COMMODORE 64/128
INTERFACE CARTRIDGE
FOR NL-10

USERS MANUAL

COMMODORE 64/128

INTERFACE CARTRIDGE FOR NL-10

USER'S MANUAL

PN 80820516

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

NL-10: Star Micronics Co., Ltd.
Commodore 64/128: Commodore Business Machines, Inc.

Table of Contents

CHAPTER 1

INTRODUCTION

Subjects we'll cover in Chapter 1 include—
- **Installing the interface cartridge;**
- **Connecting the printer;**
- **Installing the ribbon and paper;**
- **Extra functions with the control panel.**

This interface cartridge contains all the electronics your printer needs to talk to a computer. To use your printer with a different computer, just install a different cartridge. There is an interface (I/F) cartridge for each popular computer on the market.

This interface cartridge offers full compatibility with any Commodore on the market. It combines the speed and efficiency of a proven winner with the character sets and printer codes used by the Commodore!

INSTALLING THE INTERFACE CARTRIDGE

If you have the correct interface cartridge for your computer, we can start by turning the printer around. Facing the back of it, you'll notice an opening at the left end. This is where the cartridge goes.

Fit the interface cartridge into the space as shown, with the round connector toward you and slide it in all the way (don't force it). If the connector is seated snugly in its socket, you should be able to tighten the screw easily. This done, connect the cable from your computer and you're ready to go.

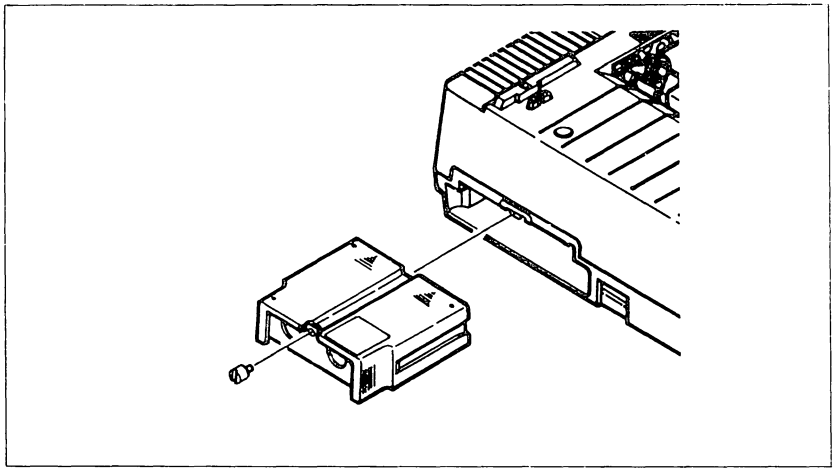


Figure 1-1. Slide the interface cartridge into the slot and fix it with a screw.

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 6-pin DIN cable to either of the two connector sockets located on the interface cartridge. This cable is "keyed," so it is impossible for you to connect it incorrectly. This means that you should be able to plug in the cable with only a slight pressure — don't force it! If you have to force it, it's in the wrong way. Push too hard and you'll ruin your cable.
3. Connect the other end of the cable to your computer by the Serial Port connector at the back of your computer. Make sure that you are "keying" the pins properly into the connector with six holes.

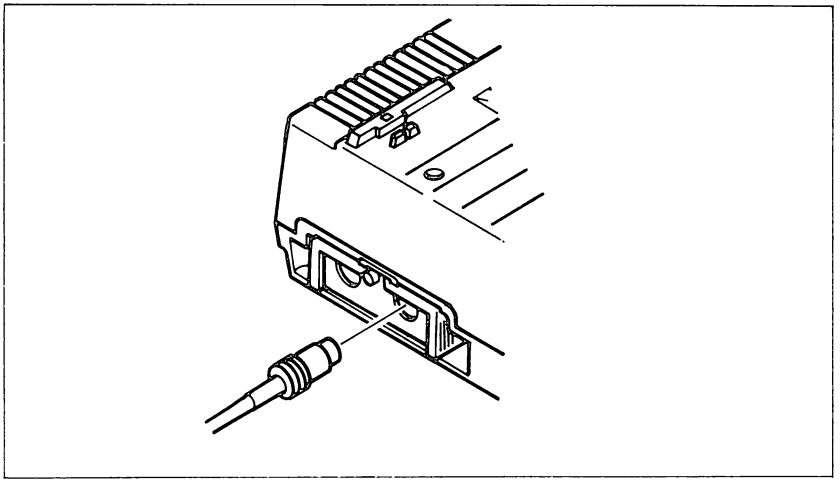


Figure 1-2. This is how you connect the cable.

INSTALLING THE RIBBON AND PAPER

After you have connected the cable, it is necessary to install the ink ribbon and the paper. They are described in your printer's manual, so please follow the procedures.

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.

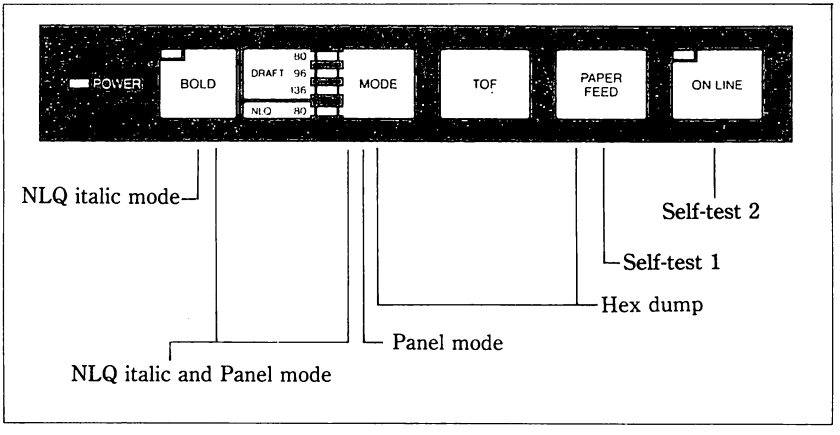


Figure 1-3. Extra functions while turning on the printer.

Self-tests

Your printer has two built-in programs that print out sample lines of letters, numbers, and other characters—to show you that everything's in good working order. Self-test 1 is the long version, and self-test 2 is the short one. You can use either one to show the characters available in the printer, to check the installation of the ribbon and paper, and to check the adjustment of the head-to-platen gap.

Best of all, you don't have to wait another minute—you can print the self-test without hooking up the printer to your computer! It's as simple as 1, 2, 3...

1. Plug in the printer (don't turn it on yet).
2. Insert a sheet of paper (or sprocket paper, either one).
3. While holding down one of the following keys, turn on the power switch:

Paper Feed key—Self-test 1 (long)
On Line key—Self-test 2 (short)

Figure 1-4. The self-tests show you what the printer can print.

Were you surprised? It's fast, isn't it? About 120 characters a second, to be exact.

When you want to terminate the self-test function, turn off the power switch.

■ 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 2, 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.

■ NLQ italic mode

Sometimes, you may want to print with italic characters with NLQ mode as the power-on default. You can set the NLQ 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.

■ NLQ italic and Panel mode

You can combine with the "Panel" mode and the "NLQ 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.

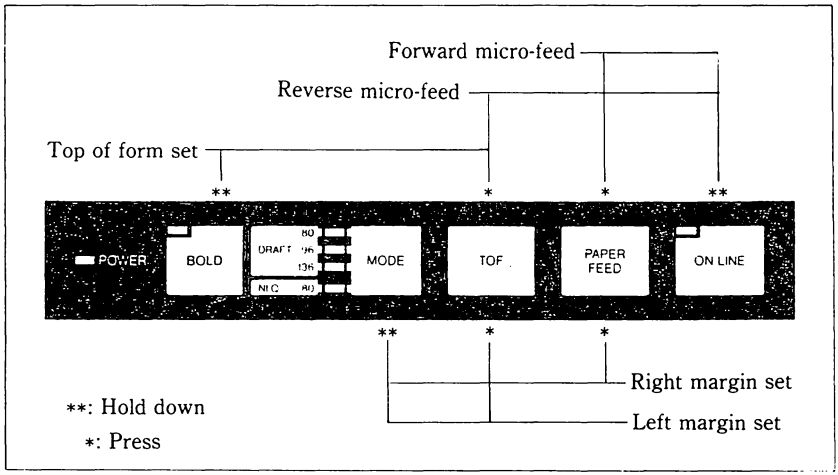


Figure 1-5. You can set many functions by the combinations of the control panel keys while in the Off Line mode.

■ 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
   TOF key — Reverse micro-feed
3. When you can set the print start position, release the Paper Feed key or the TOF key first, then release the On Line key.

■ Setting the top of form

When you turn on the printer, the top of form is automatically set to the current position. If you want to change the position, you can re-set it by the following procedures.

1. Set the Off Line mode by pressing the On Line key.
2. While holding the Bold key, press the TOF key. Your printer acknowledges the new top of form with the sound of beep.

■ Setting the left and right margins

As you'll learn in Chapter 3, 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 Mode key, press one of the following keys.
   TOF 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 2

BASIC PRINTING

Subjects we'll cover in Chapter 2 include—

- **BASIC commands that affect your 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 —**
  - **Underlining,**
  - **Boldface and emphasized text,**
  - **Subscripts and superscripts,**
  - **Printing in italics,**
  - **Reverse printing,**
  - **Mixing print modes.**

The rest of this book will show you enough BASIC programming to use your printer. The book assumes that you know a little programming but, if you're new to computers, don't worry — you'll pick it up quickly.

We're not going to try to make an expert out of you, just get you started. We chose BASIC because it is easy to learn and to use. Also, more personal computer users know BASIC than any other programming language.

Just a quick note — don't be offended by some of the vocabulary used in this manual. Not that there are any four-letter words but computer people, like other specialists, have their own jargon and some of the words may sound a little strange to newcomers ("unlisten" for instance).

COMMODORE COMMANDS

Commodore BASIC has four commands that will use to operate your printer — OPEN, CMD, PRINT #, and CLOSE. PRINT# is used with other commands, which we'll call printer commands when we define them later.

■ The OPEN command

When you want to print something, you first have to tell the computer that the printer is an output device. You can this by typing:

OPEN *lfn, dn, sa*

The OPEN command assigns a file number to a physical device. The *lfn*, or *logical file number*, may be any number from 1 to 255 — it doesn't matter which one you choose as long as you use it consistently until you **close** the file. The *dn*, or *device number*, tells the computer where you want to send the data — use 4 for the printer unless you set it to 5 using DIP switch 1-3. The last parameter is *sa*, which is used to select certain printer functions (see Appendix D).

■ The CMD command

Every CMD command is associated with an OPEN statement and causes the computer to send all of its output to the device with the specified logical device number (4 or 5 for the printer). (Normally it sends output to the display.)

Before you can use the CMD command, you must **open** a file. All output generated by the PRINT or LIST command will be stored in this file until you use a PRINT# command to "unlisten" the printer (break the connection between the printer and the computer [see the PRINT# command]). Here is the format of the command —

CMD *lfn*

You have to use the same value of *lfn* that you used in the OPEN statement. Unlike the PRINT command, the CMD command leaves the line or bus to the receiving device open. The line or bus to the receiving device (in this case, the printer) is said to be listening.

To re-direct output back to the screen, use the PRINT# command to send a blank line to the CMD device before you close the file. (If a syntax error occurs, output will not be redirected back to the screen.) Devices are not unlistened when an error occurs, so you should PRINT# a blank line after an error occurs.

■ The PRINT# command

The PRINT# command works just like the PRINT command except that it sends output to the printer instead of to the screen. After printing the specified data, the line or bus to the printer is unlistened. The syntax of the PRINT# command is:

PRINT# *lfn, data*

You should PRINT# an empty (blank) line before you **close** a file used by the CMD command to make sure that the printer is unlistened.

■ The CLOSE command

Every file that you **open** must be **closed** before you leave the program. After printing to the file, **close** it as follows:

CLOSE *lfn*

Commodore BASIC will let you have up to ten files open at a time. It's a good idea to close a file soon as you are finished with it. This way you always have the maximum number of files available and you avoid file errors.

■ Your printer's first words

We've learned something about communicating with our printer. Now we need to use what we know to print something from a BASIC program. Generally, computers use about the same procedure for printing program output as they do for listing a program.

Let's try to use what we've learned. Enter this short program into your computer:

```
10 REM    DEMO OF PRINTING
20 OPEN4,4
30 PRINT#4, "HEY!  THIS IS GREAT."
40 CLOSE4
RUN
```

Now that you have printed your honest opinion of your new printer, let's go on something more challenging.

ASCII CODES AND PRINTER COMMANDS

■ What is an ASCII code?

Your computer and printer talk to each other by exchanging electronic numbers. Most numbers from 0 to 255 have a certain meaning for computer equipment — 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 makes the printer buzz.

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 several ways to write an ASCII code, depending on how you are using it. We'll mention some of them soon in "Control codes:" For example, the number "0" is represented by the decimal number 48 in ASCII. Sometimes these codes are treated like regular numbers. Appendix B shows all of the ASCII codes.

BASIC uses ASCII codes with the CHR$ function. To print the character represented by an ASCII code, send PRINT CHR$ (*ASCII Code*) to the printer. The BASIC statement PRINT CHR$(48) will print a "0".

Try this little program and see what we mean:

```
10 REM    DEMO OF PRINTING AN ASCII CHARACTER
20 OPEN4,4
30 PRINT#4, CHR$(49)
40 CLOSE4
RUN
```

That should print a "1". If you check the chart in Appendix B, you'll see that 49 is the decimal ASCII code for "1".

ASCII codes may be written as hex numbers. "Hex" is short for *hexadecimal* and refers to a base-16 number (the numbers we use in everyday life are based on 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. (If there is no "&H" the number is a decimal number.) The ASCII code for the number "1" (49 in decimal) is &H31 in hex.

Of course, most of the time we don't need to bother with these codes — they are just the way that our computers handle information. We'll use some of them when we make our printer "jump through the hoop" later.

■ Control codes

Some of the ASCII codes don't have their own keys on the keyboard. The most important of these have values below 32 and control many of the printer's functions, so we call them control codes.

Let's try one right now.

```
10 REM    RING THE BELL
20 OPEN4,4
30 PRINT#4, CHR$(7)
40 CLOSE4
RUN
```

That's the printer's bell (we call it that even though it sounds like a buzzer). We'll learn more about it later ("The printer's bell"). We just wanted to show you a code that makes the printer do something you can noticeable.

Even though control codes don't have their own keys, we can enter them from the keyboard — hold down the "control" key and press another key at the same time. The other key that is pressed determine what code is sent. Pressing the control (CTRL) key and A sends ASCII code 1, CTRL B sends ASCII code 2, and so on.

There are four common ways of referring to a control code: the name of the code, the decimal ASCII value, the hexadecimal ASCII value, and the "CTRL—" value (see Appendix B). For example, the ASCII code that causes the printer to advance the paper one line is decimal 10. You can refer to this by any of the following names:

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

■ Escape code

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 Standard 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〉. (We'll also write other code names this way.)

〈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

As we introduce you to each new command, we will print it in the following form as shown here:

```
CHR$(27);CHR$(87);CHR$(49)
```

This is the command that turns on expanded printing. CHR$(27), as we mentioned earlier, is the escape code (〈ESC〉). As you'll learn in next section, this printer has three character sets. So an ASCII value represents three different characters depending on the character set.

For example, CHR$(87) prints "W" with the graphics character set and the ASCII character set, and "w" with the business character set. (For details, refer to Appendix B.)

By using the correct character in place of the CHR$ number, you can shorten the command string as shown below.

With the ASCII character mode:
CHR$(27);CHR$(87);CHR$(49)
CHR$(27);"W";"1"

With the graphics mode of Commodore mode:
CHR$(27);CHR$(87);CHR$(49)
CHR$(27);"W";"1"

With the business mode of Commodore mode:
CHR$(27);CHR$(87);CHR$(49)
CHR$(27);"w";"1"

These six commands will have the same result.

THE OPERATING MODE

■ ASCII mode vs Commodore mode

Your printer has two basic modes of operation: the Commodore mode and the ASCII mode. Normally you will use the Commodore mode, which is set automatically when you turn on the printer by setting the DIP switch 1-5 on. The ASCII operating mode — which has no functional relation to ASCII codes — provides certain other functions. You can switch the operating modes as follows.

Table 2-1
Operating mode commands

Function	Control code
Select ASCII operating mode	CHR$(27);CHR$(93);CHR$(49)
Select Commodore operating mode	CHR$(27);CHR$(93);CHR$(48)

The main difference between the two is the character set that the printer uses, but there are other differences: five printer commands do one thing in Commodore mode and something else in ASCII mode; five other printer commands work only in Commodore mode. We'll explain these as we go along.

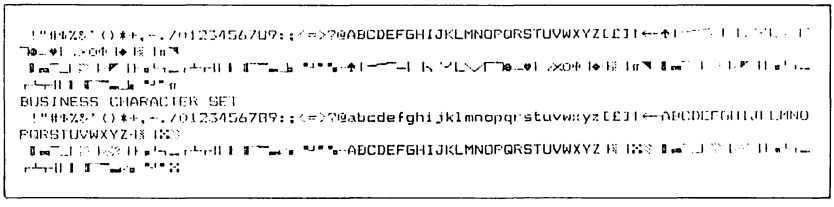
■ Character sets of the Commodore mode

The Commodore mode actually has two character sets that you can choose, so we really have three character sets at our

command. Try this program to see the Commodore character sets:

```
10 REM     CHARACTER SET
20 A$=""
30 FOR I=32 TO 127
40 A$=A$+CHR$(I)
50 NEXT I
60 B$=""
70 FOR I=160 TO 255
80 B$=B$+CHR$(I)
90 NEXT I
100 OPEN4,4
110 PRINT#4, CHR$(27);CHR$(93);CHR$(48);
120 PRINT#4, "GRAPHICS CHARACTER SET"
130 PRINT#4, CHR$(145);A$
140 PRINT#4, CHR$(145);B$
150 PRINT#4, "BUSINESS CHARACTER SET"
160 PRINT#4, CHR$(17);A$
170 PRINT#4, CHR$(17);B$
180 CLOSE4
```

When you run this program you should get like this:



In this program, line 110 selects the Commodore operating mode with the ⟨ESC⟩ CHR$(93) CHR$(48) command and lines 130 and 140 selects the graphics character set with CHR$(145). Then the characters are printed with the graphics character set. Lines 160 and 170 selects the business character set with CHR$(17) and this time the characters are printed with business character set.

Now turn off the printer and set the DIP switch 1-5 on. Then try this program.

```
10 REM     CHARACTER SET (2)
20 FOR I=32 TO 127
30 A$=A$+CHR$(I)
40 NEXT I
```

```
50 FOR J=160 TO 255
60 B$=B$+CHR$(J)
70 NEXT J
80 OPEN4,4
90 PRINT#4, A$
100 PRINT#4, B$
110 CLOSE4
120 OPEN7,4,7
130 PRINT#7, A$
140 PRINT#7, B$
150 CLOSE7
```

When you run this program, you should get the same result as the previous one.

When you turn on the printer with the Commodore operating mode, the graphics character set is selected.

The secondary address assignment of 7 in the OPEN statement changes all the graphics mode characters into business mode characters, as shown the program.

Table 2-2
Character set commands

Function	Control code
Select graphics character set	CHR$(145)
Select business character set	CHR$(17)

THE PRINT MODE

Now, we'll discuss the print mode which is completely different from the operating mode, which we just discussed. Don't confuse the two — the operating mode refers to large-scale printer operation (the character set, available functions, and so on); the print mode is smaller in scale, mostly determining character appearance and spacing.

■ Draft printing and NLQ printing

Basically, your printer can operate in either the draft mode or the NLQ print mode. Draft mode, which is set automatically when you turn on your printer, is good enough for most work. And draft mode printing is fast — 120 characters per second!

But for those special jobs that demand your best, you'll want

to use the NLQ mode, which provides "Near Letter Quality" characters. NLQ printing is perfect for correspondence, reports, and so forth — it takes a keen eye to tell that a dot-matrix printer did the work. NLQ printing is much slower than draft mode — 30 characters per second — but its quality is worth waiting for. To turn NLQ printing on or off, enter the following:

Table 2-3
Near letter quality commands

Function	Control code
Near letter quality ON	CHR$(27);CHR$(120);CHR$(49)
Near letter quality OFF	CHR$(27);CHR$(120);CHR$(48)

Here's what NLQ printing looks like:

```
NEW
10 REM    DEMO OF NLQ PRINTING
20 OPEN4,4
30 PRINT#4, CHR$(27);CHR$(120);CHR$(49);
40 PRINT#4, "THIS LINE SHOWS NEAR LETTER QUALITY! "
50 PRINT#4, CHR$(27);CHR$(120);CHR$(48);
60 PRINT#4, "THIS LINE SHOWS STANDARD DRAFT PRINT."
70 CLOSE4
```

When you run this program you should get this:

```
THIS LINE SHOWS NEAR LETTER QUALITY!
THIS LINE SHOWS STANDARD DRAFT PRINT.
```

To set NLQ mode manually, press the ON LINE key to put the printer off line (the ON LINE indicator goes out), then press the MODE key until the NLQ indicator lights. Press the ON LINE key again to set the printer on line (the ON LINE indicator lights again).

Of course, you can set one of the print pitches for the draft mode manually the same way. Or you can set them through software (we'll see how in the section on character spacing).

■ Underlining

Sometimes you want to emphasize special words. Print them with underline. Like the other special effects, underlining stays on until it is turned off. The control codes are shown in Table 2-4.

Table 2-4
Underline commands

Function	Control code
Underline ON	CHR$(27);CHR$(45);CHR$(49)
Underline OFF	CHR$(27);CHR$(45);CHR$(48)

Try this program:

```
10 REM    DEMO OF UNDERLINING
20 OPEN4,4
30 PRINT#4, "DEMONSTRATION OF ";
40 PRINT#4, CHR$(27);CHR$(45);CHR$(49);
50 PRINT#4, "UNDERLINED";
60 PRINT#4, CHR$(27);CHR$(45);CHR$(48);
70 PRINT#4, " PRINTING."
80 CLOSE4
```

Here is what you should get:

```
DEMONSTRATION OF UNDERLINED PRINTING.
```

In this program underline is turned on in line 40 with ⟨ESC⟩ CHR$(45) CHR$(49), and then off in line 60 with ⟨ESC⟩ CHR$(45) CHR$(48). There's a new little wrinkle in this program, though. It all printed on one line. The semicolons at the end of the first four 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.

■ Character spacing — fixed and proportional

Fixed spacing

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. And if you have squeeze a lot of information on a page, there is condensed printing which, at 17 characters per inch, gives you 136 columns of printing across an 8 1/2 inch page.

To set pica, elite, or condensed pitch by software, you have to send the printer some control codes as shown in the table below.

Table 2-5
Print pitch commands

Pitch	Characters/inch	Control code
Pica	10	CHR$(27);CHR$(80)
		CHR$(18)[ASCII mode only]
Elite	12	CHR$(27);CHR$(77)
Condensed	17	CHR$(27);CHR$(15)
		CHR$(15)[ASCII mode only]

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

```
10 REM     DEMO OF ALL PITCHES
20 OPEN4,4
30 PRINT#4, CHR$(27);CHR$(15);
40 PRINT#4, "THIS LINE IS CONDENSED PITCH."
50 PRINT#4, CHR$(27);CHR$(77);
60 PRINT#4, "THIS LINE IS ELITE PITCH."
70 PRINT#4, CHR$(27);CHR$(80);
80 PRINT#4, "THIS LINE IS PICA PITCH (NORMAL)."
90 CLOSE4
```

When you run this program you should get this:

```
THIS LINE IS CONDENSED PITCH.
THIS LINE IS ELITE PITCH.
THIS LINE IS PICA PITCH (NORMAL).
```

Line 30 turns on condensed pitch with ⟨ESC⟩ CHR$(15). Line 40 prints a line at 17 characters per inch. The ⟨ESC⟩ CHR$(77) in line 50 changes the printer to elite pitch and line 60 prints in elite pitch. Line 70 resets the printer to pica pitch and line 80 prints a line in pica pitch.

Pica pitch and condensed pitch can be set with "shortcut" codes in the ASCII mode. CHR$(18) sets pica pitch and CHR$(15) sets condensed pitch. You cannot set elite pitch with a single code.

Proportional spacing

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 printers), 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 2-6
Proportional commands

Function	Control code
Proportional ON	CHR$(27);CHR$(112);CHR$(49)
Proportional OFF	CHR$(27);CHR$(112);CHR$(48)

Try this program to see how the proportional spacing works.

```
10 REM    PROPORTIONAL SPACING
20 OPEN4,4
30 PRINT#4, CHR$(27);CHR$(112);CHR$(49);
40 PRINT#4, CHR$(27);CHR$(15);
50 PRINT#4, "THIS IS PROPORTIONAL WITH CONDENSED."
60 PRINT#4, CHR$(27);CHR$(77);
70 PRINT#4, "THIS IS PROPORTIONAL WITH ELITE."
80 PRINT#4, CHR$(27);CHR$(80);
90 PRINT#4, "THIS IS PROPORTIONAL WITH PICA."
100 PRINT#4, CHR$(27);CHR$(112);CHR$(48);
110 PRINT#4, "THIS IS NORMAL PICA PITCH."
120 CLOSE4
```

When you run this program you should get this:

```
THIS IS PROPORTIONAL WITH CONDENSED.
THIS IS PROPORTIONAL WITH ELITE.
THIS IS PROPORTIONAL WITH PICA.
THIS IS NORMAL PICA PITCH.
```

Line 30 turns on the proportional spacing with 〈ESC〉 CHR$(112) CHR$(49). Line 40 turns on condensed pitch and line 50 prints a line with condensed proportional pitch. Then, line 60 changes the printer to elite pitch and line 70 prints in elite proportional pitch. Line 80 resets the printer to pica pitch and line 90 prints a line in pica proportional pitch. Finally, line 100 resets the proportional spacing and line 110 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.

■ Expanded printing

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 REM    DEMO OF EXPANDED PRINT
20 OPEN4,4
30 PRINT#4, CHR$(27);CHR$(93);CHR$(49);
40 PRINT#4, "DEMONSTRATION OF ";
50 PRINT#4, CHR$(14);
60 PRINT#4, "EXPANDED";
70 PRINT#4, CHR$(20);
80 PRINT#4, " PRINTING."
90 PRINT#4, "NOTICE THAT ";
100 PRINT#4, CHR$(14);
110 PRINT#4, "EXPANDED PRINTING"
120 PRINT#4, "AUTOMATICALLY TURNS OFF AT THE END
    OF LINE."
130 CLOSE4
```

```
DEMONSTRATION OF EXPANDED PRINTING.
NOTICE THAT EXPANDED PRINTING
AUTOMATICALLY TURNS OFF AT THE END OF LINE.
```

In this program line 30 selects the ASCII mode, because this program doesn't work in the Commodore mode. Expanded print set with CHR$(14) is automatically cancelled at the end of the line. This is convenient in many applications, such as for one line titles. Note that you don't need to put an ⟨ESC⟩ in front of the CHR$(14).

You can also cancel one line expanded print *before* a carriage return with CHR$(20), as done in line 70.

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

```
10 REM    DEMO OF PREMANENT EXPANDED PRINT
20 OPEN4,4
30 PRINT#4, CHR$(27);CHR$(87);CHR$(49);
40 PRINT#4, "PERMANENT EXPANDED"
50 PRINT#4, "PRINT STAYS ON UNTIL"
60 PRINT#4, CHR$(27);CHR$(87);CHR$(48)
70 PRINT#4, "TURNED OFF."
80 CLOSE4
```

Now the results look like this:

```
PERMANENT EXPANDED
PRINT STAYS ON UNTIL
TURNED OFF.
```

When you turn on expanded print with ⟨ESC⟩ CHR$(87) CHR$(49) it stays on until you turn it off with ⟨ESC⟩ CHR$(87) CHR$(48).

Table 2-7
Expanded print commands

Function	Control code
One line expanded ON	CHR$(14)[ASCII mode only]
One line expanded OFF	CHR$(20)[ASCII mode only]
Expanded ON	CHR$(27);CHR$(87);CHR$(49)
	CHR$(14)[Commodore mode only]
Expanded OFF	CHR$(27);CHR$(87);CHR$(48)
	CHR$(15)[Commodore mode only]

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 2-8
Print emphasis commands

Function	Control code
Boldface ON	CHR$(27);CHR$(71)
Boldface OFF	CHR$(27);CHR$(72)
Emphasized ON	CHR$(27);CHR$(69)
Emphasized OFF	CHR$(27);CHR$(70)

Try them now with this little program:

```
10 REM     DEMO OF BOLDFACE AND EMPHASIZED
20 OPEN4,4
30 PRINT#4, CHR$(27);CHR$(71);
40 PRINT#4, "THIS LINE IS BOLDFACE PRINTING."
50 PRINT#4, CHR$(27);CHR$(69);
60 PRINT#4, "THIS LINE IS EMPHASIZED BOLDFACE
   PRINTING."
70 PRINT#4, CHR$(27);CHR$(72);
80 PRINT#4, "THIS LINE IS EMPHASIZED PRINTING."
90 PRINT#4, CHR$(27);CHR$(70);
100 PRINT#4, "THIS LINE IS NORMAL PRINTING."
110 CLOSE4
```

Run this program. The results will look like this.

```
THIS LINE IS BOLDFACE PRINTING.
THIS LINE IS EMPHASIZED BOLDFACE PRINTING.
THIS LINE IS EMPHASIZED PRINTING.
THIS LINE IS NORMAL PRINTING.
```

Line 30 turns on boldface with ⟨ESC⟩ CHR$(71) and line 40 prints a line of text. In line 50 emphasized is turned on with ⟨ESC⟩ CHR$(69). Line 60 prints a line of text in boldface *and* emphasized. Line 70 then turns boldface off with ⟨ESC⟩ CHR$(72) so that line 80 can print in emphasized only. Finally, line 90 turns emphasized off, so that 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?

■ Superscripts and subscripts

This 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 2-9 has the commands for using superscripts and subscripts.

Table 2-9
Superscripts and subscripts commands

Function	Control code
Superscript ON	CHR$(27);CHR$(83);CHR$(48)
Subscript ON	CHR$(27);CHR$(83);CHR$(49)
Superscript and subscript OFF	CHR$(27);CHR$(84)

Try this program to see them work:

```
10 REM    DEMO OF SUPER AND SUBSCRITPS
20 OPEN4,4
30 PRINT#4, "LOOK!  ";
40 PRINT#4, CHR$(27);CHR$(83);CHR$(48);
50 PRINT#4, "SUPERSCRIPTS";
60 PRINT#4, CHR$(27);CHR$(84);
70 PRINT#4, " & ";
80 PRINT#4, CHR$(27);CHR$(83);CHR$(49);
90 PRINT#4, "SUBSCRIPTS ";
```

```
100 PRINT#4, CHR$(27);CHR$(84);
110 PRINT#4, "ON ONE LINE."
120 CLOSE4
```

```
LOOK! SUPERSCRIPTS & SUBSCRIPTS ON ONE LINE.
```

Here line 40 turns on superscripts with ⟨ESC⟩ CHR$(83) CHR$(48). It's turned off in line 60 with ⟨ESC⟩ CHR$(84). Then, between printing text, subscripts are turned on in line 80 with ⟨ESC⟩ CHR$(83) CHR$(49), and finally off in line 100. Again, everything prints on one line because of the semicolons.

■ Printing in italics

Italic letters are the letters that are slanted on the right. Your printer can print the NLQ characters in italic as well as the roman (standard) letters we have been using. Italics can be used to give extra emphasis to certain words. The commands to turn on and off are shown in Table 2-10.

Table 2-10
Italic commands

Function	Control code
Italic ON	CHR$(27);CHR$(52)
Italic OFF	CHR$(27);CHR$(53)

Use this program to see italic characters:

```
10 REM    DEMO OF ITALICS AND ROMAN
20 OPEN4,4
30 PRINT#4, CHR$(27);CHR$(120);CHR$(49);
40 PRINT#4, "THESE ARE ";
50 PRINT#4, CHR$(27);CHR$(52);
60 PRINT#4, "ITALIC CHARACTERS";
70 PRINT#4, CHR$(27);CHR$(53);
80 PRINT#4, " AND ROMAN CHARACTERS."
90 CLOSE4
```

Here is what you should get:

```
THESE ARE ITALIC CHARACTERS AND ROMAN CHARACTERS.
```

This program is easy; line 30 selects the NLQ characters and line 50 turns italic on with ⟨ESC⟩ CHR$(52). Then line 70 turns it off with ⟨ESC⟩ CHR$(53).

■ Reverse printing

In Commodore mode, you can also print white letters on a black background. We call this *reverse printing*. The control codes are shown in Table 2-11.

Table 2-11
Reverse printing commands

Function	Control code
Reverse printing ON	CHR$(18)
Reverse printing OFF	CHR$(146)

Try this program to see how it works.

```
10 REM    DEMO OF REVERSE PRINTING
20 OPEN4,4
30 PRINT#4, CHR$(27);CHR$(93);CHR$(48);
40 PRINT#4, "DEMONSTRATION ";
50 PRINT#4, CHR$(18);
60 PRINT#4, "OF REVERSE";
70 PRINT#4, CHR$(146);
80 PRINT#4, " PRINTING."
90 CLOSE4
```

```
DEMONSTRATION ■■■■■■■■■■ PRINTING.
```

In this program line 30 selects the Commodore mode, because the reverse printing don't work in the ASCII mode. When you turn on reverse printing with CHR$(18), it stays on until you turn it off with CHR$(146).

WARNINGS

1) Do not use reverse printing for more than five consecutive lines. This is very hard on the print head and may damage it.
2) You cannot print true descenders in reverse printing.

■ 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:

CHR$(27);CHR$(33);CHR$(*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 2-12 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 2-12
Values of mixing print styles for Master Print

Bit	Print style	Decimal value
1	Elite pitch	1
2	Proportional pitch	2
3	Condensed pitch	4
4	Emphasized	8
5	Boldface	16
6	Expanded	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
n=	49

The command would look like this:

```
CHR$(27);CHR$(33);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 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 and Condensed are turned off.
2) When two modes conflict, the one of lesser priority is cancelled. For example, Condensed and Emphasized can not be printed at the same time, printing is in Emphasized.
3) Elite cancels Emphasized.
4) Underline, and Expanded Modes combine with any print modes.
5) Emphasized will not mix with Elite or Condensed.

MEMO

CHAPTER 3

FORMATTING TEXT

Subjects we'll cover in Chapter 3 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 2 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 1-1 is off; 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 1-1 on so that your printer will add the line feed for you. When you have DIP switch 1-1 on 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 1-1 off.

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 CHR$(27);CHR$(10), 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 3-1
Line feed commands

Function	Control code
Return print head to left margin	CHR$(13)
Advance paper one line	CHR$(10)
Reverse paper one line	CHR$(27);CHR$(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:

```
NEW
10 REM     DEMO OF LINE SPACING
20 OPEN4,4
30 FOR I=1 TO 25
40 IF I=13 THEN 70
50 PRINT#4, CHR$(27);CHR$(65);CHR$(I);
60 PRINT#4, "THIS LINE SPACING IS SET TO";I
70 NEXT I
80 PRINT#4, "LINE SPACING IS SET TO 1/6 INCH
   (NORMAL)."
90 PRINT#4, CHR$(27);CHR$(50)
100 CLOSE4
```

This is what you will get:

```
THIS LINE SPACING IS SET TO 1
THIS LINE SPACING IS SET TO 2
THIS LINE SPACING IS SET TO 3
THIS LINE SPACING IS SET TO 4
THIS LINE SPACING IS SET TO 5
THIS LINE SPACING IS SET TO 6
THIS LINE SPACING IS SET TO 7
THIS LINE SPACING IS SET TO 8
THIS LINE SPACING IS SET TO 9
THIS LINE SPACING IS SET TO 10
THIS LINE SPACING IS SET TO 11
THIS LINE SPACING IS SET TO 12
THIS LINE SPACING IS SET TO 14
THIS LINE SPACING IS SET TO 15
THIS LINE SPACING IS SET TO 16
THIS LINE SPACING IS SET TO 17
THIS LINE SPACING IS SET TO 18
THIS LINE SPACING IS SET TO 19
THIS LINE SPACING IS SET TO 20
THIS LINE SPACING IS SET TO 21
THIS LINE SPACING IS SET TO 22
THIS LINE SPACING IS SET TO 23
THIS LINE SPACING IS SET TO 24
THIS LINE SPACING IS SET TO 25
LINE SPACING IS SET TO 1/6 INCH (NORMAL).
```

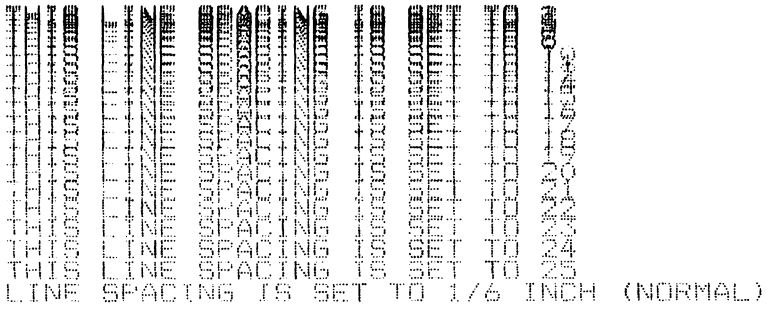
Line 50 changes the line spacing. The command CHR$(27);CHR$(65);CHR$(*n*) changes the line spacing to *n*/72 of an inch. The loop that is started in line 30 increases the value of *n* (the variable I in this program) each time it is executed. So the line spacing increases as the program continues. Line 40 just shortcuts the loop when I=13, since BASIC won't let us send CHR$(13) without adding an unwanted CHR$(10) to it. Finally, the CHR$(27);CHR$(50) in line 90 resets the line spacing to 6 lines per inch. This is a shortcut that is the same as CHR$(27);CHR$(65);CHR$(12).

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 CHR$(27);CHR$(51);CHR$(*n*) command sets the line spacing in increments of 1/216 inch. Change line 50 in your program so it is like this:

```
50 PRINT#4, CHR$(27);CHR$(51);CHR$(I);
```

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



The program works just the same as before, but the line spacing are just one-third what they were. This is because CHR$(27);CHR$(51);CHR$(*n*) sets the line spacing to *n*/216 inch.

Table 3-2 shows all the line spacing commands, including several "shortcut" commands for commonly used line spacings.

Table 3-2
Line spacing commands

Function	Control code
Set line spacing to 1/8 inch	CHR$(27);CHR$(48)
Set line spacing to 7/72 inch	CHR$(27);CHR$(49)
Set line spacing to 1/6 inch	CHR$(27);CHR$(50)
Set line spacing to *n*/72 inch	CHR$(27);CHR$(65);CHR$(*n*)
Set line spacing to *n*/216 inch	CHR$(27);CHR$(51);CHR$(*n*)
One-time line feed of *n*/216 inch	CHR$(27);CHR$(74);CHR$(*n*)

■ 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 command does just that.

The CHR$(27);CHR$(74);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 REM    DEMO OF ONE-TIME LINE FEEDS
20 OPEN4,4
30 PRINT#4, "LINE NUMBER 1."
40 PRINT#4, "LINE NUMBER 2.";
50 REM    ONE-TIME LINE FEED
60 PRINT#4, CHR$(27);CHR$(74);CHR$(100);
70 PRINT#4, "LINE NUMBER 3."
80 PRINT#4, "LINE NUMBER 4."
90 CLOSE4
```

Here is what your printer will produce:

```
LINE NUMBER 1.
LINE NUMBER 2.


                    LINE NUMBER 3.
LINE NUMBER 4.
```

The CHR$(27);CHR$(74);CHR$(100) in line 60 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 40 and line 60 end with semicolons. This prevents the normal line feed from occurring.

PAGE CONTROL

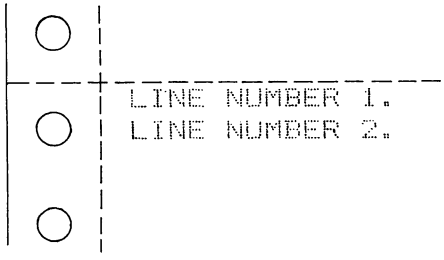
Now that we have seen how to control line spacing, we can go on to page control — positioning the printing on the page and adjusting the paper length.

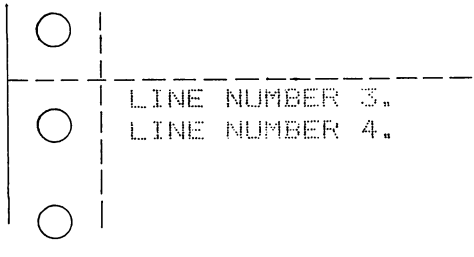
■ Form feed

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 50 and 60 to this:

```
50 REM     FORM FEED
60 PRINT#4, 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 the separated manual. When you run the program, the results will look like this:





The form feed (CHR$(12)) in line 60 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 used, 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>— CHR$(27);CHR$(12).

Table 3-3
Form feed commands

Function	Control code
Advance paper to top of next page	CHR$(12)
Reverse paper to top of current page	CHR$(27);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 3-4
Form length control

Function	Control code
Set the page length to *n* lines	CHR$(27);CHR$(67);CHR$(*n*)
Set the page length to *n* inches	CHR$(27);CHR$(67);CHR$(0) CHR$(*n*)

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 REM     VARIABLE FORM LENGTH
20 OPEN4,4
30 PRINT#4, CHR$(27);CHR$(67);CHR$(0);CHR$(7);
40 PRINT#4, "PAY TO THE ORDER OF:"
50 PRINT#4, CHR$(12);
60 PRINT#4, "PAY TO THE ORDER OF:"
70 CLOSE4,4
```

This program should print "PAY TO THE ORDER OF:" twice, and they should be 7 inches apart. Line 30 sets the form length to 7 inches. After line 40 prints, line 50 sends a form feed advance the paper to the top of the next form. Line 60 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 3-5
Top and bottom margin commands

Function	Control code
Set top margin to *n* lines	CHR$(27);CHR$(114);CHR$(*n*)
Set bottom margin to *n* lines	CHR$(27);CHR$(78);CHR$(*n*)
Set bottom margin to 6 lines	CHR$(147)
Cancel top and bottom margins	CHR$(27);CHR$(79)
	CHR$(19)

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 CHR$(27);CHR$(114);CHR$(*n*), the value of *n* tells the printer what line to start printing on. When you set the bottom margin with CHR$(27);CHR$(78);CHR$(*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 REM     DEMO OF TOP AND BOTTOM MARGINS
20 OPEN4,4
70 PRINT#4, CHR$(12); :REM    FORM FEED
80 FOR I=1 TO 150
90 PRINT#4, "THIS IS LINE";I
100 NEXT I
120 PRINT#4, CHR$(12); :REM    FORM FEED
```

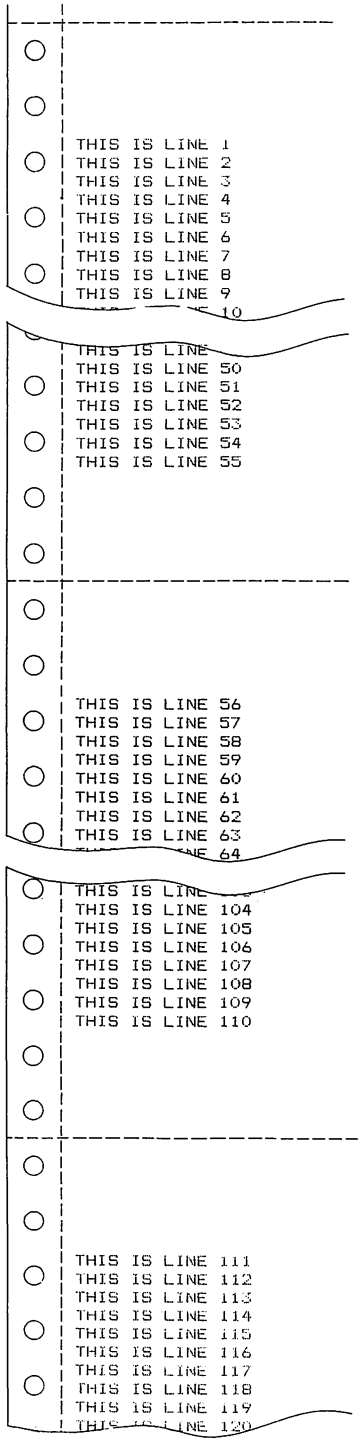
When you run this program it will print 150 lines right down the page and across the perforations. When it's done line 120 sends a form feed to advance to the top of the next page. Look at the lines that have printed near the perforations. Separate the sheets and see if any of the lines have been torn 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.)

```
30 REM   LEAVE 6 BLANK LINES AT BOTTOM OF PAGE.
40 PRINT#4, CHR$(27);CHR$(78);CHR$(6);
50 REM   START TOP OF PAGE AT LINE 6.
60 PRINT#4, CHR$(27);CHR$(114);CHR$(6);
110 PRINT#4, CHR$(27);CHR$(79); :REM   CLEAR
    TOP & 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 60 sets the top margin, line 40 sets the bottom margin, and line 110 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 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 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

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 3-6
Left and right margin commands

Function	Control code
Set left margin at column *n*	CHR$(27);CHR$(108);CHR$(*n*)
Set right margin at column *n*	CHR$(27);CHR$(81);CHR$(*n*)

Try setting the margins with this program:

```
10 REM    DEMO OF MARGINS
20 OPEN4,4
30 GOSUB 80
40 PRINT#4, CHR$(27);CHR$(108);CHR$(10);
50 PRINT#4, CHR$(27);CHR$(81);CHR$(70);
60 GOSUB 80
70 CLOSE4:END
80 FOR I=1 TO 80
90 PRINT#4, "X";
100 NEXT I
110 PRINT#4
120 RETURN
```

The first thing that this program does is to branch to the subroutine that starts in line 80 . 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 40 sets the left margin to 10, and line 50 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:

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
          XXXXXXXXXXXXXXXXXXXX
```

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) with the ASCII mode.

Try this program to see how the default tabs work.

```
NEW
10 REM     DEMO OF HORIZONTAL TABBING
20 OPEN4,4
30 PRINT#4, CHR$(27);CHR$(93);CHR$(49);
50 PRINT#4, "ONE";CHR$(9);"TWO";CHR$(9);"THREE"
   ;CHR$(9);"FOUR"
60 CLOSE4
```

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:

```
40 PRINT#4, CHR$(27);CHR$(68);CHR$(7);CHR$(14)
   ;CHR$(21);CHR$(0)
```

CHR$(27);CHR$(68) 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 3-7
Horizontal tab commands

Function	Control code
Advance to next tab position	CHR$(9) [ASCII mode only]
Set tabs at *n1*, *n2*, etc.	CHR$(27);CHR$(68);CHR$(*n1*); CHR$(*n2*);·····;CHR$(0)

■ 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 REM     DEMO OF VERTICAL TABS
20 OPEN4,4
50 PRINT#4, CHR$(11);"FIRST TAB."
60 PRINT#4, CHR$(11);"SECOND TAB."
70 PRINT#4, CHR$(11);"THIRD TAB."
80 PRINT#4, CHR$(11);"FOURTH TAB."
100 CLOSE4
```

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

```
30 PRINT#4, CHR$(27);CHR$(66);CHR$(10);
40 PRINT#4, CHR$(20);CHR$(40);CHR$(50);CHR$(0);
```

CHR$(27);CHR$(66) 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.

```
90 PRINT#4, CHR$(11);"FIFTH TAB."
```

Now when you run the program the first page looks just like before, but line 90 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 3-8
Vertical tab commands

Function	Control code
Advance paper to next tab position	CHR$(11)
Set vertical tabs at *n1*, *n2*, etc.	CHR$(27);CHR$(66);CHR$(*n1*); CHR$(*n2*);·····;CHR$(0)

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 3-9
Aligning commands

Function	Control code
Left-aligned printing	CHR$(27);CHR$(97);CHR$(0)
Centered printing	CHR$(27);CHR$(97);CHR$(1)
Right-aligned printing	CHR$(27);CHR$(97);CHR$(2)

Try this program to see how easy it is.

```
NEW
10 REM     ALIGNING AND CENTERING TEXT
20 OPEN4,4
30 PRINT#4, CHR$(27);CHR$(108);CHR$(20);
40 PRINT#4, CHR$(27);CHR$(81);CHR$(60);
50 PRINT#4, CHR$(27);CHR$(97);CHR$(0);
```

```
60 PRINT#4, "THIS LINE IS LEFT-ALIGNED."
70 PRINT#4, CHR$(27);CHR$(97);CHR$(1);
80 PRINT#4, "THIS LINE IS CENTERED."
90 PRINT#4, CHR$(27);CHR$(97);CHR$(2);
100 PRINT#4, "THIS LINE IS RIGHT-ALIGNED."
110 CLOSE4
RUN
```

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 4

SPECIAL FEATURES OF THE PRINTER

Subjects we'll cover in Chapter 4 include—

- **Bell;**
- **Master reset;**
- **Backspace;**
- **Printing zeroes;**
- **International character sets;**
- **Printing BIG characters;**
- **The optional sheet feeder;**
- **Quotation marks;**
- **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:

```
10 OPEN4,4
20 PRINT#4, CHR$(7)
30 CLOSE4
```

■ 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 〈ESC〉 CHR$(64) will reset all of the printer's features to the power on condition (as determined by the DIP switches), with two exceptions. Those exceptions are that 〈ESC〉 CHR$(64) 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 Panel mode", or "NLQ Italic" by the control panel settings at the power on, these functions will be remain with this control code.

■ Backspace

Backspace (CHR$(8)) "backs up" the printhead in the ASCII mode 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.

The following program shows how this code works.

```
10 REM    BACKSPACE
20 OPEN4,4
30 PRINT#4, CHR$(27);CHR$(93);CHR$(49);
40 PRINT#4, "BACKSPACE DOES NOT";
50 PRINT#4, CHR$(8);CHR$(8);CHR$(8);
60 PRINT#4, "=== WORK."
70 CLOSE4
```

Here is what this program will print:

```
BACKSPACE DOES NOT WORK.
```

The backspace codes in line 50 move the printhead a total of three spaces to the left so that the first part of line 60 will overprint the word "NOT".

■ 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).

Table 4-1
Some miscellaneous commands

Function	Control code
Sound bell	CHR$(7)
Master reset	CHR$(27);CHR$(64)
Move printhead back one space	CHR$(8)[ASCII mode only]
Print "slash zero"	CHR$(27);CHR$(126);CHR$(49)
Print "normal zero"	CHR$(27);CHR$(126);CHR$(48)

■ International character sets

Your printer is a multi-lingual printer for it can speak in eight languages! This printer changes languages by changing many 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 4-2.

Table 4-2
International character set commands

Country	Control code
Commodore standard	CHR$(27);CHR$(82);CHR$(0)
U.S.A	CHR$(27);CHR$(82);CHR$(1)
Germany	CHR$(27);CHR$(82);CHR$(2)
Demark	CHR$(27);CHR$(82);CHR$(3)
France	CHR$(27);CHR$(82);CHR$(4)
Sweden	CHR$(27);CHR$(82);CHR$(5)
Italy	CHR$(27);CHR$(82);CHR$(6)
Spain	CHR$(27);CHR$(82);CHR$(7)

The characters that change are shown beneath their ASCII code in Table 4-3.

Table 4-3
International character sets

Country	35	36	64	91	92	93	123	124	125	126	219	220	221	222
(In case of graphics characters)														
Commodore standard	#	$	@	[	£	]	┼	▒	|	π	┼	▒	|	π
U.S.A	#	$	@	[	\	]	┼	▒	|	π	┼	▒	|	π
Germany	#	$	§	Ä	Ö	Ü	┼	▒	|	π	┼	▒	|	π
Demark	#	$	@	Æ	Ø	Å	┼	▒	|	π	┼	▒	|	π
France	#	$	à	°	Ç	§	┼	▒	|	π	┼	▒	|	π
Sweden	#	¤	É	Ä	Ö	Å	┼	▒	|	π	┼	▒	|	π
Italy	#	$	@	°	\	é	┼	▒	|	π	┼	▒	|	π
Spain	₧	$	@	¡	Ñ	¿	┼	▒	|	π	┼	▒	|	π
(In case of business characters)														
Commodore standard	#	$	@	[	£	]	┼	▒	|	▚	┼	▒	|	▚
U.S.A	#	$	@	[	\	]	{	¦	}	~	{	¦	}	~
Germany	#	$	§	Ä	Ö	Ü	ä	ö	ü	ß	ä	ö	ü	ß
Demark	#	$	@	æ	ø	å	æ	ø	å	~	Æ	Ø	Å	~
France	#	$	à	°	Ç	§	é	ù	è	¨	é	ù	è	¨
Sweden	#	¤	É	Ä	Ö	Å	ä	ö	å	ü	ä	ö	å	ü
Italy	#	$	@	°	\	é	à	ò	è	ì	à	ò	è	ì
Spain	₧	$	@	¡	Ñ	¿	¨	ñ	}	~	¨	ñ	}	~
(In case of ASCII characters)														
Commodore standard	#	$	@	[	\	]	{	¦	}	~	┼	▒	|	▚
U.S.A	#	$	@	[	\	]	{	¦	}	~	{	¦	}	~
Germany	#	$	§	Ä	Ö	Ü	ä	ö	ü	ß	ä	ö	ü	ß
Demark	#	$	@	Æ	Ø	Å	æ	ø	å	~	æ	ø	å	~
France	#	$	à	°	Ç	§	é	ù	è	¨	é	ù	è	¨
Sweden	#	¤	É	Ä	Ö	Å	ä	ö	å	ü	ä	ö	å	ü
Italy	#	$	@	°	\	é	à	ò	è	ì	à	ò	è	ì
Spain	₧	$	@	¡	Ñ	¿	¨	ñ	}	~	¨	ñ	}	~

■ 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.

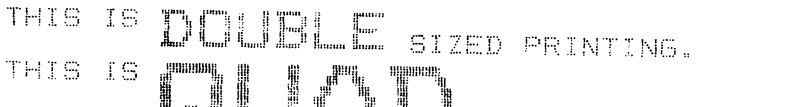
Table 4-4
Big character commands

Function	Control code
Double-high enlarged print	CHR$(27);CHR$(104);CHR$(1)
Quad-high enlarged print	CHR$(27);CHR$(104);CHR$(2)
Double-high lower-half enlarged print	CHR$(27);CHR$(104);CHR$(3)
Double-high upper-half enlarged print	CHR$(27);CHR$(104);CHR$(4)
Quad-high lower-half enlarged print	CHR$(27);CHR$(104);CHR$(5)
Quad-high upper-half enlarged print	CHR$(27);CHR$(104);CHR$(6)
Cancel enlarged print	CHR$(27);CHR$(104);CHR$(0)

Try this program to see the big characters.

```
10 REM   DEMO OF BIG CHARACTERS
20 OPEN4,4
30 PRINT#4, "THIS IS ";
40 PRINT#4, CHR$(27);CHR$(104);CHR$(1);
50 PRINT#4, "DOUBLE";
60 PRINT#4, CHR$(27);CHR$(104);CHR$(0);
70 PRINT#4, " SIZED PRINTING."
80 PRINT#4, "THIS IS ";
90 PRINT#4, CHR$(27);CHR$(104);CHR$(2);
100 PRINT#4, "QUAD";
110 PRINT#4, CHR$(27);CHR$(104);CHR$(0);
120 PRINT#4, " SIZED PRINTING."
130 CLOSE4
```

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



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 REM   DEMO OF BIG CHARACTERS (2)
20 OPEN4,4
30 PRINT#4, "THIS IS ";
40 PRINT#4, CHR$(27);CHR$(65);CHR$(7);
50 PRINT#4, CHR$(27);CHR$(10);
60 PRINT#4, "        ";
70 PRINT#4, CHR$(27);CHR$(104);CHR$(1);
80 PRINT#4, "DOUBLE";
90 PRINT#4, CHR$(27);CHR$(104);CHR$(0);
100 PRINT#4, CHR$(27);CHR$(65);CHR$(1);
110 PRINT#4, CHR$(27);CHR$(10);
120 PRINT#4, "                    ";
130 PRINT#4, " SIZED PRINTING.";
140 PRINT#4, CHR$(27);CHR$(50)
150 PRINT#4, : PRINT#4
160 PRINT#4, "THIS IS ";
170 PRINT#4, CHR$(27);CHR$(65);CHR$(21);
180 PRINT#4, CHR$(27);CHR$(10);
190 PRINT#4, "        ";
200 PRINT#4, CHR$(27);CHR$(104);CHR$(2);
210 PRINT#4, "QUAD";
220 PRINT#4, CHR$(27);CHR$(104);CHR$(0);
230 PRINT#4, CHR$(27);CHR$(65);CHR$(3);
240 PRINT#4, CHR$(27);CHR$(10);
250 PRINT#4, "                        ";
260 PRINT#4, " SIZED PRINTING."
270 PRINT#4, CHR$(27);CHR$(64)
280 CLOSE4
```

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

■ 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 that 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 4-5
Automatic sheet feeder commands

Function	Control code
Select automatic feed mode	CHR$(27);CHR$(25);CHR$(4)
Cancel automatic feed mode	CHR$(27);CHR$(25);CHR$(0)
Insert paper	CHR$(27);CHR$(25);CHR$(1)
Eject paper	CHR$(27);CHR$(25);CHR$(82)

When the automatic sheet feeder is installed, you must set the DIP switch 1-2 on to detect the paper-out.

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

■ Printing quotation marks

Perhaps when we learned to print words directly by placing them in quotation marks you wondered how to print the quotation marks themselves. These marks can only be printed with CHR$(34). Quotation marks can make your program's output much more effective, but use them logically and don't overdo it. Try this program —

```
10 REM    DEMO OF QUOTATION MARKS
20 OPEN4,4
30 PRINT#4, CHR$(34);"ABCD";CHR$(34)
40 GOSUB 120
50 PRINT#4, CHR$(34);"PRINTER;CHR$(34)
60 GOSUB 120
70 PRINT#4, CHR$(34);EFGH;CHR$(34)
80 GOSUB 120
90 FOR I=1 TO 8:PRINT#4: NEXT I
```

```
100 CLOSE4
110 END
120 FOR J=1 TO 2
130 PRINT#4: NEXT J
140 RETURN
```

When you run the program you will get like this:

```
"ABCD"

"PRINTER;CHR$(34)

" 0 "
```

Line 30 prints "ABCD". Line 50 prints "PRINTER; CHR$(34)" because both PRINTER and CHR$(34) are enclosed in their own quotes. Line 70 prints "0" because EFGH is seen by the computer as a numeric variable since it is not enclosed within quotes and its contents are naturally zero.

If an odd number of quotation marks have been transmitted, control characters are made visible. This can be particularly useful when you are making a listing a BASIC program containing control characters in quotation marks instead of using the CHR$ function.

■ 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 jargonese *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 in 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 4-6.

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.

Table 4-6
Macro instruction commands

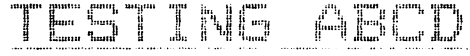
Function	Control code
Define macro	CHR$(27);CHR$(43);...(*codes you include*) ...;CHR$(30)
Use macro	CHR$(27);CHR$(43);CHR$(1)

```
10 OPEN4,4
20 PRINT#4, CHR$(27);CHR$(43);
30 PRINT#4, CHR$(27);CHR$(104);CHR$(0);
40 PRINT#4, CHR$(27);CHR$(33);CHR$(0);
50 PRINT#4, CHR$(27);CHR$(84);
60 PRINT#4, CHR$(27);CHR$(50);
70 PRINT#4, CHR$(27);CHR$(97);CHR$(0);
80 PRINT#4, CHR$(30)
90 CLOSE4
```

In this program, we started to define macro in line 20. Line 30 cancels the big character printing. Line 40 sets the normal pica, and also this command cancels the proportional pitch, expanded print, boldface, emphasized, and the underlining. Line 50 cancels the superscripts and the subscripts. Line 60 sets the line spacing to 1/6 inch. Line 70 sets the left-aligned printing. Then, line 80 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 70. When line 90 prints the style is "plain vanilla" because the macro has reset it.

```
10 OPEN4,4
20 PRINT#4, CHR$(27);CHR$(81);CHR$(40);
30 PRINT#4, CHR$(27);CHR$(97);CHR$(2);
40 PRINT#4, CHR$(27);CHR$(45);CHR$(49);
50 PRINT#4, CHR$(27);CHR$(104);CHR$(1);
60 PRINT#4, "TESTING ABCD"
70 PRINT#4, CHR$(27);CHR$(43);CHR$(1);
80 PRINT#4, "TESTING ABCD"
90 CLOSE4
```



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 and Mode keys and turn on the printer.

```
10 OPEN4,4
20 FOR I=0 TO 127
30 A$=A$+CHR$(I)
40 NEXT I
50 FOR J=128 TO 255
60 B$=B$+CHR$(J)
70 NEXT J
80 PRINT#4, A$;B$
90 CLOSE4
```

Your printer should give you the following information. (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     `abcdefghijklmno
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   A8 A9 AA AB AC AD AE AF     ................
B0 B1 B2 B3 B4 B5 B6 B7   B8 B9 BA BB BC BD BE BF     ................
C0 C1 C2 C3 C4 C5 C6 C7   C8 C9 CA CB CC CD CE CF     ................
D0 D1 D2 D3 D4 D5 D6 D7   D8 D9 DA DB DC DD DE DF     ................
E0 E1 E2 E3 E4 E5 E6 E7   E8 E9 EA EB EC ED EE EF     ................
F0 F1 F2 F3 F4 F5 F6 F7   F8 F9 FA FB FC FD FE FF     ................
0D                                                    .
```

MEMO

CHAPTER 5

CREATING YOUR OWN CHARACTERS

Subjects we'll cover in Chapter 5 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 4, 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 5-1 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 5-2 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).

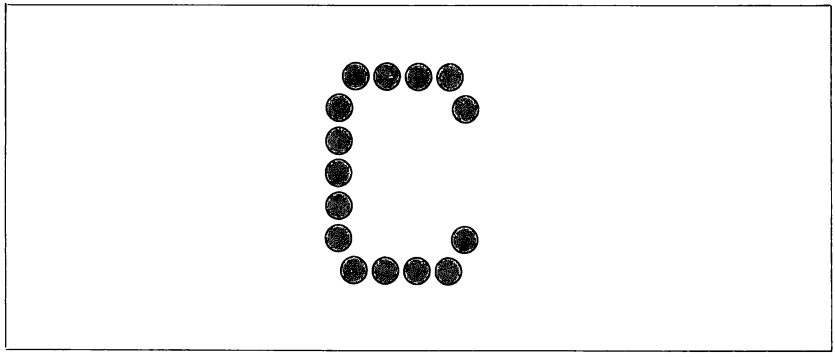


Figure 5-1. The letter "C" is created by printing 15 dots.

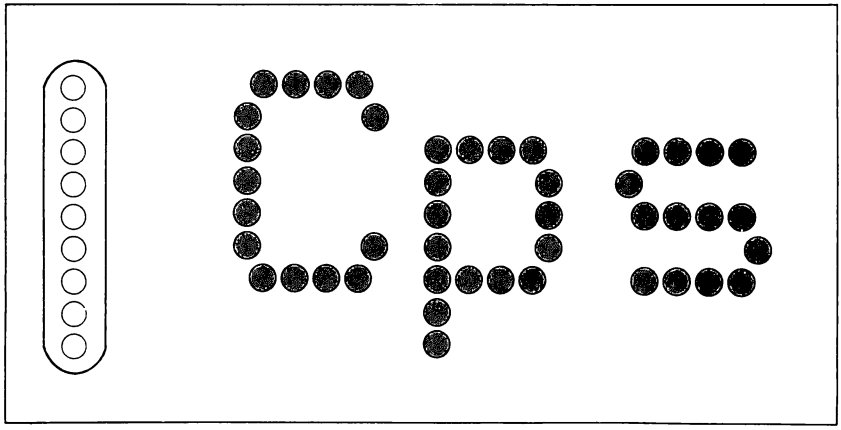


Figure 5-2. As the printhead moves across the page, each of the wires prints one row of dots.

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, the block graphics and special chara-

cters, 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 5-3. 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.

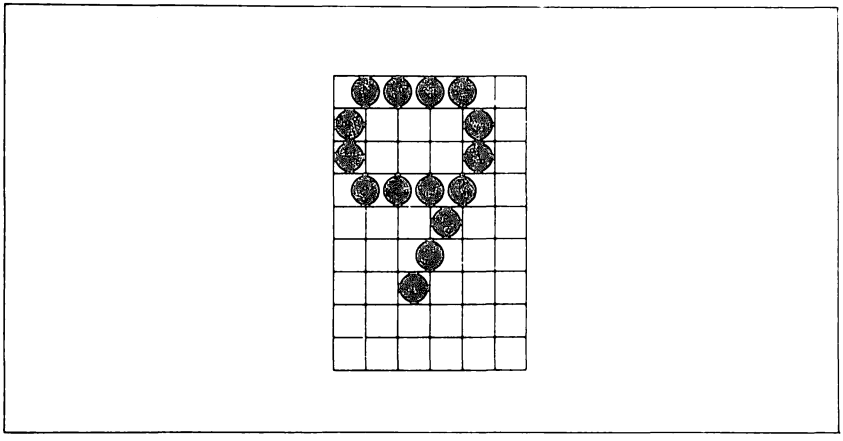


Figure 5-3. Dots can be inside boxes or straddle the vertical lines of the grid.

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 5-4 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 5-5. Later in this chapter we'll use this character to create a small graph.)

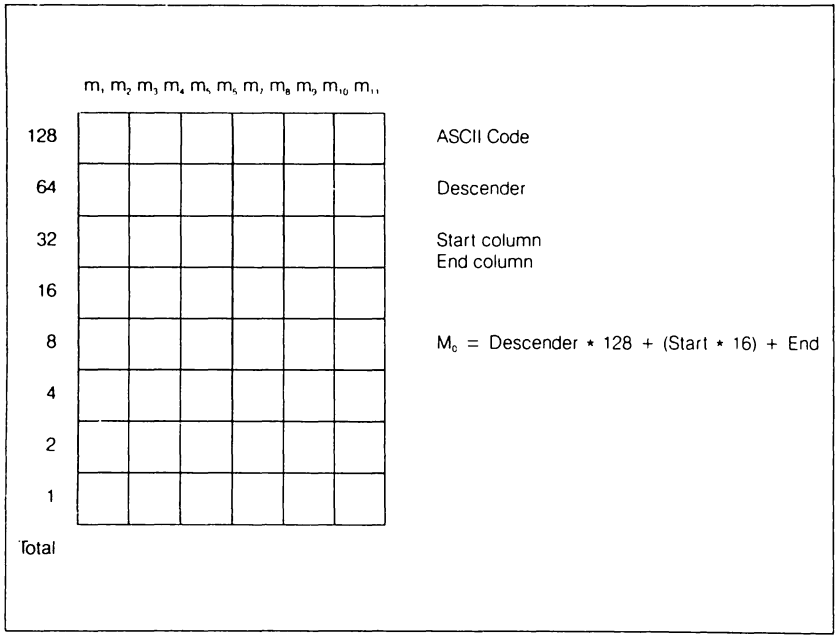


Figure 5-4. Use this grid (or one similar to it) to define your own draft characters.

You'll notice that Figure 5-4 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 show in Figure 5-3: it's only eight boxes high. Which leads us to ...

■ Rule 1: Draft download characters are eight dots high

As you noticed in Figure 5-2, 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 5-5, we've written in a "1" on out grid.

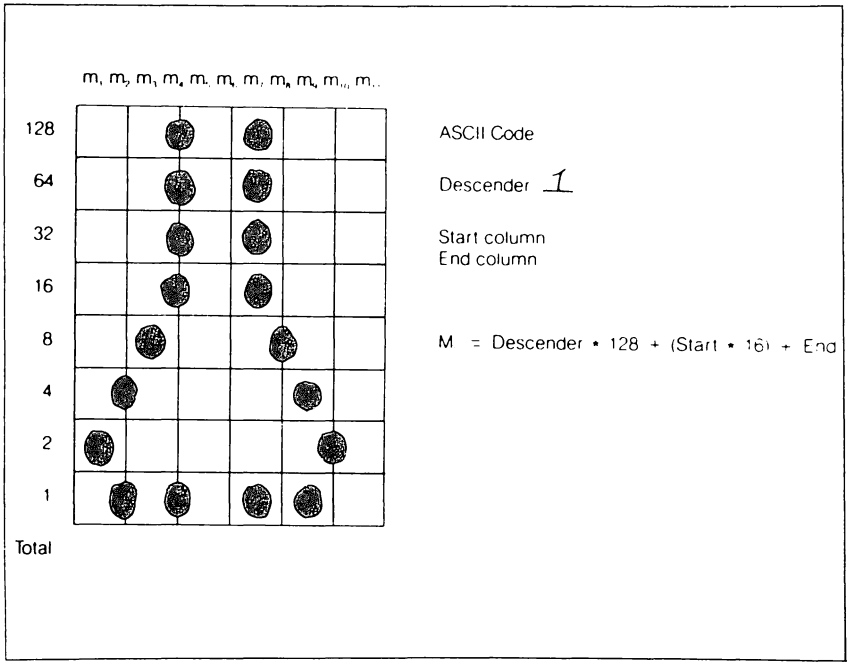


Figure 5-5. We've designed a character and decided that it would not be a descender, hence the "1" written in.

■ Rule 2: Dots cannot overlap

As you can see in Figure 5-5 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 5-6? 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 5-6. (Actually, you can define it, but when it prints, your printer will leave out the overlapping dots, so that it would print like Figure 5-5.)

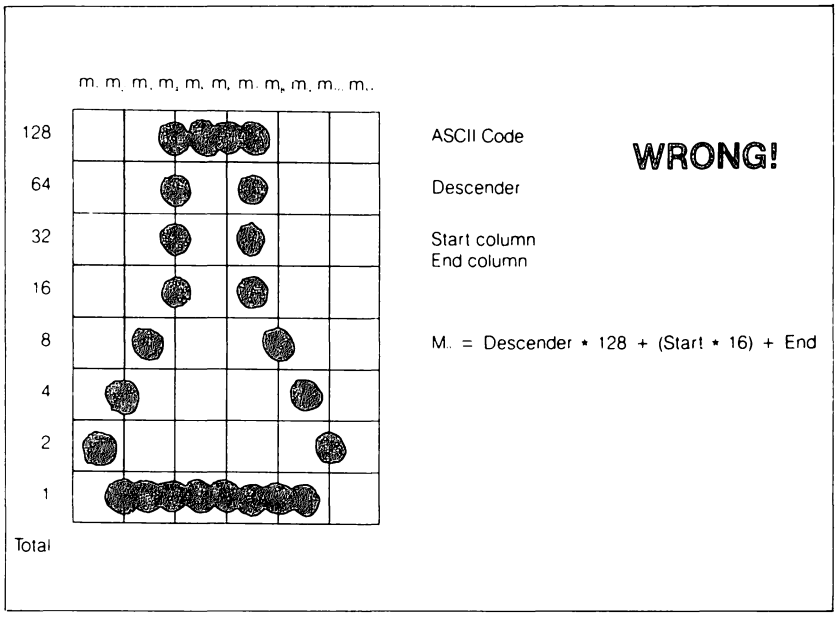


Figure 5-6. Dots cannot overlap; those in immediately adjacent "half columns" will be ignored when the character is printed.

■ 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 5-7, 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.

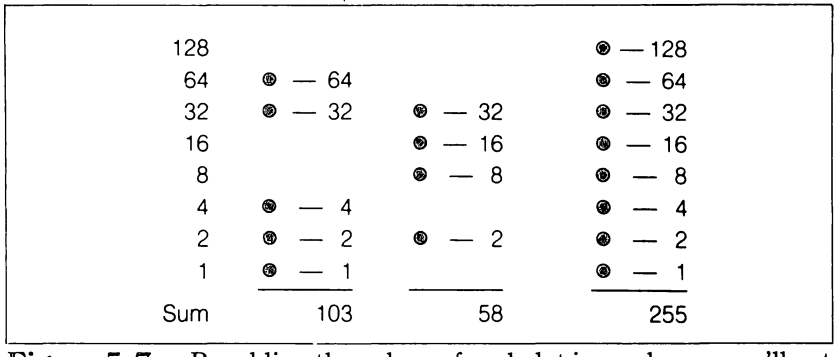


Figure 5-7. By adding the values of each dot in a column, you'll get a unique description for any combination of dots.

So add up the values of the dots in each column using this system. In Figure 5-8 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.

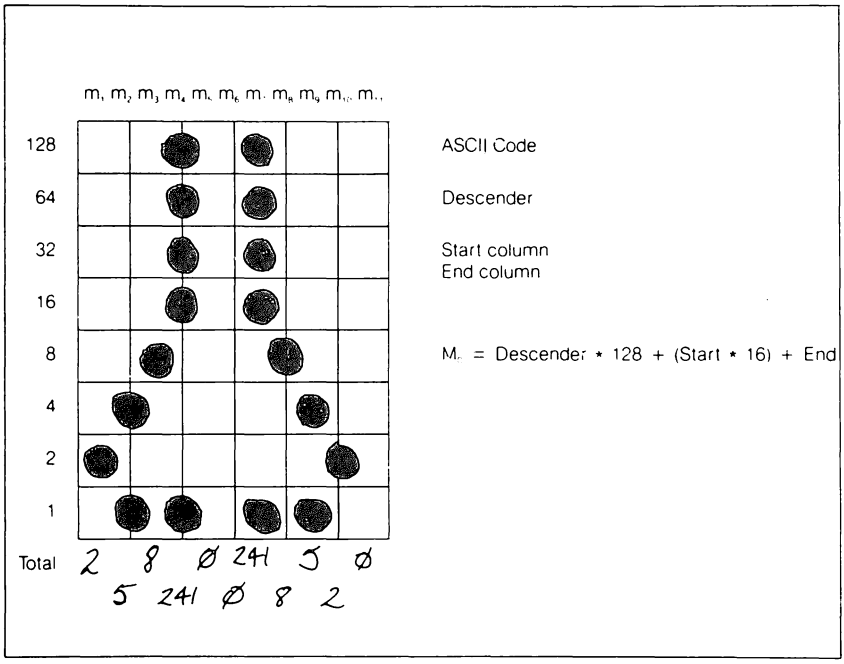


Figure 5-8. Add the values of the dots in each column and write the sum of each column at the bottom.

■ Assigning a value to your character

We've done a pretty through job of designing and describing a user-defined character. But this printer has room for 96 download characters — how does it knows 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 define any positions between 32 to 127. 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.

Except for the limitation that download characters must be between 32 and 127, 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 5-9 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).

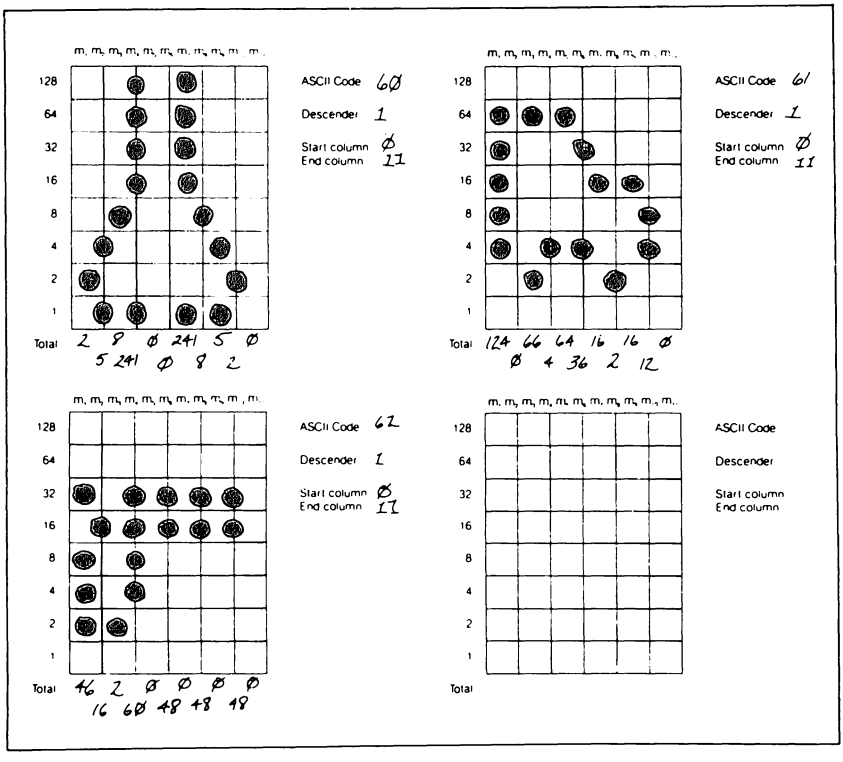


Figure 5-9. Character designs for the three graph symbols.

■ 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:

CHR$(27);CHR$(38);CHR$(0);CHR$(*n1*);CHR$(*n2*);CHR$(*m0*)
;CHR$(*m1*);CHR$(*m2*);CHR$(*m3*);CHR$(*m4*);CHR$(*m5*)
;CHR$(*m6*);CHR$(*m7*);CHR$(*m8*);CHR$(*m9*);CHR$(*m10*);CHR$(*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$(38) CHR$(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 descussed earlier: to use the top eight pins, this bit should be 1; to use the bottom eight pins this bit should be 0. Figure 5-10 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.)

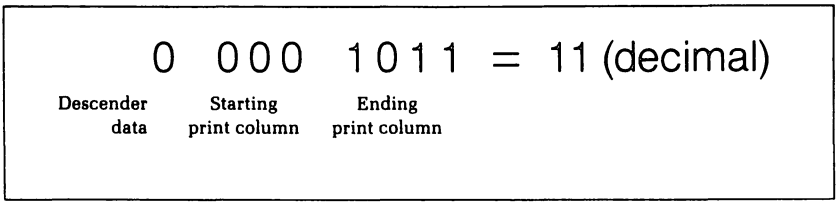


Figure 5-10. The attribute byte (*m0*) for our flask character.

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. Enter the program and run it.

```
10 OPEN4,4
20 PRINT#4, CHR$(27);CHR$(38);CHR$(0);CHR$(60);
   CHR$(62);
30 FOR N=60 TO 62
40 FOR M=0 TO 11
50 READ MM
60 PRINT#4, CHR$(MM);
70 NEXT M
80 NEXT N
90 PRINT#4
100 CLOSE4 : END
110 DATA 139,  2,  5,  8,241,  0,  0,241,  8,  5,
     2,  0
120 DATA 139,124,  0, 66,  4, 64, 36, 16,  2, 16,
    12,  0
130 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:

CHR$(27);CHR$(37);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 OPEN4,4
20 PRINT#4, CHR$(27);CHR$(37);CHR$(49);CHR$(0);
30 PRINT#4, CHR$(60);CHR$(61);CHR$(62)
40 PRINT#4, CHR$(27);CHR$(37);CHR$(48);CHR$(0)
50 CLOSE4
```

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

(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 OPEN4,4
20 PRINT#4, CHR$(27);CHR$(37);CHR$(49);CHR$(0)
30 FOR I=32 TO 127
40 PRINT#4, CHR$(I);
50 NEXT I
60 PRINT#4
70 FOR I=160 TO 255
80 PRINT#4, CHR$(I);
90 NEXT I
100 PRINT#4, CHR$(27);CHR$(37);CHR$(48);CHR$(0)
110 CLOSE4
```

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:

CHR$(27);CHR$(58);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:

```
15 PRINT#4, CHR$(27);CHR$(58);CHR$(0);
   CHR$(0);CHR$(0)
```

Now try the download printout test program again. Your results look like Figure 5-11.

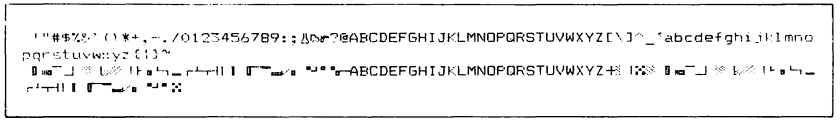


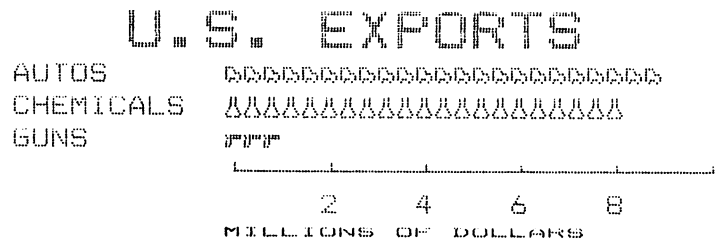
Figure 5-11. Printout of the download character set, into which all the ASCII characters have been copied, and the 〈, = and 〉 have been changed.

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 OPEN4,4
20 PRINT#4, CHR$(27);CHR$(58);CHR$(0);
   CHR$(0);CHR$(0);
30 PRINT#4, CHR$(27);CHR$(38);CHR$(0);
   CHR$(60);CHR$(62);
40 FOR N=60 TO 62
50 FOR M=0 TO 11
```

```
60 READ MM
70 PRINT#4, CHR$(MM);
80 NEXT M
90 NEXT N
100 PRINT#4
110 DATA 139,  2,  5,  8,241,  0,  0,241,
      8,  5,  2,  0
120 DATA 139,124,  0, 66,  4, 64, 36, 16,
      2, 16, 12,  0
130 DATA 139, 46, 16,  2, 60,  0, 48,  0,
     48,  0, 48,  0
140 PRINT#4, CHR$(27);CHR$(93);CHR$(49);
150 PRINT#4, CHR$(27);CHR$(68);CHR$(11);
    CHR$(0)
160 PRINT#4, CHR$(27);CHR$(104);CHR$(1);
170 PRINT#4, "   U.S. EXPORTS"
180 PRINT#4, CHR$(27);CHR$(104);CHR$(0);
190 PRINT#4, CHR$(27);CHR$(37);CHR$(49);
    CHR$(0);
200 PRINT#4, "AUTOS";CHR$(9);
210 FOR I=0.4 TO 9.3 STEP 0.4
220 PRINT#4, CHR$(61);
230 NEXT I
240 PRINT#4
250 PRINT#4, "CHEMICALS";CHR$(9);
260 FOR I=0.4 TO 8.7 STEP 0.4
270 PRINT#4, CHR$(60);
280 NEXT I
290 PRINT#4
300 PRINT#4, "GUNS";CHR$(9);
310 FOR I=0.4 TO 1.4 STEP 0.4
320 PRINT#4, CHR$(62);
330 NEXT I
340 PRINT#4
350 PRINT#4, CHR$(27);CHR$(37);CHR$(48);
    CHR$(0);
360 PRINT#4, CHR$(9);CHR$(173);CHR$(192);
    CHR$(192);
370 SCALE$=CHR$(192)+CHR$(192)+CHR$(177)+
    CHR$(192)+CHR$(192)
380 FOR I=2 TO 8 STEP 2
390 PRINT#4, SCALE$;
400 NEXT I
410 PRINT#4, CHR$(192);CHR$(192);CHR$(189)
420 PRINT#4, CHR$(9);"  ";
430 FOR I=2 TO 8 STEP 2
440 PRINT#4, "  ";I;
```

```
450 NEXT I
460 PRINT#4
470 PRINT#4, CHR$(27);CHR$(83);CHR$(48);
480 PRINT#4, CHR$(9);"MILLIONS OF DOLLARS"
490 PRINT#4, CHR$(27);CHR$(84)
500 CLOSE4
510 END
```



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⟩ CHR$(64) command, which initializes the printer, does not destroy the contents of download RAM.

DEFINING PROPORTIONAL CHARACTERS

Except for the actual width, defining characters for proportinal 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 norrow 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, spcify a starting column of 6 and an ending column of 8, even thougu those are both within the acceptable range). If you inadvertantly 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: (descender * 128) + (start * 16) + 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$(38);CHR$(0);CHR$(*n1*);CHR$(*n2*);CHR$(*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 2, 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 23 dots horizontally, you plan your designs on a different grid than the one you used for draft characters. Make up some grids (photocopy Figure 5-12 if you wish) and get ready to be creative!

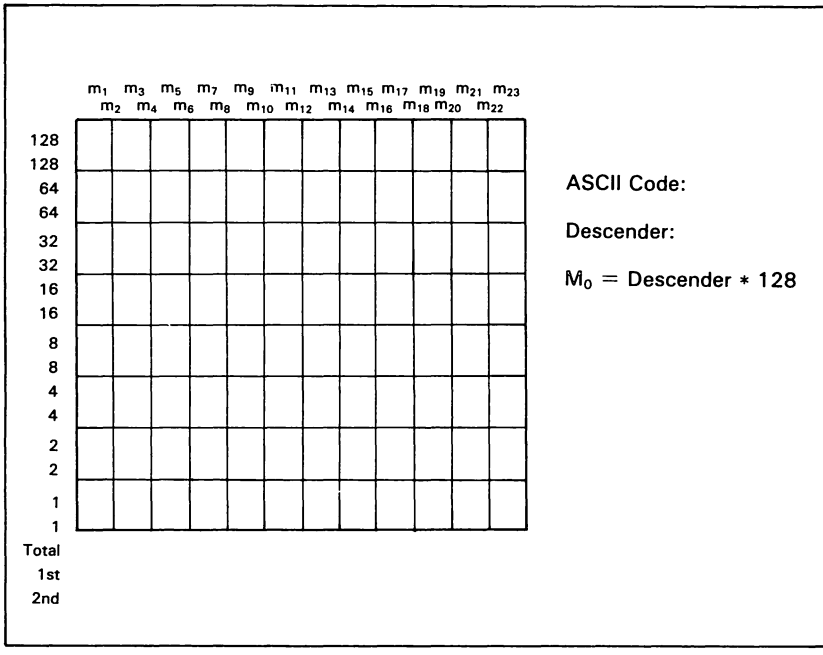


Figure 5-12. Use this grid (or one similar to it) to define your own NLQ characters.

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 5-13.

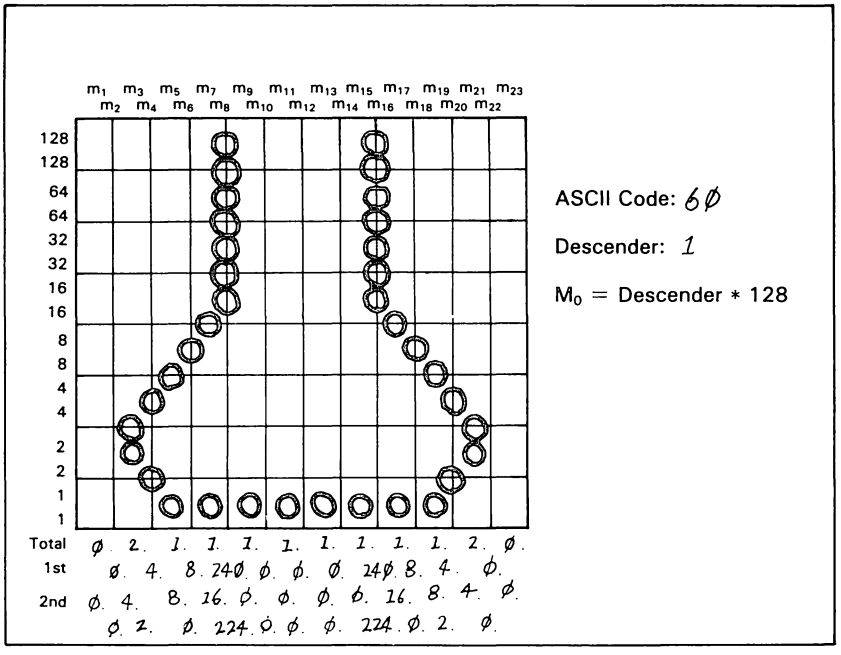


Figure 5-13. Add the values of the dots in each box and line column and write the sum of each column at the bottom.

Now we'll show you how to use the NLQ character definition with a flask as shown in Figure 5-13. Figure 5-13 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.

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 OPEN4,4
20 PRINT#4, CHR$(27);CHR$(120);CHR$(49);
30 PRINT#4, CHR$(27);CHR$(38);CHR$(0);
   CHR$(60);CHR$(60);
40 FOR M=0 TO 46
50 READ MM
60 PRINT#4, CHR$(MM);
70 NEXT M
80 PRINT#4, CHR$(27);CHR$(120);CHR$(48)
90 CLOSE4
```

```
100 DATA 128,  0,  0,  2,  4,  1,  8,  1,
    240,  1,  0,  1
110 DATA   0,  1,  0,  1,240,  1,  8,  1,
      4,  2,  0,  0
120 DATA   0,  0,  4,  2,  8,  0, 16,224,
      0,  0,  0,  0
130 DATA   0,  0,  0,224, 16,  0,  8,  2,
      4,  0,  0
```

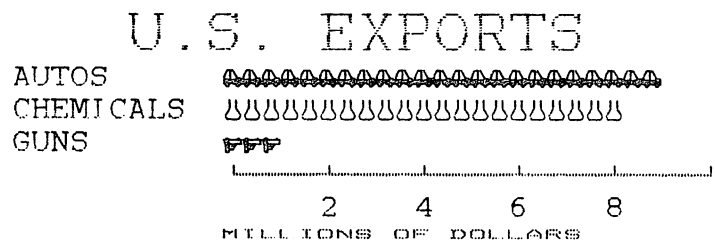
When you want to print the defined character, you must select the NLQ mode first, then select the download characters. If you don't select the NLQ mode, the download characters are not printed even you selected the download character set.

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

```
10 OPEN4,4
20 PRINT#4, CHR$(27);CHR$(120);CHR$(49);
30 PRINT#4, CHR$(27);CHR$(58);CHR$(0);
   CHR$(0);CHR$(0);
40 PRINT#4, CHR$(27);CHR$(38);CHR$(0);
   CHR$(60);CHR$(62);
50 FOR N=60 TO 62
60 FOR M=0 TO 46
70 READ MM
80 PRINT#4, CHR$(MM);
90 NEXT M
100 NEXT N
110 PRINT#4, CHR$(27);CHR$(120);CHR$(48)
120 DATA 128,  0,  0,  2,  4,  1,  8,  1,
    240,  1,  0,  1
130 DATA   0,  1,  0,  1,240,  1,  8,  1,
      4,  2,  0,  0
140 DATA   0,  0,  4,  2,  8,  0, 16,224,
      0,  0,  0,  0
150 DATA   0,  0,  0,224, 16,  0,  8,  2,
      4,  0,  0
160 DATA 128, 14, 16, 38,  1, 70,  1, 70,
      0,126,  0,  0
170 DATA  70,  0, 38,  0, 22,  9,  6,  9,
      6,  8,  6,  0
180 DATA  28, 32, 14, 64, 14,  0, 14,  0,
    124,  0,  0, 12
190 DATA  64, 12, 32, 14, 16, 14,  0, 14,
      0, 12,  0
```

```
200 DATA 128,  0, 24,  7, 56,  1, 56,  7,
      8,  2,  8,  6
210 DATA   8,  2,  8,  4,  8,  0,  8, 32,
      8,  0, 24,  0
220 DATA  32, 16, 14, 48, 14, 48, 14, 32,
      0, 32,  0, 40
230 DATA   0, 36,  0, 40,  0, 32,  0, 32,
      0, 48,  0
240 PRINT#4, CHR$(27);CHR$(120);CHR$(49);
250 PRINT#4, CHR$(27);CHR$(93);CHR$(49);
260 PRINT#4, CHR$(27);CHR$(68);CHR$(11);
    CHR$(0)
270 PRINT#4, CHR$(27);CHR$(104);CHR$(1);
280 PRINT#4, "   U.S. EXPORTS"
290 PRINT#4, CHR$(27);CHR$(104);CHR$(0);
300 PRINT#4, CHR$(27);CHR$(37);CHR$(49);
    CHR$(0);
310 PRINT#4, "AUTOS";CHR$(9);
320 FOR I=0.4 TO 9.3 STEP 0.4
330 PRINT#4, CHR$(61);
340 NEXT I
350 PRINT#4
360 PRINT#4, "CHEMICALS";CHR$(9);
370 FOR I=0.4 TO 8.7 STEP 0.4
380 PRINT#4, CHR$(60);
390 NEXT I
400 PRINT#4
410 PRINT#4, "GUNS";CHR$(9);
420 FOR I=0.4 TO 1.4 STEP 0.4
430 PRINT#4, CHR$(62);
440 NEXT I
450 PRINT#4
460 PRINT#4, CHR$(27);CHR$(37);CHR$(48);
    CHR$(0);
470 PRINT#4, CHR$(9);CHR$(173);CHR$(192);
    CHR$(192);
480 SCALE$=CHR$(192)+CHR$(192)+CHR$(177)+
    CHR$(192)+CHR$(192)
490 FOR I=2 TO 8 STEP 2
500 PRINT#4, SCALE$;
510 NEXT I
520 PRINT#4, CHR$(192);CHR$(192);CHR$(189)
530 PRINT#4, CHR$(9);"  ";
540 FOR I=2 TO 8 STEP 2
550 PRINT#4, "  ";I;
560 NEXT I
570 PRINT#4, CHR$(27);CHR$(120);CHR$(48)
580 PRINT#4, CHR$(27);CHR$(83);CHR$(48);
```

```
590 PRINT#4, CHR$(9);"MILLIONS OF DOLLARS"
600 PRINT#4, CHR$(27);CHR$(84)
610 CLOSE4
620 END
```



CHAPTER 6

DOT GRAPHICS

Subjects we'll cover in Chapter 6 include--

- **Comparing dot graphics with download characters;**
- **Dot graphics commands;**
- **Calculating graphics command data;**
- **Mixing text and graphics;**
- **Printing a design or logo;**
- **Creating bar charts;**
- **Using the printer as a plotter;**
- **High-resolution graphics.**

In Chapter 5 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 this 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 graphic 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 5. The principles for dot graphics are the same as those for draft 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 same seven or eight pins of the print head, depending on whether you are using graphics commands.

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. 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 graphic 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 (your 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.)

GRAPHICS COMMANDS/DATA

The dot graphics commands

Dot graphics can be printed in any of three resolutions: normal, double-density, and quad-density (high-resolution). Normal dot graphics can print 60 dots per inch (this is good enough for most purposes). Double density can print twice that (120 dots per inch — for logos and letter-heads). You use quad density for high-resolution graphics (240 dots per inch) — computer art perhaps.

In addition, normal dot graphics can be printed in reverse field — a white design on a black background with the Commodore mode.

There are seven dot graphics commands, the main tools you'll use to create your own graphics patterns. Three of them can be used only in the Commodore mode. The commands look like this:

Table 6-1
Dot graphics commands

Function	Control code
7-pin normal density (60 dots/inch)	CHR$(8);CHR$(*m1*);CHR$(*m2*);... [Commodore mode only]
7-pin double density (120 dots/inch)	CHR$(9);CHR$(*m1*);CHR$(*m2*);... [Commodore mode only]
7-pin reverse field, normal density (60 dots/inch)	CHR$(27);CHR$(18);CHR$(*m1*);CHR$(*m2*);... [Commodore mode only]
8-pin normal density (60 dots/inch)	CHR$(27);CHR$(75);CHR$(*n1*);CHR$(*n2*); CHR$(*m1*);CHR$(*m2*);...
8-pin double density (120 dots/inch)	CHR$(27);CHR$(76);CHR$(*n1*);CHR$(*n2*); CHR$(*m1*);CHR$(*m2*);...
8-pin double density with double speed (120 dots/inch)	CHR$(27);CHR$(89);CHR$(*n1*);CHR$(*n2*); CHR$(*m1*);CHR$(*m2*);...
8-pin quad density (240 dots/inch)	CHR$(27);CHR$(90);CHR$(*n1*);CHR$(*n2*); CHR$(*m1*);CHR$(*m2*);...

Any of these commands tells the printer that you want the dot graphics mode. Since the last four commands begin with the 〈ESC〉 code, we'll call them the 8-pin graphics codes to distinguish them from the first three commands. The first three commands we'll call the 7-pin graphics commands.

The 7-pin graphics commands have the differences listed below. —

(1) Don't have *n1* or *n2*.
(2) Control only seven print wires.
(3) Must have 128 added to their graphics data.
(4) Can only be used in the Commodore mode.

Trying to use the 7-pin graphics commands in the ASCII mode will have various embarassing effects. In ASCII mode, CHR$(9) generates a horizontal tab that may do strange things to your printout, and CHR$(8) backs up one character position. 〈ESC〉 CHR$(18) is ignored if encountered.

When you use the 8-pin graphics commands, you have to tell the printer how many bytes of graphics data to expect next, expressed as *n1* and *n2*. (The next section of this chapter will explain how to calculate *n1* and *n2*.)

The graphics data (*m1, m2, m3, ...*) tell the printer which print wires to operate. Each of the print wires has an address, or value that makes it move (see next figure). We'll see how to

calculate graphics data in the section "Specifying graphics data."

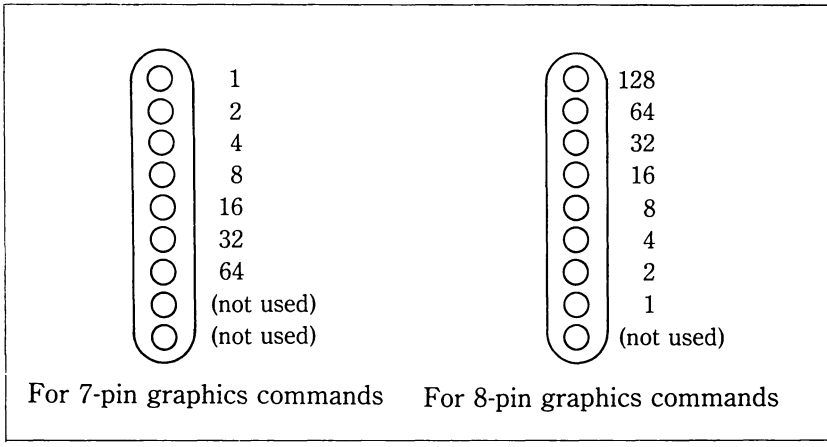


Figure 6-1. There are two sets of print wire addresses.

Notice that you can print double-density graphics at either of two speeds. Double-speed graphics can save you a lot of time when you are debugging a program or when you are in a hurry (for instance, when proofreading the rough draft of a document that contains graphics). "High-Res Graphics" will show us how to work with double- and quad-density graphics, but first we have to cover the basics of normal-density graphics.

■ Specifying the amount of data

In the 8-pin graphics commands, we have to tell the printer how many bytes of graphics data there are (*n1* and *n2*) before we send it the data. (The 7-pin graphics commands don't have an *n1* or *n2*.)

The first thing to do is to figure out how many dots wide your graphics image is (remember, 60 dots to the inch in normal density). Once that's done, we have to convert one number (the number of dots) into two numbers (*n1* and *n2*). This isn't so hard if you use Table 6-2.

Table 6-2
Calculating *n1* and *n2* for the 8-pin graphics commands.

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

Let's try some examples. For 128 dots, we see that 128 is between 1 and 256 (the first line in the table). This means that *n1* = 128 and *n2* = 0. If we have 1791 dots, *n1* = 255 (1791 − 1536) and *n2* = 6. The maximum number of dots that can be printed in one line is 1920 (in quad density), so you should always be able to find *n1* and *n2* from Table 6-2.

The next step is to calculate the graphics data.

■ Specifying graphics data

Next we'll see how to calculate the data to put into the graphics commands. The procedures for the 7-pin graphics commands and the 8-pin graphics commands are different. Let's go over the 7-pin graphics commands.

The 7-pin graphics commands

Suppose we want to operate the first (top), third, and fourth print wires. First, find the addresses of all the print wires to be operated (1, 4, and 8) and add their values (1 + 4 + 8 = 13). Since this information will be used in a 7-pin graphics command, we have to add 128 to tell the printer that what follows is graphics data. In this example, we get 13 + 128, or 141. Thus, to operate the first, third, and fourth print wires, we send CHR$(141). (Don't forget that you can't specify the eighth print wire in these commands.)

The 8-pin graphics commands

Now we'll find the data byte that will operate the same print wires from any of the 8-pin graphics commands. Referring to Figure 6-1, we see that the addresses for the first, third, and fourth wires from the top are different from the above example:

128 (top), 32, and 16. (Notice that the 8-pin graphics commands can control eight print wires.) Adding the addresses (128 + 32 + 16), we get 176. We do not add 128 to the address total when using the 8-pin graphics commands, so sending CHR$(176) will do the job.

Here's a short program to demonstrate what we've been talking about. SAVE the program after you run it — we'll be using it again in just a minute.

```
10 REM     DEMO OF DOT GRAPHICS
20 PI = 3.14159
30 WID = 100
40 OPEN 4,4
50 PRINT#4, CHR$(27);CHR$(75);
60 PRINT#4, CHR$(WID-256*INT(WID/256));
70 PRINT#4, CHR$(INT(WID/256));
80 FOR I = 0 TO WID-1
90 J=1+SIN(I*PI/32)
100 PRINT#4, CHR$(2↑INT((J)*3.5+.5));
110 NEXT I
120 PRINT#4
130 CLOSE4
```

Your printout should look like this:

In lines 50 to 70, we specify normal dot graphics and tell the printer that we'll be sending 100 bytes of graphics data. The loop of lines 80 to 110 plots 100 points along a curve (a sine wave).

■ Mixing text and graphics

We can also mix text and graphics in a line. This lets us label charts and graphs, and even insert fancy graphics in text. Try adding these two lines to the program we just ran:

```
45 PRINT#4, "WOW! ";
115 PRINT#4, "THIS IS GREAT! ";
```

Remember that all graphics data specified in a command must fit on the same line. (The graphics mode with one of the 8-pin graphics commands is turned off at the end of each line even if you specify more than one line of graphics.) Change line 30 as follows to see what happens when you overflow the line.

```
30 WID = 1000
```

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

As you can see, the printer plotted the curve 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 print wire, you can print nearly anything that can be drawn by hand — and probably better, if you're like most computer users! You can use this for creating computer art or drawing maps. Or, as we'll show you here, you can use dot graphics to print a logo or letterhead for the top of your correspondence.

Creating your own design can be a lot of fun. The best way to start is to sketch your idea on graph paper. But first draw a horizontal line every eight rows (seven with 7-pin graphics commands) on your paper. Since eight dots can be printed each time, one strip corresponds to one pass of the print head.

It may be helpful to write the dot values (128, 64, 32, etc.) along the left edge of each strip. After you've sketched your design and know which dots you want printed, add up the values in each column of the strip. The result is one byte of print data.

In the program below, we've taken the logo graphics information and put it into DATA statements. The program itself is short and simple, and its printout follows (SAVE the program — we'll use it later).

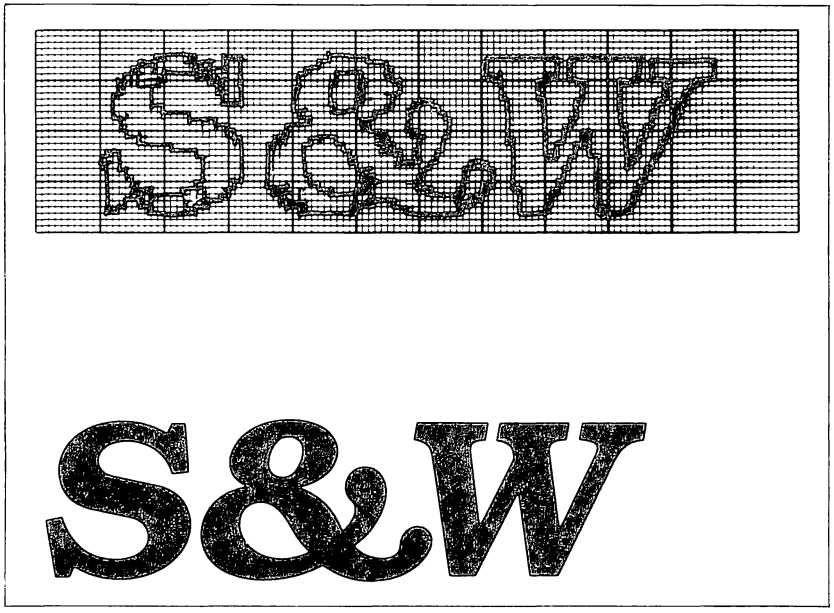


Figure 6-2. You have to sketch your design on graph paper before you can calculate the graphics data.

```
100 REM     PRINT S&W LOGO
110 GRAPHIC$ = CHR$(27)+CHR$(75)
120 DIM L$(4)
130 REM     READ DATA
140 OPEN4,4
150 FOR ROW = 1 TO 4
160 FOR COLUMN = 1 TO 100
170 READ P
180 L$(ROW) = L$(ROW) + CHR$(P)
190 NEXT COLUMN
200 NEXT ROW
210 REM     PRINT LOGO
220 PRINT#4, CHR$(27);CHR$(65);CHR$(8);
230 FOR ROW = 1 TO 4
240 PRINT#4, GRAPHIC$;CHR$(100);CHR$(0);
250 PRINT#4, L$(ROW)
260 NEXT ROW
270 PRINT#4, CHR$(27);CHR$(50)
280 CLOSE4 : END
290 REM     ROW1
300 DATA    0,  0,  0,  0,  1,  3,  7,  7,  7, 15
310 DATA   14, 14, 14, 14, 14,  7,  7,  3,  3, 15
```

```
320 DATA  15, 15,  0,  0,  0,  0,  0,  0,  0,  0
330 DATA   0,  1,  3,  3,  7,  7, 15, 14, 14, 14
340 DATA  14, 15,  7,  7,  7,  3,  0,  0,  0,  0
350 DATA   0,  0,  0,  0,  0,  0,  0,  0,  0,  0
360 DATA   0,  6,  7,  7,  7,  7,  7,  7,  7,  7
370 DATA   6,  6,  0,  0,  7,  7,  7,  7,  7,  7
380 DATA   7,  7,  7,  7,  7,  0,  0,  7,  7,  7
390 DATA   7,  7,  7,  7,  7,  7,  7,  0,  0,  0
400 REM    ROW2
410 DATA   0,  0, 60,255,255,255,255,255,143, 15
420 DATA   7,  7,  7,  7,  3,  3,  3,131,193,241
430 DATA 240,240,  0,  0,  0,  0,  0,  0,  0,  1
440 DATA 121,253,253,255,255,255,143,  7,  7,  7
450 DATA  31,253,252,248,248,240,192,  0,  7, 15
460 DATA  31, 31, 15,  7,  3,  0,  0,  0,  0,  0
470 DATA   0,  0,  0,224,255,255,255,255,255, 31
480 DATA   0,  0,  0,  1,  3, 31,255,255,255,255
490 DATA 255,255,  1,  0,  0,  0,  1,  7, 31,255
500 DATA 252,240,192,128,  0,  0,  0,  0,  0,  0
510 REM    ROW3
520 DATA   0, 31, 31,  3,129,128,192,192,192,192
530 DATA 192,224,224,224,224,240,255,255,255,255
540 DATA 255,127,  0,  0,  0,  0, 63,127,255,255
550 DATA 255,255,193,128,128,128,128,192,224,240
560 DATA 252,255,255,255,127, 63, 31,  7,  7, 31
570 DATA 254,252,248,224,128,  0,  0,  3,  7,  7
580 DATA   7,  3,  0,  0,192,255,255,255,255,255
590 DATA  15, 15, 63,252,240,192,  0,240,255,255
600 DATA 255,255,255,  7, 15,127,252,240,192,  0
610 DATA   0,  0,  0,  0,  0,  0,  0,  0,  0,  0
620 REM    ROW4
630 DATA   0,248,248,240,224,224,112,112, 56, 56
640 DATA  56, 56, 56,120,120,240,240,224,224,192
650 DATA 128,  0,  0,  0,  0,  0,192,224,240,240
660 DATA 240,248,248,248,120,120, 56, 56, 56, 56
670 DATA  48,112,224,224,224,224,240,240,248,248
680 DATA 120,120, 56, 56, 56, 56,120,240,224,224
690 DATA 192,128,  0,  0,  0,128,248,248,248,248
700 DATA 240,192,  0,  0,  0,  0,  0,  0,240,248
710 DATA 248,248,248,240,192,  0,  0,  0,  0,  0
720 DATA   0,  0,  0,  0,  0,  0,  0,  0,  0,  0
```

Figure 6-3. A normal logo.

CREATING BAR CHARTS

Now we'll show you a special function available in the 7-pin graphics mode. Using the repeat function, you can create effective and striking bar charts. The command has the following format.

CHR$(8);CHR$(26);CHR$(*m*);CHR$(*n*)

This code sequence causes the printer to repeat dot graphics data in the Commodore mode. Parameter *m* is a binary number with a value between 0 and 255, and specifies the desired number of times the specified byte of graphics data (*n*) is repeated.

You can repeat a byte of graphics data up to 256 times (use *m* = 0 for 256 repetitions). To repeat it more than that, you have to use the sequence again. Try this program:

```
10 REM    DEMO OF REPEAT GRAPHICS.
20 OPEN4,4
30 FOR I = 1 TO 6
40 READ N
50 PRINT#4, CHR$(8);CHR$(26);CHR$(N);CHR$(238)
60 PRINT#4, CHR$(15)
70 NEXT I
80 FOR J = 1 TO 6
90 PRINT#4
100 NEXT J
110 CLOSE4
120 DATA  34, 57, 75, 93,120,183
```

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

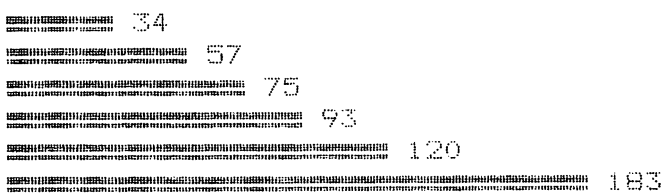


Figure 6-4. Bar graphs are an effective way to present information.

USING THE PRINTER AS A GRAPHICS PLOTTER

This section of the manual gets into more advanced BASIC than we've used so far. Beginners, gird your loins! It's time to do a little studying — taken slowly, the program in this section can be is understood. If you can master these techniques, you'll have a wealth of terrific business graphs, charts, and function plots at your command.

The best way to plot a graph is to program an array in memory. The array is your graph paper. The first thing to do is to decide how big your graph (and your array) will be. (If you plan to plot an entire page of data, you'd better have a lot of memory in your computer — a graph of normal density and eight inches on a side takes up 32K bytes!)

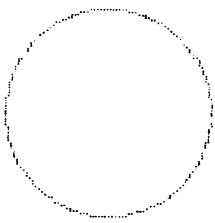
Here's a program that graphs a circle. As we will see, by changing a few lines, you can use it to plot virtually any shape.

```
1000 REM    PLOTTING PROGRAM
1010 REM    SET PROGRAM CONSTANTS
1020 MC% = 75              : MR% = 14
1030 DIM BT%(MC%,MR%)
1040 MK%(1) = 64           : MK%(4) = 8
1050 MK%(2) = 32           : MK%(5) = 4
1060 MK%(3) = 16           : MK%(6) = 2
1070 LX = 20               : LY = 20
1080 XF = 72/LX            : YF = 87/LY
2000 REM    PLOT CURVE
2010 RAD = 9
2020 X1 = 19               : Y1 = 10
2030 FOR AG = 0 TO 360 STEP 10
2040 RG = AG*6.28/360
2050 X2 = RAD*COS(RG)+10
2060 Y2 = RAD*SIN(RG)+10
2070 GOSUB 4000
2080 NEXT AG
3000 REM    SEND BIT IMAGE MAP TO PRINTER
3010 OPEN4,4
3020 PRINT#4, CHR$(27);CHR$(65);CHR$(6)
3030 FOR RW = 0 TO MR%   : RW% = RW
3040 A$ = ""
3050 PRINT#4, CHR$(27);CHR$(75);
3060 PRINT#4, CHR$(MC%);CHR$(0);
3070 FOR CL = 1 TO MC%   : CL% = CL
3080 A$ = A$ + CHR$(BT%(CL%,RW%))
3090 NEXT CL
```

```
3100 PRINT#4, A$;" "
3110 NEXT RW
3120 PRINT#4, CHR$(27);CHR$(50)
3130 CLOSE4 : END
4000 REM    DRAW A LINE FROM X1,Y1 TO X2,Y2
4010 XL = X2 - X1          : YL = Y2 - Y1
4020 NX = ABS(XL*XF)       : NY = ABS(YL*YF)
4030 IF NX < NY THEN NX = NY
4040 NS% = INT(NX+1)
4050 DX = XL/NS%           : DY = YL/NS%
4060 FOR I = 1 TO NS%
4070 X1 = X1 + DX          : Y1 = Y1 + DY
4080 GOSUB 5000
4090 NEXT I
4100 RETURN
5000 REM    PLOT A POINT AT X1,Y1
5010 XX = X1*XF            : YY = Y1*YF
5020 CL% = INT(XX) + 1
5030 RW% = INT(YY/6)
5040 XT% = INT(YY - RW%*6) + 1
5050 BT%(CL%,RW%) = BT%(CL%,RW%) OR MK%(XT%)
5060 RETURN
```

In this program, we created an array called BT%, which is dimensioned in line 1030. You'll note that, instead of using numeric constants to dimension the array, we used the variables MC% and MR%. This way, if your computer has enough memory and you want to plot a larger image, all you have to change are the values in line 1020.

The array MK% contains the values of the dots (In order to make this program compatible with as many computers as possible, we're using only six print wires — but you can use eight wires for many computers.) Lines 1070 and 1080 define other variables: LX, XF, LY, YF (all scaling factors). By changing these variables, you can change the size of your printed image or distort it (make a circle into an ellipse, for instance). Experiment a bit and see what you can do!



Most of the calculations are done by the subroutine that starts in line 2000. This is where you put the formula that you want to graph. By changing only the lines after 2000, you can plot any function you can imagine. (You'll find some examples at the end of the chapter.)

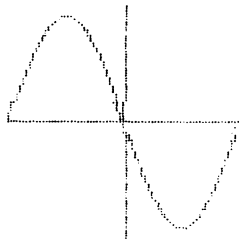
The subroutine calculates the starting and end points of lines that make up the plot (the lines in our circle are very short — sometimes the starting and end points are the same). The coordinates of the starting point are X1 and Y1. Those of the end point are X2 and Y2. After these values are calculated, the subroutine calls another subroutine starting at line 4000, which determines the individual points between them.

After the coordinates of all these points are found, the subroutine starting at line 5000 is called. This routine turns on individual dots in our array BT%. (Keep in mind that no printing has been done yet — the computer is still drawing on the "graph paper" in memory.) Line 5050 shows how to use the logical OR function to turn on a dot.

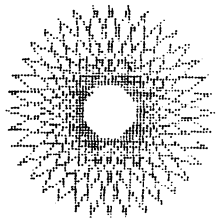
When all the dots have been plotted in memory, actual printing begins with line 3000. We first set the spacing to 6/72 inch using the ⟨ESC⟩ "A" command so there are no spaces between the rows of dots. Then the loop from line 3030 to 3110 prints the image one six-dot-high line at a time. The variable A$ is used to build a string of all the columns of BT% in each row.

As you can see by taking the program in small pieces, graphics programming does not have to be difficult. If you want to try some variations of this program, try these (replace the lines after 2000 with those given below). The printout for each program is given below the listing.

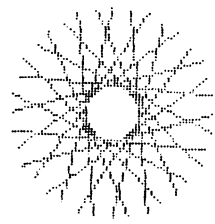
```
2000 REM    PLOT CURVE
2010 X1=0 :Y1=10 :X2=20 :Y2=10 :GOSUB 4000
2020 X1=10 :Y1=0 :X2=10 :Y2=20 :GOSUB 4000
2030 X1=0 :Y1=10 : FOR X2=0 TO 20 STEP .2
2040 Y2=10-9*SIN(3.14159*X2/10) :GOSUB 4000
2050 NEXT X2
```



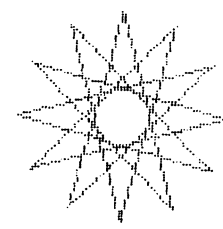
```
2000 REM     PLOT CURVE
2010 RAD = 9
2020 FOR AG = 0 TO 360 STEP 10 :AG% = AG
2030 RG = AG%*6.28/360
2040 R2 = (AG%+150)*6.28/360
2050 X1 = RAD*COS(RG)+10 :Y1 = RAD*SIN(RG)+10
2060 X2 = RAD*COS(R2)+10 :Y2 = RAD*SIN(R2)+10
2070 GOSUB 4000
2080 NEXT AG
```



STEP 10



STEP 20



STEP 30

Notice the different designs you can create simply by changing the step value in line 2020.

HIGH-RES GRAPHICS

So far, all of our dot graphics operations have been done in normal density. Now we take up high-resolution, or high-res, graphics which are based on double- and quad-density printing. Extremely detailed graphs and final drafts, for instance, demand high-res graphics. Your printer has four density modes.

The format for the high-res commands is the same as that for the CHR$(27);CHR$(75) command, as are the calculations for *n1*, *n2*, *m1*, *m2*, etc. Using them is easy — in fact the best way to

demonstrate their differences is to show you the same function plotted in four different density modes.

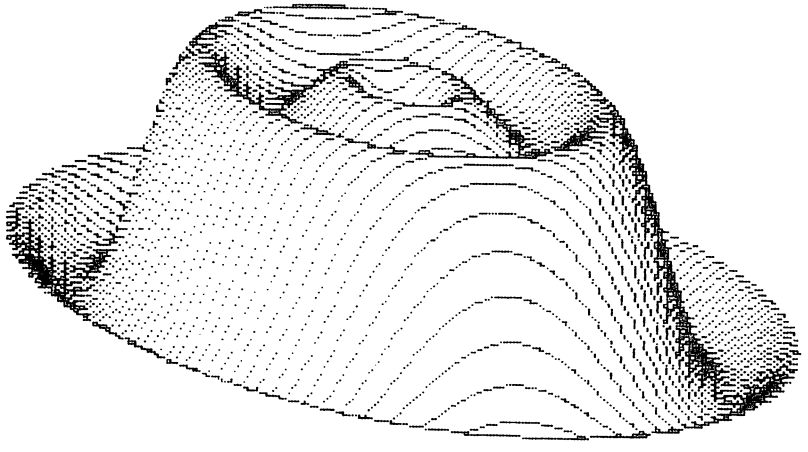


Figure 6-5. Normal-density graphics.

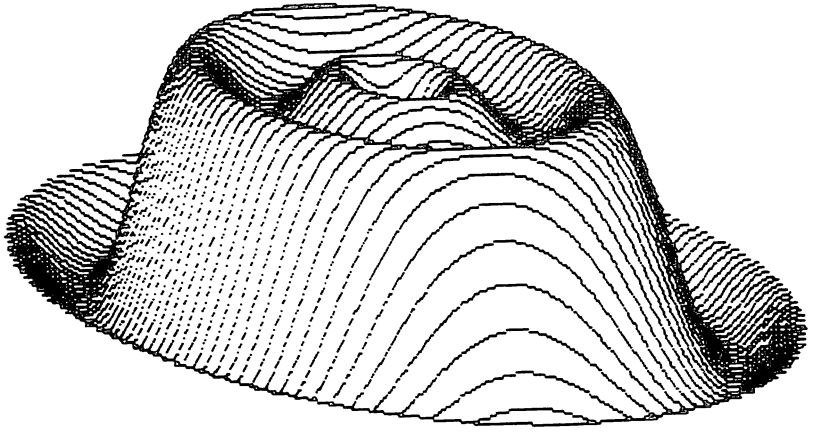


Figure 6-6. Double-density graphics.

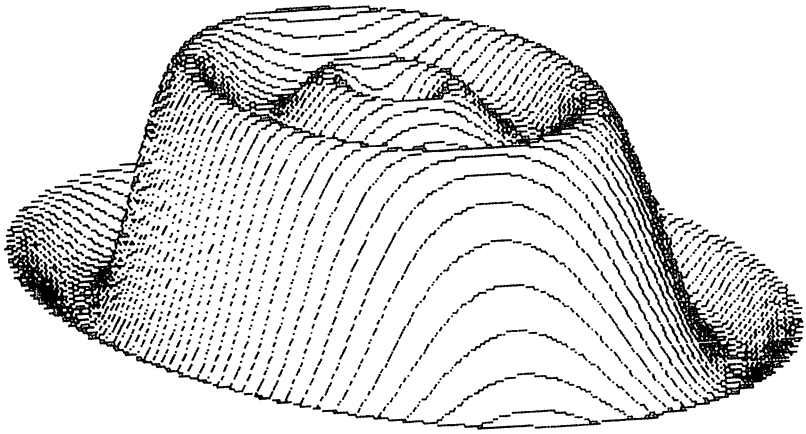


Figure 6-7. Double-density double-speed graphics.

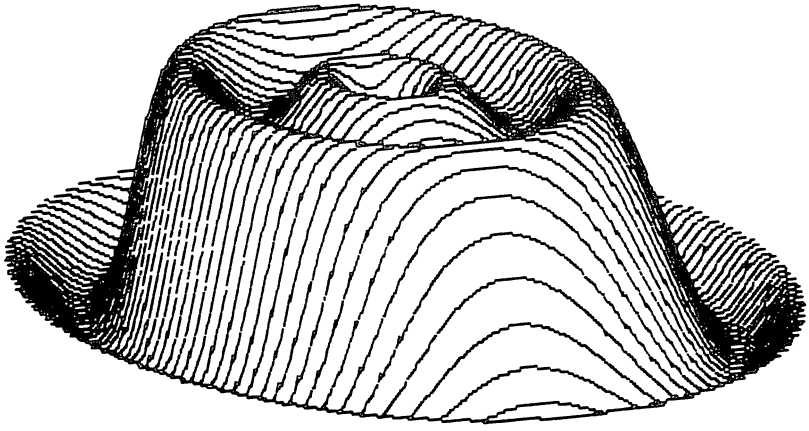


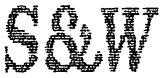
Figure 6-8. Quad-density graphics.

If quad density looks so great, why not use it all the time? Well, let's try an experiment and see. Using the logo demonstration program you entered earlier, change line 110 to try each of the density modes:

CHR$(75) → CHR$(76)	(Double density, normal speed)
CHR$(75) → CHR$(89)	(Double density, double speed)
CHR$(75) → CHR$(90)	(Quad density)

You should get something like the following:

Nomal density



Double density

Double density with double speed

Quadruple desity

As you can see, the different modes seem to condense the printed image. So, to get the same image in a higher density, you have to plot more points. This takes twice as much memory for your array, twice as much computing time, and twice as much printing time. Clearly, the choice involves a trade-off between print quality and the limitations of your computer and on your time.

A final note on double-speed graphics

Double-density double-speed graphics are a unique shortcut for program development. Although this mode takes up just as much memory and computing time as regular double-density graphics, 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 same number of dots are plotted as in normal-density graphics. The advantage, of course, is that you can write and debug your programs at double speed, then change to regular double-density graphics for great printout.

MEMO

APPENDIX A

DIP SWITCH SETTINGS

A dual-in-line set of switches (collectively called a [one] DIP switch) controls some of the functions of the printer. The DIP switch actually contains eight individual switches. Figure A-1 shows the DIP switch.

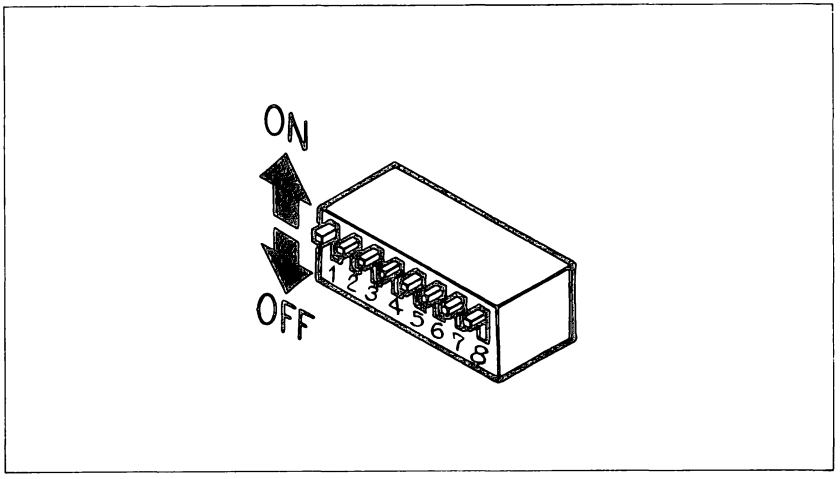


Figure A-1. The DIP switch is several small switches in one package.

The DIP switch is readily accessible from the rear of the printer. To set one of the switches, use a ball-point pen to move the switch lever gently. The on position is upwards, and off is downwards.

Warning: Never set a DIP switch when the power is on. Turn off both the printer and your computer first.

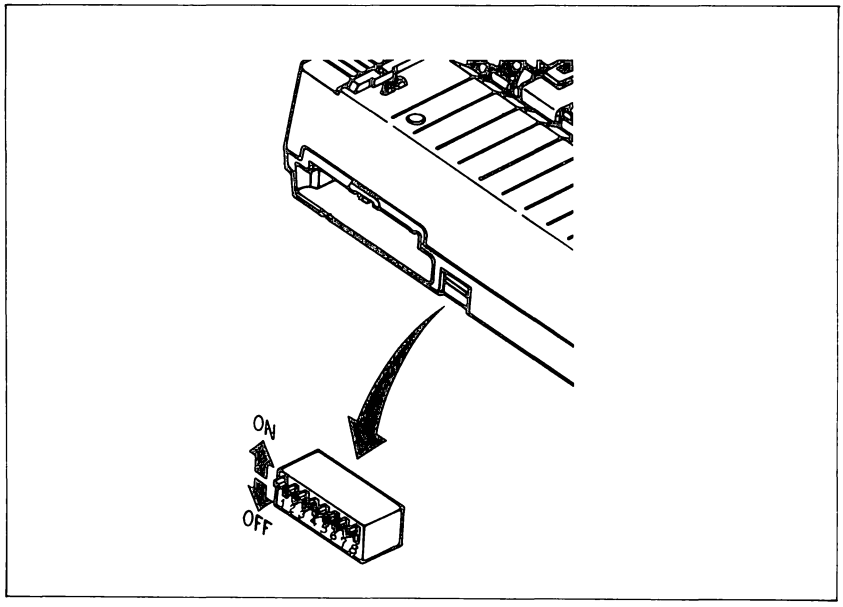


Figure A-2. The DIP switch is located on the back of the printer.

Table A-1 summarizes the functions of the switches.

Table A-1
DIP switch settings

Switch	ON	OFF
1-1	Auto LF with CR	LF must be from host
1-2	Paper-out detected	Paper-out not detected
1-3	Device no. 4	Device no. 5
1-4	11″ page length	12″ page length
1-5	Commodore mode	ASCII mode
1-6	International character set selection — see Table A-2.	
1-7		
1-8		

SWITCH FUNCTIONS

Switch **Function**

1-1 When this switch is off, the computer must send a line feed command every time the paper is to advance. When this switch is on, the printer will automatically advance the paper one line every time

it receives a carriage return. (Most BASICs send a line feed with every carriage return, therefore, this switch should usually be off.)

1-2 This switch disables the paper-out detector. If the switch is on, the printer will signal the computer when it runs out of paper and will stop printing. If the switch is off, the printer will ignore the paper-out detector and will continue printing.

1-3 This switch selects the device number for this printer. If the switch is on, the device number is set to 4. If the switch is off, the device number is set to 5.

1-4 This switch sets the default page length. When the switch is on, the page length is 11 inches. When the switch is off, the page length is 12 inches.

1-5 This switch selects the operating mode. If this switch is on, the operating mode is set to the Commodore mode and the characters are compatible with the Commodore computers. If this switch is set off, the operating mode is set to the ASCII mode and the characters are set to Standard ASCII characters.

1-6～1-8 These three switches determine the default international character set as shown in Table A-2.

Table A-2
International character sets

Switch	Commodore	USA	Germany	Denmark	France	Sweden	Italy	Spain
1-6	ON	OFF	ON	OFF	ON	OFF	ON	OFF
1-7	ON	ON	OFF	OFF	ON	ON	OFF	OFF
1-8	ON	ON	ON	ON	OFF	OFF	OFF	OFF

MEMO

APPENDIX B

ASCII CODES AND CONVERSION CHART

Standard ASCII Codes			Control	Character mode		
Decimal	Hex	Binary	Character	Graphics	Business	ASCII
0	00	0000 0000	Ctrl-@			
1	01	0000 0001	Ctrl-A	($1)	($1)	($1)
2	02	0000 0010	Ctrl-B	($2)	($2)	($2)
3	03	0000 0011	Ctrl-C	($3)	($3)	($3)
4	04	0000 0100	Ctrl-D	($4)	($4)	($4)
5	05	0000 0101	Ctrl-E	(WHT)	(WHT)	(WHT)
6	06	0000 0110	Ctrl-F	($6)	($6)	($6)
7	07	0000 0111	Ctrl-G	($7)	($7)	($7)
8	08	0000 1000	Ctrl-H	(DISH)	(DISH)	(DISH)
9	09	0000 1001	Ctrl-I	(ENSH)	(ENSH)	(ENSH)
10	0A	0000 1010	Ctrl-J			
11	0B	0000 1011	Ctrl-K	($11)	($11)	($11)
12	0C	0000 1100	Ctrl-L	($12)	($12)	($12)
13	0D	0000 1101	Ctrl-M			
14	0E	0000 1110	Ctrl-N	(SWLC)	(SWLC)	(SWLC)
15	0F	0000 1111	Ctrl-O	($15)	($15)	($15)
16	10	0001 0000	Ctrl-P	($16)	($16)	($16)
17	11	0001 0001	Ctrl-Q	(DOWN)	(DOWN)	(DOWN)
18	12	0001 0010	Ctrl-R	(RVS)	(RVS)	(RVS)
19	13	0001 0011	Ctrl-S	(HOME)	(HOME)	(HOME)
20	14	0001 0100	Ctrl-T	(DEL)	(DEL)	(DEL)
21	15	0001 0101	Ctrl-U	($21)	($21)	($21)
22	16	0001 0110	Ctrl-V	($22)	($22)	($22)
23	17	0001 0111	Ctrl-W	($23)	($23)	($23)
24	18	0001 1000	Ctrl-X	($24)	($24)	($24)
25	19	0001 1001	Ctrl-Y	($25)	($25)	($25)
26	1A	0001 1010	Ctrl-Z	($26)	($26)	($26)
27	1B	0001 1011		(ESC)	(ESC)	(ESC)
28	1C	0001 1100		(RED)	(RED)	(RED)
29	1D	0001 1101		(RGHT)	(RGHT)	(RGHT)
30	1E	0001 1110		(GRN)	(GRN)	(GRN)
31	1F	0001 1111		(BUL)	(BUL)	(BUL)
32	20	0010 0000		SP	SP	SP

Standard ASCII Codes			Character mode		
Decimal	Hex	Binary	Graphics	Business	ASCII
33	21	0010 0001	!	!	!
34	22	0010 0010	"	"	"
35	23	0010 0011	#	#	#
36	24	0010 0100	$	$	$
37	25	0010 0101	%	%	%
38	26	0010 0110	&	&	&
39	27	0010 0111	'	'	'
40	28	0010 1000	(	(	(
41	29	0010 1001	)	)	)
42	2A	0010 1010	*	*	*
43	2B	0010 1011	+	+	+
44	2C	0010 1100	,	,	,
45	2D	0010 1101	-	-	-
46	2E	0010 1110	.	.	.
47	2F	0010 1111	/	/	/
48	30	0011 0000	0	0	0
49	31	0011 0001	1	1	1
50	32	0011 0010	2	2	2
51	33	0011 0011	3	3	3
52	34	0011 0100	4	4	4
53	35	0011 0101	5	5	5
54	36	0011 0110	6	6	6
55	37	0011 0111	7	7	7
56	38	0011 1000	8	8	8
57	39	0011 1001	9	9	9
58	3A	0011 1010	:	:	:
59	3B	0011 1011	;	;	;
60	3C	0011 1100	<	<	<
61	3D	0011 1101	=	=	=
62	3E	0011 1110	>	>	>
63	3F	0011 1111	?	?	?
64	40	0100 0000	@	@	@
65	41	0100 0001	A	a	A
66	42	0100 0010	B	b	B
67	43	0100 0011	C	c	C
68	44	0100 0100	D	d	D
69	45	0100 0101	E	e	E
70	46	0100 0110	F	f	F
71	47	0100 0111	G	g	G
72	48	0100 1000	H	h	H
73	49	0100 1001	I	i	I
74	4A	0100 1010	J	j	J
75	4B	0100 1011	K	k	K
76	4C	0100 1100	L	l	L

Standard ASCII Codes			Character mode		
Decimal	Hex	Binary	Graphics	Business	ASCII
77	4D	0100 1101	M	m	M
78	4E	0100 1110	N	n	N
79	4F	0100 1111	O	o	O
80	50	0101 0000	P	p	P
81	51	0101 0001	Q	q	Q
82	52	0101 0010	R	r	R
83	53	0101 0011	S	s	S
84	54	0101 0100	T	t	T
85	55	0101 0101	U	u	U
86	56	0101 0110	V	v	V
87	57	0101 0111	W	w	W
88	58	0101 1000	X	x	X
89	59	0101 1001	Y	y	Y
90	5A	0101 1010	Z	z	Z
91	5B	0101 1011	[	{	[
92	5C	0101 1100	£	£	\
93	5D	0101 1101	]	}	]
94	5E	0101 1110	↑	↑	^
95	5F	0101 1111	←	←	
96	60	0110 0000	—	—	`
97	61	0110 0001	▚	A	a
98	62	0110 0010	▌	B	b
99	63	0110 0011	▀	C	c
100	64	0110 0100	▀	D	d
101	65	0110 0101	▝	E	e
102	66	0110 0110	▄	F	f
103	67	0110 0111	▐	G	g
104	68	0110 1000	▐	H	h
105	69	0110 1001	▗	I	i
106	6A	0110 1010	▗	J	j
107	6B	0110 1011	▟	K	k
108	6C	0110 1100	▖	L	l
109	6D	0110 1101	▚	M	m
110	6E	0110 1110	▟	N	n
111	6F	0110 1111	▐	O	o
112	70	0111 0000	▝	P	p
113	71	0111 0001	▜	Q	q
114	72	0111 0010	▄	R	r
115	73	0111 0011	▜	S	s
116	74	0111 0100	▐	T	t
117	75	0111 0101	▗	U	u
118	76	0111 0110	▚	V	v
119	77	0111 0111	▖	W	w
120	78	0111 1000	▜	X	x

Standard ASCII Codes			Character mode		
Decimal	Hex	Binary	Graphics	Business	ASCII
121	79	0111 1001			
122	7A	0111 1010			
123	7B	0111 1011			
124	7C	0111 1100			
125	7D	0111 1101			
126	7E	0111 1110			
127	7F	0111 1111			SP
128	80	1000 0000	($128)	($128)	($128)
129	81	1000 0001	(ORNG)	(ORNG)	(ORNG)
130	82	1000 0010	($130)	($130)	($130)
131	83	1000 0011	($131)	($131)	($131)
132	84	1000 0100	($132)	($132)	($132)
133	85	1000 0101	(F1)	(F1)	(F1)
134	86	1000 0110	(F3)	(F3)	(F3)
135	87	1000 0111	(F5)	(F5)	(F5)
136	88	1000 1000	(F7)	(F7)	(F7)
137	89	1000 1001	(F2)	(F2)	(F2)
138	8A	1000 1010	(F4)	(F4)	(F4)
139	8B	1000 1011	(F6)	(F6)	(F6)
140	8C	1000 1100	(F8)	(F8)	(F8)
141	8D	1000 1101	(SHRT)	(SHRT)	(SHRT)
142	8E	1000 1110	(SWUC)	(SWUC)	(SWUC)
143	8F	1000 1111	($143)	($143)	($143)
144	90	1001 0000	(BLK)	(BLK)	(BLK)
145	91	1001 0001	(UP)	(UP)	(UP)
146	92	1001 0010	(OFF)	(OFF)	(OFF)
147	93	1001 0011	(CLR)	(CLR)	(CLR)
148	94	1001 0100	(INST)	(INST)	(INST)
149	95	1001 0101	(BRN)	(BRN)	(BRN)
150	96	1001 0110	(LRED)	(LRED)	(LRED)
151	97	1001 0111	(GRY1)	(GRY1)	(GRY1)
152	98	1001 1000	(GRY2)	(GRY2)	(GRY2)
153	99	1001 1001	(LGRN)	(LGRN)	(LGRN)
154	9A	1001 1010	(LBLU)	(LBLU)	(LBLU)
155	9B	1001 1011	(GRY3)	(GRY3)	(GRY3)
156	9C	1001 1100	(PUR)	(PUR)	(PUR)
157	9D	1001 1101	(LEFT)	(LEFT)	(LEFT)
158	9E	1001 1110	(YEL)	(YEL)	(YEL)
159	9F	1001 1111	(CYN)	(CYN)	(CYN)
160	A0	1010 0000	SP	SP	SP
161	A1	1010 0001			
162	A2	1010 0010			
163	A3	1010 0011			
164	A4	1010 0100			

Standard ASCII Codes			Character mode		
Decimal	Hex	Binary	Graphics	Business	ASCII
165	A5	1010 0101	▌	▌	▌
166	A6	1010 0110	▒	▒	▒
167	A7	1010 0111	▌	▌	▌
168	A8	1010 1000	▒	▒	▒
169	A9	1010 1001	╣	▒	▒
170	A A	1010 1010	▌	▌	▌
171	A B	1010 1011	╠	╠	╠
172	A C	1010 1100	▐	▐	▐
173	A D	1010 1101	└	└	└
174	A E	1010 1110	┐	┐	┐
175	A F	1010 1111	▀	▀	▀
176	B0	1011 0000	┌	┌	┌
177	B1	1011 0001	┴	┴	┴
178	B2	1011 0010	┬	┬	┬
179	B3	1011 0011	┤	┤	┤
180	B4	1011 0100	▌	▌	▌
181	B5	1011 0101	▌	▌	▌
182	B6	1011 0110	▐	▐	▐
183	B7	1011 0111	▪	▪	▪
184	B8	1011 1000	▀	▀	▀
185	B9	1011 1001	▀	▀	▀
186	B A	1011 1010	▁▌	▚	▚
187	B B	1011 1011	▐	▐	▐
188	B C	1011 1100	▫	▫	▫
189	B D	1011 1101	▁▌	▁▌	▁▌
190	B E	1011 1110	▫	▫	▫
191	B F	1011 1111	▫▐	▫▐	▫▐
192	C0	1100 0000	▬	▬	▬
193	C1	1100 0001	╨	A	A
194	C2	1100 0010	▌	B	B
195	C3	1100 0011	▬	C	C
196	C4	1100 0100	▬	D	D
197	C5	1100 0101	▪	E	E
198	C6	1100 0110	▬	F	F
199	C7	1100 0111	▌	G	G
200	C8	1100 1000	▌	H	H
201	C9	1100 1001	▖	I	I
202	C A	1100 1010	▖	J	J
203	C B	1100 1011	▗	K	K
204	C C	1100 1100	▙	L	L
205	C D	1100 1101	▚	M	M
206	C E	1100 1110	▗	N	N
207	C F	1100 1111	▀	O	O
208	D0	1101 0000	▀▌	P	P

Standard ASCII Codes			Character mode		
Decimal	Hex	Binary	Graphics	Business	ASCII
209	D1	1101 0001			
210	D2	1101 0010			
211	D3	1101 0011			
212	D4	1101 0100			
213	D5	1101 0101			
214	D6	1101 0110			
215	D7	1101 0111			
216	D8	1101 1000			
217	D9	1101 1001			
218	DA	1101 1010			
219	DB	1101 1011			
220	DC	1101 1100			
221	DD	1101 1101			
222	DE	1101 1110			
223	DF	1101 1111			
224	E0	1110 0000			
225	E1	1110 0001			
226	E2	1110 0010			
227	E3	1110 0011			
228	E4	1110 0100			
229	E5	1110 0101			
230	E6	1110 0110			
231	E7	1110 0111			
232	E8	1110 1000			
233	E9	1110 1001			
234	EA	1110 1010			
235	EB	1110 1011			
236	EC	1110 1100			
237	ED	1110 1101			
238	EE	1110 1110			
239	EF	1110 1111			
240	F0	1111 0000			
241	F1	1111 0001			
242	F2	1111 0010			
243	F3	1111 0011			
244	F4	1111 0100			
245	F5	1111 0101			
246	F6	1111 0110			
247	F7	1111 0111			
248	F8	1111 1000			
249	F9	1111 1001			
250	FA	1111 1010			
251	FB	1111 1011			
252	FC	1111 1100			

Standard ASCII Codes			Character mode		
Decimal	Hex	Binary	Graphics	Business	ASCII
253	F D	1111 1101	▪	▪	▪
254	F E	1111 1110	■	■	■
255	F F	1111 1111	▀	▀	▀

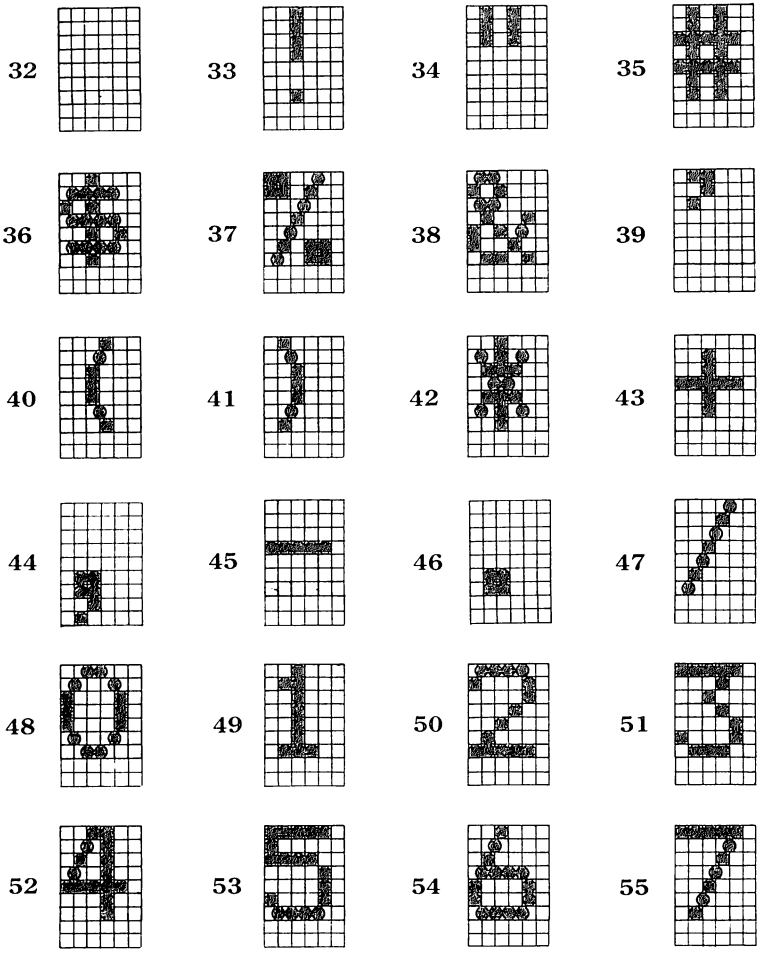
MEMO

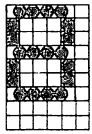
APPENDIX C

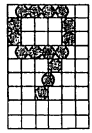
CHARACTER FONTS

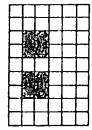
COMMODORE OPERATING MODE

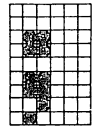
■ Graphics characters

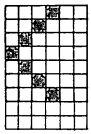


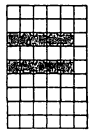
56 

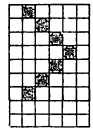
57 

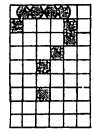
58 

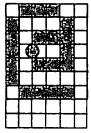
59 

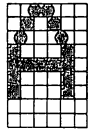
60 

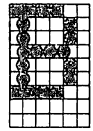
61 

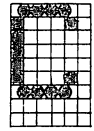
62 

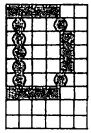
63 

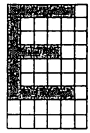
64 

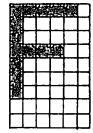
65 

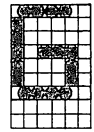
66 

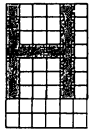
67 

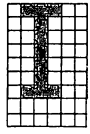
68 

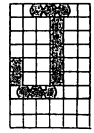
69 

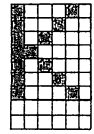
70 

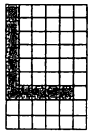
71 

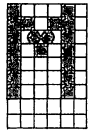
72 

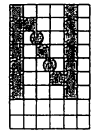
73 

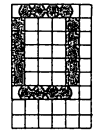
74 

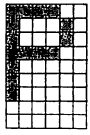
75 

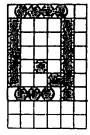
76 

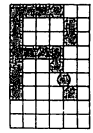
77 

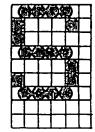
78 

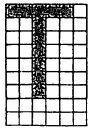
79 

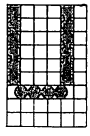
80 

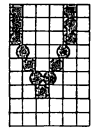
81 

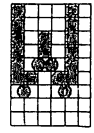
82 

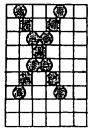
83 

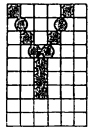
84 

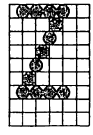
85 

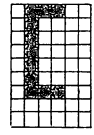
86 

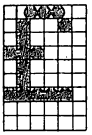
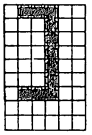
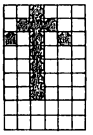
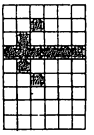
87 

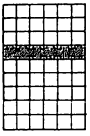
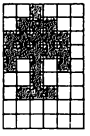
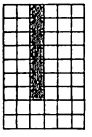
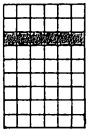
88 

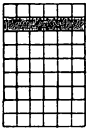
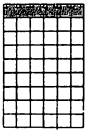
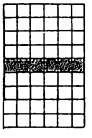
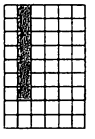
89 

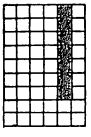
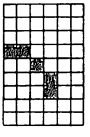
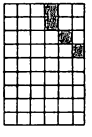
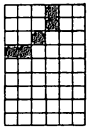
90 

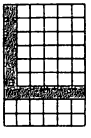
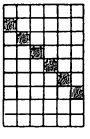
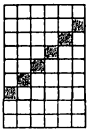
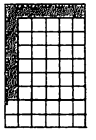
91 

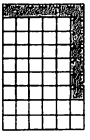
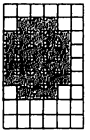
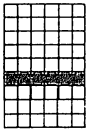
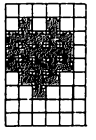
92 
93 
94 
95 

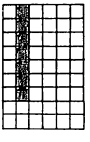
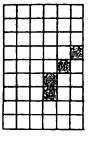
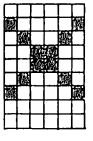
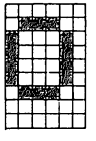
96 
97 
98 
99 

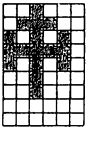
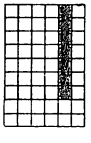
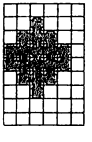
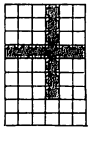
100 
101 
102 
103 

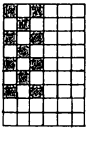
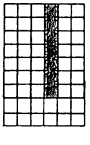
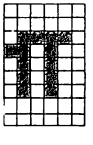
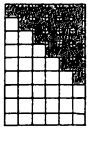
104 
105 
106 
107 

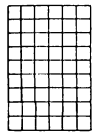
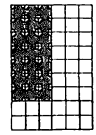
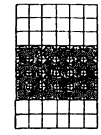
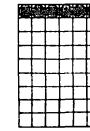
108 
109 
110 
111 

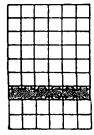
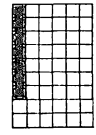
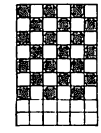
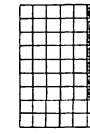
112 
113 
114 
115 

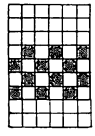
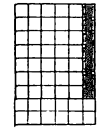
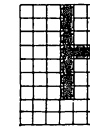
116 
117 
118 
119 

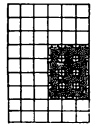
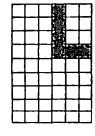
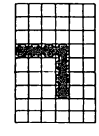
120 
121 
122 
123 

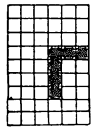
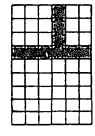
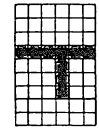
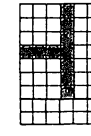
124 
125 
126 
127 

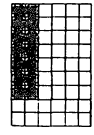
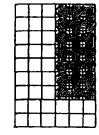
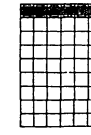
160 
161 
162 
163 

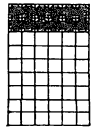
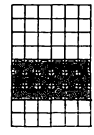
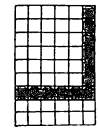
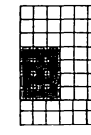
164 
165 
166 
167 

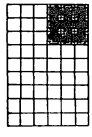
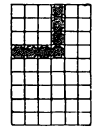
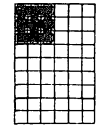
168 
169 
170 
171 

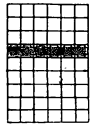
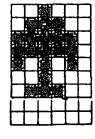
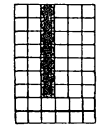
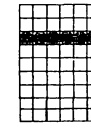
172 
173 
174 
175 

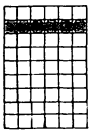
176 
177 
178 
179 

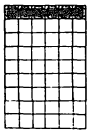
180 
181 
182 
183 

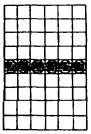
184 
185 
186 
187 

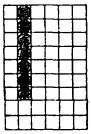
188 
189 
190 
191 

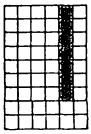
192 
193 
194 
195 

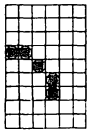
196 

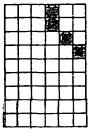
197 

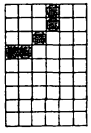
198 

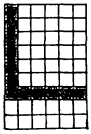
199 

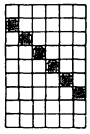
200 

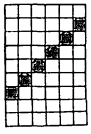
201 

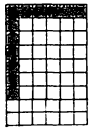
202 

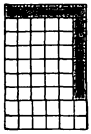
203 

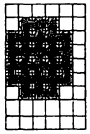
204 

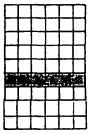
205 

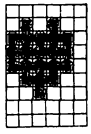
206 

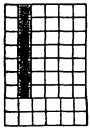
207 

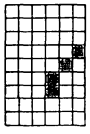
208 

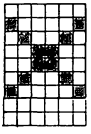
209 

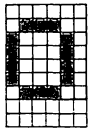
210 

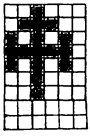
211 

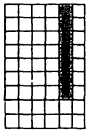
212 

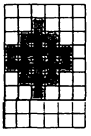
213 

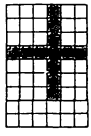
214 

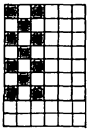
215 

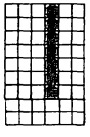
216 

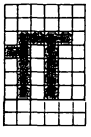
217 

218 

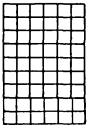
219 

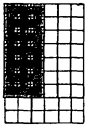
220 

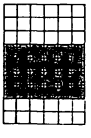
221 

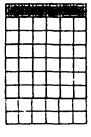
222 

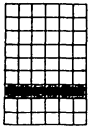
223 

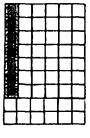
224 

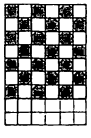
225 

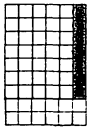
226 

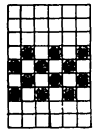
227 

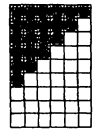
228 

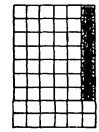
229 

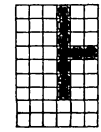
230 

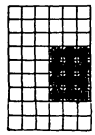
231 

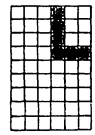
232 

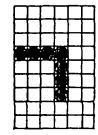
233 

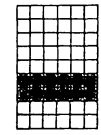
234 

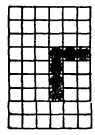
235 

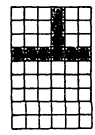
236 

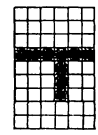
237 

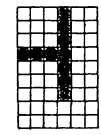
238 

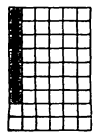
239 

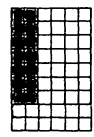
240 

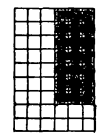
241 

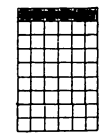
242 

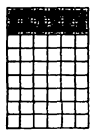
243 

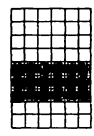
244 

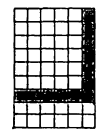
245 

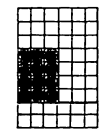
246 

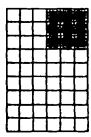
247 

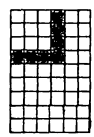
248 

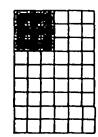
249 

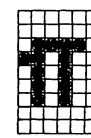
250 

251 

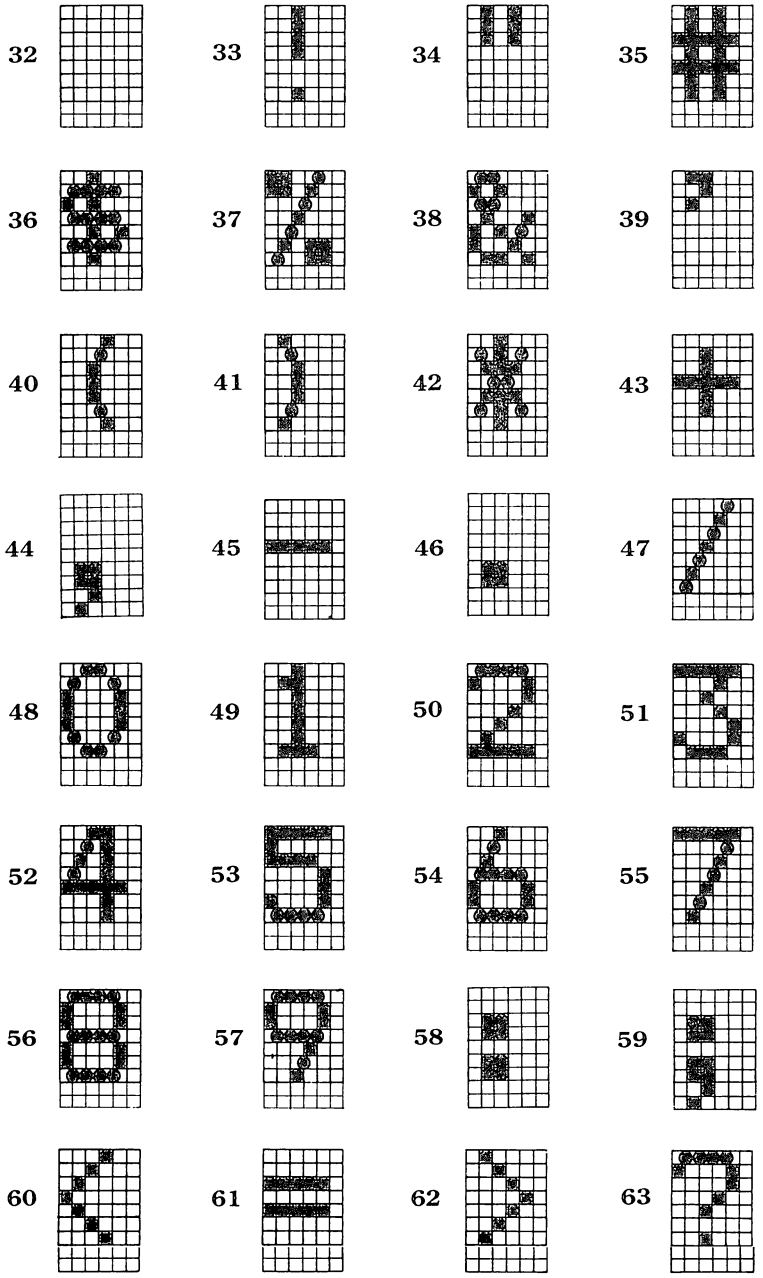
252 

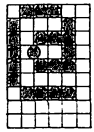
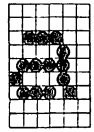
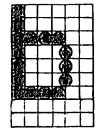
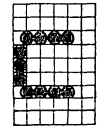
253 

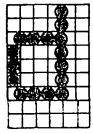
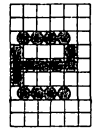
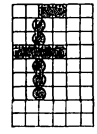
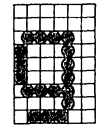
254 

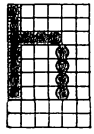
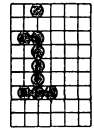
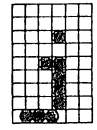
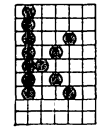
255 

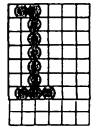
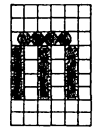
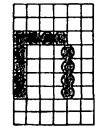
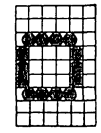
Business characters

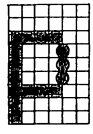
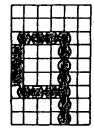
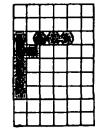
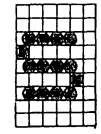


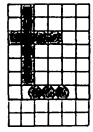
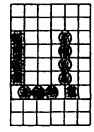
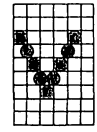
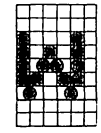
64 
65 
66 
67 

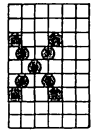
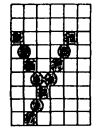
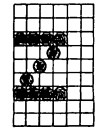
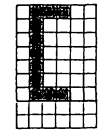
68 
69 
70 
71 

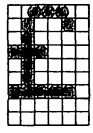
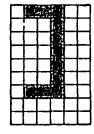
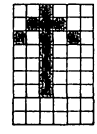
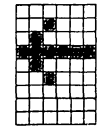
72 
73 
74 
75 

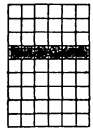
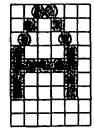
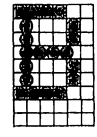
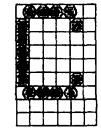
76 
77 
78 
79 

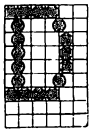
80 
81 
82 
83 

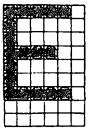
84 
85 
86 
87 

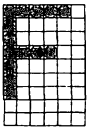
88 
89 
90 
91 

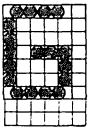
92 
93 
94 
95 

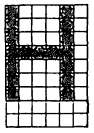
96 
97 
98 
99 

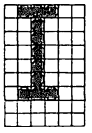
100 

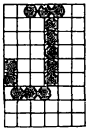
101 

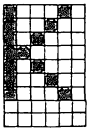
102 

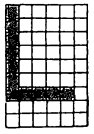
103 

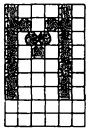
104 

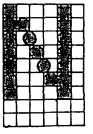
105 

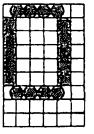
106 

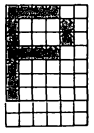
107 

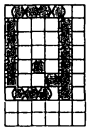
108 

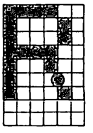
109 

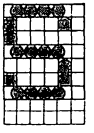
110 

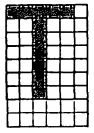
111 

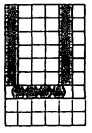
112 

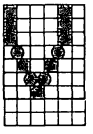
113 

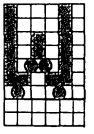
114 

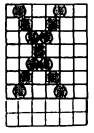
115 

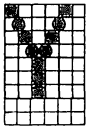
116 

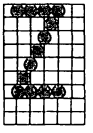
117 

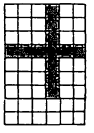
118 

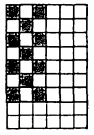
119 

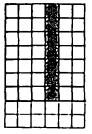
120 

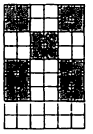
121 

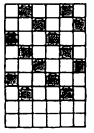
122 

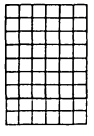
123 

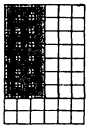
124 

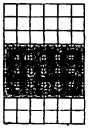
125 

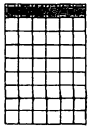
126 

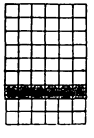
127 

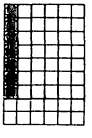
160 

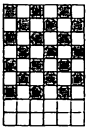
161 

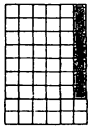
162 

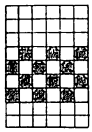
163 

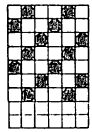
164 

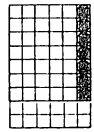
165 

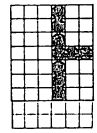
166 

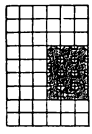
167 

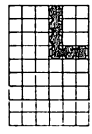
168 

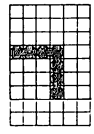
169 

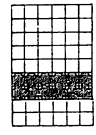
170 

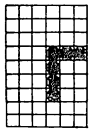
171 

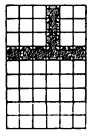
172 

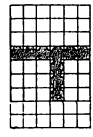
173 

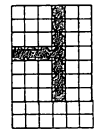
174 

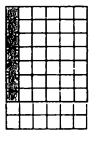
175 

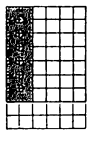
176 

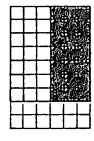
177 

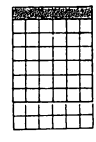
178 

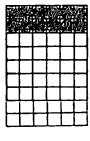
179 

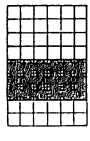
180 

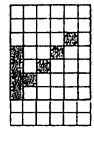
181 

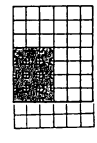
182 

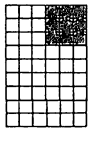
183 

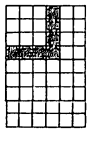
184 

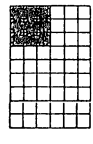
185 

186 

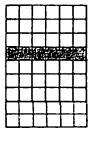
187 

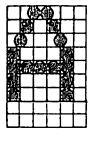
188 

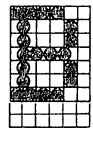
189 

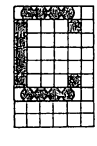
190 

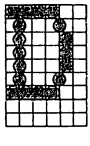
191 

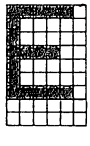
192 

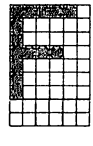
193 

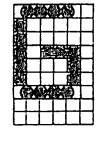
194 

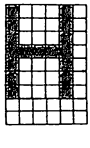
195 

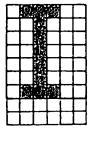
196 

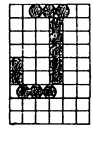
197 

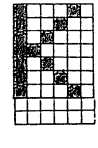
198 

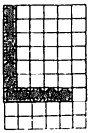
199 

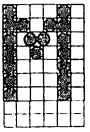
200 

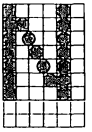
201 

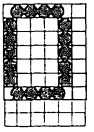
202 

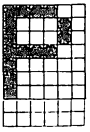
203 

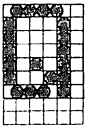
204 

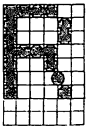
205 

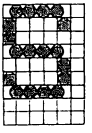
206 

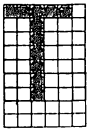
207 

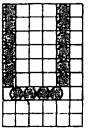
208 

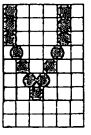
209 

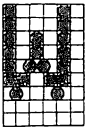
210 

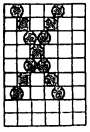
211 

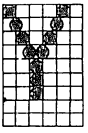
212 

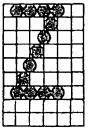
213 

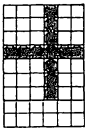
214 

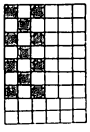
215 

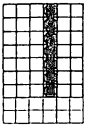
216 

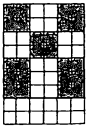
217 

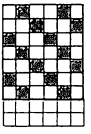
218 

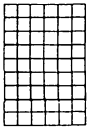
219 

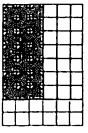
220 

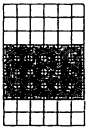
221 

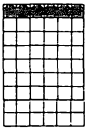
222 

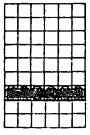
223 

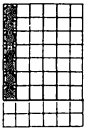
224 

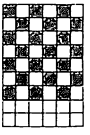
225 

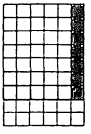
226 

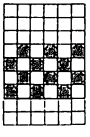
227 

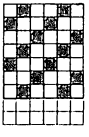
228 

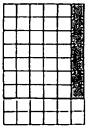
229 

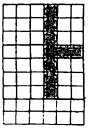
230 

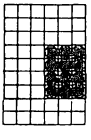
231 

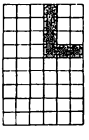
232 

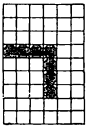
233 

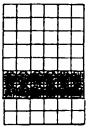
234 

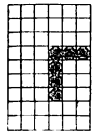
235 

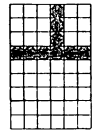
236 

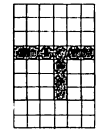
237 

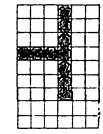
238 

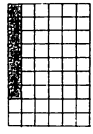
239 

240 

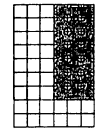
241 

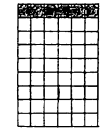
242 

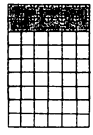
243 

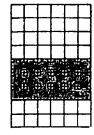
244 

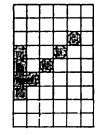
245 

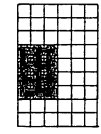
246 

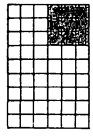
247 

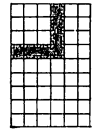
248 

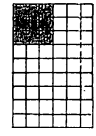
249 

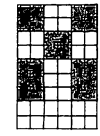
250 

251 

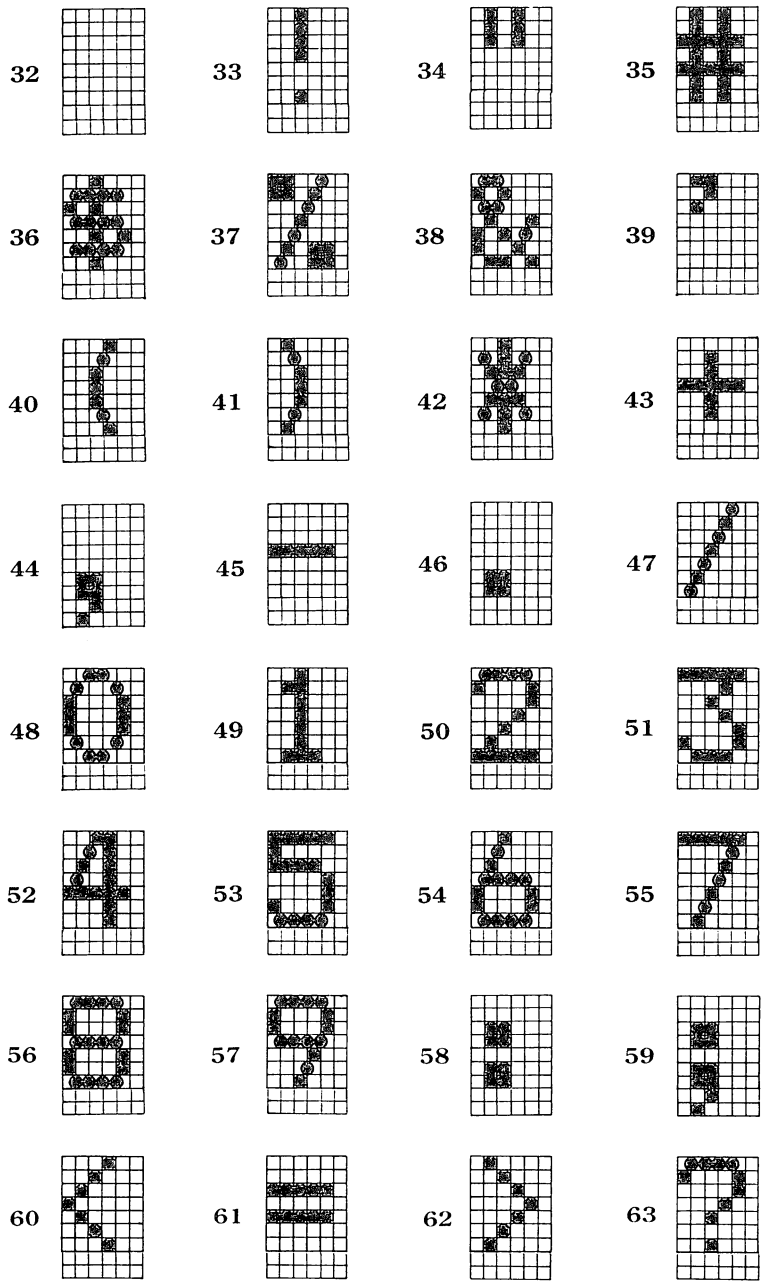
252 

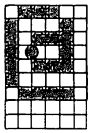
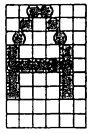
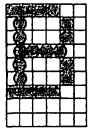
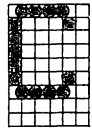
253 

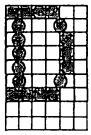
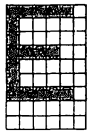
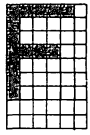
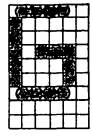
254 

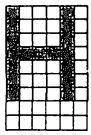
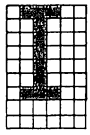
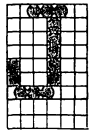
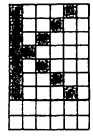
255 

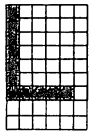
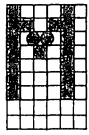
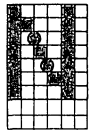
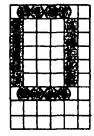
ASCII OPERATING MODE

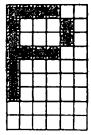
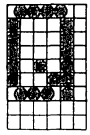
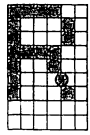
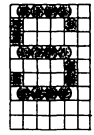


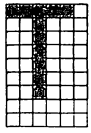
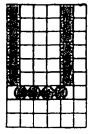
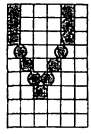
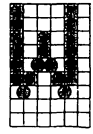
64 
65 
66 
67 

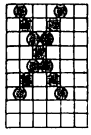
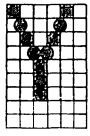
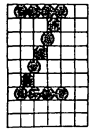
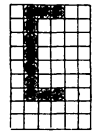
68 
69 
70 
71 

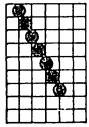
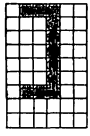
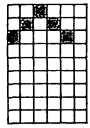
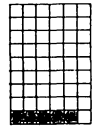
72 
73 
74 
75 

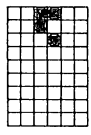
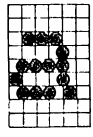
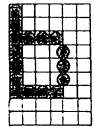
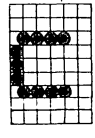
76 
77 
78 
79 

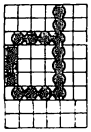
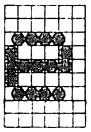
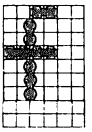
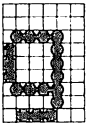
80 
81 
82 
83 

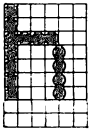
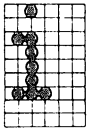
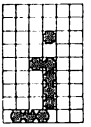
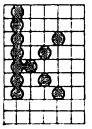
84 
85 
86 
87 

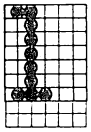
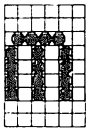
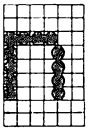
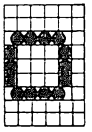
88 
89 
90 
91 

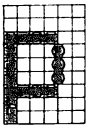
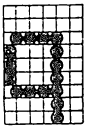
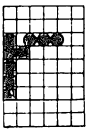
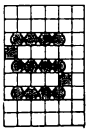
92 
93 
94 
95 

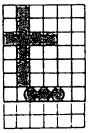
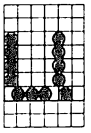
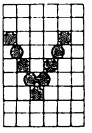
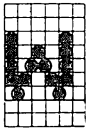
96 
97 
98 
99 

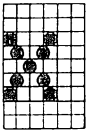
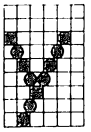
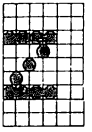
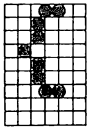
100 
101 
102 
103 

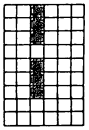
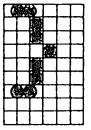
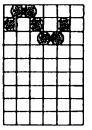
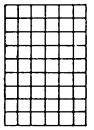
104 
105 
106 
107 

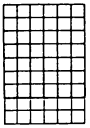
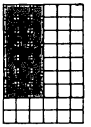
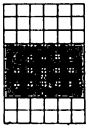
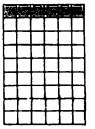
108 
109 
110 
111 

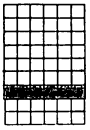
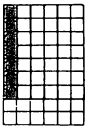
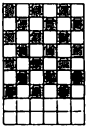
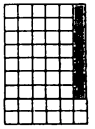
112 
113 
114 
115 

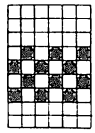
116 
117 
118 
119 

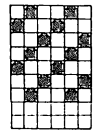
120 
121 
122 
123 

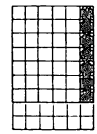
124 
125 
126 
127 

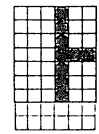
160 
161 
162 
163 

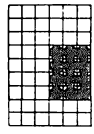
164 
165 
166 
167 

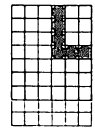
168 

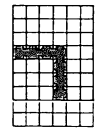
169 

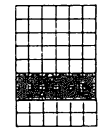
170 

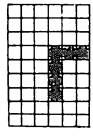
171 

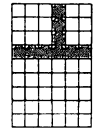
172 

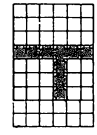
173 

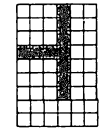
174 

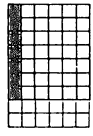
175 

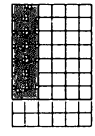
176 

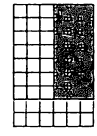
177 

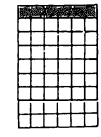
178 

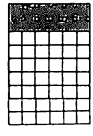
179 

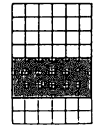
180 

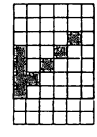
181 

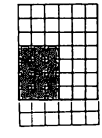
182 

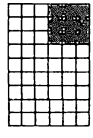
183 

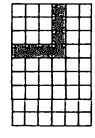
184 

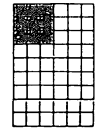
185 

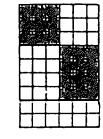
186 

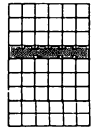
187 

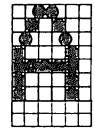
188 

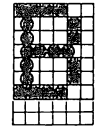
189 

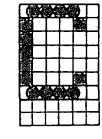
190 

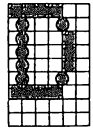
191 

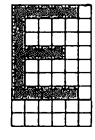
192 

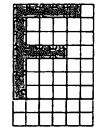
193 

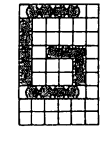
194 

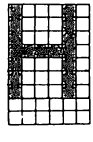
195 

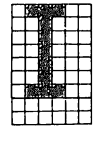
196 

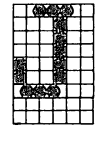
197 

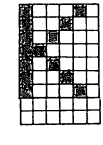
198 

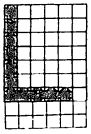
199 

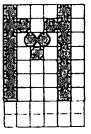
200 

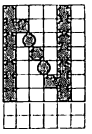
201 

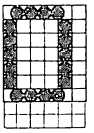
202 

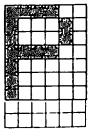
203 

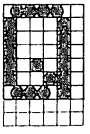
204 

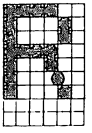
205 

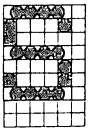
206 

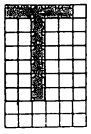
207 

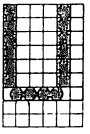
208 

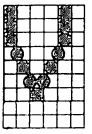
209 

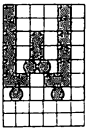
210 

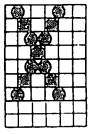
211 

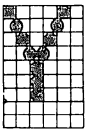
212 

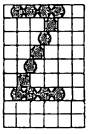
213 

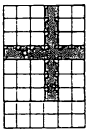
214 

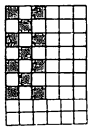
215 

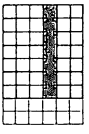
216 

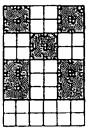
217 

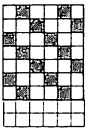
218 

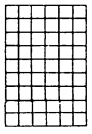
219 

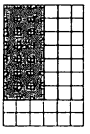
220 

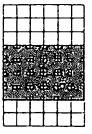
221 

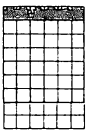
222 

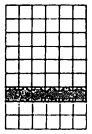
223 

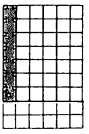
224 

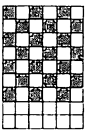
225 

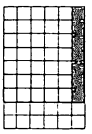
226 

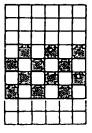
227 

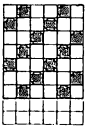
228 

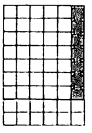
229 

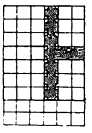
230 

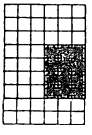
231 

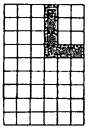
232 

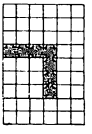
233 

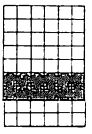
234 

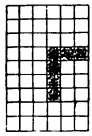
235 

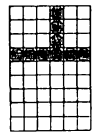
236 

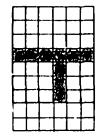
237 

238 

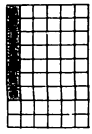
239 

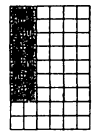
240 

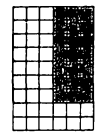
241 

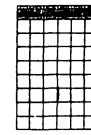
242 

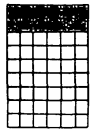
243 

244 

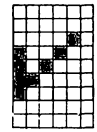
245 

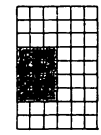
246 

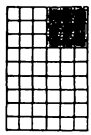
247 

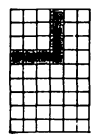
248 

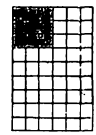
249 

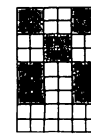
250 

251 

252 

253 

254 

255 

APPENDIX D

FUNCTION CODES

The purpose of this Appendix is to provide a quick reference for the various functions available on this printer. Codes are described in the following format.

PURPOSE	**Tells what the function code does.**
BASIC function	Gives the BASIC implementation of the ASCII code for the desired function.
REMARKS	Briefly describes how the command is used.
SEE	Tells where details of the command may be found.

Several commands require you to specify a value or values. In these cases, we have used an "*n*" or "*m*" to indicate a variable. You should insert the ASCII code for the proper value here.

SETTING THE OPERATING MODE; CHOOSING A CHARACTER SET

These commands allow you to select the operating mode of the printer and any of its many character sets.

■ Operating mode

PURPOSE	**Selects the ASCII operating mode.**
BASIC function	CHR$(27);CHR$(93);CHR$(49)
REMARKS	This command puts the printer in the ASCII operating mode so that it uses the ASCII character set and the ASCII mode commands. You can select the ASCII operating mode as the power-on default by turning DIP switch 1-5 off.
SEE	Chapter 2

PURPOSE **Selects the Commodore operating mode.**

BASIC function CHR$(27);CHR$(93);CHR$(48)

REMARKS This command puts the printer in the Commodore operating mode so that it uses the graphics or business character sets and the Commodore mode commands. You can select the Commodore operating mode as the power-on default by turning DIP switch 1-5 on.

SEE Chapter 2

■ Character sets

PURPOSE **Selects the "business mode" character set.**

BASIC function CHR$(17)

REMARKS This command selects the business mode character set and cancels the graphics mode character set. (You can also do this by specifying 7 as the secondary address in the OPEN statement.) This character set is available only in the Commodore operating mode.

SEE Chapter 2

PURPOSE **Selects the "graphics mode" character set.**

BASIC function CHR$(145)

REMARKS This command selects the graphics mode character set and cancels the business mode character set. This character set is available only in the Commodore operating mode.

SEE Chapter 2

PURPOSE **Selects an international character set.**

BASIC function CHR$(27);CHR$(82);CHR$(*n*)

REMARKS This command selects the international character set according to the value of *n* as shown below.

n Character set	*n* Character set
0 Commodore standard	4 France
1 U.S.A.	5 Sweden
2 Germany	6 Italy
3 Denmark	7 Spain

You can also change the default character set to any of the above by setting DIP switches 1-6, 1-7 and 1-8 as described in Appendix A.

SETTING THE PRINT MODE

These commands are used to control the print mode: character pitch and expanded printing, emphasis, boldface, and underlining.

■ Print mode command

PURPOSE **Sets the master print mode.**

BASIC function CHR$(27);CHR$(33);CHR$(*n*)

REMARKS This is a powerful command that allows the user to set several printing characteristics at one time: print pitch, expanded printing, emphasizing, boldface, underlining, and any combination of these as determined by *n*, a number from 0 to 255. (See Table 2-12 for details.)

SEE Chapter 2

■ Character pitch

PURPOSE	**Sets the print pitch to pica.**
BASIC function	CHR$(27);CHR$(80)
REMARKS	This command causes printing to be done in pica pitch with 80 characters per line. This command is ignored when the "Panel" mode is selected at the power-on.
SEE	Chapter 2

PURPOSE	**Sets the print pitch to pica.**
BASIC function	CHR$(18)
REMARKS	This command causes printing to be done in pica pitch with 80 characters per line under the ASCII operating mode. This command is ignored when the "Panel" mode is selected at the power-on.
SEE	Chapter 2

PURPOSE	**Sets the print pitch to elite.**
BASIC function	CHR$(27);CHR$(77)
REMARKS	This command causes printing to be done in elite pitch with 96 characters per line (NLQ characters are not printed in elite pitch). This command is ignored when the "Panel" mode is selected at the power-on.
SEE	Chapter 2

PURPOSE	**Sets the print pitch to condensed.**
BASIC function	CHR$(27);CHR$(15)
REMARKS	This command causes printing to be done in condensed pitch with 136 characters per line (NLQ characters are not printed in condensed). This command is ignored when the "Panel" mode is selected at the power-on.
SEE	Chapter 2

PURPOSE	**Sets the print pitch to condensed.**
BASIC function	CHR$(15)
REMARKS	This command causes printing to be done in condensed pitch with 136 characters per line under the ASCII operating mode (NLQ characters are not printed in condensed pitch). This command is ignored when the "Panel" mode is selected at the power-on.
SEE	Chapter 2

PURPOSE	**Turns on expanded printing.**
BASIC function	CHR$(27);CHR$(87);CHR$(49)
REMARKS	This command causes characters to be printed twice as wide as normally (half the current pitch) until expanded printing is cancelled.
SEE	Chapter 2

PURPOSE	**Turns on expanded printing.**
BASIC function	CHR$(14)
REMARKS	This command causes characters to be printed twice as wide as normally under the Commodore operating mode until expanded printing is cancelled.
SEE	Chapter 2

PURPOSE	**Turns off expanded printing.**
BASIC function	CHR$(27);CHR$(87);CHR$(48)
REMARKS	This command resets the character pitch to what it was before expanded printing was set.
SEE	Chapter 2

PURPOSE	**Turns off expanded printing.**
BASIC function	CHR$(15)
REMARKS	This command resets the character pitch to what it was before expanded printing was set under the Commodore operating mode.
SEE	Chapter 2

PURPOSE	**Turns on expanded printing for the remainder of the current line.**
BASIC function	CHR$(14)
REMARKS	This command causes characters to be printed twice as wide as normally until a carriage return is sent under the ASCII operating mode. It also cancelled with CHR$(20).
SEE	Chapter 2

PURPOSE	**Turns off expanded printing.**
BASIC function	CHR$(20)
REMARKS	This command cancels one line expanded print set with CHR$(14) under the ASCII operating mode.
SEE	Chapter 2

■ Emphasis and boldface

PURPOSE	**Turns on emphasized printing.**
BASIC function	CHR$(27);CHR$(69)
REMARKS	This command causes characters to be emphasized until emphasized printing is turned off.
SEE	Chapter 2
PURPOSE	**Turns off emphasized printing.**
BASIC function	CHR$(27);CHR$(70)
REMARKS	This command cancels emphasized printing.
SEE	Chapter 2
PURPOSE	**Turns on boldface printing.**
BASIC function	CHR$(27);CHR$(71)
REMARKS	This command causes characters to be printed in boldface until boldface is cancelled. Boldface cannot be used with superscripts, subscripts, or reverse characters. This command is ignored when the "Panel" mode is selected at the power-on.
SEE	Chapter 2
PURPOSE	**Turns off boldface printing.**
BASIC function	CHR$(27);CHR$(72)
REMARKS	This command turns off boldface printing and returns the printer to normal printing. This command is ignored when the "Panel" mode is selected at the power-on.
SEE	Chapter 2

■ Special types of printing

PURPOSE	**Turns on NLQ printing.**
BASIC function	CHR$(27);CHR$(120);CHR$(49)
REMARKS	This command causes the printer to print near letter quality (NLQ) characters until NLQ mode is cancelled. NLQ mode cannot be used with any other special printing functions except underlining, expanded printing, and big character printing. This command is ignored when the "Panel" mode is selected at the power-on.
SEE	Chapter 2
PURPOSE	**Turns off NLQ printing.**
BASIC function	CHR$(27);CHR$(120);CHR$(48)
REMARKS	This command cancels NLQ printing and returns the printer to the draft mode. This command is ignored when the "Panel" mode is selected at the power-on.
SEE	Chapter 2
PURPOSE	**Turns on proportional printing.**
BASIC function	CHR$(27);CHR$(112);CHR$(49)
REMARKS	This command causes draft characters to be printed with proportional spacing until proportional printing is cancelled.
SEE	Chapter 2
PURPOSE	**Turns off proportional printing.**
BASIC function	CHR$(27);CHR$(112);CHR$(48)
REMARKS	This command cancels the proportional printing and returns to the fixed pitch printing.
SEE	Chapter 2

PURPOSE	**Turns on underlining.**
BASIC function	CHR$(27);CHR$(45);CHR$(49)
REMARKS	This command underlines the following characters until underlining is cancelled.
SEE	Chapter 2

PURPOSE	**Turns off underlining.**
BASIC function	CHR$(27);CHR$(45);CHR$(48)
REMARKS	This command stops underlining.
SEE	Chapter 2

PURPOSE	**Prints superscripts.**
BASIC function	CHR$(27);CHR$(83);CHR$(48)
REMARKS	This command raises the following characters and prints them as superscripts until superscripting is cancelled. Superscripts are printed from left to right only and in boldface. Superscripts cannot be used with NLQ or reverse printing.
SEE	Chapter 2

PURPOSE	**Prints subscripts.**
BASIC function	CHR$(27);CHR$(83);CHR$(49)
REMARKS	This command lowers the following characters and prints them as subscripts until subscripting is cancelled. All conditions described for superscripts also apply to subscripts.
SEE	Chapter 2

PURPOSE	**Cancels a superscript or subscript.**
BASIC function	CHR$(27);CHR$(84)
REMARKS	This command stops printing of superscripts or subscripts and sets normal printing. It also cancels uni-directional printing and boldface, which are set automatically for superscripts and subscripts.
SEE	Chapter 2
PURPOSE	**Turns on italics.**
BASIC function	CHR$(27);CHR$(52)
REMARKS	This command causes NLQ characters to be printed in italics until italic printing is cancelled.
SEE	Chapter 2
PURPOSE	**Turns off italic printing.**
BASIC function	CHR$(27);CHR$(53)
REMARKS	This command cancels italics.
SEE	Chapter 2
PURPOSE	**Turns on reverse printing.**
BASIC function	CHR$(18)
REMARKS	This command causes all characters to be printed white on a black background in pica pitch until reverse printing is cancelled under the Commodore operating mode. ***Warning:*** Do not use reverse printing for more than five consecutive lines — extended printing in this mode will damage the print head.
SEE	Chapter 2

PURPOSE	**Turns off reverse printing.**
BASIC function	CHR$(146)
REMARKS	This command resets the print mode to whatever it was before reverse printing was turned on.
SEE	Chapter 2

CONTROLLING THE VERTICAL PRINT POSITION

These commands are used to move the paper relative to the print head. By moving the paper up or down, the print head, in effect, moves the opposite direction (down or up) on the page.

■ Line feed and reverse line feed

PURPOSE	**Advances the paper one line (Line Feed).**
BASIC function	CHR$(10)
REMARKS	The actual distance advanced by the line feed is set through various codes which can be sent (see below). When DIP switch 1-1 is "on" a line feed is automatically generated whenever the printer receives a carriage return.
SEE	Chapter 3

PURPOSE	**Reverse the paper one line.**
BASIC function	CHR$(27);CHR$(10)
REMARKS	This command causes the printer to reverse the paper (in effect moving the print head up on the sheet) one line. The actual distance travelled is set through various codes which can be sent (see below).
SEE	Chapter 3

PURPOSE	**Sets line spacing to 1/8 inch.**
BASIC function	CHR$(27);CHR$(48)
REMARKS	This command sets the distance the paper advances or reverses during all subsequent line feeds to 1/8 inch.
SEE	Chapter 3

PURPOSE	**Sets line spacing to 7/72 inch.**
BASIC function	CHR$(27);CHR$(49)
REMARKS	This command sets the actual distance the paper advances or reverses during all subsequent line feeds to 7/72 inch.
SEE	Chapter 3

PURPOSE	**Sets line spacing to 1/6 inch.**
BASIC function	CHR$(27);CHR$(50)
REMARKS	This command sets the actual distance the paper advances or reverses during all subsequent line feeds to 1/6 inch.
SEE	Chapter 3

PURPOSE	**Sets line spacing to *n*/216 inch.**
BASIC function	CHR$(27);CHR$(51);CHR$(*n*)
REMARKS	This command sets the actual distance the paper advances or reverses during all subsequent line feeds to *n*/216 inch. The value of *n* must be between 1 and 255.
SEE	Chapter 3

PURPOSE	**Sets line spacing to *n*/72 inch.**
BASIC function	CHR$(27);CHR$(65);CHR$(*n*)
REMARKS	This command sets the actual distance the paper advances or reverses during all subsequent line feeds to *n*/72 inch. The value of *n* must be between 1 and 255.
SEE	Chapter 3

PURPOSE	**Sends a one-time paper feed of *n*/216 inch.**
BASIC function	CHR$(27);CHR$(74);CHR$(*n*)
REMARKS	This command causes the printer to advance the paper *n*/216 inch. It does not change the current value of line spacing and it does not cause a carriage return. The value of *n* must be between 1 and 255.
SEE	Chapter 3

■ Form feed and related commands

PURPOSE	**Advances the paper to the top of the next page (form feed).**
BASIC function	CHR$(12)
REMARKS	The actual length of a page ejected by a form feed is set either by the setting of DIP switch 1-4 or through various codes which can be sent (see below). This command works as the ejecting paper command when the optional automatic sheet feeder is installed.
SEE	Chapter 3

PURPOSE	**Reverses the paper to the top of the current page.**
BASIC function	CHR$(27);CHR$(12)
REMARKS	This command causes the printer to reverse the paper to the top of the current printing page (or form). This command is ignored when the optional automatic sheet feeder is installed.
SEE	Chapter 3

PURPOSE	**Sets page length to *n* inches.**
BASIC function	CHR$(27);CHR$(67);CHR$(0);CHR$(*n*)
REMARKS	This command sets the length of all subsequent pages to *n* inches. The value of *n* must be between 1 and 32. You can select a power-on default form length of 11 inches or 12 inches by setting DIP switch 1-4. This command is ignored when the optional automatic sheet feeder is installed.
SEE	Chapter 3

PURPOSE	**Sets page length to *n* lines.**
BASIC function	CHR$(27);CHR$(67);CHR$(*n*)
REMARKS	This command sets the length of all subsequent pages to *n* lines. The value of *n* must be between 1 and 255. This command is ignored when the optional automatic sheet feeder is installed.
SEE	Chapter 3

■ Top/bottom margins and vertical tabs

PURPOSE	**Sets the top margin to *n* lines.**
BASIC function	CHR$(27);CHR$(114);CHR$(*n*)
REMARKS	This command sets the top margin to *n* lines. Printing begins on the (*n* + 1)th line on the page. This command is ignored when the optional automatic sheet feeder is installed.
SEE	Chapter 3

PURPOSE	**Sets the bottom margin to *n* lines.**
BASIC function	CHR$(27);CHR$(78);CHR$(*n*)
REMARKS	This command sets the bottom margin to *n* lines. The printer will generate a form feed whenever there are *n* lines left on the page. This command is ignored when the optional automatic sheet feeder is installed. The value of *n* must be between 1 and 255.
SEE	Chapter 3

PURPOSE	**Sets the bottom margin to 6 lines.**
BASIC function	CHR$(147)
REMARKS	This command sets the bottom margin to 6 lines. The printer will generate a form feed whenever there are 6 lines left on the page. This command is ignored when the optional automatic sheet feeder is installed.
SEE	Chapter 3

PURPOSE	**Cancels top and bottom margins.**
BASIC function	CHR$(27);CHR$(79)
REMARKS	This command cancels both the top and the bottom margins.
SEE	Chapter 3

PURPOSE	**Cancels top and bottom margins.**
BASIC function	CHR$(19)
REMARKS	This command cancels the top and the bottom margins.
SEE	Chapter 3

PURPOSE	**Advances paper to the next vertical tab position.**
BASIC function	CHR$(11)
REMARKS	This command causes the paper to be advanced to the next vertical tab position, or the top of the next page, whichever it finds first. If the vertical tab positions are not set, this command works as a line feed command.
SEE	Chapter 3

PURPOSE	**Sets vertical tab positions.**
BASIC function	CHR$(27);CHR$(66);CHR$(*n1*); CHR$(*n2*);.....;CHR$(0)
REMARKS	This command cancels all current vertical tab positions and sets those defined at lines *n1, n2, n3,* etc. The maximum number of vertical tab positions allowed is 20. The ASCII 0 character is used as a command terminator. Each vertical tab position must be specified in ascending order.
SEE	Chapter 3

CONTROLLING THE HORIZONTAL PRINT POSITION

This section described commands that move the print head and restrict its printing range (such as setting margins and tabs).

PURPOSE	**Returns print head to the left margin (carriage return).**
BASIC function	CHR$(13)
REMARKS	This command returns the print head to the left margin. If DIP switch 1-1 has been set on, then this command will also cause a line feed character to be generated after the carriage return, thereby advancing to the beginning of the next print line automatically.
SEE	Chapter 3

PURPOSE	**Sets the left margin.**
BASIC function	CHR$(27);CHR$(108);CHR$(*n*)
REMARKS	This command sets the left margin to *n* characters. Each line will begin in the (*n* + 1)th character position from the left edge. The value of *n* must be between 0 and 255. You can set the left margin manually with the control panel. *NOTE:* Changing the print pitch after the left margin has been set does not change the margin — it stays in exactly the same place on the page.
SEE	Chapter 3

PURPOSE	**Sets the right margin.**
BASIC function	CHR$(27);CHR$(81);CHR$(*n*)
REMARKS	This command sets the right margin to *n*, which is the last character position that can be printed in a line. After execution of this command, any attempt to print beyond print position *n* will cause the printer to automatically generate a carriage return and a line feed before printing the remainder of the line. The value of *n* must be between 1 and 255. You can set the right margin manually with the control panel. *NOTE:* Changing the print pitch after the right margin has been set does not change the margin — it stays in exactly the same place on the page.
SEE	Chapter 3

PURPOSE	**Moves the print head to the next horizontal tab position.**
BASIC function	CHR$(9)
REMARKS	This command causes the print head to advance to the next horizontal tab position under the ASCII operating mode. The horizontal tab positions are set at power-on to print positions 8, 16, 24, etc. (to the maximum print position).
SEE	Chapter 3

PURPOSE	**Sets horizontal tab positions.**
BASIC function	CHR$(27);CHR$(68);CHR$(*n1*); CHR$(*n2*);.....;CHR$(0)
REMARKS	This command cancels all current horizontal tab positions and sets those defined at print positions *n1, n2, n3,* etc. The maximum number of horizontal tab positions allowed is 40. The ASCII 0 character is used as a command terminator. Each horizontal tab position must be specified in ascending order.
SEE	Chapter 3

PURPOSE	**Skips print position.**
BASIC function	CHR$(16);CHR$(*n1*);CHR$(*n2*)
REMARKS	This command determines the print start position from the home position (the left edge). *n1* and *n2* should be between "0" (decimal code 48) to "9" (decimal code 57).

PURPOSE	**Skips print position in dot units.**
BASIC function	CHR$(27);CHR$(16);CHR$(*n1*); CHR$(*n2*)
REMARKS	This command determines the print start position from the home position (the left edge) in dot units. The range is from 0 to 479 using $n1 \times 256 + n2$.

PURPOSE	**Moves the print head back one print position (backspace).**
BASIC function	CHR$(8)
REMARKS	This command shifts the print head one column to the left under the ASCII operating mode. If the print head is at the left margin, the command is ignored. This command can be used to overstrike characters.
SEE	Chapter 4

PURPOSE	**Sets alignment, or centering.**
BASIC function	CHR$(27);CHR$(97);CHR$(*n*)
REMARKS	This command causes the printer to format text as follows: *n* Text formatting 0 Left-aligned (ragged right margin) 1 Centered 2 Right-aligned
SEE	Chapter 3

DOWNLOAD CHARACTER COMMANDS

PURPOSE	**Defines download characters into RAM.**
BASIC function	CHR$(27);CHR$(38);CHR$(0); CHR$(*n1*);CHR$(*n2*);CHR$(*m0*); CHR$(*m1*);CHR$(*m2*);...;CHR$(*m11*) [;CHR$(*m12*);CHR$(*m13*);.....; CHR$(*m46*)]
REMARKS	This command is used to set up one or more user-defined characters and store them into RAM for later use. RAM is cleared when the power is turned off. The values of *n1* and *n2* specify the range of positions in RAM that the characters are to occupy. Valid character positions are any number between 33 and 126. Following *n2* this printer expects character data bytes for each character to be defined. The first byte, *m0*, is the attribute bytes, for it specifies whether the character is a descender (if the first bit is 0), and the proportional width of the draft character (starting and ending dot columns are defined by the low order seven bits). *m1* through *m11* determine which dots form the draft character. In case of NLQ download character, *m1* through *m46* determine which dots form the character.
SEE	Chapter 5

PURPOSE	**Copies standard character ROM font into RAM.**
BASIC function	CHR$(27);CHR$(58);CHR$(0);CHR$(0); CHR$(0)
REMARKS	This command copies the ASCII character set to the corresponding download character RAM area. This destroys any existing user-defined characters in that code range.
SEE	Chapter 5
PURPOSE	**Selects download character set.**
BASIC function	CHR$(27);CHR$(37);CHR$(49); CHR$(0)
REMARKS	This command causes the printer to select the download character set.
SEE	Chapter 5
PURPOSE	**Cancels download character set.**
BASIC function	CHR$(27);CHR$(37);CHR$(48); CHR$(0)
REMARKS	This command cancels the download character set and selects the pervious character set.
SEE	Chapter 5

DOT GRAPHICS COMMANDS

PURPOSE	**Prints 7-pin normal-density graphics.**
BASIC function	CHR$(8);CHR$(*m1*);CHR$(*m2*);.....
REMARKS	This command selects 60 dots-per-inch column-scan, 7-pin bit-image graphics mode under the Commodore operating mode. The ASCII value of graphics data bytes (*m1*, *m2*, etc.) determine which pins are fired for each character. This command also sets the line spacing to 7/72 inches. The 7-pin dot graphics mode is cancelled by the expanded select/cancel commands.
SEE	Chapter 6
PURPOSE	**Prints 7-pin double-density graphics.**
BASIC function	CHR$(9);CHR$(*m1*);CHR$(*m2*);.....
REMARKS	This command selects 120 dots-per inch column-scan, 7-pin bit-image graphics mode under the Commodore operating mode. The ASCII value of graphics data bytes (*m1*, *m2* etc.) determine which pins are fired for each character. This command also sets the line spacing to 7/72 inch. The 7-pin dot graphics mode is cancelled by the expanded select/cancel commands.
SEE	Chapter 6

PURPOSE — **Prints 7-pin reverse normal-density graphics.**

BASIC function — CHR$(27);CHR$(18);CHR$(*m1*); CHR$(*m2*);.....

REMARKS — This command selects 60-dots-per-inch column-scan, 7-pin reverse bit-image graphics mode under the Commodore operating mode. The ASCII value of graphics data bytes (*m1, m2,* etc.) determine which pins are fired for each character. This command also sets the line spacing to 7/72 inch. The 7-pin dot graphics mode is cancelled by the expanded select/cancel commands.

SEE — Chapter 6

PURPOSE — **Repeats graphics data.**

BASIC function — CHR$(26);CHR$(*n*);CHR$(*m*)

REMARKS — This command repeats the selected graphics data (*m*) for *n* times while in the 7-pin dot graphics modes.

SEE — Chapter 6

PURPOSE — **Prints 8-pin normal-density graphics.**

BASIC function — CHR$(27);CHR$(75);CHR$(*n1*); CHR$(*n2*);CHR$(*m1*);CHR$(*m2*);.....

REMARKS — This command selects 60 dots-per-inch column-scan, 8-pin bit-image graphics mode. The values of *n1* and *n2* represent the number of graphics characters to be printed, where the total number of characters = *n2* times 256 + *n1*. The correct number of graphics data bytes (*m1, m2,* etc.) must follow *n2*. The ASCII value of these characters determine which pins are fired for each character.

SEE — Chapter 6

PURPOSE **Prints 8-pin double-density graphics.**

BASIC function CHR$(27);CHR$(76);CHR$(*n1*); CHR$(*n2*);CHR$(*m1*);CHR$(*m2*);.....

REMARKS This command selects 120 dots-per-inch column-scan, 8-pin bit-image graphics mode. The values of *n1* and *n2* are the same as in 8-pin normal-density graphics. The correct number of graphics data bytes (*m1, m2,* etc.) must follow *n2*. The ASCII value of these characters determine which pins are fired for each character.

SEE Chapter 6

PURPOSE **Prints 8-pin double-density graphics with double-speed.**

BASIC function CHR$(27);CHR$(89);CHR$(*n1*); CHR$(*n2*);CHR$(*m1*);CHR$(*m2*);.....

REMARKS This command selects 120 dots-per-inch column-scan, 8-pin bit-image graphics mode with double-speed. The values of *n1* and *n2* are the same as in 8-pin normal-density graphics. The correct number of graphics data bytes (*m1, m2,* etc.) must follow *n2*. The ASCII value of these characters determine which pins are fired for each character.

SEE Chapter 6

PURPOSE **Prints 8-pin quadruple-density graphics.**

BASIC function CHR$(27);CHR$(90);CHR$(*n1*);
CHR$(*n2*);CHR$(*m1*);CHR$(*m2*);.....

REMARKS This command selects 240 dots-per-inch column-scan, 8-pin bit-image graphics mode. The values of *n1* and *n2* are the same as in 8-pin normal-density graphics. The correct number of graphics data bytes (*m1, m2,* etc.) must follow *n2*. The ASCII value of these characters determine which pins are fired for each character.

SEE Chapter 6

PURPOSE **Selects graphics modes.**

BASIC function CHR$(27);CHR$(42);CHR$(*n0*);
CHR$(*n1*);CHR$(*n2*);CHR$(*m1*);
CHR$(*m2*);.....

REMARKS This command selects one four possible 8-pin graphics modes, depending on the value of *n0*. The values of *n1* and *n2* are the same as in 8-pin normal-density graphics. The correct number of graphics data bytes (*m1, m2,* etc.) must follow *n2*. The ASCII value of these characters determine which pins are fired for each character. The value of *n0* must be between 0 and 3 as shown below:

n	Mode
0	8-pin normal-density
1	8-pin double-density
2	8-pin double-density with double-speed
3	8-pin quadruple-density

MACRO INSTRUCTION COMMANDS

PURPOSE	**Defines macro instruction.**
BASIC function	CHR$(27);CHR$(43);.....;CHR$(30)
REMARKS	This command cancels any existing macro instruction, and replaces it with the instruction defined. The maximum number of characters allowed in the macro instruction is 16. The CHR$(30) marks the end of the macro definition.
SEE	Chapter 4

PURPOSE	**Executes macro instruction.**
BASIC function	CHR$(27);CHR$(43);CHR$(1)
REMARKS	This command executes a macro instruction that was previously defined.
SEE	Chapter 4

OTHER COMMANDS

PURPOSE	**Prints "slash zero".**
BASIC function	CHR$(27);CHR$(126);CHR$(49)
REMARKS	This command causes to print "zero" with slash.
SEE	Chapter 4

PURPOSE	**Prints "normal zero".**
BASIC function	CHR$(27);CHR$(126);CHR$(48)
REMARKS	This command cancels to print "slash zero" and returns to print "normal zero".
SEE	Chapter 4

PURPOSE **Enlarges characters in whole or in part; cancels same.**

BASIC function CHR$(27);CHR$(104);CHR$(*n*)

REMARKS This special command enlarges characters following the command until the enlargement is cancelled. The values of *n* have the following effects.

n	Effect
0	Cancels enlargement
1	Double-high, double-wide
2	Quadruple-high, quadruple-wide
3	Double-high, double-wide (Lower half only)
4	Double-high, double-wide (Upper half only)
5	Quadruple-high, quadruple-wide (Lower half only)
6	Quadruple-high, quadruple-wide (Upper half only)

SEE Chapter 4

PURPOSE **Sound printer bell.**

BASIC function CHR$(7)

REMARKS This command causes the printer tone to sound for approximately one-fourth second.

SEE Chapter 4

PURPOSE **Selects auto feed mode.**

BASIC function CHR$(27);CHR$(25);CHR$(4)

REMARKS This command causes the printer to select the auto sheet feeding mode. This command is ignored when the optional automatic sheet feeder is not mounted.

SEE Chapter 4

PURPOSE	**Selects auto feed mode.**
BASIC function	CHR$(40);CHR$(40);CHR$(52); CHR$(41);CHR$(41)
REMARKS	The same as CHR$(27);CHR$(25); CHR$(4), above.

PURPOSE	**Cancels auto feed mode.**
BASIC function	CHR$(27);CHR$(25);CHR$(0)
REMARKS	This command causes the printer to cancel the auto sheet feeding mode. This command is ignored when the optional automatic sheet feeder is not mounted.
SEE	Chapter 4

PURPOSE	**Cancels auto feed mode.**
BASIC function	CHR$(40);CHR$(40);CHR$(48); CHR$(41);CHR$(41)
REMARKS	The same as CHR$(27);CHR$(25); CHR$(0), above.

PURPOSE	**Supplies paper.**
BASIC function	CHR$(27);CHR$(25);CHR$(1)
REMARKS	This command causes the printer to supply paper under non-auto sheet feeding mode. This command is ignored when the optional automatic sheet feeder is not mounted.
SEE	Chapter 4

PURPOSE	**Supplies paper.**
BASIC function	CHR$(40);CHR$(40);CHR$(49); CHR$(41);CHR$(41)
REMARKS	The same as CHR$(27);CHR$(25); CHR$(1), above.

PURPOSE	**Ejects paper.**
BASIC function	CHR$(27);CHR$(25);CHR$(82)
REMARKS	This command causes the printer to eject paper. This command is ignored when the optional automatic sheet feeder is not mounted.
SEE	Chapter 4

PURPOSE	**Ejects paper.**
BASIC function	CHR$(40);CHR$(40);CHR$(82); CHR$(41);CHR$(41)
REMARKS	The same as CHR$(27);CHR$(25); CHR$(82), above.

PURPOSE	**Resets the printer.**
BASIC function	CHR$(27);CHR$(64)
REMARKS	This command reinitializes the printer. The print buffer is cleared, and the form length, operating mode, and international character set are all reset to the values defined by their respective DIP switches. The main difference between this command and turning the printer off and back on is that download character RAM and the macro instruction are preserved with this command.
SEE	Chapter 4

APPENDIX E

TECHNICAL SPECIFICATIONS

Interface	Commodore serial
Printing direction	Bidirectional, logic seeking Unidirectional in dot graphics
Character sets	Near Letter Quality (NLQ) characters Upper- and lower-case letters, numbers, and symbols Draft-quality character set Upper- and lower-case letters, numbers, symbols, and block graphics
Character matrix	18 × 23 dots, NLQ 9 × 11 dots, standard draft 7 × 11 dots, block graphics 7 or 8 × 480 dots, dot graphics 7 or 8 × 960 dots, dot graphics 8 × 1920 dots, dot graphics
Line spacing	1/6 or 1/8 inch standard *n*/72 or *n*/216 inch programmable
Page width	
Expanded Pica (5 CPI)	40 char./line
Expanded Elite (6 CPI)	48 char./line
Expanded Condensed (8.5 CPI)	68 char./line
Pica (10 CPI)	80 char./line
Elite (12 CPI)	96 char./line
Condensed (17 CPI)	136 char./line

Special features	Near Letter Quality printing Short form tear-off Easy access format switches Self-test Hex dump Skip over perforation

MEMO

INDEX

Consumer Response

Star Micronics Co., Ltd. invites your suggestions and comments on this publication and the printer that it was written for. Please address your correspondence to:

For Worldwide Headquarters:
STAR MICRONICS CO., LTD.
194 Nakayoshida
Shizuoka, JAPAN 422-91
Attn: Product Manager

For American Market:
STAR MICRONICS AMERICA INC.
P.O. Box 1630
El Toro, California, U.S.A. 92630
Attn: Product Manager

For European Market:
STAR MICRONICS DEUTSCHLAND GMBH
Frankfurter Allee 1-3.
D-6236 Eschborn/Ts., W. Germany
Attn: Product Manager

For U.K Market:
STAR MICRONICS U.K. LTD.
Craven House, 4th. Floor
40 Uxbridge Road, Ealing W.5, London
Attn: Product Manager

For Asian Market:
STAR MICRONICS (S.E.A.) LTD.
Rm 2407-8 Sincere Building;
173 Des Voeux Road Central, Hong Kong
Attn: Product Manager

star **COMMODORE** I/F CARTRIDGE FOR NL-10 ***USERS MANUAL***

star™
micronics

THE POWER BEHIND THE PRINTED WORD.

PRINTED IN JAPAN