

# **NX-10**

## **USER'S MANUAL**

NOT INTENDED FOR SALE

PN 80820128

## Federal Communications Commission Radio Frequency Interference Statement

This equipment generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna
- Relocate the computer with respect to the receiver
- Move the computer away from the receiver
- Plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet, prepared by the Federal Communications Commission helpful: "How to Identify and Resolve Radio-TV Interference Problems." This booklet is available from the U.S. Government Printing Office, Washington, D.C., 20402, Stock No. 004-000-00345-4.

For compliance with Federal Noise Interference Standard, this equipment requires a shielded cable.

*This statement will be applied only for the printers marketed in U.S.A.*

### Self Declaration

Radio interferences regarding this equipment has been eliminated according to Vfg 1046/1984 announced by the DBP.

DBP has been informed about the introduction of this special equipment and has been conceded the right to examine the whole series.

It is upon the responsibility of the user to assume that his own assembled system is in accordance with the technical regulations under Vfg 1046/1984.

To observe FTZ-regulations it is necessary, to establish all connections to the printer with shielded cable.

The equipment may only be opened by qualified service representatives.

*This statement will be applied only for the printers marketed in West Germany.*

### Trademark Acknowledgement

**NX-10, grafstar:** Star Micronics Co., Ltd.

**Apple, Apple II, Apple II +, Apple II e, Applesoft:** Apple computer Inc.

**Commodore C-64:** Commodore Business Machines, Inc.

**Compaq:** Compaq Computer Corporation

**CP/M:** Digital Research

**IBM Personal Computer, IBM PC:** International Business Machines Corp.

**Kaypro:** Kaypro Corporation

**Microsoft BASIC:** Microsoft Corporation

**Osborne 1:** Osborne Computer Corporation

**TRS-80:** Radio Shack, a division of Tandy Corporation

### NOTICE

- All rights reserved. Reproduction of any part of this manual in any form whatsoever, without STAR's express permission is forbidden.
- The contents of this manual are subject to change without notice.
- All efforts have been made to ensure the accuracy of the contents of this manual at the time of going to press. However, should any errors be detected, STAR would be greatly appreciate being informed of them.
- The above notwithstanding, STAR can assume no responsibility for any errors in this manual.

©Copyright 1986 Star Micronics Co., Ltd.

# Table of Contents

<b>Chapter 1</b>	<b>Setting Up Your Printer</b>	<b>1</b>
	Where shall we put it?	
	What have we here?	
	Removing the printer cover	
	Removing the packing tube	
	Installing the ribbon cartridge	
<b>Chapter 2</b>	<b>Getting to Know Your Printer</b>	<b>7</b>
	Controls and parts of the printer	
	Parts of the printer	
	Controls and indicators	
	Extra functions	
	Other controls	
	Selecting and loading paper	
	Loading single sheets	
	Loading sprocket-feed paper	
	Adjusting the print head	
	Connecting the printer	
	Extra functions with the control panel	
	Self-tests	
	Hex dump	
	Panel mode	
	Italic mode	
	Italic and Panel mode	
	Setting print start position	
	Setting the left and right margins	

<b>Chapter 3</b>	<b>Basic Printing</b>	<b>23</b>
	Some basics of BASIC	
	A new language!	
	First steps	
	ASCII codes and the CHR\$ function	
	Control codes	
	The escape codes	
	A note on command syntax	
	Some special kinds of text	
	Near Letter Quality characters	
	Italic printing	
	Underlining	
	Superscripts and subscripts	
	Changing the print pitch	
	Expanded print	
	Condensed print	
	Proportional printing	
	Making words stand out	
	Mixing print modes	
<b>Chapter 4</b>	<b>Formatting Text</b>	<b>41</b>
	Lines and line spacing	
	Starting a new line	
	Reverse line feeds	
	Changing the line spacing	
	Moving down the page without a carriage return	
	Page control	
	Form feed	
	Reverse form feed	
	Changing the page length	
	Top and bottom margins	
	Setting left and right margins	
	Horizontal and vertical tabs	
	Horizontal tabs	
	One-time horizontal tabs	
	Vertical tabs	
	Vertical tab channels	
	Centering and aligning text	

<b>Chapter 5</b>	<b>Special Features of the Printer</b>	<b>61</b>
	Now hear this	
	Resetting the printer	
	Putting your printer to sleep	
	Printing the bottom of the sheet	
	Backspace, delete, and cancel text	
	Printing zeroes	
	Immediate-print	
	Adjusting the width of space between characters	
	Uni-directional printing	
	The seven bit dilemma	
	Block graphics characters and special symbols	
	International character sets	
	Printing characters in the control code area	
	Printing BIG characters	
	The optional sheet feeder	
	The macro control code	
	Reading a hex dump	
<b>Chapter 6</b>	<b>Creating Your Own Characters</b>	<b>81</b>
	Dot matrix printing	
	The print matrix	
	Defining your own characters	
	Rule 1: Draft download characters are eight dots high	
	Rule 2: Dots cannot overlap	
	Add up each column of dots	
	Assigning a value to your character	
	Download character definition command	
	Printing download characters	
	Defining proportional characters	
	Defining NLQ download characters	

<b>Chapter 7</b>	<b>Dot Graphics</b>	<b>101</b>
	Comparing dot graphics with download characters	
	Using the dot graphics commands	
	Specifying the number of columns of dots	
	Specifying the graphics data	
	Combining text and graphics	
	Printing a design or logo	
	Plotting with your printer	
	How the program works	
	High resolution graphics	
	Compatibility with existing software	
	More graphics programming tips	
	Redefining alternate graphics codes	
	9-pin graphics mode	
<b>Chapter 8</b>	<b>Caring for Your Printer</b>	<b>123</b>
	Cleaning the printer	
	Replacing the ribbon	
	Replacing the print head	
<b>Appendix A</b>	<b>DIP Switch Settings</b>	<b>131</b>
	Switch functions	
<b>Appendix B</b>	<b>ASCII Codes and Conversion Chart</b>	<b>135</b>
<b>Appendix C</b>	<b>Character Fonts</b>	<b>143</b>
	Roman characters	
	Standard characters (Set #1 and Set #2)	
	Special characters (Set #2 only)	
	International characters	
	Italic characters	
	Standard characters (Set #1 and Set #2)	
	Special characters (Set #2 only)	

<b>Appendix D</b>	<b>Function Codes</b>	<b>165</b>
	Commands to control print style	
	Font style controls	
	Font pitch controls	
	Special print modes	
	Controlling the vertical print position	
	Line feed and reverse line feed	
	Form feed and related commands	
	Top/bottom margins and vertical tabs	
	Controlling the horizontal print position	
	Download character commands	
	Dot graphics commands	
	Macro instruction commands	
	Other commands	
<b>Appendix E</b>	<b>Command Summary in Numeric Order</b>	<b>209</b>
<b>Appendix F</b>	<b>Technical Specifications</b>	<b>213</b>
<b>Appendix G</b>	<b>The Parallel Interface</b>	<b>217</b>
	Functions of the Connector Signals	
<b>Appendix H</b>	<b>Connecting with Computer</b>	<b>221</b>
	Connecting with IBM-PC and COMPAQ	
	BASIC programming	
	Listing programs	
	Connecting with Apple II computers	
	Applesoft BASIC	
	Listing programs	
	Connecting with TRS-80 computers	
	TRS-80 BASIC	
	Listing programs	
	Connecting with Kaypro, Osborne, and other CP/M computers	
	Using MBASIC	
	Listing programs	

---

---

# CHAPTER 1

# SETTING UP YOUR

# PRINTER

---

---

**Subjects we'll cover in Chapter 1 include—**

- **Choosing a suitable place for your printer;**
- **Unpacking your new printer;**
- **Setting it up.**

## **WHERE SHALL WE PUT IT?**

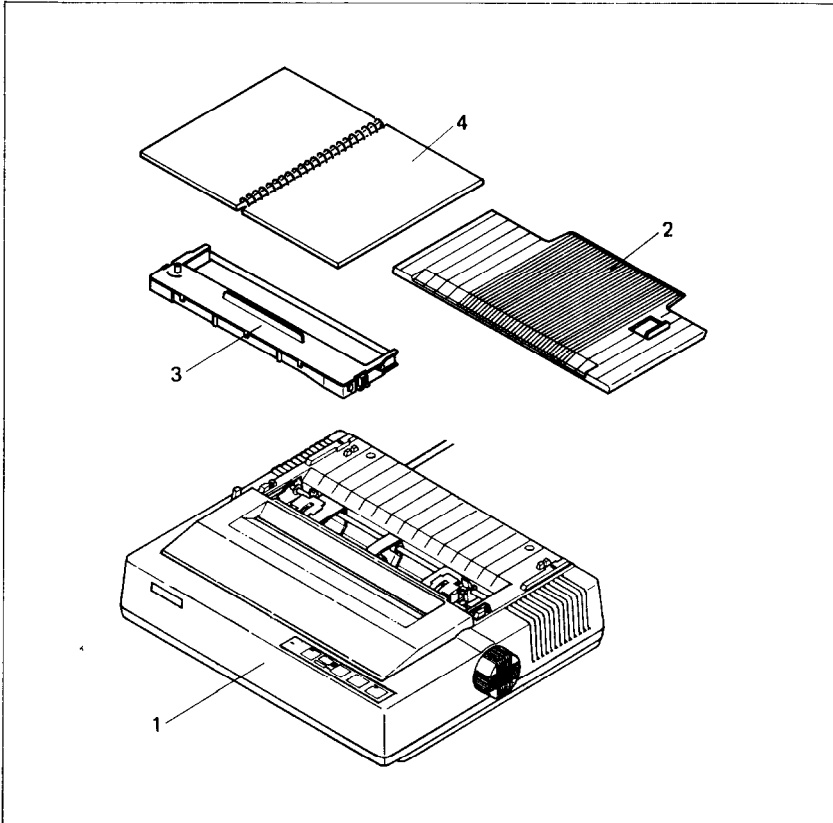
Before you do anything else, give some thought to where you'll be using your printer. Obviously, it will be somewhere near your computer. And both printer and computer will lead longer, healthier lives if they like their surroundings. For instance, we recommend...

- Using the printer on a flat surface.
- Keeping it out of direct sunlight and away from heat-producing units.
- Using it only in temperatures where you are comfortable.
- Avoiding areas with a lot of dust, grease, or humidity.
- Giving it "clean" electricity. Don't connect it to the same circuit used by large, noise-producing appliances (such as refrigerators).
- The line voltage should be the same voltage that's specified on the identification plate — within 10% of the recommended voltage.



## WHAT HAVE WE HERE?

Now let's take a look at what's in the carton. Open it up and check each item in the box against Figure 1-1. There should be four items.



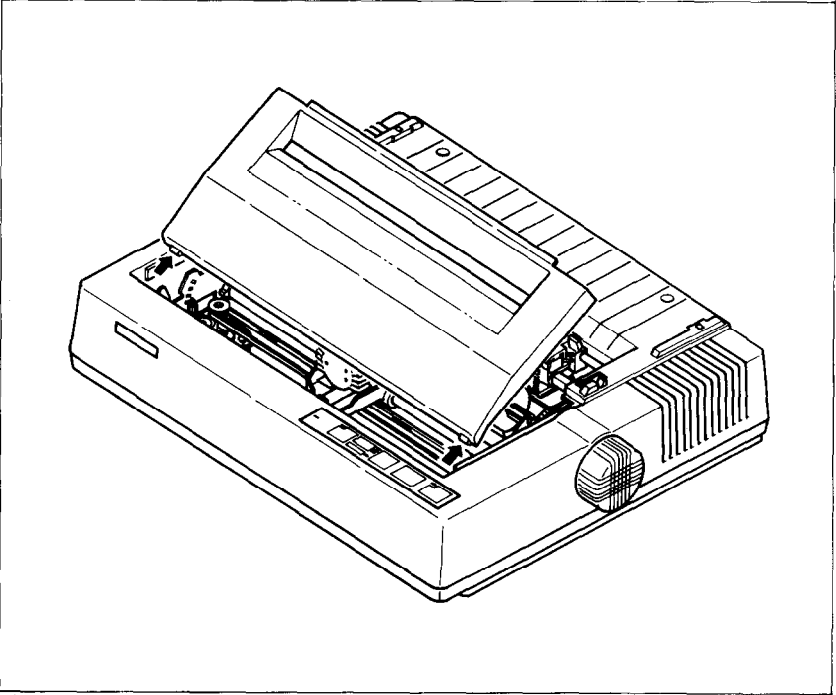
**Figure 1-1.** Check to make sure you have all four items: 1) printer, 2) paper guide, 3) ribbon cartridge, and 4) user's manual.

Let's move on the next step.

### ■ Removing the printer cover

The cover is important for two reasons — it keeps dust and dirt away from the printer's delicate "innards," and it quiets the printer's operation. Don't take off the cover except when you have to change the ribbon, or to make an adjustment.

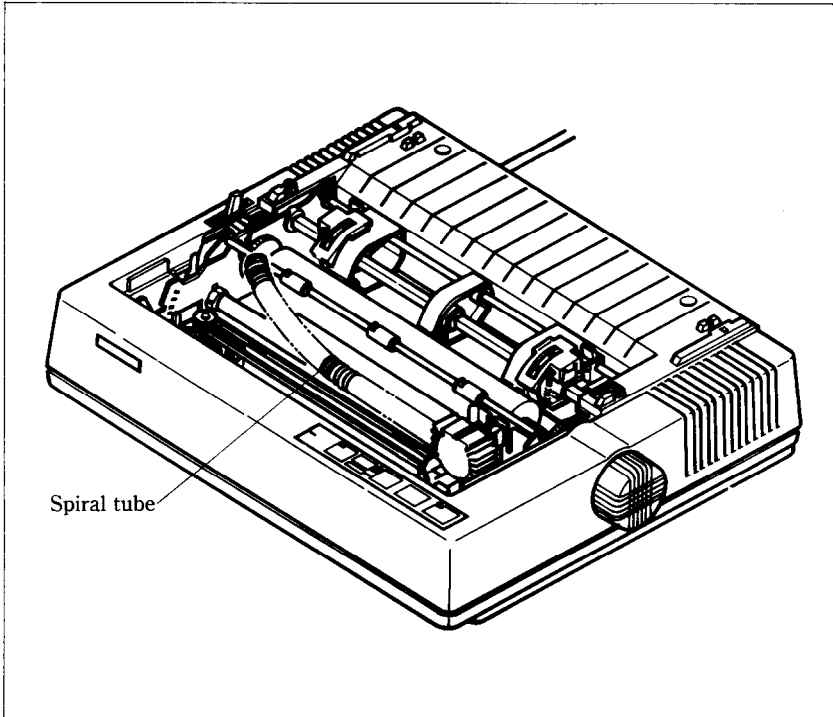
Removing the printer cover is easy. Lift up the back of the cover to disengage the two tabs at the front and then lift it off the rest of the way. To replace it, just slide the tabs in at the front and lower it into place. Figure 1-2 shows the proper position and movement for both removing and replacing the cover.



**Figure 1-2.** Remove the printer cover by lifting carefully.

■ Removing the packing tube

The printer is shipped with a protective spiral tube to keep the print head from being damaged in transit. We have to remove this tube. First, remove the printer cover. See the tube on the carriage rail (Figure 1-3). Pull it off carefully.



**Figure 1-3.** Remove the protective tube from the carriage rail.

Up to this point, we've been clearing the decks for action, so to speak. Only one more thing left to do before we can start printing — install the ribbon cartridge.

#### ■ Installing the ribbon cartridge

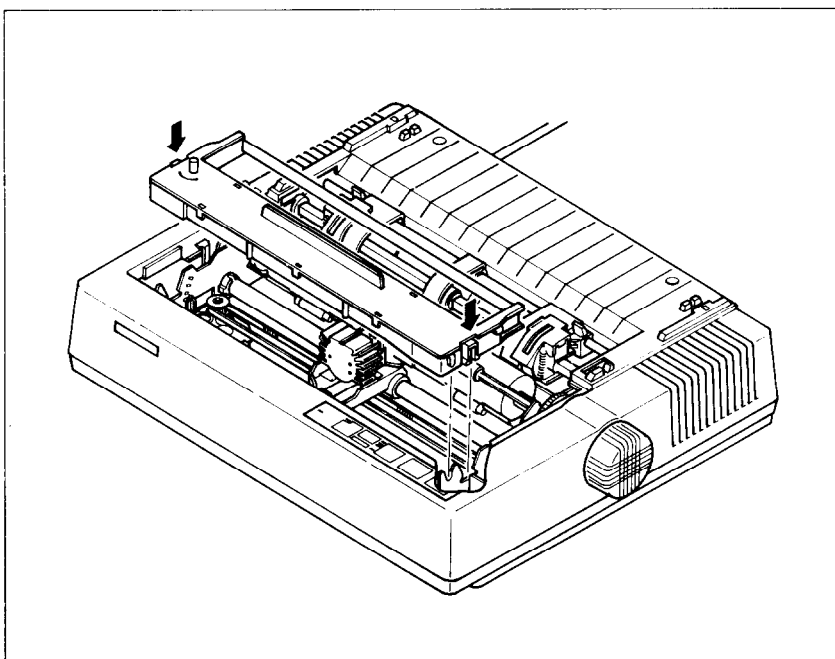
This printer uses a neat, easy-to-change ribbon cartridge so you don't have to spend a lot of time threading a ribbon. And getting your hands all dirty to boot.

Telling you how to put in a ribbon is like explaining how to tie your shoelaces — it takes a lot longer to tell than to do. You can just follow the illustrations if you wish; they'll tell you all you really need to know.

Or, if you feel better following written instructions, read on.

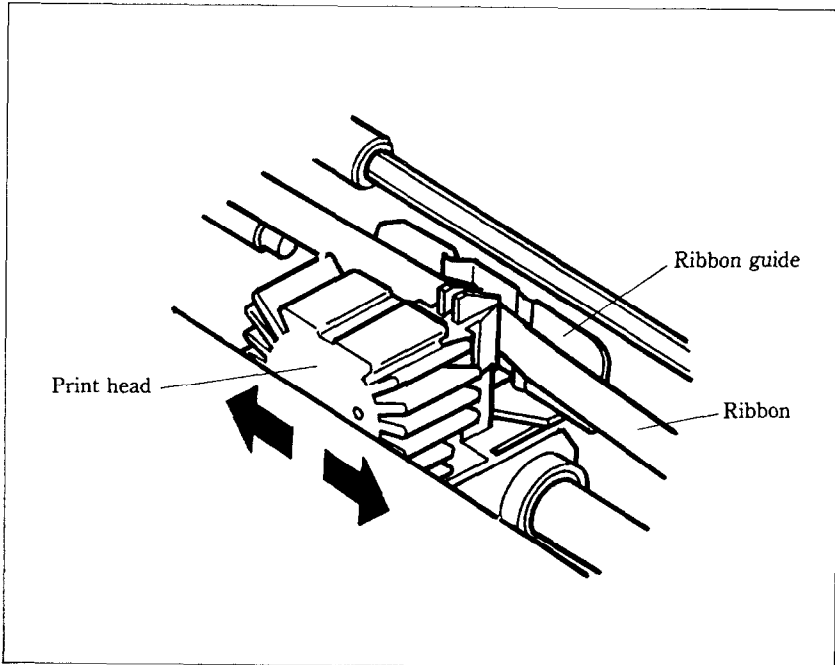
1. Turn *off* the power and remove the printer cover.
2. Now slide the print head gently to the center of the printer.

**Warning:** The print head gets hot during operation, so let it cool off before you touch it.



**Figure 1-4.** Press the cartridge into place until the holding springs snap into place.

3. Using the guide holders as a fulcrum with the ribbon facing away from you, as shown in Figure 1-4, lightly press the cartridge down until the two holder springs snap shut to hold the cartridge firmly in place.
4. Check that the cartridge fits so that the drive pins engage the cartridge teeth.
5. Gently slide the print head carriage manually all the way to your right or left until the ribbon automatically slips down into its proper place between the print head and the silver ribbon guide.
6. Put the printer cover back on and you're finished. (A special switch on the printer prevents the printer from working when the cover is off.)



**Figure 1-5.** All you have to do is to slide the print head carriage manually to your right or left, then the ribbon slips down by itself into its proper position.

---

---

# CHAPTER 2

## GETTING TO KNOW YOUR PRINTER

---

---

**Subjects we'll cover in Chapter 2 include —**

- **Parts of the printer — what they're for and how to use them;**
- **Paper selection and loading;**
- **Adjustment;**
- **Connecting your printer to the computer;**
- **Extra functions with the control panel.**

### **CONTROLS AND PARTS OF THE PRINTER**

#### ■ **Parts of the printer**

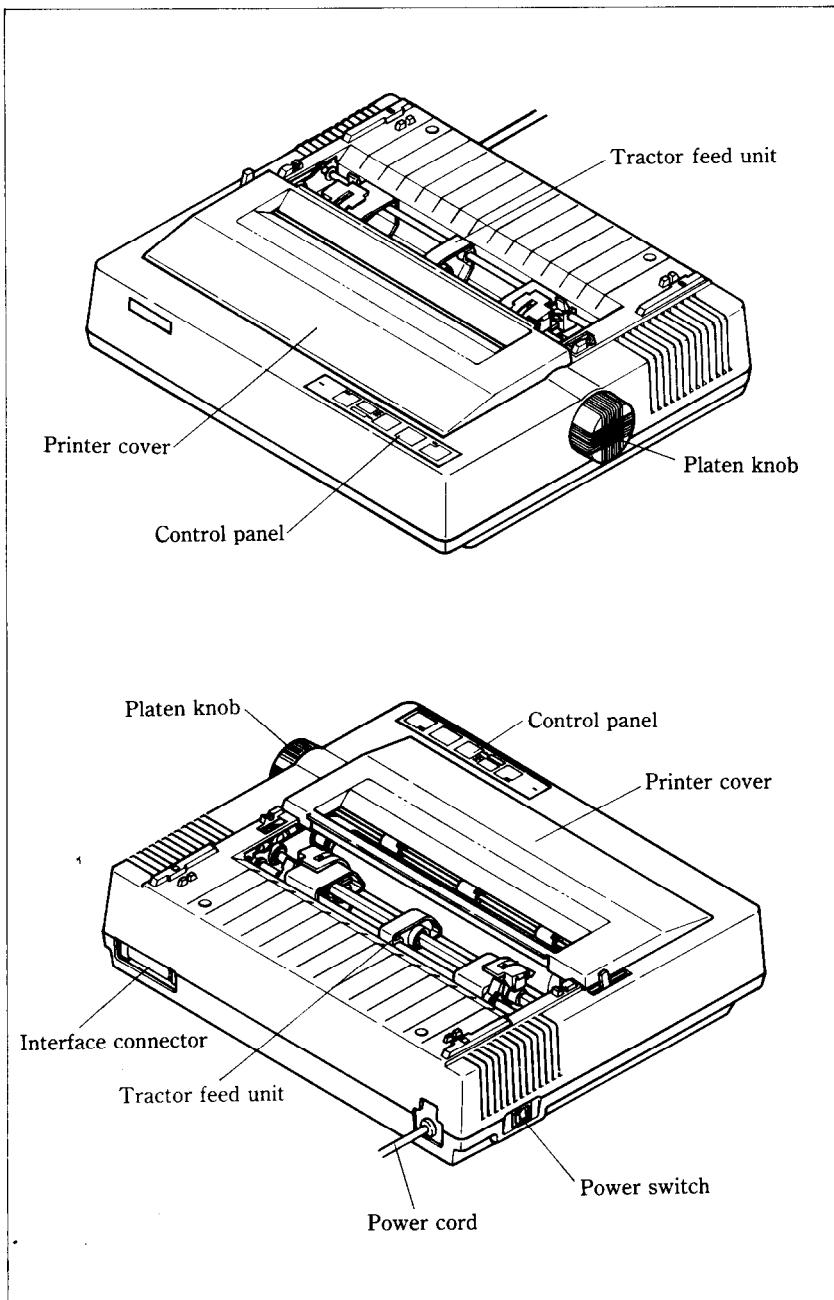
First, we'll go over the parts of the printer. You saw most of these when you unpacked your printer. Now we'll give you a brief explanation of what they do. (For details on the initial set-up of this printer, with all components in place, see Chapter 1.)

**PRINTER COVER** — This protects the ribbon and the print head from dust and dirt, and cuts down the sound of the printer.

**PAPER GUIDE** — As you've guessed, this flat plastic molding guides the paper during printing (it is raised for single sheets and lies flat for sprocket-feed paper).

**POWER CORD** — This cord connects the printer to its power source, usually a wall outlet. It's located at the left rear of the printer.

**PRINT HEAD** — This is the unit that does the actual printing. Like the strike lever in a typewriter, tiny, stiff wires in the print head hit the paper through a ribbon.



**Figure 2-1.** Front and rear views of the printer

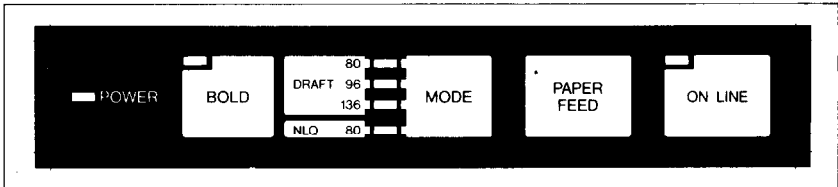
**TRACTOR FEED UNIT** — The drive gear and sprockets of the tractor feed unit move sprocket-feed paper through the printer.

**PLATEN** — This is the rubber cylinder that carries paper to the print head.

**INTERFACE CONNECTOR** — On the back of the printer, this connector is used to connect your computer to the printer.

### ■ Controls and indicators

Now let's take a tour around the controls, starting with the control panel on the right. There are seven indicators and four keys on the control panel.



**Figure 2-2.** The control panel.

**POWER INDICATOR** — Glows red when the power is *on* or blinks when the printer is out of paper or some other error occurs.

**BOLD KEY and INDICATOR** — Pressing this key selects boldface printing.

**DRAFT INDICATORS (80, 96, 136)** — Glows green to indicate the number of characters per line when the printer is in the draft mode (set by the Mode key or the software control).

**NLQ INDICATOR** — Glows green when the printer is printing in the Near Letter Quality (NLQ) mode (set by the Mode key or the software control).

**MODE KEY** — Changes the print mode every time it is pressed.

**PAPER FEED KEY**—Advances the paper one line at a time when the On Line indicator is off. If you hold the key down, you'll get consecutive line feeds, one after the other. If you push the On Line key while holding this key, you can advance the paper to the top of the next page or a new form.

**ON LINE KEY and INDICATOR** — Glows green when the printer can receive data from your computer (on line). When the printer is off line, it sends a signal to the computer indicating that it cannot accept data. Pressing the On Line key switches the printer on line and off line.



### ■ Extra functions

There are eleven more functions that are not directly specified on the control panel. You can use six of them by pressing a key (or keys) when you turn on the power; five others are activated by pressing two control keys at once.

For details, please refer to the end of this chapter.

### ■ Other controls

There are other controls, not connected to the control panel board. Some of the more important ones are:

**POWER SWITCH** — The switch to turn the printer on is at the back, on the left side.

**PLATEN KNOB** — This knob is at the middle of the right side and lets you turn the platen by hand, just like a typewriter.

*Caution:* Turn this knob only with the Power switch *off*. Turning it with the power on could damage the platen drive gears.

**RELEASE LEVER** — The release lever is on top of the printer, near the left rear corner. You'll be using it often — it controls how strongly the paper is held against the platen. The release lever has four positions: the top setting is used for inserting paper, the next is for single sheets, the third is for sprocket-feed paper, and the bottom one is used when adjusting the paper.

**PAPER BAIL** — The bail is the movable bar that holds the paper against the platen. Its position depends on the setting of the release lever.

**DIP SWITCHES** — These are a set of ten switches and a set of six switches that are used in interfacing the printer to your computer. There are also switches to set page length, character style, perforation skipping, and selecting the international character sets. See Appendix A for a complete list and explanation.





## SELECTING AND LOADING PAPER

Your printer can handle the two basic types of paper—single sheets (stationery, envelopes, multipart business forms, etc.) and continuous paper (fan-folded perforated paper).

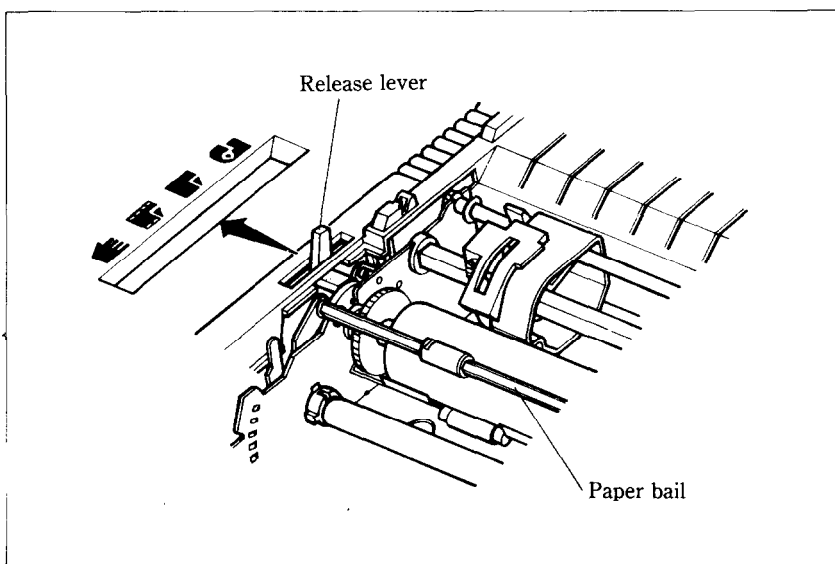
This is a good place to tell you about the release lever, which

you'll be using often. This lever controls the pressure of the paper against the platen.

It has four settings:

-  — The top setting is used when you want to load paper in the printer.
-  — The second setting is for normal single sheets.
-  — The third is for sprocket-feed paper.
-  — The bottom one is used when you want to release the paper completely to adjust it.

The paper bail holds the paper against the platen according to the setting of the release lever. The bail is opened when the release lever is in the top position, closed when the lever is in the second or third positions, and opened by the printer when the lever is in the bottom position.



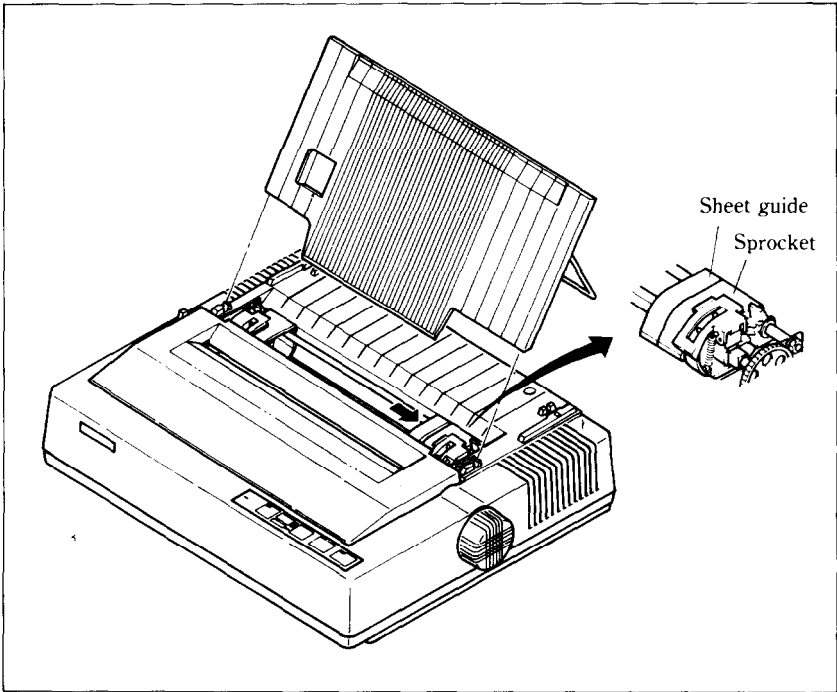
**Figure 2-3.** Use friction feed for single sheets and tractor feed for continuous paper.

### ■ Loading single sheets

Now, instead of feeding the paper in *manually* by turning the platen knob, we're going to use the release lever (the printer must be turned *on*). Remember what we told you about this lever? It allows you to advance the paper according to its position.

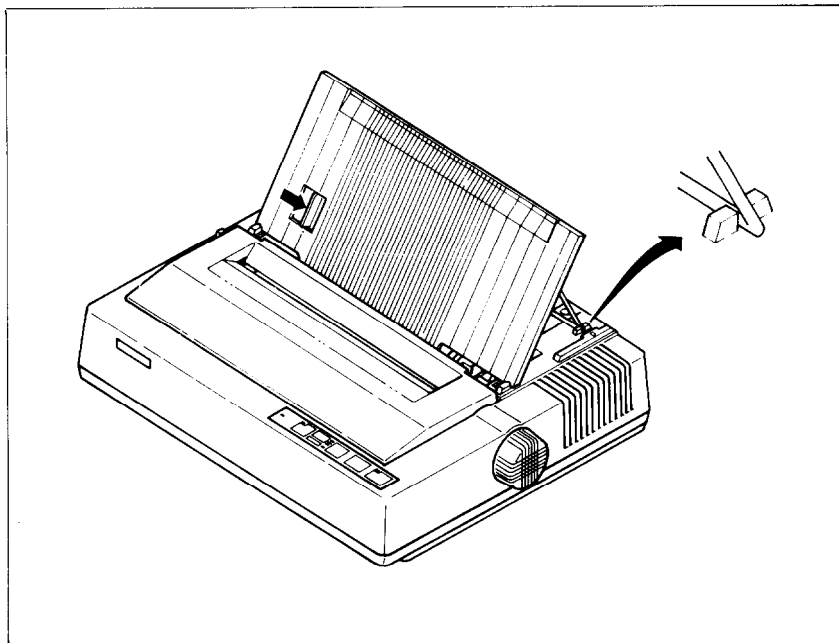
OK? Let's start.

1. Move the sprockets on the tractor feed unit all the way to the ends.
2. Slide the sheet guide all the way to the right.
3. With the ribs of the paper guide toward you and the sliding adjuster at the bottom left, insert the guide into the slot at the back of the printer cover.

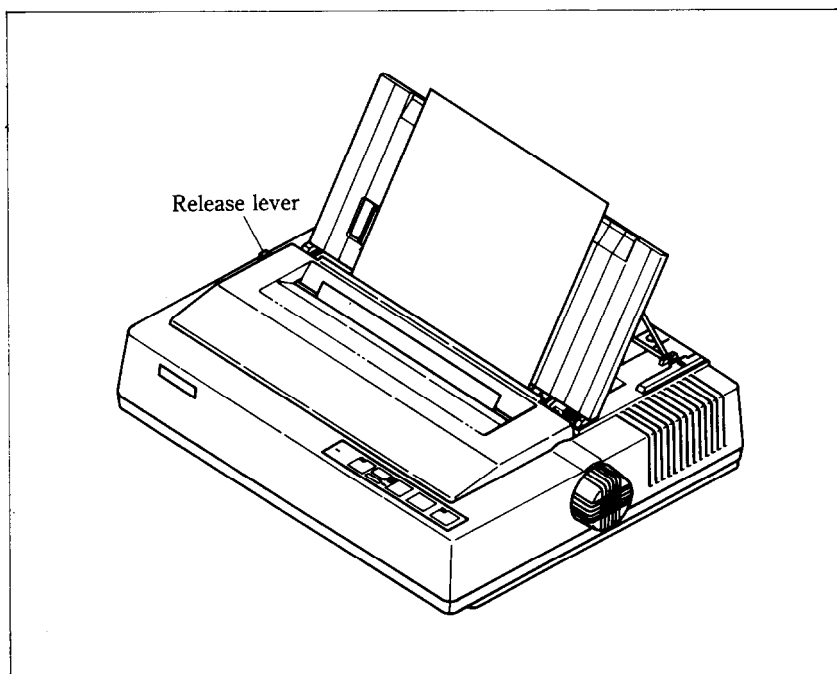


**Figure 2-4.** Raise the paper guide for loading single sheets.

4. Insert the guide stand in the notches provided for it and use it to prop up the paper guide.
5. Position the adjuster at the triangle marked on the guide for standard margins, or set it according to your requirements.
6. Set the release lever for single sheets—the second position from the top.
7. Align a sheet of paper on the guide along the adjuster.
8. Turn on the power switch. (The Power indicator will blink because there is no paper.)



**Figure 2-5.** Prop up the paper guide by inserting the guide stand in the notches provided for it.



**Figure 2-6.** You can insert a sheet of paper by using the release lever.

9. Now set the release lever to the auto-feed setting—the top position. The printer will advance the paper automatically.
10. When the paper stops, set the release lever for single sheets again.

### **To align paper that is not in straight—**

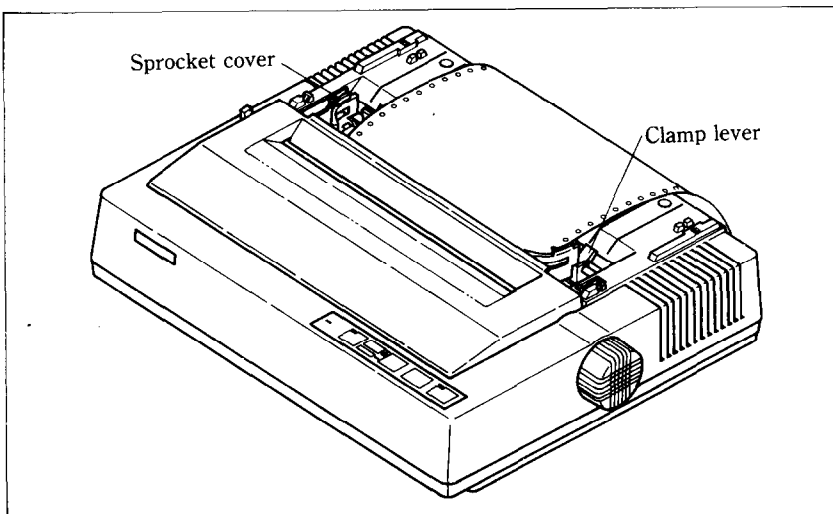
- Set the release lever to the adjustment setting.
- Straighten the paper and adjust it for the margins you want.
- Move the release lever back to its original setting.

### **■ Loading sprocket-feed paper**

This is the familiar computer paper, with the holes along the sides and perforations between the sheets. It is also called sprocket, punched, fan-fold, or just plain “computer paper.” It can be as narrow as 4”, and up to 10” wide.

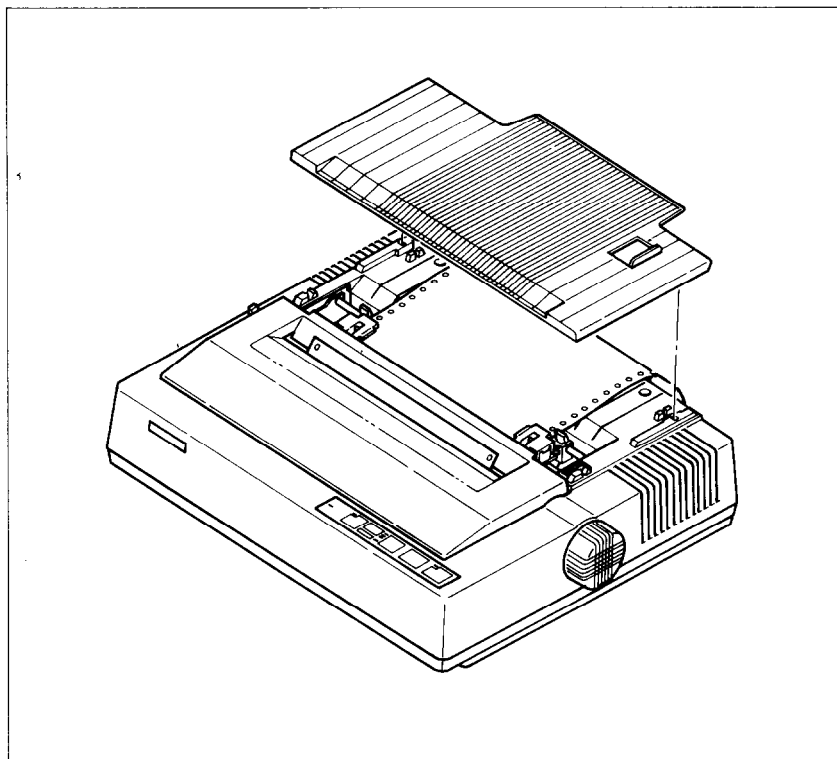
Figure 2-7 shows the tractor unit. Please identify the parts of the tractor.

1. Turn on the printer and set the release lever for sprocket-feed paper.
2. If the paper guide is installed, remove it.
3. Place a stack of fan-fold paper behind the printer.
4. Open the sprocket covers, on the right and left sprocket units, as shown in Figure 2-7.



**Figure 2-7.** Open the sprocket covers to expose the sprocket teeth.

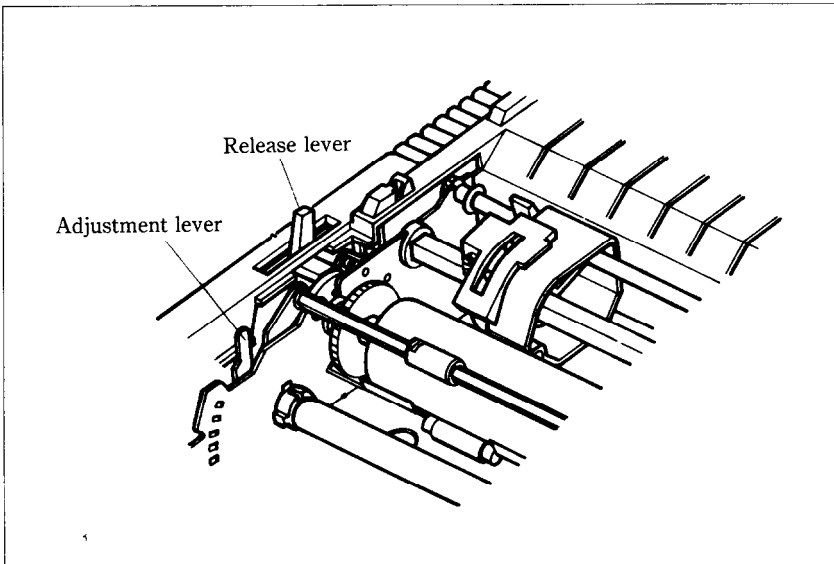
5. Flip the clamp levers backward. This allows the two sprocket units to move freely right and left so you can align them with the holes in the paper.
6. Feed the end of the paper into the slot next to the platen cover plate.
7. Fit the holes in the paper over the sprocket pins so the paper is even, and clamp the sprocket units in place.
8. Check that the paper is still positioned correctly, and close the sprocket covers (Figure 2-8).
9. The Power indicator should be blinking. Turn the platen knob to feed the paper until the indicator stops blinking.
10. Now turn the platen knob the other way just a bit, until the indicator starts blinking again.
11. Set the release lever to the auto-feed setting. The printer will advance the paper automatically.
12. When the paper stops, set the release lever for sprocket-feed paper again.
13. Finally, install the paper guide on the printer (Figure 2-8). Now you're ready to roll!



**Figure 2-8.** Ready to run with sprocket-feed paper.

## ADJUSTING THE PRINT HEAD

The distance between the print head and the platen must be adjusted to accommodate papers of different thicknesses. To make this adjustment, move the adjustment lever, which is under the printer cover and immediately in front of the release lever (Figure 2-9). Pulling the adjustment lever towards you will widen the gap; pushing it away from you will narrow the gap.



**Figure 2-9.** The adjustment lever allows for different thicknesses of paper.

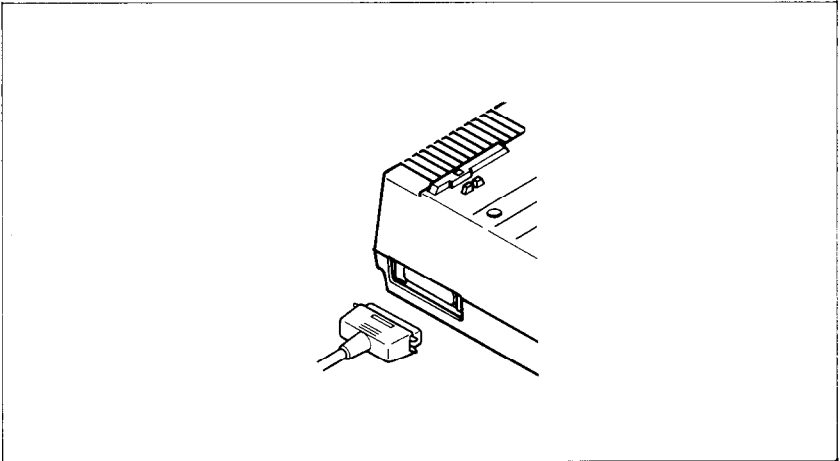
There are four positions; you can feel the lever clicking into the various notches. The first step (illustrated) is the one most commonly used for single sheets of paper.

You shouldn't have any trouble finding the right gap setting for your paper. If necessary, experiment; you'll soon find the best position for the paper you're using.

## CONNECTING THE PRINTER

Now that you have assembled your printer, it's time to use it for what you bought it for — print information from your computer. But first you have to connect it to your computer. Please follow the instructions in the order listed below.

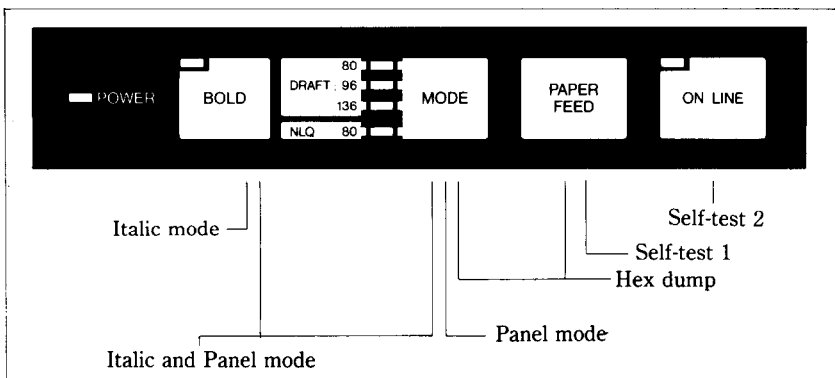
1. Make sure both your computer and printer are turned off.
2. Connect one end of the interface cable to the connector socket at the right rear of the printer as shown in Figure 2-10.
3. Connect the other end of the cable to your computer as described in the computer manual.



**Figure 2-10.** This is how you connect the cable.

## EXTRA FUNCTIONS WITH THE CONTROL PANEL

There are many functions that are not directly specified on the control panel. In this section, we'll show you these extra functions.



**Figure 2-11.** Extra functions while turning on the printer.





### ■ Hex dump

Can you guess what a “hex dump” is? No, it’s not where witches throw away useless spells. A hex dump is an advanced ability of your printer that you can use, in certain cases, to find a problem with your system. Fortunately, such problems rarely arise but the hex dump is available if one does. We’ll go over hex dump in Chapter 4. Right now, we’ll just tell you how to make a hex dump:

1. Plug in the printer (don’t turn it on yet).
2. Insert a sheet of paper, as you did for the self-tests.
3. While holding down both the Paper Feed and Mode keys, turn on the power switch.

### ■ Panel mode

As you’ll learn in Chapter 5, this printer has many software controls. But if you want to print in one mode, ignoring the control codes, the “Panel” mode takes effect for you. To set the “Panel” mode, follow the procedures:

1. Plug in the printer (don’t turn it on yet).
2. While holding down the Mode key, turn on the power switch.

Notice that this mode stays on until you turn off the printer.

### ■ Italic mode

Sometimes, you may want to print with italic characters with Draft mode as the power-on default. You can set the Italic mode with the following procedures:

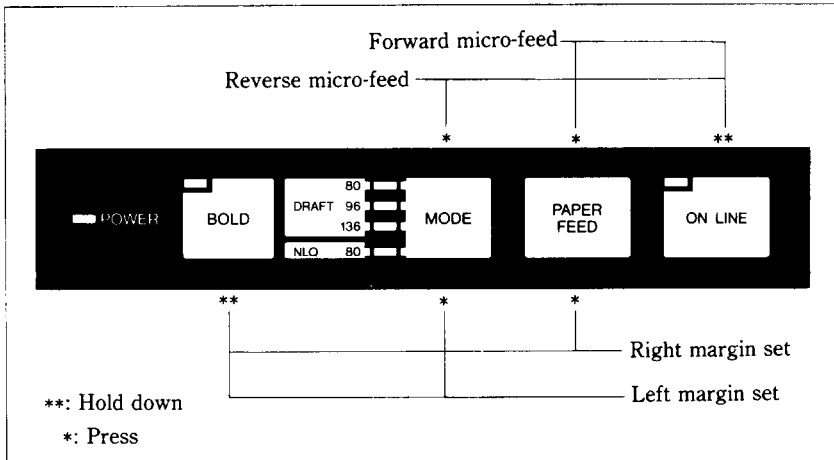
1. Plug in the printer (don’t turn it on yet).
2. While holding the Bold key, turn on the power switch.

This mode stays on until you send the cancel command to your printer. This mode re-activates when you send the reset command to your printer.

### ■ Italic and Panel mode

You can combine with the “Panel” mode and the “Italic” mode at a time. To set these modes at a time follow the procedures below:

1. Plug in the printer (don't turn it on yet).
2. While holding both the Mode and Bold keys, turn on the power switch.



**Figure 2-13.** You can set many functions by the combinations of the control panel keys while in the Off Line mode.

#### ■ Setting print start position

When you want to align the print start position, you can set it by the micro-feed operation with the control panel, instead of turning the platen knob manually.

1. Set the Off Line mode by pressing the On Line key.
2. While holding down the On Line key, press one of the following keys.
  - Paper Feed key — Forward micro-feed
  - Mode key — Reverse micro-feed
3. When you can set the print start position, release the Paper Feed key or the Mode key first, then release the On Line key.

#### ■ Setting the left and right margins

As you'll learn in Chapter 4, you can set the left and right margins with control codes. In addition, you can set them by the following procedures.

1. Set the Off Line mode by pressing the On Line key.
2. While holding the Bold key, press one of the following keys.

Mode key — Left margin set

Paper Feed key — Right margin set

3. While holding the two keys, the print head moves across the page step-by-step.
4. When the print head goes to the position where you want to set margin, release the two keys. So the printer acknowledges the margin with the sound of beep.

**MEMO**

---

---

## CHAPTER 3

# BASIC PRINTING

---

---

Subjects we'll cover in Chapter 3 include—

- Listing BASIC programs on the printer;
- How a program prints things;
- Control codes, escape codes, and command syntax;
- Near letter quality (NLQ) characters;
- Fixed and proportional character spacing;
- Special printing —  
    Printing in italics,  
    Underlining,  
    Superscripts and subscripts,  
    Boldface and emphasized text,  
    Mixing print modes.

To show you how to control your printer from a program, we choose BASIC because it is easy to learn and easy to use. Also, more personal computer users program in BASIC than in any other language.

The rest of this manual will show you a little BASIC — just enough for you to use your printer. We're not going to try to make you an expert programmer, though, only get you started. There are many excellent books that will teach you BASIC, so if you discover that you like to program you should have no trouble learning more about it.

### SOME BASICS OF BASIC

#### ■ A new language!

Many people who meet BASIC for the first time are intimidated. Some are put off by the idea of learning any new

language, perhaps recalling the rigors of high-school Latin. Others are unnerved by anything having to do with computers.

Well don't be! In the first place, BASIC may well be the easiest language you could learn: it has a vary limited vocabulary, a simple but precise grammar, and its dialects — unlike those of English — usually different from each other only in minor detail. Without programming, a computer is a useless collection of chips and wire — why should we think of it as something special? Anyway, computers are here to stay — let's accept them with good grace.

### ■ First steps

The first things that a beginner learns to do are to list a program and to print a character string. Certainly these are the easiest operations one can do, but even they may depend on what computer you have. In Microsoft BASIC, we can list all the steps in a program by entering LIST. This lists them on the CRT screen; if we want to print them on a printer, we prefix the command with an L (enter LLIST).

The Microsoft BASIC command for outputting information is PRINT. Like the LIST command, this displays the information on the CRT screen so we have to add an L (→ LPRINT) if we want to use the printer. Just put whatever you want to print between quotes and after LPRINT (anything enclosed in quotes is called a character string). For example, we would use LPRINT "Hello!" to output "Hello!" to the printer. We'll see later how to LPRINT things other than character strings.

We started with Microsoft BASIC because it is the most widely used version of BASIC around. The programs in this manual are written in Microsoft BASIC so they should run on most computers. But if strange things happen when you try to run a program, check the BASIC manual that came with your computer.

Let's talk about Apple II computers for a minute. These enormously popular computers use their own brand of BASIC. To use an Apple II, enter the following —

```
PR#1      PR#1
LIST      PRINT "Hello! "
PR#0      PR#0
```

The PR#1 tells the Apple to send everything to the printer, the LIST or PRINT command sends it, and the PR#0 returns

output to the screen.

Other computers, notably the Commodore C-64, require you to open the printer as a numbered device then to direct output to that device. For example, you might type the following to print "Hello!" or to list a program in the C-64's memory.

```
OPEN4,4    OPEN4,4
CMD4       CMD4
LIST       PRINT "Hello!"
CLOSE4     CLOSE4
```

Here, the first line says that the printer is device number 4, the second directs output to it, the third does the printing, and the last line closes down device number 4.

Appendix H gives more information about listing programs on the various computers. Find the part that applies to your computer and try it.

Now that we know how to address the printer, let's try listing a BASIC program. Load a program into memory ready to program printer operation — just as soon as we learn a little bit about the ASCII codes.

### ■ ASCII codes and the CHR\$ function

You talk to your computer in BASIC, but your computer and your printer talk to each other in what are known as ASCII codes. In the ASCII code, each number from 0 to 255 has a particular meaning — 36, for example, makes the printer print a dollar sign. Some numbers cause the printer to do other things, too. For instance, sending a 7 sounds the printer's bell.

Taken together, these numbers and their meanings make up the ASCII code (pronounced *ask-key*), which stands for the *American Standard Code for Information Interchange*. There are ASCII codes for all the letters of the alphabet (upper case and lower case), 0 to 9, most punctuation marks, and some (but not all) of the functions of the printer.

There are a number of different ways to represent an ASCII code, depending on how you are using it. For example, the ASCII codes for the letter "A" are 65 (decimal) or &H41 (hexadecimal). Or you can just call it "A". Appendix B shows all of the ASCII codes.

BASIC uses the CHR\$ function to represent ASCII characters and many functions. To print the letter "A" we



would enter `LPRINT CHR$(65)`. To make the printer's bell sound, we would `LPRINT CHR$(7)`. In general, we print a character by entering `LPRINT CHR$(ASCII code)` to the printer.

We can also use hex ASCII codes. Although we use only decimal ASCII codes in this manual, you should understand at least what a hex code is. "Hex" is short for hexadecimal and refers to a base-16 number (the numbers we use in everyday life are base 10). Since the hex system needs 16 digits, it uses the numerals 0 through 9 and also the letters A through F. You can always tell that a number is in hexadecimal by the "&H" immediately preceding it. The ASCII code for the letter "A" (65 in decimal) is &H41 in hex.

### ■ Control codes

ASCII codes with values of 32 or less do not have their own keys. These codes control many of the printer's functions, so we call them control codes. To enter a control code from the keyboard, we have to press two keys at the same time — the "control" key and one other. The other key determines what code is sent — pressing the control (CTRL) key and A sends ASCII code 1, CTRL B sends ASCII code 2, and so on.

Your printer has a lot of control codes to let you do some really nifty things. Let's try one that we've mentioned several times already:

```
10 ' Demo of ASCII code
20 LPRINT CHR$(7)
30 END
RUN
```

That's the printer's bell (we call it that even though it sounds like a buzzer). We'll learn more about it in later (we just wanted to show you a control code that would get your attention right away).

There are four common ways of referring to a control code: the name of the code or its abbreviation, the decimal ASCII value, the hexadecimal ASCII value, and the "CTRL-" value. For example, the ASCII code that causes the printer to advance the paper one line is decimal 10. This code may be referred to by any of the following.

line feed	—the name of the code
⟨LF⟩	—its abbreviation
ASCII 10	—its decimal value
ASCII &H0A	—its hexadecimal value (the &H signifies hex)
CHR\$(10)	—the way it's used in BASIC
CTRL-J	—the way you send it from a keyboard

Of course, most of the time we don't need to bother with these. Our computers are smart enough to know that when we press the "A" key we want to print the letter "A" — they take care of all the intermediate steps.

Appendix B is a table that shows the various names for each code so you can convert back and forth. The microcomputer world is not very consistent in describing ASCII codes, so it's important that you have a basic knowledge of them.

### ■ The escape codes

Back when the ASCII system was set up, computer equipment was relatively simple and thirty-three control codes were considered sufficient at the time. The American Standards people realized that, eventually, more control codes would be needed so they included the escape (ESC) code to allow almost any number of additional codes to be defined when they became necessary.

ESC allows us to "escape" from the ordinary set of control codes so we can specify additional functions and other information needed for a printer function. In this manual, we'll write the ESC code inside broken brackets, like this — ⟨ESC⟩.

⟨ESC⟩ — decimal 27 — is always followed by at least one other number; it is never used alone. The whole series of related numbers is called an escape sequence.

### ■ A note on command syntax

Because the readers of this manual will be running such a wide variety of applications on so many different computers, we just can't show the exact way of sending codes to the printer for each one of them. Instead, as we introduce you to each new command, we will show the commands as in this example:

```
⟨ESC⟩ "W" 1
```

This command that turns on expanded printing.  $\langle \text{ESC} \rangle$ , as we mentioned earlier, is the escape code (which is ASCII code 27). A letter or number in quotes (such as the "W" above) means that the character should be sent to the printer (without the quotes). In our example, you should send a capital W following the escape code. In BASIC, you could do this in a couple of ways: by sending the character itself (e.g, LPRINT "W");, or by using the CHR\$ function to send the ASCII code for the character (e.g. LPRINT CHR\$(87);).

Many of printer commands end with a 1 or 0. When shown as in the above example (i.e. no quotes and no "CHR\$"), you can use either ASCII code 1 (i.e. CHR\$(1)) or the character "1" (which is ASCII code 49). The same idea applies to commands ending with 0.

So for our example above, any of these BASIC statements will have the same result:

```
LPRINT CHR$(27); "W"; CHR$(1)
LPRINT CHR$(27); "W"; CHR$(49)
LPRINT CHR$(27); "w1"
```

Even though, there are many commands that require the use of ASCII code 0; the character "0" (ASCII code 48) cannot be substituted. In these cases, instead of an unadorned 0 we will show CHR\$(0) each time these commands are referenced.

That's it for the basics. You are now ready to learn how to use the many features of your printer.

## **SOME SPECIAL KINDS OF TEXT**

If you looked carefully at your printer's self test, you noticed that it can print in italics. But there's more! Your printer can underline characters, print superscripts and subscripts, and perhaps most exciting, print near letter quality characters.

### **■ Near Letter Quality characters**

This printer's Near Letter Quality (sometimes abbreviated as NLQ) character set is ideal for correspondence and other important printing, for it takes a keen eye to detect that it is from a dot matrix printer. Normally, your printer prints draft quality characters. This is adequate for most work and it prints fastest.

But for the final printout, try NLQ. The program below shows how.

```

10 ' Demo of NLQ character set
20 LPRINT CHR$(27);"x1";
30 LPRINT "This line shows NEAR LETTER QUALITY!"
40 LPRINT CHR$(27);"x0";
50 LPRINT "This line shows standard print."

```

In this program, line 20 selects NLQ characters with <ESC> "x"1 command. Line 30 prints a sample before line 40 switches printer back to draft printing with an <ESC> "x"0. When you run the program you should get this:

```

This line shows NEAR LETTER QUALITY!
This line shows standard print.

```

**Table 3-1**  
Near letter quality commands

Function	Control code
Near letter quality ON	<ESC> "x"1
Near letter quality OFF	<ESC> "x"0

### ■ Italic printing

*Italic* letters are letters that are slanted to the right. Your printer can print all of its letters except NLQ characters in *italic* as well as the roman (standard) letters you are accustomed to. Italics can be used to give extra emphasis to certain words. The command codes to turn italic on and off are shown in Table 3-2.

**Table 3-2**  
Italic commands

Function	Control code
Italic ON	<ESC> "4"
Italic OFF	<ESC> "5"

Use this program to see italic characters:

```

10 ' Demo of italic and roman
20 LPRINT CHR$(27);"4";
30 LPRINT "This line is in ITALIC characters."
40 LPRINT CHR$(27);"5";
50 LPRINT "This line is in ROMAN characters."

```

Here is what you should get:

```

This line is in ITALIC characters.
This line is in ROMAN characters.

```

In this program, line 20 turns italic on with  $\langle \text{ESC} \rangle$  "4", and line 40 turns italic off with  $\langle \text{ESC} \rangle$  "5".

### ■ Underlining

Not only can your printer print all styles of printing in both roman and italic, but it can underline them too. The control codes are shown in Table 3-3.

**Table 3-3**  
**Underline commands**

Function	Control code
Underline ON	$\langle \text{ESC} \rangle$ "-1"
Underline OFF	$\langle \text{ESC} \rangle$ "-0"

Again, that's simple. Let's try it with this program:

```

10 ' Demo of underlining
20 LPRINT CHR$(27);"-1";
30 LPRINT "This phrase is UNDERLINED;";
40 LPRINT CHR$(27);"-0";
50 LPRINT " this is not."

```

It should come out like this:

```

This phrase is UNDERLINED; this is not.

```

In this program underline is turned on in line 20 with `<ESC> “-”1`, and then off in line 40 with `<ESC> “-”0`. There’s a new little wrinkle in this program, though. The semicolons at the end of the first three lines told BASIC that those lines were to be continued. Therefore, BASIC didn’t send a carriage return and line feed at the end of those lines. We just did this to illustrate that all these control codes can be used in the middle of a line. It’s easy to underline or *italicize* only part of a line.

### ■ Superscripts and subscripts

Your printer can print in two different heights of characters. The smaller characters are called *superscripts* and *subscripts* and are half the height of normal characters. *Superscripts* print even with the tops of regular printing while *subscripts* print even with the bottom of regular printing. They are frequently used to reference footnotes, and in mathematical formulas.

Table 3-4 has the codes for using superscripts and subscripts.

**Table 3-4**  
**Superscripts and subscripts commands**

Function	Control code
Superscript ON	<code>&lt;ESC&gt;“S”0</code>
Subscript ON	<code>&lt;ESC&gt;“S”1</code>
Super and subscript OFF	<code>&lt;ESC&gt;“T”</code>

Try this program to see them work:

```

10 ' Demo of superscripts and subscripts
20 LPRINT "Look! ";
30 LPRINT CHR$(27);"S0";
40 LPRINT "SUPERSCRIPTS ";
50 LPRINT CHR$(27);"T";
60 LPRINT "& ";
70 LPRINT CHR$(27);"S1";
80 LPRINT "SUBSCRIPTS ";
90 LPRINT CHR$(27);"T";
100 LPRINT "on one line."

```

Look! <sup>SUPERSCRIPTS</sup> & <sub>SUBSCRIPTS</sub> on one line.

Here line 30 turns on superscripts with `<ESC> "S"0`. It's turned off in line 50 with `<ESC> "T"`. Then between printing text, subscripts are turned on in line 70 with `<ESC> "S"1`, and finally off in line 90. Again, everything prints on one line because of the semicolons.

## CHANGING THE PRINT PITCH

In "printer talk," the number of characters that can be printed in one inch is called the print pitch or character pitch. Normally, your printer is set for 10 characters per inch, which is called *pica* (and is the same as the pica pitch on some typewriters). This works out to 80 characters per line.

You can also print 12 characters per inch (*elite pitch*). This gives you 96 characters per line.

You can set these pitches by using the Mode key on the control panel manually, or by software as shown in the table below.

**Table 3-5**  
**Print pitch commands**

Pitch	Characters/inch	Control code
Pica	10	<code>&lt;ESC&gt; "P"</code>
Elite	12	<code>&lt;ESC&gt; "M"</code>

Try this program to see how these two pitches work. Be sure to set the printer to draft mode.

```
10 ' Demo of pica and elite pitches
20 LPRINT CHR$(27);"M";
30 LPRINT "This line is ELITE pitch."
40 LPRINT CHR$(27);"P";
50 LPRINT "This line is PICA pitch (normal)."
```

When you run this program you should get this:

```
This line is ELITE pitch.
This line is PICA pitch (normal).
```

Line 20 turns on elite pitch with `<ESC> "M"`. Line 30 prints a line at 12 characters per inch. The `<ESC> "P"` in line 40 resets the printer to pica pitch and line 50 prints a line in pica pitch.

### ■ Expanded print

Each of the print pitches can be enlarged to twice its normal width. This is called expanded print. Try this program to see how it works:

```

10 ' Demo of expanded print
20 LPRINT "Demonstration of ";
30 LPRINT CHR$(14);
40 LPRINT "EXPANDED";
50 LPRINT CHR$(20);
60 LPRINT " printing."
70 LPRINT "Notice that ";
80 LPRINT CHR$(14);
90 LPRINT "EXPANDED mode"
100 LPRINT "automatically turns off at the end
    of a line."

```

Demonstration of **EXPANDED** printing.  
 Notice that **EXPANDED mode**  
 automatically turns off at the end of a line.

Expanded print set with `CHR$(14)` is automatically cancelled at the end of the line. This is convenient in many applications, such as for one line titles. Note that you didn't need to put an `<ESC>` in front of the `CHR$(14)`, although `<ESC> CHR$(14)` works just the same.

You can also cancel one line expanded print *before* a carriage return with `CHR$(20)`, as done in line 50.

Sometimes you may wish to stay in expanded print for more than one line. Change your program to this:

```

10 ' Demo of permanent expanded mode
20 LPRINT CHR$(27);"W1";
30 LPRINT "Permanent expanded"
40 LPRINT "mode stays on until"
50 LPRINT "is is ";
60 LPRINT CHR$(27);"W0";
70 LPRINT "turned off."

```



Now the results look like this:

```
Permanent expanded
mode stays on until
it is turned off.
```

When you turn on expanded print with `<ESC> "W"1` it stays on until you turn it off with `<ESC> "W"0`.

**Table 3-6**  
**Expanded print commands**

Function	Control code
One line expanded ON	CHR\$(14) or <ESC>CHR\$(14)
One line expanded OFF	CHR\$(20)
Expanded ON	<ESC>"W"1
Expanded OFF	<ESC>"W"0

### ■ Condensed print

Each of the print pitches also can be condensed to its normal width. This is called condensed print. Try this program to see how it works:

```
10 ' Demo of condensed print
20 LPRINT "Demonstration of ";
30 LPRINT CHR$(15);
40 LPRINT "CONDENSED";
50 LPRINT CHR$(18);
60 LPRINT " printing."
```

Demonstration of CONDENSED printing.

Condensed print set with `CHR$(15)` stays on until you turn it off with `CHR$(18)`. Note that you don't need to put an `<ESC>` in front of the `CHR$(15)`, although `<ESC> CHR$(15)` works just the same.

**Table 3-7**  
**Condensed print commands**

Function	Control code
Condensed ON	CHR\$(15) <i>or</i> <ESC> CHR\$(15)
Condensed OFF	CHR\$(18)

By combining expanded print and condensed print with the two pitches, this printer has eight different character widths available.

Enter this program to see how the print pitches, expanded print and condensed print can be combined:

```

10 ' Demo of various print pitches
20 LPRINT CHR$(15);
30 LPRINT CHR$(27);"M";
40 LPRINT "This line is CONDENSED ELITE pitch."
50 LPRINT CHR$(27);"P";
60 LPRINT "This line is CONDENSED PICA pitch."
70 LPRINT CHR$(18);
80 LPRINT CHR$(27);"M";
90 LPRINT "This line is NORMAL ELITE pitch."
100 LPRINT CHR$(27);"P";
110 LPRINT "This line is NORMAL PICA pitch."
120 LPRINT CHR$(27);"Wl";
130 LPRINT CHR$(15);
140 LPRINT CHR$(27);"M";
150 LPRINT "This line is EXPANDED CONDENSED
    ELITE."
160 LPRINT CHR$(27);"P";
170 LPRINT "This line is EXPANDED CONDENSED
    PICA."
180 LPRINT CHR$(18);
190 LPRINT CHR$(27);"M";
200 LPRINT "This is EXPANDED ELITE."
210 LPRINT CHR$(27);"P";
220 LPRINT "This is EXPANDED PICA."
230 LPRINT CHR$(27);"WO"
240 END

```

Here's what you should get from this program:

This line is CONDENSED ELITE pitch.

This line is CONDENSED PICA pitch.

This line is NORMAL ELITE pitch.

This line is NORMAL PICA pitch.

**This line is EXPANDED CONDENSED ELITE.**

**This line is EXPANDED CONDENSED PICA.**

**This is EXPANDED ELITE.**

**THIS IS EXPANDED PICA.**

### ■ Proportional printing

Have you ever noticed in books and magazines? Doesn't it look nice? The main reason is that each character is given an amount of space proportional to its actual width. A typewriter (and most printer), on the other hand, give every character the same amount of space, no matter how wide it is. (Pica pitch, for example, gives a "w" and an "i" 1/10 of an inch each. Look these letters closely and you'll see that a "w" is two or three times as wide as an "i".)

Well, you too enjoy professional-looking proportional printing. You can turn proportional printing on and off with the following command.

**Table 3-8**  
**Proportional commands**

Function	Control code
Proportional ON	<ESC>"p"1
Proportional OFF	<ESC>"p"0

Try this program to see how the proportional spacing works.

```
10 ' Demo of proportional printing
20 LPRINT CHR$(27);"M";
30 LPRINT "This line is NORMAL ELITE printing."
40 LPRINT CHR$(27);"p1";
50 LPRINT "This line is PROPORTIONAL ELITE."
60 LPRINT CHR$(27);"P";
70 LPRINT "This line is PROPORTIONAL PICA."
80 LPRINT CHR$(27);"p0";
90 LPRINT "This line is NORMAL PICA printing."
100 END
```

When you run this program you should get this:

```
This line is NORMAL ELITE printing.
This line is PROPORTIONAL ELITE.
This line is PROPORTIONAL PICA.
This line is NORMAL PICA printing.
```

Line 20 selects the elite pitch and line 40 turns on the proportional printing with `<ESC>“p”1`. Line 50 prints a line with proportional elite pitch. Then, line 60 selects the pica pitch, so that line 70 prints a line with proportional pica pitch. Finally, line 80 resets the proportional printing and line 90 prints a line in normal pica pitch.

**NOTE:** When you change the print pitch by the MODE key on the control panel, this proportional spacing should be automatically cancelled.

## MAKING WORDS STAND OUT

Your printer has very good print density when it's just printing regularly. But sometimes you may want something to stand out from the rest of the page. This printer provides two ways to do this: boldface and emphasized print. Both of these go over the characters twice, but they use slightly different methods to darken the characters. Let's try them and see what the difference is.

The following table shows the control codes for getting into and out of boldface and emphasized modes.

**Table 3-9**  
**Print emphasis commands**

Function	Control code
Boldface ON	<code>&lt;ESC&gt;“G”</code>
Boldface OFF	<code>&lt;ESC&gt;“H”</code>
Emphasized ON	<code>&lt;ESC&gt;“E”</code>
Emphasized OFF	<code>&lt;ESC&gt;“F”</code>

Try them now with this little program:

```
10 ' Demo of boldface and emphasized
20 LPRINT CHR$(27);"G";
30 LPRINT "This line is BOLDFACE printing."
40 LPRINT CHR$(27);"E";
50 LPRINT "This line is BOLDFACE and
    EMPHASIZED."
60 LPRINT CHR$(27);"H";
70 LPRINT "This line is EMPHASIZED printing."
80 LPRINT CHR$(27);"F";
90 LPRINT "This line is normal printing."
100 END
```

Run this program. The results will look like this:

```
This line is BOLDFACE printing.
This line is BOLDFACE and EMPHASIZED.
This line is EMPHASIZED printing.
This line is normal printing.
```

Line 20 turns on boldface with  $\langle \text{ESC} \rangle$ "G" and line 30 prints a line of text. In line 40 emphasized is turned on with  $\langle \text{ESC} \rangle$ "E". Line 50 prints a line of text in boldface *and* emphasized. Line 60 then turns boldface off with  $\langle \text{ESC} \rangle$ "H" so that line 70 can print in emphasized only. Finally, line 80 turns emphasized off, so your printer is set for normal printing.

Look closely at the different lines of printing. In the line of boldface printing each character has been printed twice, and they are moved down just slightly the second time they are printed. In emphasized printing, they are moved slightly to the right the second time your printer prints. The last line combined both of these so that each character was printed 4 times. Now that's pretty nice printing, isn't it?

## MIXING PRINT MODES

We have learned how to use the various print modes individually and together. Now we'll see how to combine them

more efficiently.

You have at your disposal a unique command that lets you choose any valid combination of print modes and pitch. This is the Master Print mode command. It looks like this:

⟨ESC⟩ “!” *n*

Here, the value of *n* defines the print style to be selected. The value of *n* can range from 0 to 255, which is the range of values that can be stored in one eight-bit byte. If you look at each bit in this byte, you'll find that each one represents a printing style variation. Adding the binary values of the selected bits gives the value of *n* for a particular combination of print styles.

Table 3-10 shows the decimal values of the bits in the Master Print byte. To calculate the value *n* for a particular combination of printing styles, just add the values of the features that you want to combine.

**Table 3-10**  
**Values of mixing print styles for Master Print**

Bit	Print style	Decimal value
1	Elite print	1
2	Proportional print	2
3	Condensed print	4
4	Emphasized print	8
5	Boldface print	16
6	Expanded print	32
7	(Not used)	
8	Underline	128

For example, if you want to select elite expanded boldface print, you would calculate the value of *n* like this:

Elite	1
Boldface	16
Expanded	32
<i>n</i> =	<u>49</u>

The command would look like this:

CHR\$(27);“!”;CHR\$(49)

To better understand the way the print modes work, consider that each mode except pica (pica is the default) has a separate switch that can be turned on and off via software. Once the switch is on, it stays on until turned off. When two modes that conflict are turned on at the same time, the printer must choose which one to use.

For example, suppose you turn on both Elite and Emphasized modes. Since these cannot combine the printer must make a choice; in this case, the printer chooses Elite.

### *Summary notes*

- 1) Pica is the default pitch and is active when Elite is turned off.
- 2) When two modes conflict, the one of lesser priority is cancelled. For example, Condensed and Emphasized cannot be printed at the same time, printing is Emphasized.
- 3) Elite cancels Emphasized.
- 4) Underline, and Expanded modes combine with any print modes.
- 5) Emphasized will not mix with Elite or Condensed.

---

---

# CHAPTER 4

## FORMATTING TEXT

---

---

Subjects we'll cover in Chapter 4 include—

- The carriage return and line feed;
- The amount of space between lines;
- Moving to the next page;
- The number of lines on a printed page;
- Horizontal and vertical tabs;
- Setting margins—left, right, top and bottom;
- Centering and aligning.

Chapter 3 showed us all the basic techniques of using the printer. Now we're ready for the more advanced ones. We'll concentrate on changing the appearance of the page to suit our needs.

### LINES AND LINE SPACING

#### ■ Starting a new line

Up until now the only time we have thought about printing on a new line is when we *didn't* want it to happen. We learned that putting a semicolon (;) at the end of a BASIC line will *not* end the line of printing. So somehow, the computer telling the printer when to end one line and start another.

There are two codes that are used to end one line and start another. They are *carriage return* (CHR\$(13)) and *line feed* (CHR\$(10)). Like the escape code, they have been given abbreviations which you'll find many texts (including this one): <CR> and <LF>. The codes are simple, but their action is a little confusing (especially with BASIC). Carriage return is the easiest. Each time that the printer receives a CHR\$(13) it returns the print head to the left margin. It does not advance the paper (if DIP switch 1-8 is on; see below).



Line feed is more complicated. Each time the printer receives a CHR\$(10) it both advances the paper one line and returns the print head to the left margin, ready to start a new line.

Now to add a little confusion—most (but not all) versions of BASIC add a line feed (CHR\$(10)) to every carriage return (CHR\$(13)) that they send. If your version of BASIC doesn't do this, then you should turn DIP switch 1-8 off so that your printer will add the line feed for you. When you have DIP switch 1-8 off the printer will do the same thing when it receives a carriage return as it does when it receives a line feed.

If you find that your printer double spaces when it should single space, then you probably need to turn DIP switch 1-8 on.

### ■ Reverse line feeds

Your printer has a unique capability: it can move the paper up or down! Its unique tractor design allows the paper to be fed in either direction without jamming. This allows you to move around the page at will. You can use this feature to print several columns of text side by side, or print a graph and then move back up and insert descriptive legends. As you experiment you're bound to come up with more uses!

The simplest form of reverse paper feeding is a reverse line feed. The code is <ESC><LF>, which causes the paper to move down (in effect, moving the printing *up*) one line. A "line" used in a reverse line feed is the same size as a line in a regular line feed (this is normally 1/6 inch). When you change the line spacing (which you'll read about next), you change it for both forward and reverse line feeds.

**Table 4-1**  
**Line feed commands**

Function	Control code
Return print head to left margin	CHR\$(13)
Advance paper one line	CHR\$(10)
Reverse paper one line	<ESC>CHR\$(10)

### ■ Changing the line spacing

When you turn your printer on the line spacing is set to 6 lines per inch. This is fine for most printing applications, but sometimes you may want something different. Your printer makes it easy to set the line spacing to whatever you want.

Try this program to see how easy it is to change the line spacing:



Line 40 changes the line spacing. The command  $\langle \text{ESC} \rangle "A" \text{CHR}\$(n)$  changes the line spacing to  $n/72$  of an inch. The loop that is started in line 20 increases the value of  $n$  (the variable  $I$  in this program) each time it is executed. So the line spacing increases as the program continues. Line 30 just shortcuts the loop when  $I=13$ , since BASIC won't let us send  $\text{CHR}\$(13)$  without adding an unwanted  $\text{CHR}\$(10)$  to it. Finally, the  $\langle \text{ESC} \rangle "2"$  in line 80 resets the line spacing to 6 lines per inch. This is a shortcut that is the same as  $\langle \text{ESC} \rangle "A" \text{CHR}\$(12)$ .

When you run this program with the DIP switch 1-6 off (IBM mode), you cannot get the printout as shown above.

The  $\langle \text{ESC} \rangle "A" \text{CHR}\$(n)$  command in IBM mode only defines the line spacing as  $n/72$  of an inch; the  $\langle \text{ESC} \rangle "2"$  command changes the line spacing to the amount defined by the previous  $\langle \text{ESC} \rangle "A"$ .

So, you need to change the following lines to the previous program as shown below for the IBM mode:

```
40 LPRINT CHR$(27);"A";CHR$(I);CHR$(27);"2";  
80 LPRINT CHR$(27);"A";CHR$(12);CHR$(27);"2"
```

You may wonder why they picked  $1/72$  of an inch as the increment for the line spacing command. There's a good reason: the dots that the printer makes are  $1/72$  inch apart. So this means that you can vary the line spacing in increments as fine as one dot — unless you want finer spacing, like one third dot spacing.

The  $\langle \text{ESC} \rangle "3" \text{CHR}\$(n)$  command sets the line spacing in increments of  $1/216$  inch. Change line 40 in your program so it is like this:

```
40 LPRINT CHR$(27);"3";CHR$(I);
```

and run the program again. Now the results will look like this:

THIS LINE SPACING IS SET TO 1/6 INCH (NORMAL).

The program works just the same as before, but the line spacing are just one-third what they were. This is because `<ESC>“3”CHR$(n)` sets the line spacing to  $n/216$  inch.

Table 4-2 shows all the line spacing commands, including several “shortcut” commands for commonly used line spacings.

**Table 4-2**  
**Line spacing commands**

Function	Control code
Set line spacing to 1/8 inch	<code>&lt;ESC&gt;“0”</code>
Set line spacing to 7/72 inch	<code>&lt;ESC&gt;“1”</code>
Set line spacing to 1/6 inch or use <code>&lt;ESC&gt;“A”</code> definition	<code>&lt;ESC&gt;“2”</code>
Set or define line spacing to $n/72$ inch	<code>&lt;ESC&gt;“A”CHR\$(n)</code>
Set line spacing to $n/216$ inch	<code>&lt;ESC&gt;“3”CHR\$(n)</code>
One-time line feed of $n/216$ inch	<code>&lt;ESC&gt;“J”CHR\$(n)</code>
One-time reverse line feed of $n/216$ inch	<code>&lt;ESC&gt;“j”CHR\$(n)</code>

#### ■ Moving down the page without a carriage return

So far, all the commands that move the paper also move the print head to the left margin. And normally this is what you want. Sometimes, though, you may wish to move down the page without moving the printhead back to the left margin. The following commands do just that.

The `<ESC>“J”CHR$(n)` command causes the printer to make one line feed of  $n/216$  inch, but does not *change* the setting of the line spacing. Try this program to see how it works:

```
NEW
10 ' Demo of one-time line feeds
20 LPRINT "LINE NUMBER 1."
30 LPRINT "LINE NUMBER 2.";
40 ' One-time line feed
50 LPRINT CHR$(27);"J";CHR$(100);
60 LPRINT "LINE NUMBER 3."
70 LPRINT "LINE NUMBER 4."
80 END
```

Here is what your printer will produce:

```
LINE NUMBER 1.
LINE NUMBER 2.

                LINE NUMBER 3.
LINE NUMBER 4..
```

The `<ESC>"J"CHR$(100)` in line 50 changes the spacing to 100/216 inches for one line only without moving the printhead. The rest of the lines printed with the normal line spacing. Notice that both line 30 and line 50 end with semicolons. This prevents the normal line feed from occurring.

The `<ESC>"j"CHR$(n)` command works the same way except that the paper moves in the opposite direction. Try this simple change to your program and see what a difference it makes!

```
40 ' One-time reverse line feed
50 LPRINT CHR$(27);"j";CHR$(100);

                LINE NUMBER 3.
LINE NUMBER 4.
LINE NUMBER 1.
LINE NUMBER 2.
```

## PAGE CONTROL

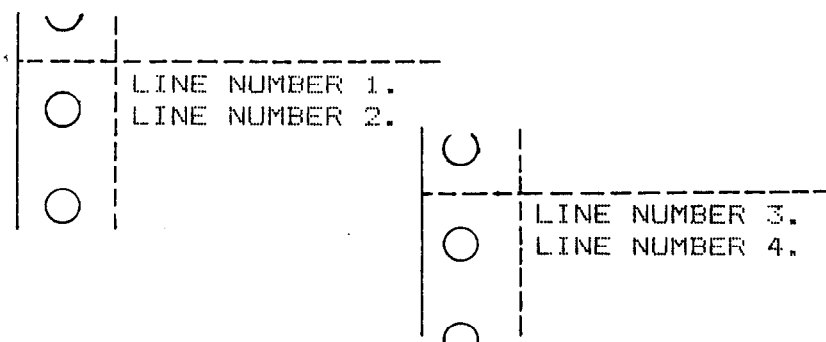
Now that we have seen how to control line spacing, we can go on to page control — positioning the printing on the page and adjusting the paper length.

### ■ Form feed

The simplest forms control code is the *form feed*. Form feed (or <FF>) is CHR\$(12) and causes the printer to move the paper to the top of the next sheet. Try it by changing lines 40 and 50 to this:

```
40 ' form feed
50 LPRINT CHR$(12);
```

Before you run the program, turn your printer off and adjust the paper so that the top of the sheet is even with the top of the ribbon guide on the print head, then turn the printer back on. If you don't remember how to do this, review Chapter 2. When you run the program, the results will look like this:



The form feed (CHR\$(12)) in line 50 caused the printer to move to the top of a new page before printing the last two lines.

### ■ Reverse form feed

Just as your printer can perform a reverse line feed, it can do a reverse form feed. This code moves the paper so that the print head is positioned at the top of the current page. This can be us-

ed, for example, to print text in a multi-column magazine format; print the first column, then reverse form feed back to the top of the page to start the second column. The code for reverse form feed is easy to remember: `<ESC> <FF>`.

**Table 4-3**  
**Form feed commands**

Function	Control code
Advance paper to top of next page	CHR\$(12)
Reverse paper to top of current page	<ESC>CHR\$(12)

#### ■ Changing the page length

You may have some computer forms that you wish to use with this printer that are not 11 inches high. That's no problem, because you can tell your printer how high the forms are that you are using. There are two commands for doing this, shown in this table.

**Table 4-4**  
**Form length control**

Function	Control code
Set the page length to <i>n</i> lines	<ESC>"C"CHR\$( <i>n</i> )
Set the page length to <i>n</i> inches	<ESC>"C"CHR\$(0)CHR\$( <i>n</i> )

Let's set up a 7 inch high form length, which is typical of many computer checks. The following program will do it.

```
NEW
10 ' Demo of variable form lengths
20 LPRINT CHR$(27);"C";CHR$(0);CHR$(7);
30 LPRINT "PAY TO THE ORDER OF:"
40 LPRINT CHR$(12);
50 LPRINT "PAY TO THE ORDER OF:"
60 END
```

This program should print "PAY TO THE ORDER OF:" twice, and they should be 7 inches apart. Line 20 sets the form length to 7 inches. After line 30 prints, line 40 sends a form feed

advance the paper to the top of the next form. Line 50 then prints its message.

After you have run this program, turn off the printer and adjust the top of form position. When you turn the printer back on the page length will reset to its normal setting (usually 11 inches).

## TOP AND BOTTOM MARGINS

Many programs that you use a printer don't keep track of where they are printing on the page. This causes a problem when you get to the bottom of a page because these programs just keep on printing, right over the perforation. This makes it very hard to read, especially if a line happens to fall right on the perforation. And if you separate the pages then you are really in trouble.

Of course your printer has a solution to this predicament. This printer can keep track of the position on the page, and advance the paper so that you won't print too near the perforation. There are two commands to do this. One controls the space at the top of the page and the other controls the space at the bottom of the page. The control codes are given in the following table.

**Table 4-5**  
**Top and bottom margin commands**

<b>Function</b>	<b>Control code</b>
Set top margin to $n$ lines	$\langle \text{ESC} \rangle "r" \text{CHR}\$(n)$
Set bottom margin to $n$ lines	$\langle \text{ESC} \rangle "N" \text{CHR}\$(n)$
Cancel top and bottom margins	$\langle \text{ESC} \rangle "O"$

In both cases the value of  $n$  tells your printer how many lines to skip, although there is a slight difference in the usage. When you set the top margin with  $\langle \text{ESC} \rangle "r" \text{CHR}\$(n)$ , the value of  $n$  tells the printer what line to start printing on. When you set the bottom margin with  $\langle \text{ESC} \rangle "N" \text{CHR}\$(n)$ , the value of  $n$  tells the printer how many blank lines should be left at the bottom of the page.

Let's try a simple application to see how these margins work. Enter this program, which will print 150 lines *without* top and bottom margins.



```
10 ' Demo of top and bottom margins
60 LPRINT CHR$(12);    ':' form feed
70 FOR I=1 TO 150
80 LPRINT "THIS IS LINE";I
90 NEXT I
110 LPRINT CHR$(12)    ':' form feed
120 END
```

When you run this program it will print 150 lines right down the page and across the perforations. When it's done line 110 sends a form feed to advance to the top of the next page. Look at the lines that have printed near the perforations. Separate the sheets and see if any of the lines have been torn in half. These are the problems that the top and bottom margins will solve.

Now add the following lines to your program. (Don't forget the semicolons or you won't get quite the same results that we did.)

```
20 ' Leave 6 blank lines at the bottom of page
30 LPRINT CHR$(27);"N";CHR$(6);
40 ' Start top of page at line 6
50 LPRINT CHR$(27);"r";CHR$(6);
100 LPRINT CHR$(27);"O"; ' clear top and bottom
    margins
```

Now when you run the program, your printer skip the first six lines and the last six lines on each page. Always send a form feed after setting the top margin, or it will not work on the first page printed. That's because the top margin only takes effect after a form feed.

Line 50 sets the top margin, line 30 sets the bottom margin, and line 100 clears both margins when we are done.

○  
○  
○  
○ THIS IS LINE 1  
○ THIS IS LINE 2  
○ THIS IS LINE 3  
○ THIS IS LINE 4  
○ THIS IS LINE 5  
○ THIS IS LINE 6  
○ THIS IS LINE 7  
○ THIS IS LINE 8  
○ THIS IS LINE 9  
○ THIS IS LINE 10

○ THIS IS LINE 47  
○ THIS IS LINE 50  
○ THIS IS LINE 51  
○ THIS IS LINE 52  
○ THIS IS LINE 53  
○ THIS IS LINE 54  
○ THIS IS LINE 55

○  
○  
○ THIS IS LINE 56  
○ THIS IS LINE 57  
○ THIS IS LINE 58  
○ THIS IS LINE 59  
○ THIS IS LINE 60  
○ THIS IS LINE 61  
○ THIS IS LINE 62  
○ THIS IS LINE 63  
○ THIS IS LINE 64

○ THIS IS LINE 100  
○ THIS IS LINE 104  
○ THIS IS LINE 105  
○ THIS IS LINE 106  
○ THIS IS LINE 107  
○ THIS IS LINE 108  
○ THIS IS LINE 109  
○ THIS IS LINE 110

○  
○  
○ THIS IS LINE 111  
○ THIS IS LINE 112  
○ THIS IS LINE 113  
○ THIS IS LINE 114  
○ THIS IS LINE 115  
○ THIS IS LINE 116  
○ THIS IS LINE 117  
○ THIS IS LINE 118  
○ THIS IS LINE 119  
○ THIS IS LINE 120  
○ THIS IS LINE 121

## SETTING LEFT AND RIGHT MARGINS

The left and right margins of this printer work just like a typewriter — once they are set all the printing is done between them. The commands to set the margins are given in the following table:

**Table 4-6**  
**Left and right margin commands**

Function	Control code
Set left margin at column <i>n</i>	⟨ESC⟩“I”CHR\$( <i>n</i> )
Set right margin at column <i>n</i>	⟨ESC⟩“Q”CHR\$( <i>n</i> )
Set left margin at column <i>n1</i> and right margin at column <i>n2</i>	⟨ESC⟩“X”CHR\$( <i>n1</i> ) CHR\$( <i>n2</i> )

Try setting the margins with this program:

```

10 ' Demo of margins
20 GOSUB 70
30 LPRINT CHR$(27);"I";CHR$(10);
40 LPRINT CHR$(27);"Q";CHR$(70);
50 GOSUB 70
60 END
70 FOR I=1 TO 80
80 LPRINT "X";
90 NEXT I
100 LPRINT
110 RETURN

```

The first thing that this program does is to branch to the subroutine that starts in line 70 . This subroutine prints 80 X's in a row. The first time that the subroutine is used, all the X's fit in one line. Then line 30 sets the left margin to 10, and line 40 sets the right margin to 70. Once again the subroutine is used, but this time the X's won't all fit on one line since there is now only room for 60 characters between the margins.

Run the program. The results will look like this:

```

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

```

When you want to reset the margins to the default values, you have two choices. You can either turn the printer off and back on, or you can set margin values equal to the default values. This means that you should set a left margin of 0 and right margin of 80 in pica pitch.

If you change the pitch of your printing after you set your margins, the margins will not change. They stay at the same place on the page. So if you set the margins to give you 65 columns of printing when you are using pica type, then you change to elite type you will have room for more than 65 columns of elite printing between the margins.

## HORIZONTAL AND VERTICAL TABS

Suppose you need to move across the page to a certain position several times in a document. It's not much fun to type in space after space. And you don't have to — you can "tab" your way across the page.

Your printer's tabs are like those on a typewriter, but much more powerful. You have both horizontal and vertical tabs which can be used for both text and graphics — and they're really handy for indenting paragraphs and making tables.

### ■ Horizontal tabs

Horizontal tabs are set automatically every eight positions. To move the print head to the next tab position, send CHR\$(9).

Try this program to see how the default tabs work.

```
10 ' Demo of horizontal tabbing
20 LPRINT "ONE";CHR$(9);"TWO";CHR$(9);"THREE";
   CHR$(9);"FOUR"
```

Here's what you should get—

```
ONE      TWO      THREE     FOUR
```

Even though the words are different length, they are spaced out evenly by the horizontal tabs.

Now add the following line to your program to set different horizontal tabs:

```
15 LPRINT CHR$(27);"D";CHR$(7);CHR$(14);CHR$(21);
    CHR$(0)
```

⟨ESC⟩“D” is the command to begin setting horizontal tabs. It must be followed by characters representing the positions that you want the tabs set. In our program we are setting tabs in columns 7, 14, and 21. The CHR\$(0) at the end ends the string of tabs. In fact, any character that is not greater than the previous one will stop setting tabs. This means that you must put all your tab values in order, from least to greatest, or they won’t all get set.

When you run the program now it produces this:

```
ONE      TWO      THREE     FOUR
```

The words are now closer together, but still evenly spaced. Turn your printer off and on again to reset the default tabs.

**Table 4-7**  
**Horizontal tab commands**

Function	Control code
Advance to next tab position	CHR\$(9)
Set tabs at $n1$ , $n2$ , etc.	⟨ESC⟩“D”CHR\$( $n1$ ) CHR\$( $n2$ ).....CHR\$(0)

#### ■ One-time horizontal tabs

Suppose you need to move to a position across the page, but you only need to do it once. It doesn’t make much sense to set up a tab to use only one time. There must be an easier way — and of course there is.

The solution is called a *one-time tab*. Table 4-8 show the commands.

**Table 4-8**  
**One-time horizontal tabs**

Function	Control code
Absolute horizontal tab	⟨ESC⟩ “\$” CHR\$( $n1$ ) CHR\$( $n2$ )
Relative horizontal tab	⟨ESC⟩ “\” CHR\$( $n1$ ) CHR\$( $n2$ )

The absolute horizontal tab command moves the print head to a specified position on the page. The position that you want the print head to move to (measured from the left margin) is specified by the values of  $n1$  and  $n2$  in 1/60-inch units using the formula  $n1 + n2 \times 256$ .

Try this program to see how this works.

```
10 ' Demo of absolute horizontal tabs
20 FOR I=60 TO 70 STEP 2
30 LPRINT I;"+++++";
40 LPRINT CHR$(27);"$";CHR$(I);CHR$(0);
50 LPRINT "Horizontal tab."
60 NEXT I
70 END
```

In this program, the print head is positioned before the "H" in "Horizontal" is printed.

```
60 +++++ Horizontal tab.
62 +++++ Horizontal tab.
64 +++++ Horizontal tab.
66 +++++ Horizontal tab.
68 +++++ Horizontal tab.
70 +++++ Horizontal tab.
```

The relative horizontal tab command can move the print head right from the current position. The formula for calculating how much the print head moves is the same as in the absolute horizontal tab command. However, the units by which the print head actually moves vary in 1/120-inch units.

Try this program to see how this works.

```
10 FOR I=1 TO 3
20 LPRINT "Relative";
30 LPRINT CHR$(27);"\";CHR$(I*20);CHR$(0);
40 LPRINT "Horizontal";
50 LPRINT CHR$(27);"\";CHR$(I*10);CHR$(0);
60 LPRINT "Tab."
70 NEXT I
```

```
80 LPRINT
90 END
```

```
Relative Horizontal Tab.
Relative Horizontal Tab.
Relative Horizontal Tab.
```

### ■ Vertical tabs

Vertical tabs have the same kinds of uses that horizontal tabs do — they just work in the other direction. Horizontal tabs allow you to reach a specific column on the page no matter where you start from. Vertical tabs are the same. If you have a vertical tab set at line 20, a *vertical tab* (or <VT>) will move you to line 20 whether you start from line 5 or line 19.

The vertical tab is *not* set at the power-on default. If you send a CHR\$(11), which is the ASCII code for <VT>, before we have set up tabs advance the paper one line. Enter this program to see how this works.

```
10 ' Demo of vertical tabs
40 LPRINT CHR$(11);"FIRST TAB."
50 LPRINT CHR$(11);"SECOND TAB."
60 LPRINT CHR$(11);"THIRD TAB."
70 LPRINT CHR$(11);"FOURTH TAB."
```

Now, let's set some vertical tabs of our own. Add these lines to the program:

```
20 LPRINT CHR$(27);"B";CHR$(10);CHR$(20);
30 LPRINT CHR$(40);CHR$(50);CHR$(0);
```

<ESC>"B" is the command to set vertical tabs. Like the horizontal tab setting command, tab positions must be defined in ascending order. Our example sets vertical tabs at lines 10, 20, 40 and 50. Then the CHR\$(11) in each of the following lines advances the paper to the next vertical tab. The printout is shown below.

FIRST TAB.

SECOND TAB.

THIRD TAB.

FOURTH TAB.



Add one more line to the program to demonstrate one more feature of vertical tabs.

```
80 LPRINT CHR$(11);"FIFTH TAB."
```

Now when you run the program the first page looks just like before, but line 80 sends one more <VT> than there are tabs. This doesn't confuse your printer — it advances the paper to the *next* tab position which happens to be the first tab position on the next page. That's nice, isn't it?

**Table 4-9**  
**Vertical tab commands**

Function	Control code
Advance paper to next tab position	CHR\$(11)
Set vertical tabs at <i>n1</i> , <i>n2</i> , etc.	<ESC>"B"CHR\$( <i>n1</i> ) CHR\$( <i>n2</i> ).....CHR\$(0)

#### ■ Vertical tab channels

Vertical tab channels are especially helpful in two situations. The first occurs when you are writing a program to accompany a preprinted form that can accommodate various types of responses. The second occurs when you create a multipage form or report with different vertical tabs on each page.

**Table 4-10**  
**Vertical tab channel commands**

Function	Control code
Set vertical tabs at <i>n1</i> , <i>n2</i> , etc. as channel <i>n0</i>	<ESC>"b"CHR\$( <i>n0</i> ) CHR\$( <i>n1</i> ) CHR\$( <i>n2</i> ).....CHR\$(0)
Select vertical channel <i>n0</i>	<ESC>"f"CHR\$( <i>n0</i> )

You can store up to eight channels of tab stops. They are numbered from 0 to 7. If you have already stored a set using <ESC>"B" command, your printer has labelled it as channel 0.

Try this program how to see the vertical tab channels work.

```

10 ' Demo of vertical tab channels
20 LPRINT CHR$(27);"b";CHR$(1);CHR$(10);CHR$(20);
   CHR$(0);
30 LPRINT CHR$(27);"b";CHR$(2);CHR$(15);CHR$(25);
   CHR$(0);
40 LPRINT CHR$(27);"b";CHR$(3);CHR$(17);CHR$(28);
   CHR$(0);
50 ' Use vertical tab channels
60 FOR I=1 TO 3
70 LPRINT "TOP OF FORM"
80 LPRINT CHR$(27);"/";CHR$(I);
90 LPRINT CHR$(11);
100 LPRINT "1ST TAB OF CHANNEL";I
110 LPRINT CHR$(11);
120 LPRINT "2ND TAB OF CHANNEL";I
130 LPRINT CHR$(12);
140 NEXT I
150 LPRINT CHR$(27);"@"
160 END

```

When you run this program you should get like this.

TOP OF FORM	TOP OF FORM	TOP OF FORM
1ST TAB OF CHANNEL 1		
	1ST TAB OF CHANNEL 2	1ST TAB OF CHANNEL 3
2ND TAB OF CHANNEL 1		
	2ND TAB OF CHANNEL 2	2ND TAB OF CHANNEL 3

In this program we set tabs at 10 and 20 in channel 1 in line 20. In line 30 we set tabs 15 and 25 in channel 2, and in line 40 we set tabs 17 and 28 in channel 3.

Because the channels are stored, you must make the printer to recall one before you use it, so we used `<ESC>"I"CHR$(n0)` in line 80.

## CENTERING AND ALIGNING TEXT

Text can be arranged in any of three formats: left aligned (normal printing with "ragged" right margin), centered between the margins, or right aligned. These are selected by the following commands.

**Table 4-11**  
**Aligning commands**

Function	Control code
Left-aligned printing	<code>&lt;ESC&gt;"a"CHR\$(0)</code>
Centered printing	<code>&lt;ESC&gt;"a"CHR\$(1)</code>
Right-aligned printing	<code>&lt;ESC&gt;"a"CHR\$(2)</code>

Try this program to see how easy it is.

```

10 ' Demo of aligning and centering
20 LPRINT CHR$(27);"I";CHR$(20);
30 LPRINT CHR$(27);"Q";CHR$(60);
40 LPRINT CHR$(27);"a";CHR$(0);
50 LPRINT "THIS LINE IS LEFT-ALIGNED."
60 LPRINT CHR$(27);"a";CHR$(1);
70 LPRINT "THIS LINE IS CENTERED."
80 LPRINT CHR$(27);"a";CHR$(2);
90 LPRINT "THIS LINE IS RIGHT-ALIGNED."

```

When you run this program, you should get like this:

```

THIS LINE IS LEFT-ALIGNED.
      THIS LINE IS CENTERED.
                THIS LINE IS RIGHT-ALIGNED.

```

---

---

# CHAPTER 5

## SPECIAL FEATURES OF THE PRINTER

---

---

Subjects we'll cover in Chapter 5 include—

- **Printer's bell;**
- **Master reset;**
- **Uni-directional printing;**
- **International character sets;**
- **Printing BIG characters;**
- **The optional sheet feeder;**
- **Macro instruction;**
- **Reading a hex dump.**

In the previous chapters we have learned about several groups of control codes. In this chapter we will look at more control codes. These codes don't fit neatly into any of the groupings that we have studied, but they add a lot of capability to your printer. So here goes.

■ Now hear this

You may have heard the printer's *bell* if you have ever run out of paper. And you may have wondered why it's called a bell when it *beeps* instead of ringing! It's a long story that goes back to the early days of computers, when teletype machines were used for computer terminals. These mechanical marvels had a bell in them that could be heard for blocks. This bell was used to signal the operator that somethings needed attention. The code that the computer sent to the teletype machine to ring the bell was, reasonably enough, called a *bell code*. Well the name *bell code* is still with us, even if the bell has changed to a beeper, and a lot of people still call the beeper a bell, even if it doesn't sound like one. So with our trivia lesson out of the way, let's see how we can "ring the bell."

The code to sound the "bell" is `CHR$(7)`, which is ASCII code 7 or `<BEL>`. Any time your printer receives this code it will

sound the bell for a quarter of a second. This can be used to remind an operator to change the paper or to make another adjustment to the printer.

You can try this by typing:

```
LPRINT CHR$(7)
```

### ■ Resetting the printer

Up to now when we wanted to reset the printer to the power on condition we have had to either turn the printer off and then on again, or to send the specific codes that reset the particular features. There is an easier way. The control code  $\langle \text{ESC} \rangle$  “@” will reset all of the printer’s features to the power on condition (as determined by the DIP switches), with two exceptions. Those exceptions are that  $\langle \text{ESC} \rangle$  “@” will not erase any characters that you have stored in the printer’s RAM memory (Chapter 5 tells you how to create your own characters), and it won’t erase the macro if you have one stored in the printer’s RAM (this chapter will tell you how to create a macro).

In addition, if you set the “Panel” mode, “Italic” mode, or “Italic and Panel” mode by the control panel settings at the power on, these functions will be remain with this control code.

### ■ Putting your printer to sleep

You know how to put your printer *off line* with the On Line key on the control panel. Your printer has another *off line* state that can be controlled from your computer. When you turn the printer *off line* from your computer, this printer will ignore anything that you send it, except for the code to *go on line* again. CHR\$(19) is the code to turn your printer off line; CHR\$(17) turns your printer to on line status.

### ■ Printing the bottom of the sheet

Sometimes when you are using individual sheets of paper you may want to print near the bottom of a sheet. The paper-out detector usually stops the printer when you are about 1 inch from the bottom of the sheet. This is notify you if you are running out of continuous paper.

Your printer has the ability to print right to the bottom of the sheet. You can disable the paper-out detector so that it doesn’t

stop the printer. This will allow you to print to the end of the sheet, and even beyond if you are not careful. The codes to control the paper-out detector, along with the other codes that we have just learned are in the following table.

**Table 5-1**  
**Some miscellaneous commands**

Function	Control code
Sound bell	CHR\$(7)
Master reset	<ESC> "@"
Off line	CHR\$(19)
On line	CHR\$(17)
Paper-out detector off	<ESC> "8"
Paper-out detector on	<ESC> "9"
Move print head back one space	CHR\$(8)
Delete last character sent	CHR\$(127)
Cancel text in print buffer	CHR\$(24)
Print "slash zero"	<ESC> "~"1
Print "normal zero"	<ESC> "~"0
Immediate-print on	<ESC> "i"1
Immediate-print off	<ESC> "i"0

#### ■ Backspace, delete, and cancel text

Backspace (CHR\$(8)) "backs up" the printhead so that you can print two characters right on top of each other. Each time your printer receives a backspace it moves the printhead one character to the left, instead of to the right. You can *strike over* multiple letters by sending more than one backspace code.

Delete (CHR\$(127)) also "backs up" one character, but then it "erases" the previous character (it's erased from your printer's buffer, not from the paper).

Cancel text (CHR\$(24)) deletes all the text in the print buffer; that is, in the line before the delete text command. Since your printer prints one line of text at a time, only that line will be deleted.

The following program shows how these codes work.

```
10 LPRINT "BACKSPACE DOES NOT";
20 LPRINT CHR$(8);CHR$(8);CHR$(8);
30 LPRINT "=== WORK"
40 LPRINT "DELETE DOES NOT";
50 LPRINT CHR$(127);CHR$(127);CHR$(127);
```

```
60 LPRINT "WORK"  
70 LPRINT "CANCEL LINE";  
80 LPRINT CHR$(24);  
90 LPRINT "DOES NOT WORK"
```

Here is what this program will print:

```
BACKSPACE DOES NOT WORK  
DELETE DOES WORK  
DOES NOT WORK
```

The backspace codes in line 20 move the printhead a total of three spaces to the left so that the first part of line 30 will overprint the word "NOT". The delete codes in line 50 "erase" the three letters in the word "NOT" so that it doesn't even print.

In line 80, CHR\$(24) deletes the words in line 70. The semicolon at the end of line 70 prevents a line feed from causing that line to print before the printer receives the CHR\$(24) code. The text in line 90 prints as it normally would because it is after CHR\$(24).

### ■ Printing zeroes

Believe it or not, there are two types of zeroes. There is of course the type we use every day — 0 — and this is what your printer will print if you don't do anything.

The other type is used almost exclusively in computers and engineering. It is called the "slash zero" and is written like this — Ø. The line through the number is supposed to prevent you from misreading it as the letter "O". Back before high-quality printers were available, this was a good idea but you really have no need for it (although you may want to use the slash zero for special effect).

### ■ Immediate-print

This printer can print at fine rate of 120 characters per second. But it will also print more slowly at the speed of your typing. In *immediate-print* mode, the printhead prints one character at a time, as you send it. This printer also moves the paper up so that you can see the current line and then down to continue printing.

You can turn immediate-print mode on with  $\langle \text{ESC} \rangle$  "i" 1. But before looking at it, let's review the normal operation of the print buffer. Enter this program.

```
20 A$="" : INPUT " TYPE A CHARACTER ",A$
30 IF A$="" THEN 50
40 LPRINT A$; : GOTO 20
50 LPRINT : LPRINT CHR$(27);"@"
```

Now type several characters, and after each press the RETURN key. True to form, the printer just stuffs the characters into its buffer while it waits for a carriage return code. (In this program the RETURN key doesn't send a carriage return code.) To end this program and print the contents of the buffer, press RETURN alone.

Now add this line:

```
10 LPRINT CHR$(27);"i1";
```

And RUN the program. Your printer responds to your typing — immediately.

When you are finished, press RETURN alone.

### ■ Adjusting the width of space between characters

This printer provides a command that adjusts the space between the NLQ characters that it prints when the DIP switch 1-6 is set on. The  $\langle \text{ESC} \rangle$  CHR\$(32) CHR\$(*n*) command adds blank space between the characters. The units of space that are added vary in 1/60-inch units.

Try this program to see how this works.

```
10 ' Demo of adjusting spaces
20 LPRINT CHR$(27);"x1";
30 FOR I=10 TO 1 STEP -2
40 LPRINT CHR$(27);" ";CHR$(I);
50 LPRINT "This line is added";I;"spaces."
60 NEXT I
70 LPRINT "This line is Normal space."
80 END
```



This line is added 10 spaces.  
 This line is added 8 spaces.  
 This line is added 6 spaces.  
 This line is added 4 spaces.  
 This line is added 2 spaces.  
 This line is Normal space.

This command can be used to produce micro-justification, which is a method of justifying lines by increasing the space between each character.

### ■ Uni-directional printing

Uni-directional printing is a big word that means *printing in one direction only*. Your printer normally prints when the print-head is moving in both directions. But once in a while you may have an application where you are more concerned about how the vertical lines align than with how fast it prints. This printer lets you make this choice. The table below shows the commands for controlling how this printer prints.

**Table 5-2**  
**Printing directin commands**

Function	Control code
Print in one direction	<ESC> "U"1
Print in both directions	<ESC> "U"0
One time print in one direction	<ESC> "<"

Try this program to see the difference that printing in one direction makes.

```
10 ' Demo of uni-directional printing
20 LPRINT CHR$(27);"1";
30 FOR I=1 TO 10
40 LPRINT "|"
50 NEXT I
60 LPRINT : LPRINT
70 LPRINT CHR$(27);"U1";
80 FOR I=1 TO 10
90 LPRINT "|"
100 NEXT I
110 LPRINT CHR$(12);CHR$(27);"@"
```

Here is what you will get. The top line is printed bi-directionally, and the bottom is printed uni-directionally. You will have to look hard because there isn't much difference.



Let's analyze the program. Line 20 sets the line spacing to 7/72 of an inch so that the characters that we print will touch top to bottom. Lines 30 ~ 50 print 10 vertical line characters. Then line 70 sets one-direction printing and the vertical lines are printed again. Finally line 110 sends a form feed to advance the paper to the top of a new page, and then uses the master reset to restore the printer to the power on condition.

You can also set the printer to print in one direction for one line only by using the `<ESC>“<”` command. This command immediately moves the printhead to the left margin and then prints the remainder of the line from left to right.

#### ■ The seven bit dilemma

Certain computers (but not the IBM-PC fortunately!) don't have the capability to send eight bits on their parallel interface. They can only send seven bits. This would make it impossible for these computers to use this printer's block graphics characters and special symbols if our engineers hadn't thought of a solution. (All of these characters have ASCII codes greater than 127 which means that the eighth bit must be on to use them.) The solution lies in the three control codes given in the following table:

**Table 5-3**  
**Eighth bit controls**

Function	Control code
Turn the eighth bit ON	<ESC> “)”
Turn the eighth bit OFF	<ESC> “=”
Accept the eighth bit “as is” from the computer	<ESC> “#”

### ■ Block graphics characters and special symbols

Besides the upper and lower case letters and symbols that we are by now familiar with, your printer has a whole different set of characters that are for special uses. These characters include block graphics for drawing forms and graphs, and special symbols for mathematical, engineering and professional uses. The special characters are included in two character sets. The character set you normally use with IBM mode is called character set #1. The special characters are printed out when you send ASCII codes 160~255 to the printer.

Your printer also offers character set #2 which is almost the same as character set #1 except for the addition of ASCII codes 3~6, 21, and 128~159. Character set #2 is selected with <ESC> “6”; to go back to character set #1, use <ESC> “7”.

You can also specify the power-on default character set by setting DIP switch 1-7 on for character set #1 and off for character set #2. The following program will print out all of the graphics characters available:

```

10 LPRINT CHR$(27);"0";
20 LPRINT CHR$(27);"6";
30 FOR J=3 TO 6
40 LPRINT " ";J;CHR$(J);CHR$(9);
50 NEXT J
60 LPRINT " 20 ";CHR$(20)
70 LPRINT
80 LPRINT " 21 ";CHR$(21);CHR$(9);
90 LPRINT " 26 ";
100 LPRINT CHR$(26)
120 LPRINT
130 FOR J=128 TO 254 STEP 5
140 FOR I=J TO J+4
150 IF I>254 THEN 170
160 LPRINT I;CHR$(I);CHR$(9);
170 NEXT I
180 LPRINT : LPRINT
190 NEXT J

```

3 ♡	4 ♦	5 ♣	6 ♠	20 ♠
21 8	26 +			
128 9	129 ù	130 é	131 à	132 ä
133 à	134 á	135 ç	136 ë	137 è
138 è	139 ì	140 î	141 ï	142 Å
143 Å	144 é	145 æ	146 œ	147 ö
148 ö	149 ó	150 û	151 ù	152 ý
153 ö	154 ü	155 4	156 £	157 ¥
158 R	159 f	160 á	161 i	162 ó
163 ú	164 ñ	165 ñ	166 @	167 ©
168 ¢	169 ¢	170 ¢	171 ½	172 ¼
173 i	174 *	175 *	176 ☺	177 ☻
178 ☼	179	180 †	181 ‡	182 ††
183 ¶	184 †	185 ††	186 ††	187 ††
188 ¶	189 ¶	190 †	191 †	192 L
193 †	194 †	195 †	196 -	197 †
198 †	199 ††	200 ††	201 ††	202 ††
203 ††	204 ††	205 =	206 ††	207 ±
208 ††	209 ††	210 ¶	211 ††	212 ††
213 †	214 ¶	215 ††	216 ††	217 †
218 †	219 ■	220 ■	221 ■	222 ■
223 ■	224 α	225 β	226 Γ	227 π
228 Σ	229 σ	230 μ	231 τ	232 ϕ
233 θ	234 Ω	235 δ	236 ω	237 ø
238 €	239 Π	240 ≡	241 ±	242 ≥
243 ≤	244 †	245 J	246 ÷	247 ≈
248 °	249 ·	250 -	251 √	252 Π
253 ²	254 ■			

Figure 5-1. Character set #2

Figure 5-1 shows what this program will print. If your chart doesn't look like this because it has regular letters and numbers instead of the special symbols, then your computer is only using seven bits. You can get the correct printout by changing line 160 to this:

```
160 LPRINT I;CHR$(27);">";CHR$(I);CHR$(27);"=";
    CHR$(9);
```

*A note for the IBM-PC users:*

When you run this program, you cannot get the right-pointed arrow (CHR\$(26)) with the IBM-PC computers. This is because the IBM-PC does not send this code to the printer.

There is a solution to avoid this problem. Change lines 100 and 120 to the either set of the following lists.

```
100 O=INP(&H379) : IF O<128 THEN 100
110 OUT &H378,26 : OUT &H37A,5 : OUT &H37A,4
120 LPRINT : LPRINT
```

```
100 O=INP(&H3BD) : IF O<128 THEN 100
110 OUT &H3BC,26 : OUT &H3BE,5 : OUT &H3BE,5
120 LPRINT : LPRINT
```

So how are all of these strange characters used? Here is a short program that demonstrate how the graphics characters can be combined to create a figure: the 5 of clubs.

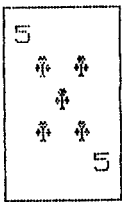
```
10 LPRINT CHR$(27);"6";
20 LPRINT CHR$(218);CHR$(196);CHR$(196);
    CHR$(196);CHR$(196);CHR$(196);CHR$(191)
30 LPRINT CHR$(179);CHR$(53);CHR$(32);CHR$(32);
    CHR$(32);CHR$(32);CHR$(179)
40 LPRINT CHR$(179);CHR$(32);CHR$(5);CHR$(32);
    CHR$(5);CHR$(32);CHR$(179)
50 LPRINT CHR$(179);CHR$(32);CHR$(32);CHR$(5);
    CHR$(32);CHR$(32);CHR$(179)
60 LPRINT CHR$(179);CHR$(32);CHR$(5);CHR$(32);
    CHR$(5);CHR$(32);CHR$(179)
70 LPRINT CHR$(179);CHR$(32);CHR$(32);CHR$(32);
    CHR$(32);CHR$(53);CHR$(179)
80 LPRINT CHR$(192);CHR$(196);CHR$(196);
    CHR$(196);CHR$(196);CHR$(196);CHR$(217)
90 LPRINT CHR$(27);"7"
```

If you have a 7-bit interface, add the following lines to the program given above:

```
15 LPRINT CHR$(27);">";  
95 LPRINT CHR$(27);"="
```

In this program line 10 selects character set #2 so that all the graphics characters can be used (including the "club" symbol). Then lines 20~80 print the 5 of clubs figure. Line 90 cancels character set #2 (which is the same as selecting character set #1).

Here is what this program prints:



#### ■ International character sets

Your printer is a multi-lingual printer for it can speak in eleven languages! Your printer changes languages by changing 12 characters that are different for the different languages. These sets of characters are called *international character sets*. The control codes to select the international character sets are given in Table 5-4.

**Table 5-4**  
**International character set commands**

Country	Control code
U.S.A	<ESC> "R"CHR\$(0)
France	<ESC> "R"CHR\$(1)
Germany	<ESC> "R"CHR\$(2)
England	<ESC> "R"CHR\$(3)
Denmark type I	<ESC> "R"CHR\$(4)
Sweden	<ESC> "R"CHR\$(5)
Italy	<ESC> "R"CHR\$(6)
Spain	<ESC> "R"CHR\$(7)
Japan	<ESC> "R"CHR\$(8)
Norway	<ESC> "R"CHR\$(9)
Denmark type II	<ESC> "R"CHR\$(10)

The characters that change are shown beneath their ASCII code in Table 5-5.

**Table 5-5**  
**International character sets**

Country	35	36	64	91	92	93	94	96	123	124	125	126
U.S.A	#	\$	@	[	\	]	^	'	{		}	~
France	#	\$	à	°	ç	§	^	'	é	ù	è	..
Germany	#	\$	§	Ä	Ö	U	^	'	ä	ö	ü	ß
England	£	\$	@	[	\	]	^	'	{		}	~
Demark type I	#	\$	@	Æ	Ø	Å	^	'	æ	ø	å	~
Sweden	#	¤	é	Ä	Ö	Å	U	é	ä	ö	å	ü
Italy	#	\$	@	°	\	é	^	ù	à	ò	è	ì
Spain	₧	\$	@	í	ñ	¿	^	'	..	ñ	}	~
Japan	#	\$	@	[	¥	]	^	'	{		}	~
Norway	#	¤	é	Æ	Ø	Å	U	é	æ	ø	Å	ü
Denmark type II	#	\$	é	Æ	Ø	Å	U	é	æ	ø	Å	ü

#### ■ Printing characters in the control code area

When you refer the Appendix B, you'll find many characters are printed in the control code area. (Remember that the low-order control codes are the ASCII codes 0 through 31 plus 127, and the high-order control codes are 128 through 159 plus 255.)

These codes don't normally print symbols on paper, rather they cause the printer to change modes. To make them print as normal symbols requires an extra command. For example, the





```

130 NEXT I
140 LPRINT CHR$(144);CHR$(145);
150 FOR I=149 TO 159
160 LPRINT CHR$(I);
170 NEXT I
180 LPRINT CHR$(27);"IO"
190 END

```

When you run this program you should get like this:

```

aèùòì"èßø"ÀÖÜäÜëé¥
aèùòì"èßø"ÀÖÜäÜëé¥

```

**Table 5-6**  
**Control code area commands**

Function	Control code
Printable code area expansion	<ESC>"6"
Control code area expansion	<ESC>"7"
Select undefined codes as characters	<ESC>"I"1
Cancel undefined codes as characters	<ESC>"I"0

### ■ Printing BIG characters

You can even enlarge your character sets for attention-grabbing headings or special effects. There are six commands you can use. Everything following any of them will be enlarged as shown below, until the cancel code is entered.

**Table 5-7**  
**Big character commands**

Function	Control code
Double-high enlarged print	<ESC>"h"CHR\$(1)
Quad-high enlarged print	<ESC>"h"CHR\$(2)
Double-high lower-half enlarged print	<ESC>"h"CHR\$(3)
Double-high upper-half enlarged print	<ESC>"h"CHR\$(4)
Quad-high lower-half enlarged print	<ESC>"h"CHR\$(5)
Quad-high upper-half enlarged print	<ESC>"h"CHR\$(6)
Cancel enlarged print	<ESC>"h"CHR\$(0)

Try this program to see the big characters.

```

10 ' Demo of BIG characters
20 LPRINT "THIS IS ";
30 LPRINT CHR$(27);"h";CHR$(1);
40 LPRINT "DOUBLE";
50 LPRINT CHR$(27);"h";CHR$(0);
60 LPRINT " SIZED PRINTING."
70 LPRINT
80 LPRINT "THIS IS ";
90 LPRINT CHR$(27);"h";CHR$(2);
100 LPRINT "QUAD";
110 LPRINT CHR$(27);"h";CHR$(0);
120 LPRINT " SIZED PRINTING."
130 END

```

When you run this program, you will get like this:

```

THIS IS  DOUBLE  SIZED PRINTING.
THIS IS  QUAD    SIZED PRINTING.

```

As you can see, when the big character command is used, the baseline for each character does not align. When you want to align the baseline, try this program:

```

10 ' Demo of aligning BIG characters
20 LPRINT "THIS IS ";
30 LPRINT CHR$(27);"j";CHR$(21);
40 LPRINT CHR$(27);"h";CHR$(1);
50 LPRINT "DOUBLE";
60 LPRINT CHR$(27);"h";CHR$(0);
70 LPRINT CHR$(27);"J";CHR$(18);
80 LPRINT " SIZED PRINTING."
90 LPRINT :LPRINT :LPRINT
100 LPRINT "THIS IS ";
110 LPRINT CHR$(27);"j";CHR$(63);
120 LPRINT CHR$(27);"h";CHR$(2);
130 LPRINT "QUAD";
140 LPRINT CHR$(27);"h";CHR$(0);
150 LPRINT CHR$(27);"J";CHR$(63);
160 LPRINT " SIZED PRINTING."
170 LPRINT :LPRINT
180 END

```

When you run this program, you will get like this:

THIS IS **DOUBLE** SIZED PRINTING.

THIS IS **QUAD** SIZED PRINTING.

### ■ The optional sheet feeder

The automatic sheet feeder is a handy option that feeds single cut sheets automatically. Work done on cut sheets looks better than done on computer paper, and you don't have to tear the "ears" off each sheet as you must with fan-fold paper.

The automatic sheet feeder feeds a new sheet automatically every time the printer receives or generates a form feed. Any time you wish, you can turn the auto-feed unit on and off by using control codes.

**Table 5-8**  
**Automatic sheet feeder commands**

Function	Control code
Select automatic feed mode	<ESC>CHR\$(25) CHR\$(4) or "((4))"
Cancel automatic feed mode	<ESC>CHR\$(25) CHR\$(0) or "((0))"
Insert paper	<ESC>CHR\$(25) CHR\$(1) or "((1))"
Eject paper	<ESC>CHR\$(25)"R" or "((R))"

When the automatic sheet feeder is installed, you must set the DIP switch 1-5 on to detect the paper-out condition.

In addition, following functions are ignored when the automatic sheet feeder is installed:

- Setting of the page length
- Top and bottom margins
- Vertical tab settings

### ■ The macro control code

The last of our group of miscellaneous codes is definitely not the least. It is a *user-defined* control code, called a *macro* control code. The term *macro* is from the jargones *macro-instruction*

which refers to an instruction that “calls,” or uses a group of normal instructions. In computer programming macro-instructions (which are similar to subroutines) save programmers a lot of time and effort. Your printer’s macro can save you a lot of time and effort also.

Here is how the printer’s macro works. You *define macro* by telling the printer what normal control codes are to be included in the macro. Then you can use the macro any time that you want and the printer will do all the things that you included the macro definition. You can include up to 16 codes in a single macro. You can even use the macro to store a frequently used word or phrase. There are two control codes for the macro: one to define it, and one to use it. They are given in the Table 5-9.

**Table 5-9**  
**Macro instruction commands**

Function	Control code
Define macro	<ESC>“+”... <i>(codes you include)</i> ... CHR\$(30)
Use macro	<ESC>“+”CHR\$(1)

To see how this works we can build a macro that will reset the printing style to normal, no matter what style it may be to start with. The following program will define a macro to do this.

```

10 LPRINT CHR$(27);"+";           'Start macro
20 LPRINT CHR$(27);"h";CHR$(0); 'Big character
   off
30 LPRINT CHR$(27);"!";CHR$(0); 'Select normal
   pica
40 LPRINT CHR$(27);"T";           'Super &
   subscripts off
50 LPRINT CHR$(27);"2";           'Set 1/6 inch
   line spacing
60 LPRINT CHR$(27);"a";CHR$(0); 'Left-aligned
   printing
70 LPRINT CHR$(30)                'End macro
   definition

```

As the comments in the program, we started to define macro in line 10. Line 20 cancels the big character printing. Line 30 sets the normal pica, and also this command cancels the propor-

tional pitch, condensed print, expanded print, boldface, emphasized, and the underlining. Line 40 cancels the superscripts and the subscripts. Line 50 sets the line spacing to 1/6 inch, and line 60 sets the left-aligned printing. Then, line 70 ends the macro definition. This printer will remember this macro until the power is turned off or until a new macro is defined. A macro can hold up to 16 bytes (characters) of information. The one that we defined contains thirteen.

Now that you have defined a macro, let's see how to use it. This program will print one line using several printing features. Then it "calls" the macro in line 60. When line 80 prints the style is "plain vanilla" because the macro has reset it.

```
10 LPRINT CHR$(27);"Q";CHR$(40);
20 LPRINT CHR$(27);"a";CHR$(2);
30 LPRINT CHR$(27);"-1";
40 LPRINT CHR$(27);"h";CHR$(1);
50 LPRINT "TESTING ABCD"
60 LPRINT CHR$(27);"+";CHR$(1);
70 LPRINT "TESTING ABCD"
80 END
```

TESTING ABCD

TESTING ABCD

### ■ Reading a hex dump

We've seen how to make a hex dump in Chapter 1, but it's not really clear what we can do with one. We need a little background first.

The BASIC in some computers changes ASCII codes before they send them to the printer. If you run into problem because of this, try this hex dump to check the ASCII codes.

First turn off the printer and run the following program. Hold down both the Paper Feed key and Mode key and turn on the printer.

```
10 FOR I=0 TO 255
20 LPRINT CHR$(I);
30 NEXT I
40 LPRINT
50 END
```

If your system passes the codes directly to the printer without changing them, you will get like this. (You can print out the last remaining line in the print buffer by putting the printer off line with the On Line key.)

```

00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  .....
10 11 12 13 14 15 16 17  18 19 1A 1B 1C 1D 1E 1F  .....
20 21 22 23 24 25 26 27  28 29 2A 2B 2C 2D 2E 2F  !"#%&'()*+,-./
30 31 32 33 34 35 36 37  38 39 3A 3B 3C 3D 3E 3F  0123456789:;<=>?
40 41 42 43 44 45 46 47  48 49 4A 4B 4C 4D 4E 4F  @ABCDEFGHIJKLMNO
50 51 52 53 54 55 56 57  58 59 5A 5B 5C 5D 5E 5F  PQRSTUVWXYZ[\]^_
60 61 62 63 64 65 66 67  68 69 6A 6B 6C 6D 6E 6F  `abcdefghijklnmo
70 71 72 73 74 75 76 77  78 79 7A 7B 7C 7D 7E 7F  pqrstuvwxyz{|}~.
80 81 82 83 84 85 86 87  88 89 8A 8B 8C 8D 8E 8F  .....
90 91 92 93 94 95 96 97  98 99 9A 9B 9C 9D 9E 9F  .....
A0 A1 A2 A3 A4 A5 A6 A7  AB A9 AA AB AC AD AE AF  .....
B0 B1 B2 B3 B4 B5 B6 B7  BB B9 BA BB BC BD BE BF  .....
C0 C1 C2 C3 C4 C5 C6 C7  CB C9 CA CB CC CD CE CF  .....
D0 D1 D2 D3 D4 D5 D6 D7  DB D9 DA DB DC DD DE DF  .....
E0 E1 E2 E3 E4 E5 E6 E7  EB E9 EA EB EC ED EE EF  .....
F0 F1 F2 F3 F4 F5 F6 F7  FB F9 FA FB FC FD FE FF  .....
0D 0A  ..

```

Most BASICs, however, are not quite that straight forward. For example, the IBM-PC prints like the following.

```

00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0A 0E  .....
0F 10 11 12 13 14 15 16  17 18 19 1B 1C 1D 1E 1F  .....
20 21 22 23 24 25 26 27  28 29 2A 2B 2C 2D 2E 2F  !"#%&'()*+,-./
30 31 32 33 34 35 36 37  38 39 3A 3B 3C 3D 3E 3F  0123456789:;<=>?
40 41 42 43 44 45 46 47  48 49 4A 4B 4C 4D 4E 4F  @ABCDEFGHIJKLMNO
50 51 52 53 54 55 56 57  58 59 5A 5B 5C 5D 5E 5F  PQRSTUVWXYZ[\]^_
60 61 62 63 64 65 66 67  68 69 6A 6B 6C 6D 6E 6F  `abcdefghijklnmo
70 71 72 73 74 75 76 77  78 79 7A 7B 7C 7D 7E 7F  pqrstuvwxyz{|}~.
80 81 82 83 84 85 86 87  88 89 8A 8B 8C 8D 8E 8F  .....
90 91 92 93 94 95 96 97  98 99 9A 9B 9C 9D 9E 9F  .....
A0 A1 A2 A3 A4 A5 A6 A7  AB A9 AA AB AC AD AE AF  .....
B0 B1 B2 B3 B4 B5 B6 B7  BB B9 BA BB BC BD BE BF  .....
C0 C1 C2 C3 C4 C5 C6 C7  CB C9 CA CB CC CD CE CF  .....
D0 D1 D2 D3 D4 D5 D6 D7  DB D9 DA DB DC DD DE DF  .....
E0 E1 E2 E3 E4 E5 E6 E7  EB E9 EA EB EC ED EE EF  .....
F0 F1 F2 F3 F4 F5 F6 F7  FB F9 FA FB FC FD FE FF  .....
0D 0A  ..

```

Notice that your printer is receiving decimal code 13 (hex 0D) is coming with hex 0A, which is really decimal 10. In addition, your printer does not receive decimal code 26 (hex 1A).

Your printer prints hex numbers 16 per line, with printing the characters on the right side. If it receives less than 16, it sits in a holding pattern, awaiting more data. Taking the printer off line

dumps the characters to the paper.

To debug a program quickly, just use the hex dump capability. Appendix B will help you translate the hex codes to ASCII equivalents.

---

---

# CHAPTER 6

## CREATING YOUR OWN CHARACTERS

---

---

Subjects we'll cover in Chapter 6 include —

- **Designing and printing your own characters;**
- **Designing proportional characters;**
- **Designing your own characters with NLQ.**

In the previous chapters of this manual you've learned how to control the printer to give dozens of different typefaces. By using various combinations of pitches, character weights, and font selections, you can create nearly any effect you want to in text. And with international character sets and the special text and big characters described in Chapter 5, you can print almost any character you think of.

But if “almost any character” isn't good enough for you, then it's a good idea you have this printer! With it you can actually create your own characters. As you'll see in this chapter, *download characters* can be used to print a logo, special characters for foreign languages, scientific and professional applications, or any other specific printing task.

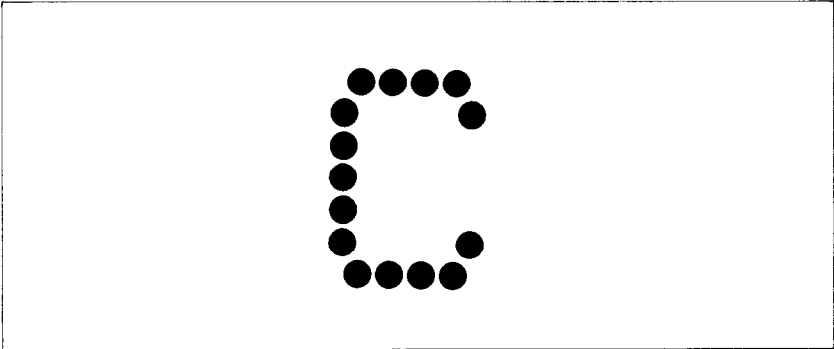
### **DOT MATRIX PRINTING**

In order to create download characters, you'll need some understanding of how dot matrix printers work. They're called “dot matrix” because each character is made up of a group of dots. Look closely at some printed characters produced by your printer and you will see the dots. Figure 6-1 shows how the letter “C” is formed by printing 15 dots.

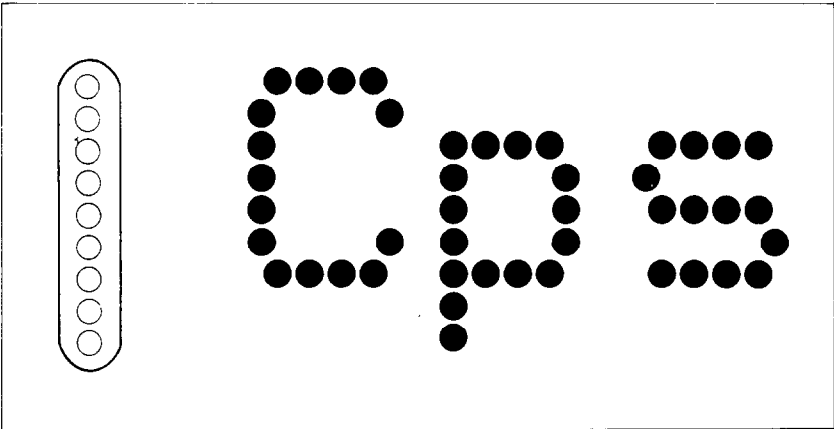
The printhead in this printer consists of nine wires stacked one atop the other. Figure 6-2 shows an enlarged schematic view of the front of the printhead, showing the ends of the wires and their relationship to the printed draft characters. As you can



see, the capital letters use the top seven wires of the printhead, and the descenders (such as the lower case “p” shown) use the bottom seven pins. As the printhead moves across the page (in either direction — that’s what is meant by bi-directional printing) it prints one column of dots at a time. Each time a dot is supposed to print an electromagnet inside the printhead causes the appropriate wire to strike the ribbon (making this printer an *impact* printer).



**Figure 6-1.** The letter “C” is created by printing 15 dots.



**Figure 6-2.** As the printhead moves across the page, each of the wires prints one row of dots.

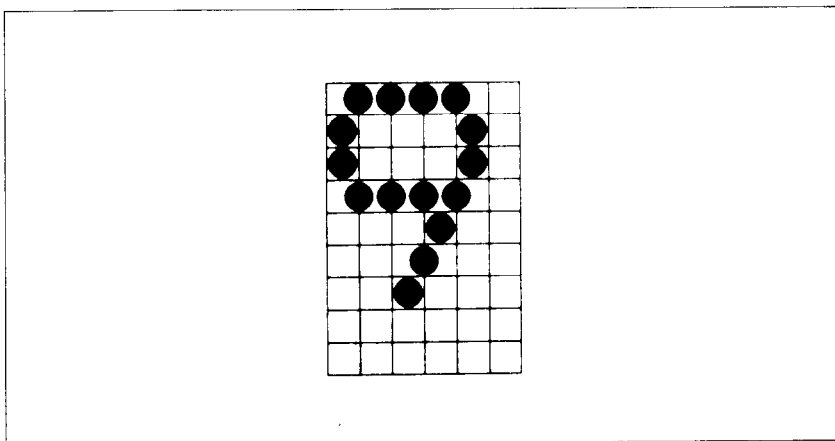
## THE PRINT MATRIX

All of the standard characters that this printer prints are formed from patterns of dots that are permanently stored in the printer’s *ROM* (read-only memory). This includes all of the standard ASCII characters, and special characters, the international

character sets, and the NLQ characters.

But there is another area of memory in this printer reserved for *user-defined* characters. These are characters that you can design and *download* into the printer. When download characters are defined they are stored in *RAM* (random access memory), which allows you to define or modify them at any time.

Each of these characters, whether it is from the standard character ROM or in download RAM, is constructed on a grid which is six “boxes” by nine “boxes” high. In addition, a dot can straddle any of the vertical lines. As an example, take a look at the enlarged “9” superimposed on the grid in Figure 6-3. As you can see, some dots are inside the boxes, and some are centered on the vertical lines. This, in effect, makes the character grid 11 dots wide by 9 dots high. To see how the rest of the characters in the standard character ROM are constructed, take a look at Appendix C.

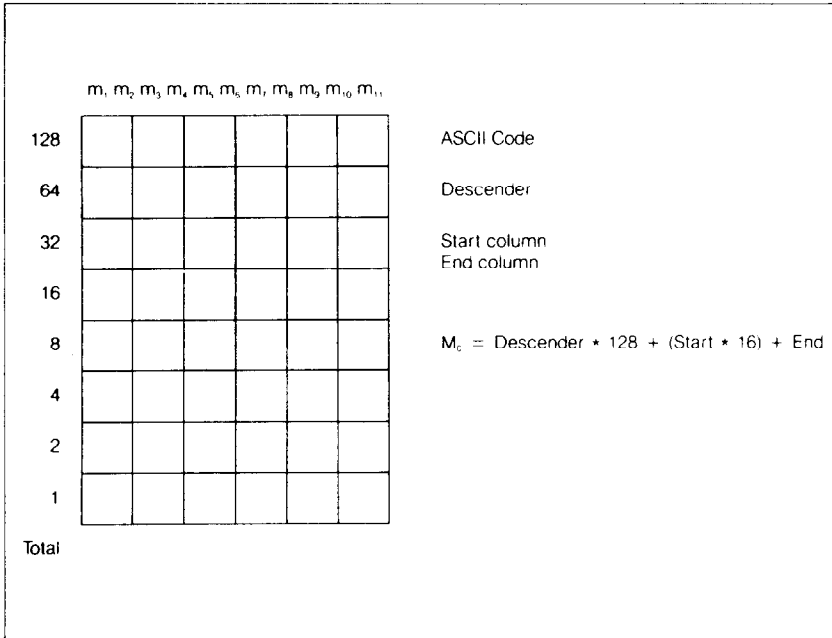


**Figure 6-3.** Dots can be inside boxes or straddle the vertical lines of the grid.

## DEFINING YOUR OWN CHARACTERS

You’ve seen how these characters are designed by using a grid to layout the dots. Now you can define characters exactly the same way. Make up some grids (photocopy Figure 6-4 if you wish) and get ready to be creative! (Just in case you are not feeling creative, and to make our explanations a little clearer, we’ll be using a picture of a chemist’s flask as an example of a draft

download character. You can see how we've laid it out in Figure 6-5. Later in this chapter we'll use this character to create a small graph.)



**Figure 6-4.** Use this grid (or one similar to it) to define your own draft characters.

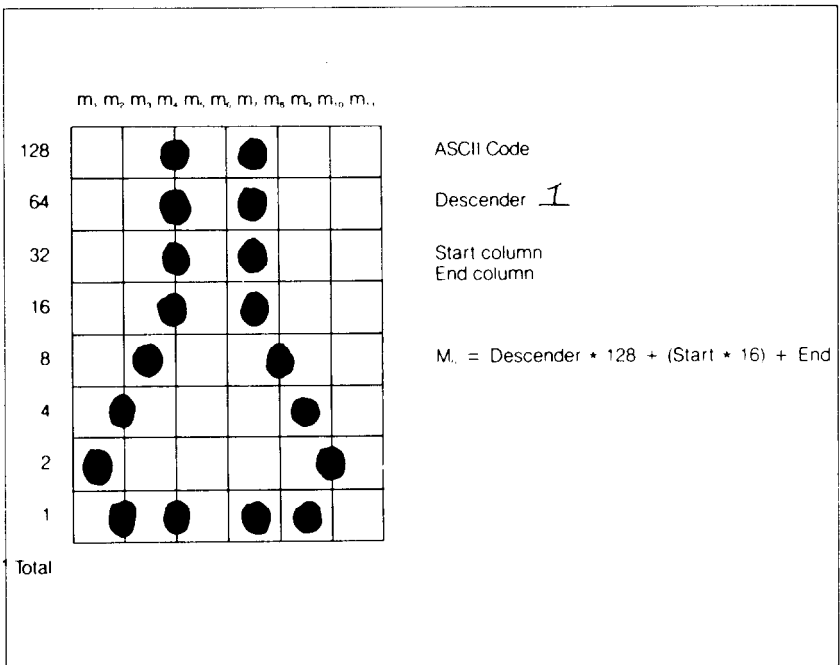
You'll notice that Figure 6-4 includes a lot of information around the grid. Don't be intimidated; we'll explain each item as we come to it in our discussion of defining and actually printing download characters. You may have noticed another difference between this grid and the one shown in Figure 6-3: it's only eight boxes high. Which leads us to ...

#### ■ Rule 1: Draft download characters are eight dots high

As you noticed in Figure 6-2, capital letters, most lowercase letters, and most special characters use only the top seven pins of the printhead. Draft download characters can go one better: they can use as many as eight of the nine wires in the printhead. So our grid is eight dots high.

It's also possible to use the bottom eight pins, just as the "g", "j", "p", "q", and "y" of the standard character sets do. These are called descenders (because the bottom of the character descends below the baseline of the rest of the characters).

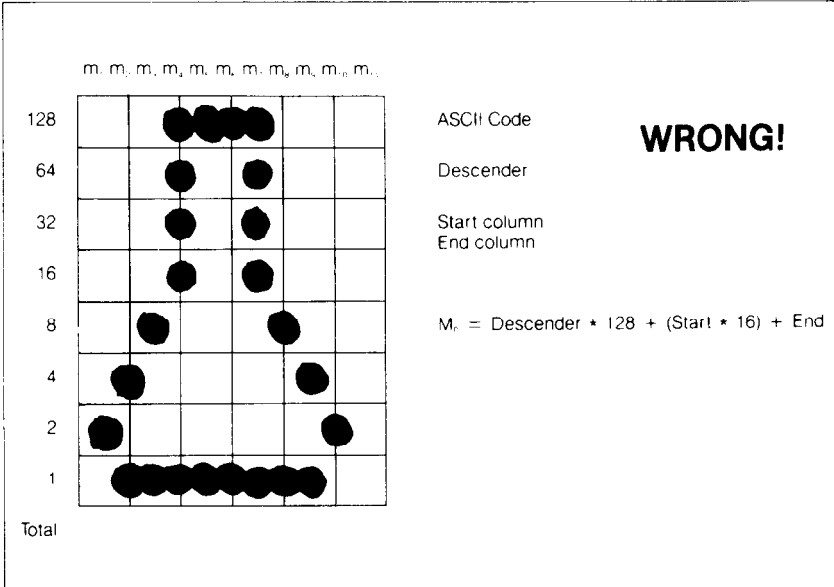
One bit in the download character definition command is to be treated as a descender or not. We'll get to the command in due time. For now, if your character uses the top eight dots, write in a one next to the word "Descender" on the layout grid; if it uses the bottom eight dots, write in a zero. In our example, we'll want to the bottom of the flask to line up with the baseline of the other characters, so it will *not* be a descender. As shown in Figure 6-5, we've written in a "1" on our grid.



**Figure 6-5.** We've designed a character and decided that it would not be a descender, hence the "1" written in.

### ■ Rule 2: Dots cannot overlap

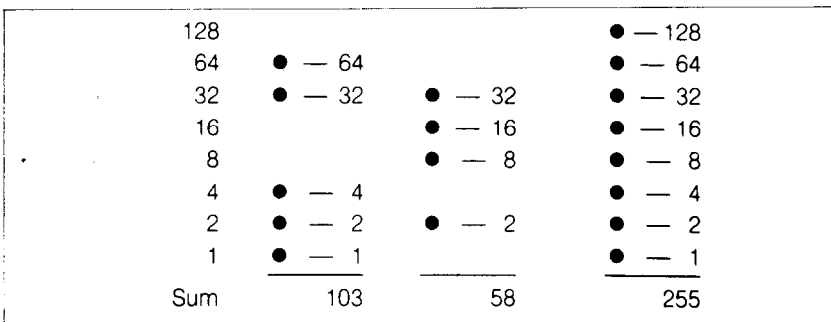
As you can see in Figure 6-5 our flask has a nearly continuous outline. But, you may ask, why not make it a *really* solid line and print all the intermediate dots, as shown in Figure 6-6? Because the dots that straddle the vertical lines in the grid actually overlap those inside the boxes. If we tried to print overlapping dots, the printhead would have to slow down and back up to print both dots — not very efficient! To avoid this inefficiency, this printer will not allow you to define a character like Figure 6-6. (Actually, you can define it, but when it prints, your printer will leave out the overlapping dots, so that it would print like Figure 6-5.)



**Figure 6-6.** Dots cannot overlap; those in immediately adjacent “half columns” will be ignored when the character is printed.

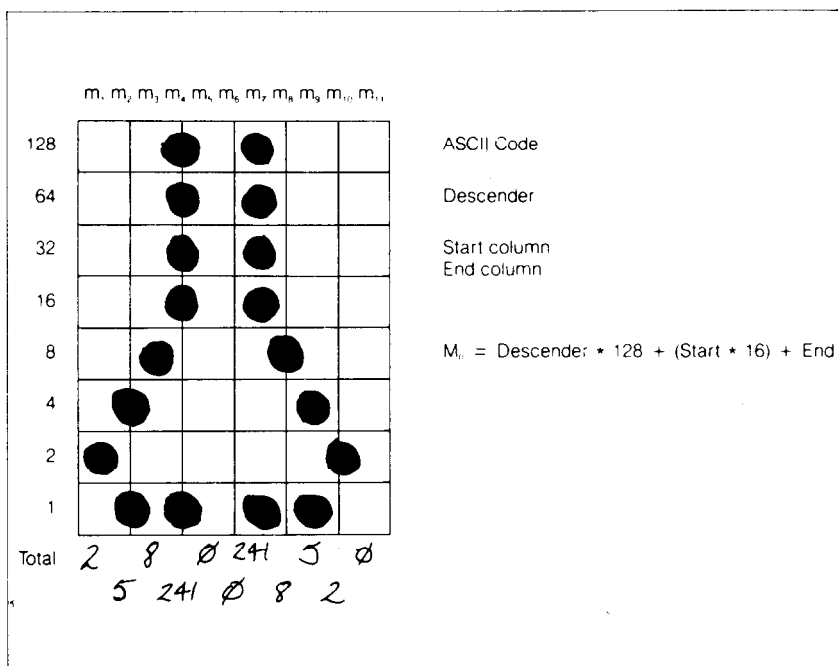
■ Add up each column of dots

Now it’s time to give our creative side a break and get down to some basic arithmetic. That’s where the numbers down the left side of the grid come in. Notice that there is a number for each row of dots and that each number is twice the number below it. By making these numbers powers of two we can take any combination of dots in a vertical column and assign them a unique value. Some examples will make this clearer. As shown in Figure 6-7, if we add the numbers for the dots that print in a column, the sum will be a number in the range of 0 to 255. Each number from 0 ~ 255 represents a unique combination of dots.



**Figure 6-7.** By adding the values of each dot in a column, you’ll get a unique description for any combination of dots.

So add up the values of the dots in each column using this system. In Figure 6-8 we've shown our grid with the sums of the columns filled in across the bottom (see if these agree with your answers!). Across the top of the grid you've probably noticed the cryptic labeling of each column: *m1*, *m2*, *m3*, etc. These labels correspond to the labels in the command syntax statement, which we'll get to shortly.



**Figure 6-8.** Add the values of the dots in each column and write the sum of each column at the bottom.

### ■ Assigning a value to your character

We've done a pretty thorough job of designing and describing a user-defined character. But this printer has room for 96 download characters — how does it know which standard character we want to print: every character is assigned a unique number.

The standard characters are assigned the ASCII codes — numbers from 0 to 255. For the download character sets you can also define any positions between 0 to 225. This means that once a character is defined and assigned a value (and the download character set is selected, you can use that character on the printer the same way you would any standard character. You can send the character with the same ASCII value. You can

also access the character from a BASIC program with the CHR\$ function.

There are no rules or restrictions on the use of numbers. This means you can use whatever is most convenient for you — perhaps seldom-used keys can be replaced by more useful characters. In our example, we'll assign the flask a value of 60, which is the code for the character “<” in the ASCII characters. A rather arbitrary selection, but this printer doesn't care!

Our chart would hardly be complete with just a picture of a chemist's flask, so in Figure 6-9 we've made completed grids for some other symbols: an automobile and a gun (quite a strange mix of characters!). The information on the grids is now complete (except for proportional width data — a more advanced topic we'll take up shortly).

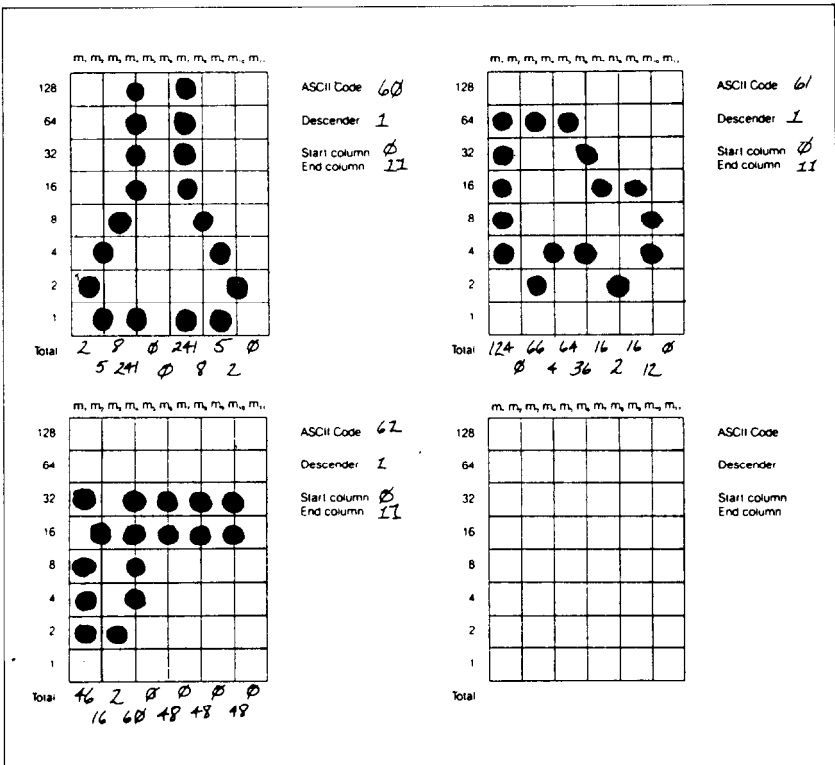


Figure 6-9. Character designs for the three graph symbols.

### ■ Download character definition command

You've read through a long explanation of download characters and we haven't even told you the command syntax yet! Now the wait is over. This is the most complex command in your printer repertoire and now you've got the necessary knowledge to implement it. Here it is:

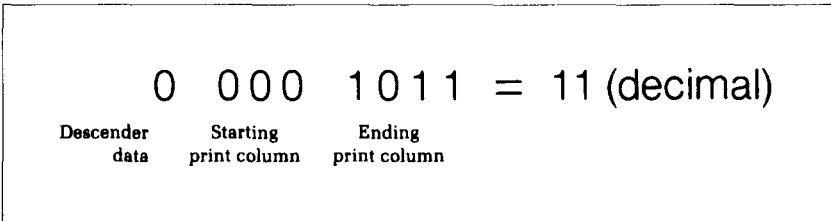
```
<ESC>“&”CHR$(0) n1 n2 m0 m1 m2 m3 m4 m5 m6 m7 m8 m9
m10 m11
```

Like the other printer's commands, it starts with an <ESC> (CHR\$(27)). The next character is an ampersand (&) (CHR\$(38)) followed by a CHR\$(0).

*n1* and *n2* are used to specify the ASCII values of the characters you are defining. The reason that there are two bytes reserved for this is that your printer allows you to define many characters with just a single command. *n1* is used to specify the beginning of a range of characters to be defined; *n2* specifies the end of the range. For instance, if you wanted to change the appearance of the numerals from 0 to 9 (which have ASCII codes 48 through 57), the command would begin with <ESC> “&” CHR\$(0) CHR\$(48) CHR\$(57) ... Of course, you can also define individual characters by making *n1* and *n2* equal.

*m0* is called the attribute byte, for it describes two attributes of the character we have designed: descender data and proportional width information. A byte consists of eight bits. In the attribute byte, the first (high order) bit is used for the descender data, and the last seven bits are used for proportional widths. We'll be discussing proportional character widths in detail later in this chapter; for now, we'll leave it at 11. The descender data was discussed earlier: to use the top eight pins, this bit should be 1; to use the bottom eight pins this bit should be 0. Figure 6-10 shows the bits of the attribute byte as we'll use them for our flask character. By now you've probably seen an easier way to determine the value of the attribute byte. Instead of translating everything to binary, merely assign the descender data a value of 128 (the value of the first bit) if you *don't want* descenders, or 0 if you *want* descenders. Then just add the descender data to the proportional width. This way, it's simply a matter of adding two decimal numbers. (In our case, it's 128 + 11 = 139.)





**Figure 6-10.** The attribute byte (*m0*) for our flask character.

You'll probably recognize *m1* ...*m11* from the top of our layout grid. That's right, each column is described by one byte. Now we've got everything we need to download one character to the printer. The complete command for our flask character is shown below:

```
CHR$(27);CHR$(38);CHR$(0);CHR$(60);CHR$(60);CHR$(139)
;CHR$(2);CHR$(5);CHR$(8);CHR$(241);CHR$(0);CHR$(0)
;CHR$(241);CHR$(8);CHR$(5);CHR$(2);CHR$(0)
```

Now let's send the information to the printer. The following program will send the character definitions for all three characters to the printer. Turn off the printer and set DIP switch 2-1 off. Then turn on the printer. Enter the program and run it.

```
10 LPRINT CHR$(27);"&";CHR$(0);CHR$(60);CHR$(62);
20 FOR N=60 TO 62
30 FOR M=0 TO 11
40 READ MM
50 LPRINT CHR$(MM);
60 NEXT M
70 NEXT N
80 LPRINT
90 END
100 DATA 139, 2, 5, 8,241, 0, 0,241, 8, 5,
      2, 0
110 DATA 139,124, 0, 66, 4, 64, 36, 16, 2, 16,
      12, 0
120 DATA 139, 46, 16, 2, 60, 0, 48, 0, 48, 0,
      48, 0
```

When you run this program, it looks like nothing happens. That's OK. We'll see why in just a moment. Save this program. We'll need it again shortly.

## PRINTING DOWNLOAD CHARACTERS

You've now defined and sent three characters to your printer. But how do you know that? If you try printing those characters now you don't get a flask, car and gun. Instead you get .. (<=). That's because the download characters are stored in a different part of the printer's memory. To tell it to look in download character RAM instead of standard character ROM it requires another command:

```
<ESC>"%"CHR$(n);CHR$(0)
```

This command is used to select the download character set (if  $n=49$ ) or to select the standard character set (if  $n=48$ ). Let's try it out. Enter this program:

```
10 LPRINT CHR$(27);"%1";CHR$(0);
20 LPRINT CHR$(60);CHR$(61);CHR$(62)
30 LPRINT CHR$(27);"%0";CHR$(0)
40 END
```

Voila! It should have printed out the three characters we defined. Your printout should look like this:

```
<ESC>"%
```

(If it doesn't, check the last program we ran for errors, then rerun it.)

Let's find out if there are any other characters in the download RAM. Try this program:

```
10 LPRINT CHR$(27);"%1";CHR$(0)
20 FOR I=32 TO 126
30 LPRINT CHR$(I);
40 NEXT I
50 LPRINT
60 FOR I=160 TO 254
70 LPRINT CHR$(I);
80 NEXT I
90 LPRINT
100 LPRINT CHR$(27);"%0";CHR$(0)
110 END
```

Nope! Just three characters in the download set. This is inconvenient for a couple of reasons. First, every time you wanted to use a download character you would have to switch back and forth between character sets. Knowing that you wouldn't want to do that, your printer won't even allow it. So we have made it an easy task to use mostly standard characters with just a few special characters thrown in. This command copies all the ASCII characters from the standard character ROM into download RAM:

```
<ESC>“:”CHR$(0);CHR$(0);CHR$(0)
```

Since it will copy *all* characters into download area, it will wipe out any characters that are already there. So it's important to send this command to the printer before you send any download characters you want to define. With that in mind, add this line to the program we used to send the characters to your printer:

```
5 LPRINT CHR$(27);“:””;CHR$(0);CHR$(0);CHR$(0)
```

Now try the download printout test program again. Your results look like Figure 6-11.

**Figure 6-11.** Printout of the download character set, into which all the ASCII characters have been copied, and the <, = and > have been changed.

To demonstrate how to use these characters, let's use this character set to print a small graph. This program, which has been built around the first program in this chapter, will do just that:

```
10 LPRINT CHR$(27);“:””;CHR$(0);CHR$(0);CHR$(0);
20 LPRINT CHR$(27);“&””;CHR$(0);CHR$(60);CHR$(62);
30 FOR N=60 TO 62
40 FOR M=0 TO 11
50 READ MM
60 LPRINT CHR$(MM);
```

```
70 NEXT M
80 NEXT N
90 LPRINT
100 DATA 139, 2, 5, 8,241, 0, 0,241,
      8, 5, 2, 0
110 DATA 139,124, 0, 66, 4, 64, 36, 16,
      2, 16, 12, 0
120 DATA 139, 46, 16, 2, 60, 0, 48, 0,
      48, 0, 48, 0
130 LPRINT CHR$(27);"D";CHR$(11);CHR$(0)
140 LPRINT CHR$(27);"h";CHR$(1);
150 LPRINT " U.S. EXPORTS"
160 LPRINT CHR$(27);"h";CHR$(0);
170 LPRINT CHR$(27);"%1";CHR$(0);
180 LPRINT "AUTOS";CHR$(9);
190 FOR I=.4 TO 9.3 STEP .4
200 LPRINT CHR$(61);
210 NEXT I
220 LPRINT
230 LPRINT "CHEMICALS";CHR$(9);
240 FOR I=.4 TO 8.7 STEP .4
250 LPRINT CHR$(60);
260 NEXT I
270 LPRINT
280 LPRINT "GUNS";CHR$(9);
290 FOR I=.4 TO 1.4 STEP .4
300 LPRINT CHR$(62);
310 NEXT I
320 LPRINT
330 LPRINT CHR$(9);"+--";
340 SCALE$="--+--"
350 FOR I=2 TO 8 STEP 2
360 LPRINT SCALE$;
370 NEXT I
380 LPRINT "---"
390 LPRINT CHR$(9);" ";
400 FOR I=2 TO 8 STEP 2
410 LPRINT " ";I;
420 NEXT I
430 LPRINT CHR$(27);"%0";CHR$(0)
440 LPRINT CHR$(27);"SO";
450 LPRINT CHR$(9);"MILLIONS OF DOLLARS"
460 LPRINT CHR$(27);"T"
470 END
```

## U.S. EXPORTS

```

AUTOS          @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
CHEMICALS     @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
GUNS          @@@@
+-----+-----+-----+-----+
                2     4     6     8
MILLIONS OF DOLLARS

```

Note that we didn't *have* to re-enter the download characters, since they were already sent to the printer with the first program. They will stay with the printer until you download new characters to replace them or turn the printer off. Even the `<ESC>` “@” command, which initializes the printer, does not destroy the contents of download RAM.

### DEFINING PROPORTIONAL CHARACTERS

Except for the actual width, defining characters for proportional printing is exactly the same as defining normal width download characters. Characters can range from 5 to 11 dots wide. This means that characters can be as narrow as one-half the normal width.

Besides being able to specify the actual width of the character, this printer allows you to specify the position in the standard grid where the character will print. You must specify the dot column in which the printed character starts and the dot column in which the character ends. Why, you may ask, would you want to define a character this way instead of merely defining the overall width of the character? Because this printer's proportional character definitions can also be used to print normal width characters, which are eleven dot columns wide. And by centering even the narrow characters in the complete grid they will look good even when you aren't printing them proportionally.

The command format for proportional character definition is exactly the same as you have learned; the only difference is the attribute byte, *m0*. As you know, the first bit of *m0* is used to specify whether the character is descender or not. The next three bits are used to specify the starting print column (acceptable values are 0 to 7). The last four bits specify the ending print column (acceptable values are 4 to 11). The minimum

character width is five dots (so you could not, for instance, specify a starting column of 6 and an ending column of 8, even though those are both within the acceptable range). If you inadvertently give an incorrect width value, however, your printer is forgiving: it will automatically revert to the default width of eleven dot columns.

Just as there was an easy trick for figuring the attribute byte earlier, you still don't need to know a thing about binary arithmetic. Merely multiply the starting column by 16, add the ending column number, and add 128 if the character is not a descender. If you prefer a formula:  $(\text{descender} * 128) + (\text{start} * 16) + \text{end}$ .

One thing to remember about defining proportional characters: a character cannot be wider than the specified width. That seems obvious enough! For example, if you specify a width of 6 for a character (starting in column 1 and ending in column 6), the seventh through eleventh of dots (if you specified any) will not print. You must, however, send information (even if it is 0) for those columns when you defined a character; your printer expects eleven characters following the `<ESC>"&"CHR$(0) n1 n2 m0` sequence.

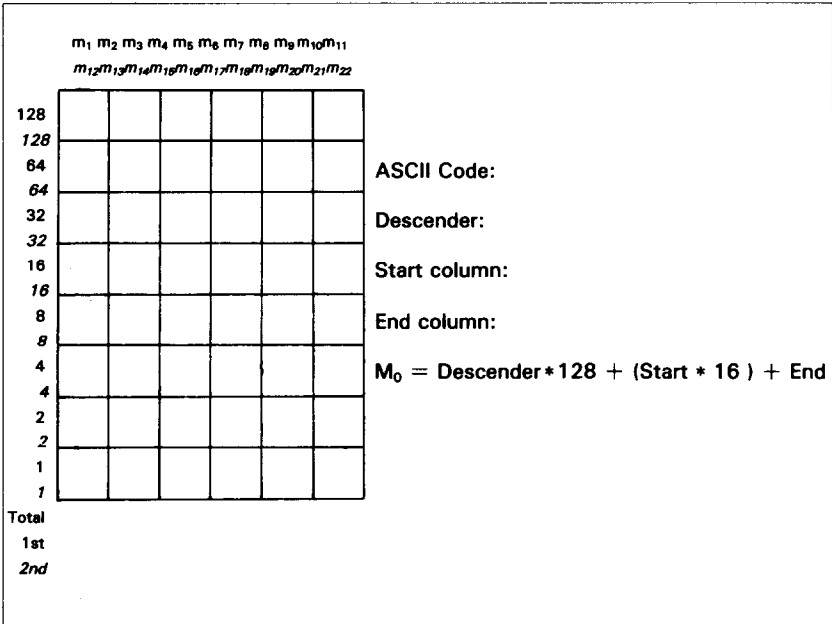
In most cases, the width you select should actually be one dot *wider* than the number of columns that the character actually occupies. This is so that there will be a space (of one dot) between characters when you print them. If you specify a width which is exactly the same as the number of columns in the character definition, the characters will touch when they print (this is sometimes desirable — for border characters or for large download characters that are more than eleven dots wide).

## DEFINING NLQ DOWNLOAD CHARACTERS

In the previous sections, we have learned how to define and print the draft download characters.

As you've learned in Chapter 3, you can print NLQ characters. You can also define the download characters with NLQ mode. Since NLQ characters use many more dots than draft characters, defining NLQ download characters is more complex than designing draft ones. If you use the grid and the program in this section, however, you will be able to design your own NLQ characters.

Because the NLQ characters can use as many as 16 dots vertically and 11 dots horizontally, you plan your designs on a different grid than the one you used for draft characters. Make up some grids (photocopy Figure 6-12 if you wish) and get ready to be creative!



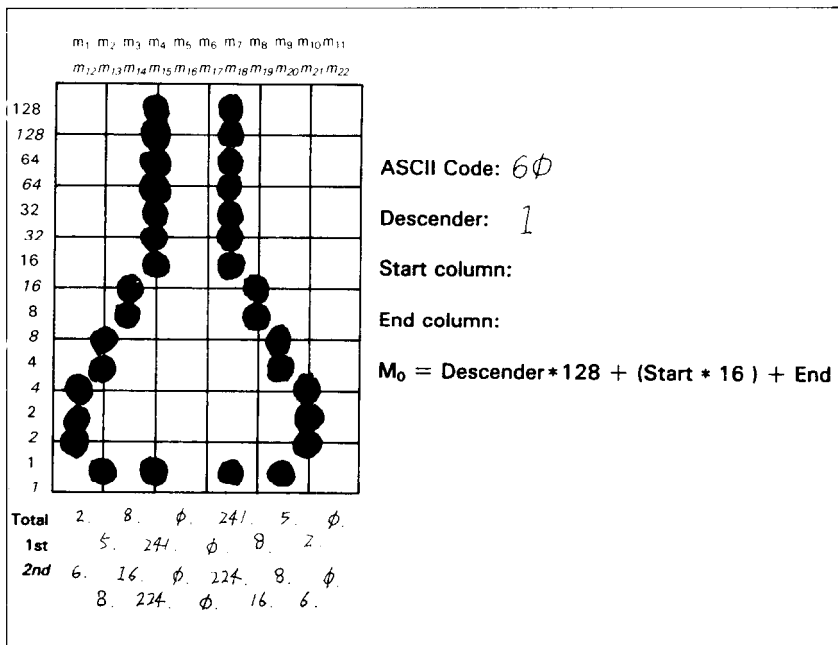
**Figure 6-12.** Use this grid (or one similar to it) to define your own NLQ characters.

As you noticed when the NLQ characters are printed, they are printing in twice; the first line of data is printed, the paper is moved up a distance of 1/2 dot, then the second data line is printed. So, we've written the numbers on the horizontal lines.

To calculate the data numbers for this column, you see which dots are used in the box and add their values together. Then you go down the dots on the horizontal lines and add their values together as shown in Figure 6-13.

Now we'll show you how to use the NLQ character definition with a flask as shown in Figure 6-13. Figure 6-13 shows the design drawn on a grid and the data numbers printed at the bottom of each column.

If you look at each column individually, you can see how the data numbers were calculated.



**Figure 6-13.** Add the values of the dots in each box and line column and write the sum of each column at the bottom.

Now enter the following program and run it. It has the data numbers for the NLQ flask character. For a character of your own, change the DATA numbers and the character definition position.

```

10 LPRINT CHR$(27);"x1";
20 LPRINT CHR$(27);"&";CHR$(0);CHR$(60);CHR$(60);
30 FOR M=0 TO 22
40 READ MM
50 LPRINT CHR$(MM);
60 NEXT M
70 LPRINT
80 LPRINT CHR$(27);"x0"
90 END
100 DATA 139, 2, 5, 8,241, 0, 0,241, 8,
      5, 2, 0
110 DATA 6, 8, 16,224, 0, 0,224, 16, 8,
      6, 0
120 DATA 139,124, 0, 66, 4, 64, 36, 16, 2,
      16, 12, 0
130 DATA 120, 4, 0, 4, 0, 96, 4, 0, 4,
      8, 0

```



```

140 DATA 139, 46, 16, 2, 60, 0, 48, 0, 48,
    0, 48, 0
150 DATA 92, 32, 4, 60, 0, 48, 0, 48, 0,
    48, 0

```

When you want to print the defined character, you must select the NLQ mode first, then select the download characters. If you don't select the NLQ mode, the download characters are not printed even you selected the download character set.

To demonstrate how to use the NLQ download characters, let's use this character set to print a small graph. Try this program.

```

10 LPRINT CHR$(27);"x1";
20 LPRINT CHR$(27);":":CHR$(0);CHR$(0);CHR$(0);
30 LPRINT CHR$(27);"&":CHR$(0);CHR$(60);CHR$(62);
40 FOR N=60 TO 62
50 FOR M=0 TO 22
60 READ MM
70 LPRINT CHR$(MM);
80 NEXT M
90 NEXT N
100 LPRINT
110 DATA 139, 2, 5, 8,241, 0, 0,241, 8,
    5, 2, 0
120 DATA 6, 8, 16,224, 0, 0,224, 16, 8,
    6, 0
130 DATA 139,124, 0, 66, 4, 64, 36, 16, 2,
    16, 12, 0
140 DATA 120, 4, 0, 4, 0, 96, 4, 0, 4,
    8, 0
150 DATA 139, 46, 16, 2, 60, 0, 48, 0, 48,
    0, 48, 0
160 DATA 92, 32, 4, 60, 0, 48, 0, 48, 0,
    48, 0
170 LPRINT CHR$(27);"D":CHR$(11);CHR$(0)
180 LPRINT CHR$(27);"h":CHR$(1);
190 LPRINT " U.S. EXPORTS"
200 LPRINT CHR$(27);"h":CHR$(0);
210 LPRINT CHR$(27);"%1":CHR$(0);
220 LPRINT "AUTOS":CHR$(9);
230 FOR I=.4 TO 9.3 STEP .4

```

```

240 LPRINT CHR$(61);
250 NEXT I
260 LPRINT
270 LPRINT "CHEMICALS";CHR$(9);
280 FOR I=.4 TO 8.7 STEP .4
290 LPRINT CHR$(60)
300 NEXT I
310 LPRINT
320 LPRINT "GUNS";CHR$(9);
330 FOR I=.4 TO 1.4 STEP .4
340 LPRINT CHR$(62);
350 NEXT I
360 LPRINT
370 LPRINT CHR$(9);"+--";
380 SCALE$="--+--"
390 FOR I=2 TO 8 STEP 2
400 LPRINT SCALE$;
410 NEXT I
420 LPRINT"--+"
430 LPRINT CHR$(9);" ";
440 FOR I=2 TO 8 STEP 2
450 LPRINT " ";I;
460 NEXT I
470 LPRINT CHR$(27);"%0";CHR$(0)
480 LPRINT CHR$(27);"S0";
490 LPRINT CHR$(9);"MILLIONS OF DOLLARS";
500 LPRINT CHR$(27);"T"
510 END

```

## U . S . EXPORTS

AUTOS	nnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
CHEMICALS	ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ
GUNS	fff
	+-----+-----+-----+-----+
	2          4          6          8
	MILLIONS OF DOLLARS

**MEMO**

---

---

## CHAPTER 7

# DOT GRAPHICS

---

---

**Subjects we'll cover in Chapter 7 include—**

- **This printer's bit image graphics capabilities;**
- **Printing a pre-defined shape;**
- **Plotting a calculated shape;**
- **High-resolution graphics.**

In Chapter 6 you were introduced to a form of computer graphics; you were able to actually define characters dot by dot. In this chapter you'll learn to use the same principles to make your printer print whole pages of dot graphics! We'll show you how to use dot graphics to create "super download characters". In addition, you'll see how your printer can be used as a graphics plotter. This can have some practical business applications as well as create some terrific computer art!

### **COMPARING DOT GRAPHICS WITH DOWNLOAD CHARACTERS**

A good understanding of dot graphics requires an understanding of how dot matrix printers work; you may want to review the first few pages in Chapter 6. The principles for dot graphics are the same as those for download characters.

There are some differences in the way they are implemented however. While download commands can be used to define a character between four and eleven columns of dots wide, dot graphics commands can be used to define a shape as narrow as one column of dots wide or as wide as 1920 dots!

There is no "descender data" with dot graphics; graphics images are always printed with the top seven or eight pins of the print head, depending on whether you have a 7-bit or 8-bit interface.

So when do you use graphics and when do you use download characters? Practically anything you can do with graphics you can do with download characters, and vice versa. A clever programmer could actually plot a mathematical curve using download characters or use strings of graphics data as user-defined characters. But why do it the hard way? There are several instances when dot graphics is clearly the best way to approach the problem:

- If the graphics image to be printed is wider than 11 dots or higher than 8 dots.
- If an image is to be printed just one time, as opposed to a frequently used “text” character.
- If you want higher resolution (this printer can print as many as 240 dots per inch in dot graphics mode; text mode, which includes download characters, prints 60 dots per inch.)

## USING THE DOT GRAPHICS COMMANDS

The command to print normal density (60 dots per inch horizontal; 72 dots per inch vertical) dot graphics uses this format:

```
<ESC>“*”CHR$(0) n1 n2 m1 m2 ...
```

Just like many of the other codes you have learned, the command starts with an escape sequence (<ESC>“\*” in this case). This is followed by CHR\$(0), which specifies normal density (the other densities are described later in this chapter). But unlike the other codes there can be any number of graphics data bytes following the command. That’s where *n1* and *n2* come in; they are used to tell the printer how many bytes of graphics data to expect.

### ■ Specifying the number of columns of dots

To figure the values of *n1* and *n2*, you’ll need to figure out how wide your graphics image will be (remember that there are 60 columns of dots per inch in normal density). Then comes the fun part: converting one number (the number of columns of dots) into two! Why is it necessary to use two numbers to tell the printer the number of graphics codes to expect? Because the

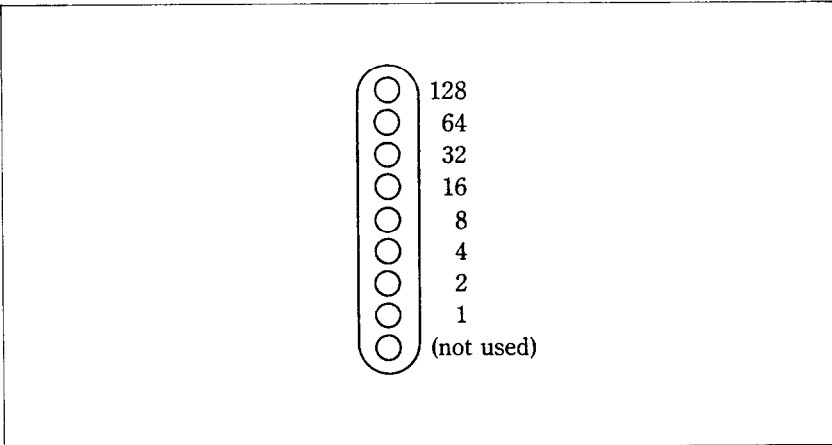
largest number we can send in one byte (that's what the BASIC CHR\$( ) function sends: one byte) is 255. And with the normal density graphics it's possible to have a graphics image as wide as 480 dots. So to figure out how many columns of graphics data to expect, your printer multiplies  $n2$  by 256 and adds the value of  $n1$  to the product. If you divide the number of columns by 256, then  $n2$  is the quotient and  $n1$  is the remainder (why not let your computer figure it out for you: if the number of columns is assigned to variable X, then  $n1 = X \text{ MOD } 256$  and  $n2 = \text{INT}(X/256)$ ). Table 7-1 might make things even easier.

**Table 7-1**  
**Calculating  $n1$  and  $n2$ .**

If the number of columns, x, ranges from:	Then $n1$ is:	and $n2$ is:
1 to 255	x	0
256 to 511	x-256	1
512 to 767	x-512	2
768 to 1023	x-768	3
1024 to 1279	x-1024	4
1280 to 1535	x-1280	5
1536 to 1791	x-1536	6
1792 to 1920	x-1792	7

#### ■ Specifying the graphics data

Now that we've told the printer how much data to expect, we better figure out how to send that information! Just as you do with download characters, with dot graphics you have control over the firing of every single pin on the print head. In Figure 7-1, you can see that we've labelled each pin on the print head with a number, as we did with download characters. And specifying pins to fire is done in the same way: to fire the second pin from the top, for instance, send a CHR\$(64). Firing several pins at once is done in a similar fashion. For example, to print the first, third, and fourth dots, add their values (128 + 32 + 16) to send this total: CHR\$(176). This is one byte of graphics data; it would replace  $m1$  in our format statement.



**Figure 7-1.** Starting with the most significant bit at the top, each pin of the print head is assigned a value which is a power of two.

A short program should demonstrate how to implement the graphics command. The program below gave us this printout:

```

10 ' Demo of dot graphics
20 PI=3.14159
30 WID=100
40 OPEN "LPT1:" AS #1 : WIDTH #1,255
50 PRINT#1, CHR$(27);"*";CHR$(0);
60 PRINT#1, CHR$(WID MOD 256);
70 PRINT#1, CHR$(INT(WID/256));
80 FOR I=0 TO WID-1
90 J=1+SIN(I*PI/32)
100 PRINT#1, CHR$(2^INT(J*3.5+.5));
110 NEXT I
120 LPRINT
130 CLOSE#1

```

In lines 50 to 70, we've selected normal density graphics and said that 100 characters of graphics data would follow. The loop between lines 80 and 110 is repeated to plot 100 points along a curve. This is an example of plotting a very simple


mathematical function (a sine wave) to create a design. Later in this chapter we'll show something more complex. The mathematical concepts (such as sine and pi) demonstrated here are not important; you don't have to be a math whiz to use this printer's graphics.

#### ■ Combining text and graphics

It's also possible to mix text and graphics in one line. This can be useful for labeling charts or graphs, or even inserting fancy graphics in text. Try adding these lines to our program:

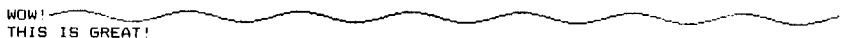
```
45 PRINT#1, "WOW! ";  
115 PRINT#1, "THIS IS GREAT! ";
```

Now if you run the program you should get a printout that looks like this:

```
WOW!  THIS IS GREAT!
```

But there is one thing to be careful of: all graphics data must print on the same line. The graphics command is turned off at the end of each line, even if you have specified that more graphics codes follow. To see what we mean, change line 30 to plot 1000 points and run the program.

```
30 WID=1000
```

```
WOW!   
THIS IS GREAT!
```

This will make the sine wave pattern long enough to go off the page.

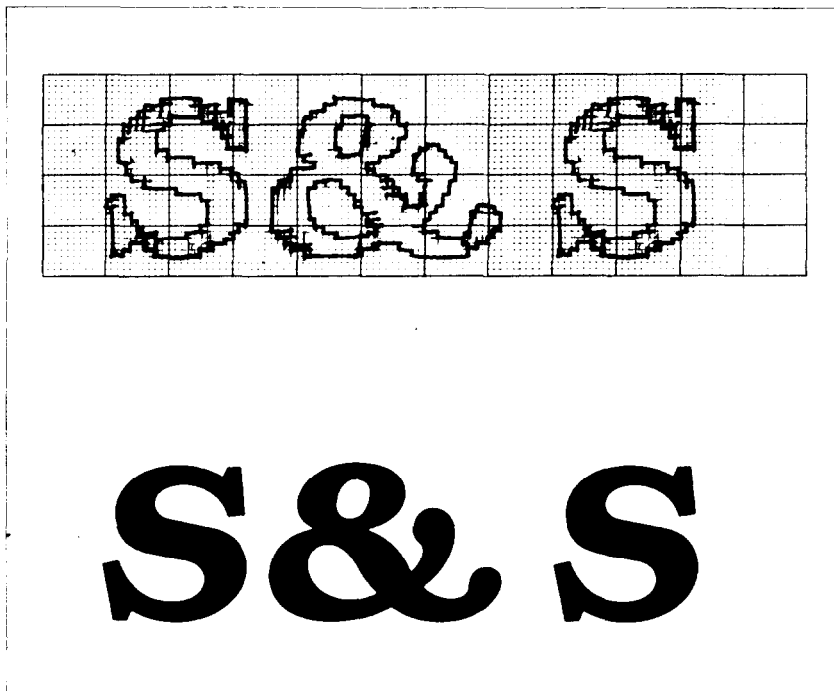
As you can see, your printer printed graphics up to the end of the line, then ignored the rest of the graphics data and returned to normal text on the next line.



## PRINTING A DESIGN OR LOGO

Since you control the firing of every pin, you can print nearly anything with your printer that can draw (and probably better, if you're like most computer users!). You can be used for creating "computer art" or drawing maps. Or, as we'll show you here, you can use dot graphics to print your logo at the top of each letter you print.

Designing an image to print with dot graphics is much like designing download characters. The best way to start is to lay out your image on graph paper. Since you can print eight rows of dots with each pass of the print head, draw a heavy horizontal line every eight rows on your graph paper. And it may be helpful to write the dot values (128,64,32, etc.) down the left side of each row. Then after you've filled in the "dots" that you want to print, it's time to get out the old calculator again! Just as you did with download characters, add up the values of each column of dots; this makes up one byte.



**Figure 7-2.** By laying out the logo on graph paper, you can calculate all of the graphics data.

In the program below, we've taken the logo graphics information and put it into BASIC DATA statements. The program itself is short and simple. The loop starting at line 100 reads the data statements into a string array variable called LOGO\$. In line 170 we change the line spacing to 8/72 inch so that the lines of graphics data will connect vertically. The actual printing is done in the loop between lines 180 and 210; line 190 sends the graphics control code to the printer and line 200 sends one line of graphics data.

The printout from the program is shown right below the program.

```

10 ' Prints S&S logo
20 LINE.8$=CHR$(27)+"A"+CHR$(8)
30 ' Set line spacing to 1/6"
40 LINE.12$=CHR$(27)+"2"
50 ' Select dot graphics
60 GRAPHIC$=CHR$(27)+CHR$(42)+CHR$(0)
70 DIM LOGO$(4)
80 WIDTH "LPT1:",255
90 ' Read data
100 FOR ROW=1 TO 4
110 FOR COLUMN=1 TO 100
120 READ P
130 LOGO$(ROW)=LOGO$(ROW)+CHR$(P)
140 NEXT COLUMN
150 NEXT ROW
160 ' Print logo
170 LPRINT LINE.8$;
180 FOR ROW=1 TO 4
190 LPRINT GRAPHIC$;CHR$(100);CHR$(0);
200 LPRINT LOGO$(ROW)
210 NEXT ROW
220 LPRINT LINE.12$
230 ' Row 1
240 DATA 0, 0, 0, 0, 1, 3, 7, 7, 7, 15
250 DATA 14, 14, 14, 14, 14, 7, 7, 3, 3, 15
260 DATA 15, 15, 0, 0, 0, 0, 0, 0, 0, 0
270 DATA 0, 1, 3, 3, 7, 7, 15, 14, 14, 14
280 DATA 14, 15, 7, 7, 7, 3, 0, 0, 0, 0
290 DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
300 DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
310 DATA 0, 0, 0, 0, 1, 3, 7, 7, 7, 15
320 DATA 14, 14, 14, 14, 14, 7, 7, 3, 3, 15
330 DATA 15, 15, 0, 0, 0, 0, 0, 0, 0, 0

```

```
340 ' Row 2
350 DATA 0, 0, 60,255,255,255,255,255,143, 15
360 DATA 7, 7, 7, 7, 3, 3, 3,131,193,241
370 DATA 240,240, 0, 0, 0, 0, 0, 0, 0, 1
380 DATA 121,253,253,255,255,255,143, 7, 7, 7
390 DATA 31,253,252,248,248,240,192, 0, 7, 15
400 DATA 31, 31, 15, 7, 3, 0, 0, 0, 0, 0
410 DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
420 DATA 0, 0, 60,255,255,255,255,255,143, 15
430 DATA 7, 7, 7, 7, 3, 3, 3,131,193,241
440 DATA 240,240, 0, 0, 0, 0, 0, 0, 0, 0
450 ' Row 3
460 DATA 0, 31, 31, 3,129,128,192,192,192,192
470 DATA 192,224,224,224,224,240,255,255,255,255
480 DATA 255,127, 0, 0, 0, 0, 63,127,255,255
490 DATA 255,255,193,128,128,128,128,192,224,240
500 DATA 252,255,255,255,127, 63, 31, 7, 7, 31
510 DATA 254,252,248,224,128, 0, 0, 3, 7, 7
520 DATA 7, 3, 0, 0, 0, 0, 0, 0, 0, 0
530 DATA 0, 31, 31, 3,129,128,192,192,192,192
540 DATA 192,224,224,224,224,240,255,255,255,255
550 DATA 255,127, 0, 0, 0, 0, 0, 0, 0, 0
560 ' Row 4
570 DATA 0,248,248,240,224,224,112,112, 56, 56
580 DATA 56, 56, 56,120,120,240,240,224,224,192
590 DATA 128, 0, 0, 0, 0, 0, 192,224,240,240
600 DATA 240,248,248,248,120,120, 56, 56, 56, 56
610 DATA 48,112,224,224,224,224,240,240,248,248
620 DATA 120,120, 56, 56, 56, 56,120,240,224,224
630 DATA 192,128, 0, 0, 0, 0, 0, 0, 0, 0
640 DATA 0,248,248,240,224,224,112,112, 56, 56
650 DATA 56, 56, 56,120,120,240,240,224,224,192
660 DATA 128, 0, 0, 0, 0, 0, 0, 0, 0, 0
```



If you are using with the IBM mode (DIP switch 1-6 off), change the following lines to the program given above.

```

20 LINE.8$=CHR$(27)+CHR$(65)+CHR$(8)+CHR$(27)+
  CHR$(50)
40 LINE.12$=CHR$(27)+CHR$(65)+CHR$(12)+CHR$(27)+
  CHR$(50)

```

## PLOTTING WITH YOUR PRINTER

This section of the manual gets into more serious BASIC programming just because it's required in order to have the computer act as a plotter driver. Don't be intimidated; while it's beyond the scope of this manual to teach BASIC, if you try the examples and take it slowly you should be doing some fancy plotting of your own before you know it.

If designing and calculating dot graphics images by laying them out on graph paper seems too tedious to you, then let the computer do the work for you! With your computer doing the calculations and your printer plotting the output, you can come up with some terrific business graphs, charts, and mathematical function plots.

The best way to do this is to set up an array in memory. This is your "graph paper." The first thing to do is to determine how big you want your output to be; this will determine the size of your array. (If you have grandiose plans to fill an entire page with plotter output, you better have lots of memory in your computer. With 60 dots per inch horizontally and 72 dots per inch vertically, it takes at least 540 bytes of memory for each square inch of plotted area. That doesn't sound so bad — but an area 8 inches square requires over 32K!)

Your array should be two-dimensional (just like graph paper) where one dimension will be the number of columns of dots and the other dimension is the number of printing lines (remember that you can have up to eight rows of dots per printed line).

Here's a program that will use calculated-shape graphics to plot a circle. As you'll see, by changing a few lines it can be used to plot virtually any shape.

```

10 ' Plotting program
20 ' Set program constants
30 MAXCOL%=75           : MAXROW%=14
40 DIM BIT%(MAXCOL%,MAXROW%)
50 MASK%(1)=64          : MASK%(4)=8
60 MASK%(2)=32          : MASK%(5)=4
70 MASK%(3)=16          : MASK%(6)=2

```

```

80 LX=20                : LY=20
90 LXFAC=72/LX         : LYFAC=87/LY
100 ' Plot curve
110 GOSUB 600
120 '
130 ' Send bit image map to printer
140 LPRINT CHR$(27);"A";CHR$(6);
150 FOR ROW%=0 TO MAXROW%
160 A$=""
170 LPRINT CHR$(27);"*";CHR$(0);CHR$(MAXCOL%);
    CHR$(0);
180 FOR COL%=1 TO MAXCOL%
190 A$=A$+CHR$(BIT%(COL%,ROW%))
200 NEXT COL%
210 LPRINT A$;" "
220 NEXT ROW%
230 LPRINT CHR$(27);"A";CHR$(12);
240 END
250 '
260 ' Subroutine to draw a line from X1,Y1 to
    X2,Y2
270 '
280 XL=X2-X1           : YL=Y2-Y1
290 NX=ABS(XL*LXFAC)   : NY=ABS(YL*LYFAC)
300 IF NX < NY THEN NX=NY
310 NS%=INT(NX+1)
320 DX=XL/NS%         : DY=YL/NS%
330 FOR I%=1 TO NS%
340 X1=X1+DX          : Y1=Y1+DY
350 GOSUB 400
360 NEXT I%
370 RETURN
400 '
410 ' Subroutine to plot a point at X1,Y1
420 '
430 XX=X1*LXFAC       : YY=Y1*LYFAC
440 COL%=INT(XX)+1
450 ROW%=INT(YY/6)
460 XIT%=INT(YY-ROW%*6)+1
470 BIT%(COL%,ROW%)=BIT%(COL%,ROW%) OR MASK%(XIT%)
480 RETURN
600 '
610 ' Subroutine to plot a circle
620 '
630 RAD=9
640 X1=19              : Y1=10
650 FOR ANG%=0 TO 360 STEP 10

```

```

660 RANG=ANG%*6.28/360
670 X2=RAD*COS(RANG)+10 : Y2=RAD*SIN(RANG)+10
680 GOSUB 250
690 NEXT ANG%
700 RETURN

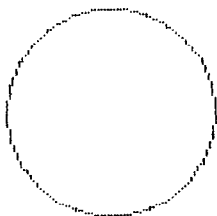
```

If you are using with the IBM mode (DIP switch 1-6 off), change the following lines to the program given above.

```

140 LPRINT CHR$(27);"A";CHR$(6);CHR$(27);"2"
230 LPRINT CHR$(27);"A";CHR$(12);CHR$(27);"2"

```



### ■ How the program works

In the program above, we've created an array called `BIT%`, which is dimensioned in line 40. You'll note that instead of using numeric constants to dimension the array, we used the variables `MAXCOL%` and `MAXROW%`. This way, if your computer has enough memory and you want to plot a larger image, all you need to change are the values in line 30. The array `MASK%` contains the values of the dots. (In order to make this program run on most computers, we're only six pins for graphics. With many computers, you can use all eight available pins.) In lines 80 and 90 we've defined some other variables you'll be interested in: `LX`, `LXFAC`, `LY`, and `LYFAC` are used as scaling factors. By changing these values, you can change the size of your printed image or even distort it (you can, for example, make our circle print as an ellipse). Experiment a little bit!

The main calculations for plotting the image are done in the subroutine starting at program line 600. This is where you put the formulas that you want to plot. By changing just the lines after 600 (with some creative mathematics!) you can plot any function — limited only by your imagination. Some examples are shown at the end of this section.

What the program section starting at line 600 actually does is

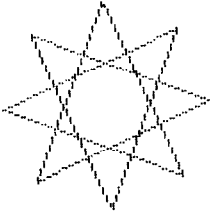
to calculate starting and ending points for a line (in our circle the “lines” are very short — sometimes the starting and ending points are the same). The coordinates of the starting point of the line are assigned to variables X1 and Y1. The line ends at point X2, Y2. When these coordinates have been calculated, a subroutine call is made to line 250. This subroutine calculates the coordinates of individual points along that line.

After these coordinates have been determined, the subroutine at line 400 is called. This routine turns “on” an individual dot in our array called BIT%. (Keep in mind that no printing has been done yet; the computer is still drawing the image on its “graph paper” in memory.) The way an individual dot is turned on is using the logical OR function in line 470.

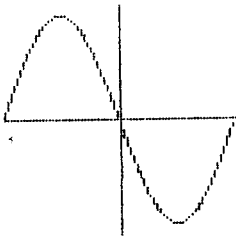
When all the points have been plotted in memory, printing begins at line 130. We first set the line spacing to 6/72 inch using the `<ESC>“A”CHR$(n)` command. This is so that there are no gaps between rows of dots. Then the loop from line 150 to line 220 prints the dot graphics image one line (which is six dots high) at a time. The variable A\$ is used to build a string of all the columns of BIT% in a given row.

As you can see, by taking the program in small pieces and analyzing it, graphics programming does not have to be difficult. If you want to try some other plots, try these (replace lines after 600 with the lines below). The printouts from each program are shown below the listing.

```
600 '
610 ' Subroutine to plot a star
620 '
630 RAD=9
640 FOR ANG%=0 TO 360 STEP 45
650 RANG=ANG%*3.14159/180
660 RANG2=(ANG%+135)*3.14159/180
670 X1=RAD*COS(RANG)+10
680 Y1=RAD*SIN(RANG)+10
690 X2=RAD*COS(RANG2)+10
700 Y2=RAD*SIN(RANG2)+10
710 GOSUB 250
720 NEXT ANG%
730 RETURN
```



```
600 '  
610 ' Subroutine to plot a sine wave  
620 '  
630 X1=0 : Y1=10 : X2=20 : Y2=10  
640 GOSUB 250  
650 X1=10 : Y1=0 : X2=10 : Y2=20  
660 GOSUB 250  
670 X1=0 : Y1=10  
680 FOR X2=0 TO 20 STEP .2  
690 Y2=10-9*SIN(3.14159*X2/10)  
700 GOSUB 250  
710 NEXT X2  
720 RETURN
```



## HIGH RESOLUTION GRAPHICS

Up until now all of the dot graphics printing we have done has been with your printer's normal density mode. This can give you some pretty sharp images at great speed. Sometimes though, you may want to create an image with even higher resolution. This printer has seven graphics modes you can use; they're summarized in Table 7-2.

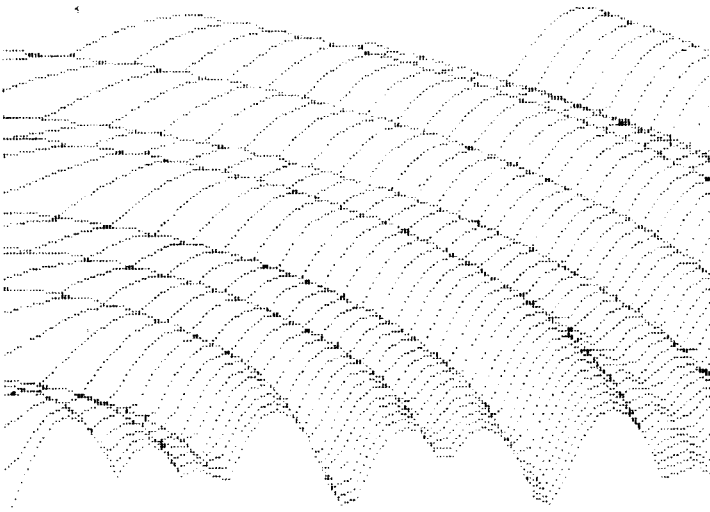


**Table 7-2**  
**Dot graphics commands**

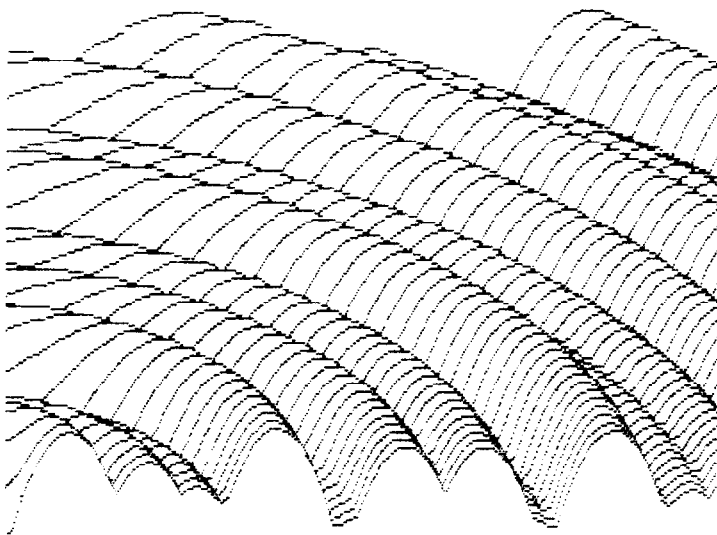
Function	Control code
Normal density (60 dots/inch)	$\langle \text{ESC} \rangle " * " \text{CHR} \$ ( 0 ) n 1 n 2 m 1 m 2 \dots$
Double density (120 dots/inch)	$\langle \text{ESC} \rangle " * " \text{CHR} \$ ( 1 ) n 1 n 2 m 1 m 2 \dots$
Double density/double speed	$\langle \text{ESC} \rangle " * " \text{CHR} \$ ( 2 ) n 1 n 2 m 1 m 2 \dots$
Quadruple density (240 dots/inch)	$\langle \text{ESC} \rangle " * " \text{CHR} \$ ( 3 ) n 1 n 2 m 1 m 2 \dots$
CRT graphics (80 dots/inch)	$\langle \text{ESC} \rangle " * " \text{CHR} \$ ( 4 ) n 1 n 2 m 1 m 2 \dots$
Plotter graphics (72 dots/inch)	$\langle \text{ESC} \rangle " * " \text{CHR} \$ ( 5 ) n 1 n 2 m 1 m 2 \dots$
CRT graphics type II (90 dots/inch)	$\langle \text{ESC} \rangle " * " \text{CHR} \$ ( 6 ) n 1 n 2 m 1 m 2 \dots$

The command syntax for all of the commands is the same — just as you have learned it for the  $\langle \text{ESC} \rangle " * " \text{CHR} \$ ( 0 )$  (normal density) command. The number of columns to be printed is  $n1 + 256 * n2$ .

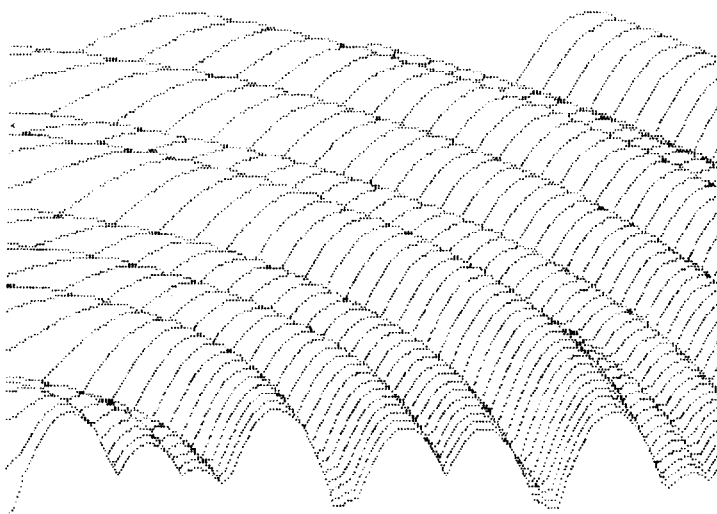
So what do these different modes do? On the following pages are actual size reproductions of printouts of the same image printed in each of the seven different graphics modes. They were all printed using the plotting program in this chapter (with a rather complex set of formulas starting at line 600!).



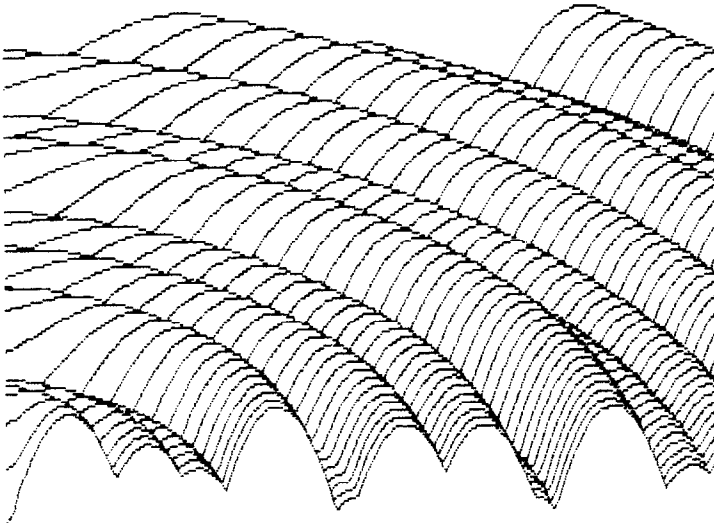
Normal density graphics



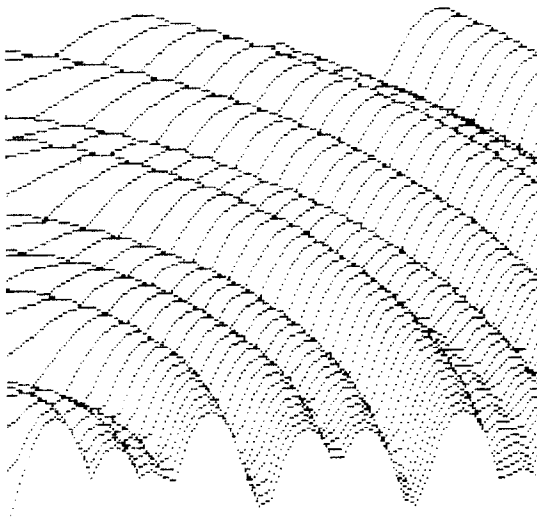
Double density graphics



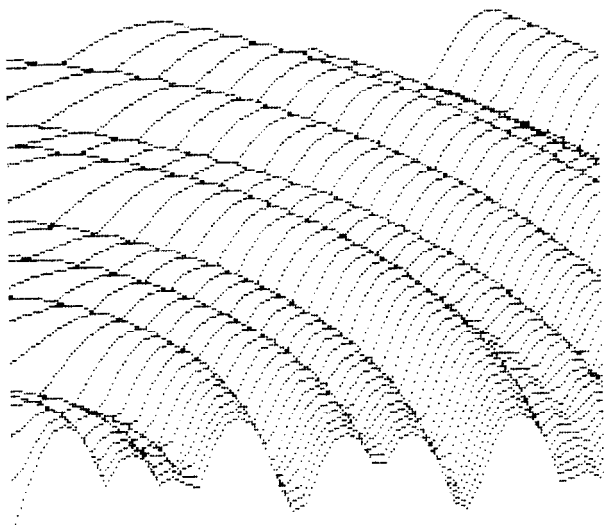
Double density/double speed



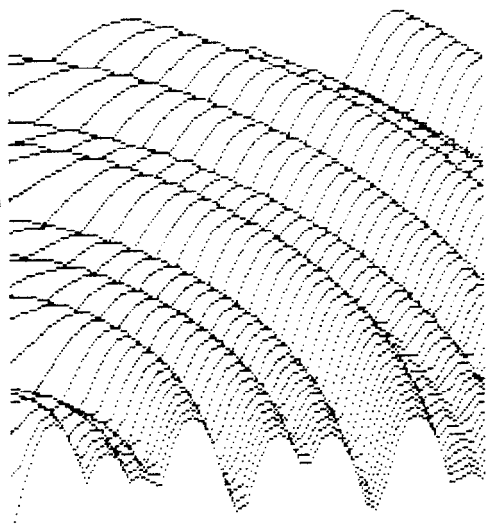
Quadruple density graphics



CRT graphics

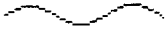


Plotter graphics

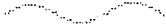


CRT graphics type II

So if quadruple density looks so great, why not use it all the time? Let's try an experiment on your printer which will show just how the different modes work. Using the first program in this chapter, change line 50 to try each of the different modes. Just change the "0" to "1", "2", "3", etc. in turn. Your printouts should look something like this:



<ESC> "\*" CHR\$(1)



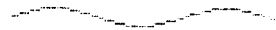
<ESC> "\*" CHR\$(2)



<ESC> "\*" CHR\$(3)



<ESC> "\*" CHR\$(4)



<ESC> "\*" CHR\$(5)



<ESC> "\*" CHR\$(6)

As you can see, the different modes seem to condense the printed image. So, to get the same image in a higher density mode, you must plot more points. This requires twice as much memory for your array, twice as much computing time, and twice as much printing time (but the results may be worth it!).

Our engineers have given programmers a unique shortcut for program development — double density double speed graphics. Although this mode requires just as much memory and com-

puting time as double density, it prints at the same speed as normal density graphics. Amazing, you say? Well, it is — until you know the secret. Every other column of dots is ignored, so the output is actually the same as normal density graphics. The advantage is that you can write and debug your programs at double speed, then change to double density graphics for terrific output.

This printer has three other densities (CRT, plotter, and CRT type II) which can be used to achieve greater compatibility with various hardware and software configurations. Plotter density can be especially for simplifying programming, since it has equal densities both the horizontal and vertical directions (72 dots per inch each way). With this feature you can plot truly round circles without any type of scaling factors.

## COMPATIBILITY WITH EXISTING SOFTWARE

With its ability to print seven different graphics densities, this printer's graphics abilities are advanced indeed. There are many programs, in fact, that are unable to use this printer's single graphics command  $\langle \text{ESC} \rangle "*" n1 n2$  for selecting the proper density. To maintain compatibility with this software, there are individual commands to select each of this printer's common

**Table 7-3**  
**Alternative graphics commands**

Density	Single command	Individual command
Normal	$\langle \text{ESC} \rangle "*" \text{CHR}\$(0) n1 n2$ $m1 m2 \dots$	$\langle \text{ESC} \rangle "K" n1 n2 m1 m2 \dots$
Double	$\langle \text{ESC} \rangle "*" \text{CHR}\$(1) n1 n2$ $m1 m2 \dots$	$\langle \text{ESC} \rangle "L" n1 n2 m1 m2 \dots$
Double with double-speed	$\langle \text{ESC} \rangle "*" \text{CHR}\$(2) n1 n2$ $m1 m2 \dots$	$\langle \text{ESC} \rangle "Y" n1 n2 m1 m2 \dots$
Quadruple	$\langle \text{ESC} \rangle "*" \text{CHR}\$(3) n1 n2$ $m1 m2 \dots$	$\langle \text{ESC} \rangle "Z" n1 n2 m1 m2 \dots$
CRT	$\langle \text{ESC} \rangle "*" \text{CHR}\$(4) n1 n2$ $m1 m2 \dots$	none
Plotter	$\langle \text{ESC} \rangle "*" \text{CHR}\$(5) n1 n2$ $m1 m2 \dots$	none
CRT type II	$\langle \text{ESC} \rangle "*" \text{CHR}\$(6) n1 n2$ $m1 m2 \dots$	none

graphics densities. These commands, which are shown in Table 7-3, can be used interchangeably with the corresponding  $\langle \text{ESC} \rangle$  “\*” command. Like the commands you are already familiar with, these new commands are followed by two bytes to specify the number of graphics data bytes to print and then the data.

## MORE GRAPHICS PROGRAMMING TIPS

At the end of this chapter, we’ll discuss two modes that the printer offers to help you solve potential graphics problems. A redefining code allows you to change the density for graphics programs that use one of the four alternate codes. The 9-pin graphics mode allows you to use all nine pins on each line and thus speed up screen dumps.

**Table 7-4**  
More graphics commands

Function	Control code
Redefine the graphics mode	$\langle \text{ESC} \rangle$ “?” <i>n0 n1</i>
9-pin graphics	$\langle \text{ESC} \rangle$ “^” CHR\$( <i>n0</i> ) <i>n1 n2</i> <i>m1 m2</i>

### ■ Redefining alternate graphics codes

Your printer provides a command to redefine one of the alternate graphics modes — K, L, Y, or Z — so that it represents any other of the seven graphics modes. The command is  $\langle \text{ESC} \rangle$  “?” *n0 n1*, where *n0* is one of the four letters, “K”, “L”, “Y”, or “Z”, and *n1* is one of the numbers used with the  $\langle \text{ESC} \rangle$  “\*” command, 0 to 7. There are several instances in which you may use this sequence.

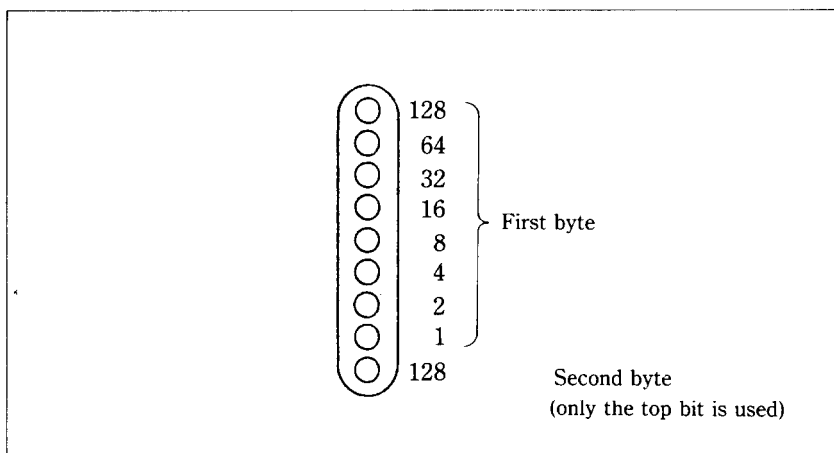
The first occurs if you have written a program to be printed in one graphics mode and now want to print it in another. If you have used concatenation to store your graphics command in one short character string, that will not be difficult. You can simply change the mode number or alternate code in the definition of the character string.

A second time you can make good use of the redefining code occurs when you want to change a program in which you have not concatenated the graphics codes. Using the  $\langle \text{ESC} \rangle$  “?” sequence allows you to change every instance of your graphics command by entering only one line.

### ■ 9-pin graphics mode

In the early part of this chapter, we said that the bottom pin of the print head is not normally used in the graphics modes. That's because most computers communicate with parallel-type peripheral devices using eight data lines. When the peripheral is a printer, each data line corresponds to one pin on the print head. Thus each byte sent will fire up to eight pins.

But the printer has 9 pins available. So how do you fire the ninth pin with only 8 data lines? In fact, do you really want to bother with just one extra pin? Well, for such graphics-intensive applications as screen dumps, printing 9 pins at a time can speed up the process considerably. For this purpose, your printer has a special 9-pin graphics mode (it won't, however, work with 7-bit interface systems). In this mode the printer takes 2 bytes to fire all 9 pins as shown in Figure 7-3.



**Figure 7-3.** Your printer takes 2 bytes to fire all 9 pins in case of the 9-pin graphics mode.

In addition, you can select the print density by the value of  $n0$ . When  $n0$  is 0 the normal density is selected, and when  $n0$  is 1 the double density is selected.

Since computers are faster than printers, there is no significant time loss in printing a single line of graphics with 9 pins. You get 9 dots per line in about the same time as you get 8 dots in the other graphics modes.



**MEMO**

## Free Manuals Download Website

<http://myh66.com>

<http://usermanuals.us>

<http://www.somanuals.com>

<http://www.4manuals.cc>

<http://www.manual-lib.com>

<http://www.404manual.com>

<http://www.luxmanual.com>

<http://aubethermostatmanual.com>

Golf course search by state

<http://golfingnear.com>

Email search by domain

<http://emailbydomain.com>

Auto manuals search

<http://auto.somanuals.com>

TV manuals search

<http://tv.somanuals.com>