have a dip switch that lets *you* decide what will happen.

In addition to control codes, most printers also react to certain "escape sequences." These are commands that begin with an escape character (control-[, decimal 27, hex $1B). For example, the two-character sequence ESC E tells an Imagewriter to switch to Elite (12-characters-per-inch) type.

Modern printers have an incredible number of features, all of which are turned on and off with either an escape sequence or one of the 33 ASCII control codes. *The only way to find out what features your printer has and how to turn them on and off is by reading the machine's manual.* There are no standard escape sequences.

What kind of features can you expect to find? The Imagewriter has six different character widths available. Each of these can be printed double-wide, which makes a total of 12 different widths that range from a big 4.5 characters-per-inch to a tiny 17 characters-per-inch. The Imagewriter also has two proportional character sets (the letters are not all the same width) and the ability to accept a custom set, which can also be proportional.

The Imagewriter includes commands for setting a left margin, for setting and clearing horizontal and vertical tabs, and for selecting how much paper is advanced by a line feed (in 1/144 of an inch increments). It will print in bold, it will underline, it allows you to reverse the direction of line feeds (this means it can roll the paper backwards), and it will print graphics. It has commands you can use to override the dip switch settings. You can choose which of seven national character sets it will use. You can choose to put a slash through the zero or leave it out.

The newest version of the Imagewriter will print in three different quality/speed combinations and can do graphics in color.

The most important parts of a printer's manual are the tables that show the meaning of the various dip switch settings, control codes, and escape sequences. I recommend you flip through your manual until you find these tables, photocopy them, and hang them on the wall next to your Beagle Bros *Peeks, Pokes and Pointers* chart.

Commercial programmers often need to write software that will work with several different types of printers. The chaos of printer commands makes this very difficult. Of significant help is a new book called the *Programmers' Handbook of Computer Printer Commands.* It's published by Cardinal Point (P.O. Box 596, Ellettsville, Ind. 47429 812-876-7811 $39.95) and includes command tables for all the printers commonly used with Apple IIs (printers from Apple, Brother, C. Itoh, Diablo, Epson, Juki, NEC, Okidata, Panasonic, Qume, Silver-Reed, Star Micronics, and 31 other companies). This book is not a suitable replacement for the printer manual your dog ate, but it is an excellent reference for programmers who need to know the command codes of a variety of printers.

Interface card peculiarities and problems

Now let's follow the printer cable back to the interface card to see how this strange creature operates. The typical Apple II interface card not only tries to be a hardware device that passes electronic signals between your computer and your printer, but also a more-or-less intellgent software device with the ability to format the data for you as it passes by. It is this "intelligence" that causes the most intractable printer problems.

Probably the most successful printer interface card for the Apple II has been the Grappler-Plus, which was designed by Milo Street (now of Street Electronics) and is produced by Orange Micro (3150 East La Palma, Suite G, Anaheim, Calif. 92806). The big seller is a parallel interface card; Orange Micro has also recently released a serial version.

The Grappler became famous because of a single strength—it could capture an image from the Apple II's high-resolution graphic screen and print it on a wide variety of dot matrix printers. Nowadays, many cards can do this. Some, such as the Fingerprint and Print-It! cards mentioned here in July and September (pages 53-54 and 69), have the additional feature of being able to print at any time—even from inside a copy-protected program.

To print graphics with a Grappler, you must first tell the card what kind of printer you have by setting some dip switches. Then you turn the interface on with a PR#1 or whatever, send a control code that tells the interface card that what follows is a command, and send a series of single letter commands.

The Grappler includes commands for printing a mixed or full-screen graphic from graphics page one or two, in standard or double size, in inverse or normal, horizontally or vertically. The control code that awakens the Grappler and all the other printer interfaces that I know about is control-I. Sending the character string "control-I GDIR2" to the Grappler tells it to print a G(raphic) D(ouble-size), in I(nverse), R(otated) from (page) 2 of your computer.

In addition to its graphics commands, the Grappler has commands for setting a left margin, a line length, for appending line feeds onto carriage returns, for dumping the *text* screen onto the printer, and for clearing the eighth data bit to zero. Like the graphics commands, these commands must begin with a control-I. The character string "control-I 12L" tells the Grappler to set a left margin of 12. After this command is executed the Grappler will send 12 space-characters after every carriage return, thus creating a left margin of 12.

There are several major areas in which printer interface cards vary. Some of these differences aren't discussed in manuals and are nearly impossible to discover without testing and experimenting with each card individually. For example, some control-I commands on some cards require a return at

ASCII Control Code Rosetta Stone

type this	numerical values: low-value CHR$	low-value hex	high-value dec	high-value hex	abbreviation	usage
control-@	0	$00	128	$80	NUL	null
control-A	1	$01	129	$81	SOH	start of heading
control-B	2	$02	130	$82	STX	start of text
control-C	3	$03	131	$83	ETX	end of text
control-D	4	$04	132	$84	EOT	end of transmission
control-E	5	$05	133	$85	ENQ	enquiry
control-F	6	$06	134	$86	ACK	acknowledge
control-G	7	$07	135	$87	BEL	bell
control-H	8	$08	136	$88	BS	backspace
control-I	9	$09	137	$89	HT	tab (horizontal)
control-J	10	$0A	138	$8A	LF	linefeed
control-K	11	$0B	139	$8B	VT	vertical tab
control-L	12	$0C	140	$8C	FF	form feed
control-M	13	$0D	141	$8D	CR	carriage return
control-N	14	$0E	142	$8E	SO	shift out
control-O	15	$0F	143	$8F	SI	shift in
control-P	16	$10	144	$90	DLE	data link escape
control-Q	17	$11	145	$91	DC1	device control-1
control-R	18	$12	146	$92	DC2	device control-2
control-S	19	$13	147	$93	DC3	device control-3
control-T	20	$14	148	$94	DC4	device control-4
control-U	21	$15	149	$95	NAK	negative acknowledge
control-V	22	$16	150	$96	SYN	synchronous idle
control-W	23	$17	151	$97	ETB	end of transmission block
control-X	24	$18	152	$98	CAN	cancel
control-Y	25	$19	153	$99	EM	end of medium
control-Z	26	$1A	154	$9A	SUB	substitute
control-[	27	$1B	155	$9B	ESC	escape
control-\	28	$1C	156	$9C	FS	file separator
control-]	29	$1D	157	$9D	GS	group separator
control-^	30	$1E	158	$9E	RS	record separator
control-_	31	$1F	159	$9F	US	unit separator
delete	127	$7F	255	$FF	DEL	delete

the end of them. Other commands, sometimes on the same card, don't, and can be strung together without a return. Commands that *require* a return usually eat it rather than pass it though to the printer. Commands that don't require returns, on the other hand, neither look for nor eat them.

The consequence of all this is that programs requiring line-by-line paper control must often be written and tested for a specific interface card. Changing cards or attempting to run the program on a computer with another card often has surprising and disappointing results. If a program uses carriage returns, changing cards can cause too much paper to be advanced. If a program doesn't use carriage returns, changing cards can cause commands to go unrecognized or totally unexpected commands to be executed.

All cards have the ability to send your characters to the screen as well as the printer. Some of them start with this feature off, however, and some with this feature on. Apple's Super Serial Card has dip switches that control how it starts up. Applesoft programmers prefer to have "video" on. If you turn on a printer from the keyboard with a PR#1 and video is off, your computer will *appear* to die. With most printers, further keystrokes won't show up on either your screen or your printer—until you press return, when the printer will spring to life. With video on, on the other hand, the characters you type will reassuringly appear on your screen.

Other than Applesoft programmers, however, no one wants video on. When a program such as AppleWorks or *Apple Writer* prints, this "feature" causes any instructions on the screen to be overwritten with mish-mash. And the open-apple-H screen-dump command in AppleWorks doesn't work well when an interface card is putting characters on the screen as fast as AppleWorks can dump them.

With Apple's own interface cards, the command for turning video off is "control-I 80N." (Again, Apple shows no partiality—the Apple IIc comes up with video off, the Super Serial Card comes up with what its dip switches say, and the Apple Parallel Card comes up on with video on. All third-party interface cards mimic one or the other of these standards.) Programmers who aren't aware that Apple's share of the interface card market is relatively small, including the wizards at Apple itself, hard wire the "control-I 80N" command into their programs. Consequently, those of us with non-Apple interface cards often have printing difficulties with such software.

On the Grappler, "control-I 80N" will, indeed, turn off video (some of the time—more on when it won't in a moment). However, this command *also* tells the Grappler and most other cards to send a carriage return at least every 80 characters. The Grappler will begin to search for carriage returns in the material being printed and if 80 characters go by without one, it will add one by its own authority. When printing on 8-inch printers at 10 characters per inch, this doesn't cause problems, because your software will provide enough carriage returns to prevent the Grappler from interfering. If you switch to narrow characters, however, so you can print a spreadsheet at 136 characters across, you will suddenly have strange problems. The lines will "wrap" at the 80th column and neither your printer nor your software manufacturer will know why.

Nowadays, when almost all software can be depended upon to send carriage returns when necessary, interface cards should mind their own business. You tell the Grappler to mind *its* own business with the command "control-I 0N" (that's zero-N, not ON). This tells the Grappler to turn off video and to forget about sending carriage returns. Unfortunately, other cards have other commands for this. "Control-I 0N" means nothing to a Super Serial Card or to an Apple IIc.

Unlike most other cards, the Super Serial Card doesn't send carriage returns unless you actually tell it to with a "control-I C" command. The Apple IIc, on the other hand, always sends carriage returns unless you tell it to ignore the data stream completely with a "control-I Z" command. (You can also poke a zero into byte 1400+slot, which is 1401 on a IIc, to stop carriage returns. This trick also works on a Super Serial Card and on a Grappler. The IIc manual says Apple guarantees this poke will always work with its equipment. I don't know how many third-party interface cards support it, however.)

Another interesting feature of the Grappler is that it expects incoming data to be in the high-value ASCII format. This means that it expects the eighth bit of each character to be set to one. This is how Applesoft itself always sends data. The Grappler has a dip switch you can set to force this eighth bit to zero when it gets sent on to a printer or to allow bit-clearing to be done under program control. (Under program control, the eighth bit is cleared unless you command the Grappler to let it pass through unmolested.)

Unfortunately, Apple's own parallel card doesn't have any eighth-bit controls. It always passes the eighth bit on just as it gets it. This means that printers that use the eighth bit for special purposes go into their special purposes when you use Apple's parallel card and Applesoft in combination. Epson owners, for example, see special graphics characters where they expect to see the grocery list or whatever.

To get around this problem, lots of commercial software bends the Applesoft standard and transmits characters to be printed with the eighth bit already cleared to zero. However, *control-Is sent to the Grappler with the eighth bit cleared pass through unrecognized.* This is why many of you find "80N" at the top of reports and documents printed by programs such as AppleWorks. Not only do you get that 80N junk at the top of all your printouts, the Grappler doesn't see the command to turn video off, so your screen turns to trash while you are printing.

There is usually no easy solution for this kind of problem. (In AppleWorks, however, you can now set the interface card command string to "control-I 0N" and the Grappler will recognize it.) It would be nice if software allowed you to specify whether the eighth bit should be set or clear when printing, but, on the other hand, having to specify such stuff can be pretty scary for new users. The only program I know of that allows this is *ASCII Express*, a communications program with so many options it sometimes causes paralysis-of-the-mind even in certified hackers.

However, it's clear that commercial-grade Apple II software must at least allow the user to specify the codes that should be sent to the interface card—a lesson Apple finally took to heart in version 1.2 of AppleWorks (free updates are available from your dealer if you have an older version). Interface cards, on the other hand, should recognize ASCII control characters whether the eighth bit is on or off, rather than suffering the 80N handicap of the Grappler.

The commands interface cards have for formatting output are much more trouble than they're worth. Today's printers and most software already handle line length, page length, and margins just fine. Enabling the card to do this stuff too is triple redundancy. (Note that if you tell your printer you want a left margin of 10, your interface card you want a left margin of 10, and your software you want a left margin of 10, you will get a left margin of 30.)

The problem is that these output-formatting features can add all kinds of extra characters to what you are sending to your printer. If you are

transmitting a stream of characters for creating graphics, for example, most printers want one byte for each eight dot positions. A complete graphic consists of thousands of bytes of data without any carriage returns at all. But some interface cards are quite content to lace your graphic with carriage returns every 80 bytes. If you are trying to print a graphic and a blip repeats itself at regular intervals throughout your picture, this is what's happening to you.

In addition to the problem of *added* characters, *missing* characters can drive you nuts quickly, too. Since control-I is used as the interface card command code, you can never send it directly to a printer. This is a pain in the dot matrix, because most printers use control-I as the control code for tabbing.

The recommended way to solve this problem is to change the card's command code. With all the cards I know about you do this by sending a control-I followed by any *control character.* (Note that all other interface card commands consist of control-I followed by a *standard, usually capital, letter.*) I use printer tabs a lot around here; "control-I control-Z" changes the Grappler command code to control-Z, which is a control code none of my equipment needs for anything else.

However, even this technique has problems. Some cards, like the Grappler, change the command code back to control-I automatically the next time the card is started with a PR# command. Other cards, like Apple's Super Serial Card, leave it where you set it. The only sure way to avoid problems when using a program with several different cards is to change the command code back to control-I before turning the printer off.

Another problem with kidnapped characters occurs when you are printing graphics. With the Grappler and many other cards, the command code has to be set to *something,* and if this code accidentally appears in your graphic character stream the card is going to eat it (and probably one or more of the following characters as well).

Apple's IIc and Super Serial Card get around both the added character and missing character problems with a "control-I Z" (zap) command. This tells the card to stop formatting output or otherwise messing with the character stream completely. After a zap command you can't send *any* command to the interface card, however, without starting over with a PR# command. The Grappler doesn't have a zap command or any other way to prevent character kidnapping that I know of, short of writing a special assembly language "driver." Of course, since the Grappler's purpose in life is to print graphics for you, perhaps this isn't a problem.

To live happily with YOUR printer interface card, you have to dive into the manual and determine the default characteristics it has when a PR# command is issued. If you have choices, via dip switches or otherwise, you probably want the card to come up with all special features, including video and carriage return insertion, turned off. If it insists on turning something on, you have to figure out the command that will turn that feature off, tape it to your forehead, and use it as a part of your "printer set-up string" in all software that asks for such a thing (and all good software should).

In addition to commands for formatting output, serial cards have commands for setting baud rates, data formats, parity, handshake protocol, and all that fun stuff. However, programmers should note that every time a PR# command is used to turn on an interface card, the card *may* revert to its default characteristics—some do, some don't. With the Apple IIc, (but not with the Super Serial Card) *this includes the default baud rate, data format, etc.* Be prepared to reset this stuff every time you use the PR# command. (***Open-Apple*** got this detail wrong in the September issue. Turn to page 73 and cross out the words "or with control-A commands" in the third italicized paragraph of column two.)

With a Super Serial Card, of course, you can also use dip switches to configure the card so its defaults match your equipment. Apple IIc owners have probably noticed, however, that their machines don't have any dip switches. On the IIc, a PR#1 command always causes reversion to the default 8N2 data format at 9600 baud, video off, line feeds added to carriage returns, and carriage returns inserted every 80 characters.

Apple IIc owners have also probably noticed that the manual that comes with that computer gives absolutely no information about how to reconfigure the machine's serial ports from inside a program. (The information is in the *Apple IIc Reference Manual,* a new version of which was to be available soon from Addison-Wesley Publishing Co at 800-238-3801, according to our April issue, page 26. I called that number in mid-October and was told the book won't be ready until at least *late* November, emphasis Addison-Wesley's.)

You *can* change the IIc's serial port defaults, however, by using the System Utilities disk that comes with the IIc. After changing the defaults, the PR#1 command will use the new defaults until you turn the computer off. Since it is painful to have to use the System Utilites disk to set these defaults every time you turn the computer on, here's a trick you can use. The System Utilities disk stores the default PR# formats for the two IIc serial ports in auxiliary memory screen holes. The IIc firmware retrieves them from there whenever a port is turned on. Bytes 1144 through 1147 hold the defaults for port 1, bytes 1148 through 1151 the defaults for port 2.

Use the System Utilities disk to get these defaults set up the way you want them. Then use the following program to retrieve what the System Utilities disk has stored in auxiliary memory:

```
100 ADR=1144 : REM use 1148 for port 2

110 IF PEEK(49176)>127 THEN S80=1 : REM remember status of 80STORE switch
120 POKE 49153,0 : REM turn on 80STORE
130 POKE 49237,0 : REM turn on auxiliary memory

140 FOR I=0 TO 3 : V(I) = PEEK(ADR+I) : NEXT
```

Ask (or tell) Uncle DOS

Proportional printing

My Juki 6100 printer has the capability to do proportional printing. Does anyone know how to do this using Apple Writer IIe?

Paul Werner
New Orleans, La.

The Juki doesn't do "proportional printing" in the sense that the Imagewriter does—it doesn't have a typeface in which the characters are all different widths. Such typefaces have a much different appearance than faces in which all the characters are the same width—what most printers and typewriters, including the Juki, have.

Either a true proportional face or a standard face can be used with either a ragged right margin or with "full" or "fill" justification, in which both the left and right margins are straight. Full justification requires that the spaces between words (and in some cases, between letters) vary in size. ***Apple Writer*** *can do full justification but it can only put whole space-characters between words. Some of the words on a line will have one space between them, some two, some even three or more. This kind of justification always looks toothy to my eye; I much prefer a ragged right margin, but this is a matter of taste.*

What the Juki does have is a variable-width space character. Using it you can avoid the toothy look by making all the spaces on a line the same size. As it comes from the factory, ***Apple Writer*** *doesn't support this feature, however. If anyone has figured out how to do it, it would be Don Lancaster, whose address and phone number are embedded elsewhere in this issue.*

Hand feeding linefeeds

I am using an Apple IIe with a Grappler-Plus, a Printer Optimizer, and Qume Sprint 5 printer. When I print a document from AppleWorks my printer seems to freeze at the end of a line and I can't bring it up unless I turn it off and on again. Then it will print a 2 or a 12 at the left margin and continues only to freeze again.

AppleWorks also keeps asking me to insert the program disk in drive 1 but looks for it only in drive 2. Why is that?

I also use *ASCII Express* to access medical data bases and print them with an Apple Scribe printer. The data bases don't send a line feed after carriage returns, but the word processor I use with this printer does, so I am constantly changing the dip switches in the printer.

I would like to send a control code from either program to change the printer. However, the control code for this function on the Scribe includes both high-value ASCII and low-value ASCII nulls (control-@). The Scribe manual says the needed control code can't be sent from Applesoft. How the hex am I supposed to enter these codes then?

Roberto Restrepo
Willow Park, Texas

Your problem with the Qume and AppleWorks is beyond me. I don't even know what a Printer Optimizer is. But the reason AppleWorks keeps looking for the program disk in drive 2 is that you have named your data disk "/APPLEWORKS"—the same name the program disk has. Use the Basic.system RENAME

```
150 POKE 49236,0 : REM revert to main memory
160 IF S80=0 THEN POKE 49152,0 : REM fix STORE80 if necessary

170 FOR I=0 TO 3 : PRINT "BYTE ";ADR+I;" HOLDS ";V(I) : NEXT
180 END
```

Once you know what these bytes hold when you have the default characteristics set the way you want them, all you have to do from inside a program is poke those values back into auxiliary memory before turning on the port. You do that like this:

```
110 IF PEEK(49176)>127 THEN S80=1 : REM remember status of 80STORE switch
120 POKE 49153,0 : REM turn on 80STORE
130 POKE 49237,0 : REM turn on auxiliary memory

140 POKE 1144,?? : REM replace ?? with the values you found
141 POKE 1145,?? : REM   using the previous program.
142 POKE 1146,?? : REM   use 1148-1151 for port 2.
143 POKE 1147,??

150 POKE 49236,0 : REM revert to main memory
160 IF S80=0 THEN POKE 49152,0 : REM fix STORE80 if necessary
```

For more information on how this trick works, see Jack Higbie's "Any Port in a Storm" in the July 1985 *inCider*, pages 28-36.

Software peculiarities and problems

Assume for the moment that your interface card and printer have excellent instruction manuals and that you have read and understood them. *It's all been a waste of time* unless the software you use can send the required control codes and escape sequences down the line.

Applesoft sends all characters with the eighth bit set to one. Apple Logo sends all characters with the eighth bit cleared to zero (now you Logo users may understand why you can't get the Grappler to print your pictures). Neither *Apple Writer* nor AppleWorks provide any way to control the eighth bit of a character. If your printer uses the eighth bit for something, you'll probably find that feature is inaccessible without special software.

From inside Applesoft, you print control characters using the CHR$ function. PRINT CHR$(27) transmits an escape (control-[, decimal 27, hex $1B). (Since Applesoft sends characters in the high-value ASCII format, however, that should really be decimal 155, hex $9B.) Great confusion arises because the ASCII control codes have so many different names (seven per code). Somewhere in this newsletter is an ASCII control-code Rosetta stone that may help you decipher the hieroglyphics found in most printer manuals.

I find the best way to handle a printer from inside an Applesoft program is to set up two special subroutines, one for turning the printer on and one for turning it off. The following routines are the ones I use on a system that has a Grappler and an Apple Dot Matrix Printer. The exact codes will be of little use to most of you, but it should give you some good ideas about how to set up your own routines. These routines were designed to turn the printer on and off without advancing any paper.

```
8000 REM send print msg to screen; turn on printer
8001 REM    P1$=character size command
8002 REM    P2$=left margin increment
8003 REM    P3$=bold/normal

8010 HOME : VTAB 13
8020 PRINT "Press <ESC> to cancel printing."
8030 PRINT "If paper jams turn off printer."
8040 PRINT CHR$(4);"PR#1"     : REM this command advances the paper one line

8050 PRINT CHR$(9);"0N";      : REM turn off screen, suppress printer's <cr>
8051 PRINT CHR$(9);CHR$(26);  : REM change card's ^I to ^Z for tabbing

8060 PRINT CHR$(27);P1$;      : REM character size
8061 PRINT CHR$(27);"L";P2$;  : REM printer's left margin command
8062 PRINT CHR$(27);P3$;      : REM bold/normal
8063 PRINT CHR$(27);"A";      : REM 6 lines per inch
8064 PRINT CHR$(27);"Y";      : REM underline off
8065 PRINT CHR$(15);          : REM elongated characters off

8080 PRINT CHR$(27);"r";      : REM reverse line feeds
8081 PRINT                    : REM roll paper back to beginning
8089 PRINT CHR$(27);"f";      : REM forward line feeds
8090 RETURN

-----------------------------------------------------------------


8900 REM turn off printer; fix controller command code
8910 PRINT CHR$(26);CHR$(9);  : REM change ^Z back to ^I

8920 PRINT CHR$(27);"r";      : REM reverse line feeds
8921 PRINT                    : REM roll paper back one line
8922 PRINT CHR$(27);"f";      : REM forward line feeds
8923 PRINT                    : REM must have <cr> before DOS 3.3 commands

8930 PRINT CHR$(4);"PR#0"     : REM modify this line for 80-column screens
8940 RETURN
```

From inside *Apple Writer*, you print control codes by embedding them in the text of your manuscript. First press control-V to enable embedding, then press the control codes you want, then press control-V again to get back to normal. Another way to do this is to put a special character, say ^, where you want a control code, then use *Apple Writer's* search and replace function to

---

command to change it to something else and the problem will go away.

The engineer that decided to make the automatic line feed function of the Scribe depend on an escape sequence like this one should be put to work on a wind-up train. For the record, the code for getting the Scribe to add line feeds onto carriage returns is ESC D High-value-control-@ Low-value-control-@. I don't know of ***any*** *program that can send that code.*

ASCII Express *has a feature that strips linefeeds from incoming messages (see section 11.8.4 of the* ***ASCII Express*** *manual, page 126). Make sure you have this option set to no. If this doesn't help, you can theoretically have your Grappler add line feeds onto carriage returns for you. The problem is that* ***ASCII Express*** *normally bypasses the software on the Grappler that does this. Section 21.3 of the* ***ASCII Express*** *manual (pages 298-299) explains how to get it to use the Grappler's software instead. Section 11.8 of the manual (pages 123-127) explains how to send control codes to the Grappler. Make sure* ***ASCII Express*** *is sending the high bit to the printer and use "control-I ON" as the "printer message." You don't have to specifically command the Grappler to add linefeeds to carriage returns because it does that by default.*

Alternatively, set the Scribe dip switch so that ***it*** *adds linefeeds to carriage returns. The Grappler is really sending the linefeeds you attribute to your word processor. Use "control-I A" as an interface card setup string inside your word processor. This code tells the Grappler to stop sending linefeeds.*

Troubleshooting AppleWorks

Recently we purchased an Olympia NP parallel printer. It has an external FINE switch that allows for correspondence quality printing. When using AppleWorks on an Apple IIe with a Grappler interface, the FINE switch works great from the data base and spreadsheet, but not from the word processor.

The FINE switch works with other word processing packages such as *Apple Writer* and *Bank Street Writer*, but with AppleWorks it is switched off at the end of the first line. If the escape code for the FINE command is substituted for the boldface command it will work till the end of a paragraph, but what about printing an entire document?

Jeff Russell
Morgantown, W.V.

I suspect that the control code you specified for the default 10 characters per inch mistakenly turns off the FINE feature. AppleWorks sends the character-per-inch code at the beginning of each line.

The easiest way to troubleshoot problems such as this one is to make a spare copy of your AppleWorks disk, boot it, and then remove and readd your printer. When adding it again, however, specify "print onto disk" rather than to a slot.

Now load a file that is giving you trouble, print it to a disk file, and load it into ***Apple Writer.*** *You will then be able to see exactly what control codes AppleWorks is putting in your file. From that you should be able to figure out what the problem is.*

What's a Panasonic?

I have a 128K Apple IIe, parallel Grappler-Plus, and a Panasonic KX-P1090 dot matrix printer. When I use AppleWorks, I am having lines truncated, spaces inserted, and lines having characters dumped to the next line.

What code to I use for the interface card setting? Do any of the preprogrammed printer codes correspond to the KX-P1090?

Michael L Groeschen
Newport, Ky

The code for the Grappler-Plus is "control-I ON". According to the ***Programmers' Handbook of Computer Printer Commands****, the control codes for the Panasonic KX-P1090 are very similar to the ones for the Epson FX-80, which is a listed AppleWorks printer.*

Dealer incompatibility

80N

I have an enhanced Apple IIe with three Disk II drives, a Star Gemini 10x printer with parallel interface, an extended 80-column card, and Z-80 Star card.

change this to a control code. This way you don't have to use control-V (this is also a good way to embed control-V itself.)

If you are going to do much of this, the best way is to use *Apple Writer's* glossary (or macro) function. You can fix it so that your printer's boldface command appears when you press open-apple-B, for example. By using the glossary you can forget your printer's arcane codes and memorize a few mnemonic letters instead.

A problem with *Apple Writer* is that it counts your control characters when determining line length. A line with several control characters may appear shorter than other lines when it is printed, since the control characters won't take up any space on the paper.

Another problem is that *Apple Writer* has no provision for sending a set up string to your interface card. If you need to use such a string, you must include it at the beginning of each document you print.

Epson printers use the ASCII null code (control-@, decimal and hex 0) for some special features. *Apple Writer*, however, doesn't provide any way to embed this code. Don Lancaster tells me that the ProDOS version of *Apple Writer* will send a null where control-underline is embedded in a document. (Lancaster is the best known source of support for *Apple Writer*. He knows more about it than your dealer, has developed and sells several packages of material that enhance it, and offers a free voice helpline. Contact him at Synergetics, Box 809, Thatcher, Ariz. 85552 602-428-4073.) People who need actually need control-underline for something, however, are always surprised by this undocumented feature. Lancaster knows how to fix *Apple Writer* so it will send your choice of nulls or control-underlines.

The ability *Apple Writer* gives you to place control characters anywhere within a document is one of its great strengths. The text you are now reading was originally written and laced with control codes using *Apple Writer*. The file was sent by modem to a typesetter, who ran the file through his equipment to produce what you are now reading, **boldface**, *italics*, `program type`, and all.

From inside AppleWorks, you place control characters within a document by using the open-apple-O command. The word processor allows you to place codes anywhere. The spreadsheet and data base allow you to specify codes only at the beginning of a document.

In the spreadsheet and in the data base report printer, pressing open-apple-O brings up a menu that allows you to specify margins, lines- and characters-per-inch, and a special printer or interface card setup string.

Within the AppleWorks word processor, however, it is impossible to directly type in a special printer or interface card control code. You must specify which of your printer's special features you want to access rather than a code. AppleWorks itself knows which codes are used with a limited number of printers. These are the ones that appear on AppleWork's internal menu of printers, which comes up when you specify information about your own equipment.

If your printer isn't listed, AppleWorks also allows you to add a "custom printer" and specify what control codes it uses for various character widths, lines-per-inch, boldface, underlining, superscripts, and subscripts. If your printer supports other special features, however, you either have to give them up or use the special tricks discussed in October's "Okidata 92 meets AppleWorks" (page 78) and September's "Correspondence quality phones" (pages 71-72).

Entering the control codes for a custom printer requires both knowledge of the control codes the printer requires and an awareness of how a printer's features can and can't be combined. For example, features such as superscript and subscript may not be available with double-wide or elongated characters. Many other conflicts occur. Each printer is different.

If your printer supports double-wide characters, this effectively doubles the number of character widths available on your printer. Include the "double-wide on" or "double-wide off" control code as a part of each code for selecting character widths.

It may help to remember that AppleWorks automatically sends the interface card control code at the beginning of each document (you can specify this code); the "superscript off" and "subscript off" codes at the end of each line (but only when the equivalent "on" code appears in that line); and the "boldface off" and "underline off" codes at the end of each paragraph. The AppleWorks character-per-inch commands not only send control codes to your printer but also control how many characters are sent between carriage returns.

AppleWorks supports the proportional typefaces in Apple's own Imagewriter. Proportional typefaces are difficult because the software must calculate not only the number of characters on a line but also the total width of the line based on the actual width of each character. AppleWorks can also fully justify the Imagewriter's proportional typefaces, with outstanding results.

AppleWorks doesn't support proportional faces with custom printers. In fact, it can't even correctly justify proportional faces with some of the listed printers. Proportional faces are ones where each character is a different width. The problem is that there are no standards for how wide the characters should be in relation to each other. The Imagewriter, for example, allows 17 dot positions for its proportional letter "M" and 12 dot positions for "a." If another printer has even slightly different proportions, say 18 and 12, there's no way AppleWorks can make the margins even. Subscribers report this happens when the proportional fonts on Epson printers are accessed with AppleWorks.

AppleWorks also asks whether your printer supports top-of-page (form feed) commands. Almost all printers do. However, you will find that the page length function within AppleWorks will not work unless you specify "no" for "accepts top-of-page commands."

As stated at the beginning of this article, printers are trouble because the number of combinations of software, interface cards, and types of printers is massive. You are the person in the best position to solve your own printer problems. You should now have some sense of the kinds of things you have to consider to do it.

Whenever I print from AppleWorks, I always get an "80N" on the first line. I have been to several stores seeking a solution to this problem and I always get the same response, "compatibility." There must be some easy way to eliminate this problem similar to "Converting CONVERT for MouseText" in your August issue.

James Lalley
Clifton, N.J.

Indeed there is. If you've already read the front part of this issue, you know the trick is to get AppleWorks version 1.2 from your dealer and to set the interface card control code to whatever your card requires. You should command the card to turn off video and to turn off all formatting. The exact command depends on what kind of card you have.

Your letter is yet another example of the low level of support Apple users get from dealers. Apple has repeatedly notified dealers of this problem and has provided various solutions for over eighteen months. Please write Apple and ask why none of the dealers you talked to knew the answer to this question. Name names and give dates if possible. Suggest that most reputable software companies provide technical support over the phone and ask why such support isn't available for AppleWorks.

Open-Apple is a trademark of Open-Apple newsletter. Apple Computer and Open-Apple are two different, unrelated, independent companies that wish everyone in the world had an Apple II.

Oh, brother!

3B 0D 3P Z

I use a IIc with a Brother EP-44. Since both were bought in Canada, I have little back-up here. Besides the dealers here do not know much about most problems. As you can see from this letter, when I use AppleWorks with this system I get a series of unrequested characters at the top of the page. Any suggestions about what to do to get rid of this?

A. Ruprecht
Riyadh, Saudi Arabia

It's good to hear that the dealers in the Mid-East aren't any better than the dealers here. I suspect your problem is caused by a control-code AppleWriter is sending at the beginning of each document that kicks the Brother into some kind of special mode for one line. Check your interface card control code specification for a stray character that may be doing this.

Here's one more trick you can try. Choose the Apple Silentype from AppleWork's internal list of printers. That particular model supports no control codes. It might work, it might not.

The only other good cheer I can offer is to remind you that even if you lived in Cupertino, the person who is the best position to solve printer problems is YOU.