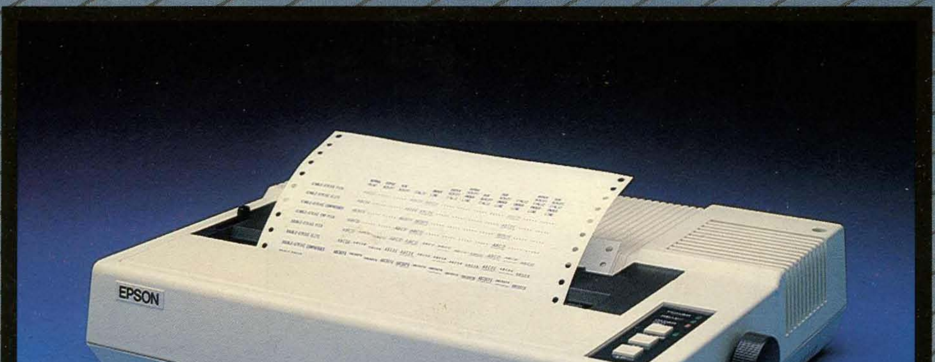
EPSON

User's Manual



FX-80 Printer

P8294014-0
Y440991010

EPSON
FX-80 PRINTER
USER'S MANUAL

By

David A. Kater

FCC COMPLIANCE STATEMENT FOR AMERICAN USERS

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 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, DC 20402. Stock No. 004-000-00345-4.

FX-80 User's Manual
The Chapters at a Glance

Table of Contents

Table of Figures

Table of Tables

Introduction

So Epson has a new printer on the market!

As people develop expertise in the use of their printers, they discover new applications, and consequently demand more from their printers. To cope with these expanding demands, printer technology is moving ahead at a breakneck pace; printer mechanisms are becoming more sophisticated, and printer brains are approaching genius levels.

We at Epson are striving to keep you one step ahead of the field so that you can tackle new printing jobs as fast as you can think of them. And with the FX-80 printer, we've done just that. This printer is packed with features just waiting for new applications.

FX-80 Features

The MX series of printers attracted enough attention to become the most popular line of printers in the industry. The FX-80 printer follows in the same grand tradition. Its power-packed assortment of features includes:

- Upward compatibility with Graftrax Plus features.
- Several different print modes that can be combined to produce a variety of print styles. These include:

  Roman and italic print fonts
  Six different print pitches
  Two kinds of bold printing

- Sixteen unique print mode combinations that can be selected by number.
- Proportionally spaced characters for professional looking documents.
- Easy-to-use Underline and Super/Subscript Modes.

- Advanced forms handling capability, including horizontal and vertical tabs, margin select, skip-over-perforation, and variable forward and reverse line feeds.
- User-definable character sets. With this powerful feature you can create your own alphabets and special symbols.
- High-resolution graphics images with six different densities. Create your own charts, diagrams, figures, and illustrations.
- International characters.
- Typewriter simulation mode.
- Program debugging mode (hexadecimal echo of codes received from the computer).
- Fast print speed—160 characters per second for rapid processing of documents.
- 2K print buffer for smoother operation.
- Epson reliability and quality.
- Built-in friction and tractor feeders with an optional tractor unit for narrow forms.
- Disposable print head.
- Easy-to-reach switch settings to customize printer features.

In short, the FX-80 is loaded with features that will challenge your ability to put it to work.

What's Inside the Printer?

The FX-80 printer contains two kinds of internal memory. There is 12K (approximately 12,000 characters) of ROM (Read Only Memory). This permanent memory contains all the logic required for the various print features as well as the patterns for the several built-in character sets. The FX-80 also contains 2K (about 2,000 characters) of RAM (Random Access Memory). RAM can be used in two different ways:

1) RAM can be used as a memory "buffer" that stores up to 2K of text and printer commands as they are received from the computer. This frees the computer for further editing while the printer continues printing.

2) RAM can be used to store a complete set of user-defined characters and symbols. Now you can create all those special symbols you wished for but your printer never had.

The FX-80 is indeed a powerful printer. But do not fear. You have your hands on an equally powerful *User's Manual* that will patiently guide you through the printer's features and warn you about pitfalls. To appreciate what we mean by pitfalls, imagine yourself in the following scenario:

WELCOME TO FX-80 ADVENTURE!

YOU ARE STANDING IN THE FAMILY ROOM NEXT TO YOUR COMPUTER DESK. THE KIDS ARE ACROSS THE ROOM PLAYING A VIDEO GAME ON THE TV. ON YOUR DESK IS A LARGE BOX WITH LETTERING.

YOUR COMMAND? *read lettering*

THE BOX IS LABELED "EPSON"

YOUR COMMAND? *open box*

INSIDE THE BOX YOU FIND A BRAND NEW FX-80 PRINTER.

A USER'S MANUAL FALLS TO THE FLOOR.

AN EERIE VOICE (YOUR CONSCIENCE MAYBE?) WHISPERS: "READ THE MANUAL FIRST."

YOUR COMMAND? *ignore manual; connect printer to computer*

KABOOOOM! THE PRINTER EXPLODES, VAPORIZING EVERYTHING WITHIN 5 FEET OF THE DESK. THE KIDS CONTINUE BLASTING ALIENS ON THE TV, UNAWARE OF YOUR FATE.

YOU ARE DEAD. PLAY AGAIN?

The result is admittedly exaggerated, but the underlying message is very clear: when all else fails, READ THE MANUAL!!

This advice is too late for our unfortunate adventurer, but who can blame him? It is always tempting (though not always wise) to start playing with a new printer the instant it is out of the box. The FX-80 is an advanced printer, complete with state-of-the-art features that can be adjusted to fit your particular needs. So it is more important than ever that you fully understand what the printer will do and how to make it do it before you flip the KABOOM switch.

"Okay, I'll spend some time learning how to set up the printer, but why should I bother with the advanced features when I can simply buy programs that will control them for me?" Because whenever a new printer comes on the market, it takes time for the professional programmers to develop applications programs utilizing the advanced features. And even more important, if and when the program of your dreams finally becomes available, it will likely require some customization to meet your EXACT needs. So learning how to control your FX-80 now will enable you to fully exploit the applications programs as they roll in. In the meantime, you can take advantage of your printer's capabilities.

Using the Manual

Now that you've decided to take the plunge, you're in for a treat. Helping you learn how to use the FX-80 to its fullest is one of the places where we really shine. This manual will take you on a carefully planned guided tour of the various features of the FX-80 printer. You can learn just how to use your computer to control the printer for a variety of applications.

This manual can be used as a reference, a tutorial study guide, or some combination thereof. The key is flexibility; the manual can be different things to different people:

Those of you who want to use the printer for one simple application (like listing BASIC programs or doing simple word processing) need only Chapter 1, the Appendixes, and a knowledge of the program you are using. If you decide to learn about the advanced features of the printer at a later time, fine. The lessons will be waiting for you.

For those who prefer to roll up their sleeves and see how the printer works firsthand, we've used sample programs to demonstrate each of the printer features.

For those who only want a quick and easy reference, the comprehensive Table of Contents, set of Appendixes, and Index provide ready access to information.

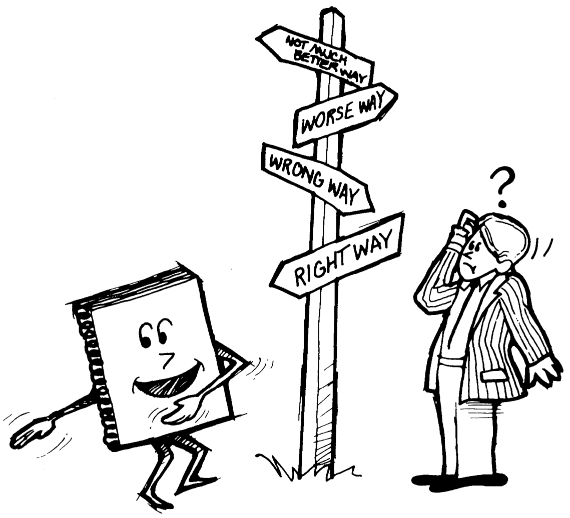
For computer professionals and other "experienced users" who simply can't wait to find out what the printer will do (regardless of the consequences), we have a special section entitled "One Easy Lesson." It gets you up and running fast, then turns you loose on a program that demonstrates several of the printer's features. This program, the Appendixes, and the Quick Reference Card will bring you quickly up to speed.

For those users who are familiar with the MX series of printers, Appendix G provides a summary of the differences between the FX-80 and the MX Graftrax Plus.

Most readers, however, will want to start with Chapter 1 and follow the manual from start to finish.

Your Personal Guide

Think of the manual as your personal guide in your exploration of the FX-80's many features.



It will teach you step by step how to operate the printer as you work through examples on your computer. Along the way, you may stumble across a few things that the BASIC language on your computer system can't handle. We try to point these out for you and suggest ways to work around them. You may want to dig a little deeper to find a solution. In any case, you'll have a much greater appreciation for the programming experts who solve these problems for us.

You'll find this manual to be a whole different breed of cat from the straitlaced reference manuals of old. Sure, you'll find all the necessary details organized in easy-to-use charts, tables, and appendixes for ready reference. But this manual does more than just tell you the correct codes to use; it shows you how to use them in carefully selected sample programs. These examples not only teach you how to operate the printer with your computer, but the hands-on style of presentation will encourage you to develop your own programs. Experiment with these programs at your own pace, and you'll soon be on your way to your own FX-80 Adventure, this one with a happy ending! For a preview of what is to come, feast your eyes on the following potpourri of print modes, user-defined characters, and graphics.

Enjoy your new printer, and by all means,

HAPPY PRINTING!

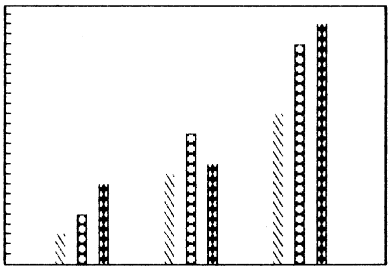
PRINT MODES

	NORMAL PRINT	SUPER SCRIPT	SUB SCRIPT	ITALIC	UNDER LINE	SUPER SCRIPT ITALIC	SUPER SCRIPT UNDER LINE	SUB SCRIPT ITALIC	SUB SCRIPT UNDER LINE	ITALIC UNDER LINE	SUPER SCRIPT ITALIC UNDER LINE	SUB SCRIPT ITALIC UNDER LINE
SINGLE-STRIKE PICA	ABCD	.....	.....	ABCD	ABCD	.....	.....	.....	.....	ABCD	.....	.....
SINGLE-STRIKE ELITE	ABCDE	.....	.....	ABCDE	ABCDE	.....	.....	.....	.....	ABCDE	.....	.....
SINGLE-STRIKE COMPRESSED	ABCDEFG	.....	.....	ABCDEFG	ABCDEFG	.....	.....	.....	.....	ABCDEFG	.....	.....
SINGLE-STRIKE EMP PICA	ABCD	.....	.....	ABCD	ABCD	.....	.....	.....	.....	ABCD	.....	.....
DOUBLE-STRIKE PICA	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD
DOUBLE-STRIKE ELITE	ABCDE	ABCDE	ABCDE	ABCDE	ABCDE	ABCDE	ABCDE	ABCDE	ABCDE	ABCDE	ABCDE	ABCDE
DOUBLE-STRIKE COMPRESSED	ABCDEFG	ABCDEFG	ABCDEFG	ABCDEFG	ABCDEFG	ABCDEFG	ABCDEFG	ABCDEFG	ABCDEFG	ABCDEFG	ABCDEFG	ABCDEFG
DOUBLE-STRIKE EMPHASIZED PICA	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD
SINGLE-STRIKE EXPANDED PICA	AB	.....	.....	AB	AB	.....	.....	.....	.....	AB	.....	.....
SINGLE-STRIKE EXPANDED ELITE	AB	.....	.....	AB	AB	.....	.....	.....	.....	AB	.....	.....
SINGLE-STRIKE EXPANDED COMPRESSED	ABC	.....	.....	ABC	ABC	.....	.....	.....	.....	ABC	.....	.....
SINGLE-STRIKE EMPHASIZED EXPANDED PICA	AB	.....	.....	AB	AB	.....	.....	.....	.....	AB	.....	.....
DOUBLE-STRIKE EXPANDED PICA	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB
DOUBLE-STRIKE EXPANDED ELITE	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB
DOUBLE-STRIKE EXPANDED COMPRESSED	ABC	ABC	ABC	ABC	ABC	ABC	ABC	ABC	ABC	ABC	ABC	ABC
DOUBLE-STRIKE EMPHASIZED EXPANDED PICA	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB	AB

DEFINED CHARACTERS

STRATA SOFTWARE

EPSON AMERICA INC



TEXT

PICA

SINGLE-STRIKE 1234567890ABCDEFGHIJKLMN / DOUBLE-STRIKE 1234567890ABCDEFGHIJKLMN
SINGLE-STRIKE EMPHASIZED 1234567890ABC / DOUBLE-STRIKE EMPHASIZED 1234567890ABC
SS EXPANDED 1234567 / **DS EXPANDED 1234567**
SS EXP EMPHASIZED 1 / DS EXP EMPHASIZED 1
UNDERLINE 1234567 / SUBSCRIPT 1234567 / SUPERSCRIPT 1234567 / *ITALICS 1234567*

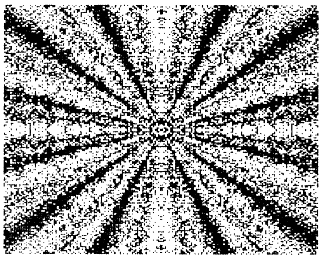
ELITE

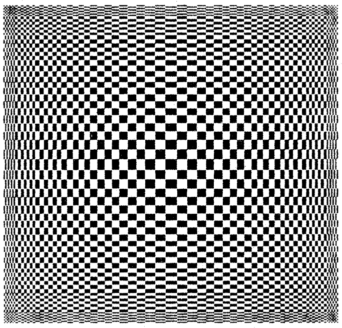
SINGLE-STRIKE 1234567890ABCDEFGHIJKLMNOPQRSTUV / **DOUBLE-STRIKE 1234567890ABCDEFGHIJKLMNOPQRSTUV**
SS EXPANDED 1234567890A / **DS EXPANDED 1234567890A**
UNDERLINE 1234567890A / SUBSCRIPT 1234567890A / SUPERSCRIPT 1234567890A / *ITALICS 1234567890A*

COMPRESSED

SINGLE-STRIKE 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnop / **DOUBLE-STRIKE 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnop**
SS EXPANDED 1234567890ABCDEFGHIJK / **DS EXPANDED 1234567890ABCDEFGHIJK**
UNDERLINE 1234567890ABCDEFGHIJK / SUBSCRIPT 1234567890ABCDEFGHIJK / SUPERSCRIPT 1234567890ABCDEFGHIJK / *ITALICS 1234567890ABCDEFGHIJK*

GRAPHICS





One Easy Lesson!!!

or . . . So who ever looks at the manual anyway?

The title of this section is a bit misleading; it takes more than "One Easy Lesson" to learn the full value of the feature-packed FX-80 printer. In fact, the more time you spend with this manual, the more your printer will cooperate with your every command. But there are always a few experts who refuse to read manuals and want to see something from their new printer RIGHT NOW—at any risk. The next few pages are especially for you experts. So go ahead, tear open the box and set that hot little item up as best you can. (You did that already? Then fasten your safety belts and prepare for a crash landing.)

If you get stuck, the proper set-up procedures are covered in Chapter 1. Those of you who've got that rare ability to wait and read the directions first should turn there now. The rest of you, in the rush of excitement that follows taking your printer out of the box, should at least follow these few simple steps to avoid getting into TOO much trouble.

First Steps

1. Make all connections with the power OFF! Connect the FX-80 printer to your computer via the printer cable (purchased separately). Some computers require a special printer interface kit (also purchased separately).

2. With the paper release lever set toward the front of the printer, insert 9½-inch fanfold paper and pull it through the paper path with the help of the paper advance knob on the right side of the printer. You may have to adjust the pin feed so that the pins line up with the holes in the paper.

Roll paper or single sheets will also work fine. Just move the pin feed out of the way, push the release lever toward the rear of the printer, and use the manual feed knob on the right side of the printer to feed the paper.

With continuous-feed paper, use the plastic paper separator to make sure that the incoming paper is separate from the outgoing paper. Check that it does not double back through and jam the feed mechanism. Remove all obstacles from the paper path so that the paper feeds freely.

If you're using single sheets, the "paper out" beeper will sound unless you shut it off via the internal switches. The switches are located under the upper right vent. (See Appendix K.)

3. Turn the printer and computer on, and load a BASIC program. Then send a listing to the printer (using LLIST, PR#1, or whatever your computer's LLIST command is). You should get a single-spaced listing. If the printout is double-spaced or printed without line spacing, change internal switch 2-4. Note that switches must be set with the power off.

Ticket Program

If you've gotten this far without turning to Chapter 1 for directions, then you are probably anxious to see what your printer can do. Type in the demo program for a ticket which is shown as Figure Easy-1. This program will unleash the wild beast in your FX-80. (If you lost us somewhere, don't worry, you haven't missed a thing. Go on ahead to Chapter 1 and we'll catch up with you there.) Now RUN it to see if you get the result shown in Figure Easy-2.

```
1Ø N=29: E$=CHR$(27): H$=CHR$(9)
2Ø LPRINT E$"1";E$"D"CHR$(26)CHR$(1);
3Ø LPRINT E$":"CHR$(Ø)CHR$(Ø)CHR$(Ø);
4Ø LPRINT E$"%"CHR$(1)CHR$(Ø);
5Ø LPRINT E$"&"CHR$(Ø)"Ø@";
6Ø FOR Y=1 TO 17: LPRINT CHR$(11);
7Ø FOR X=1 TO 11: READ D: LPRINT CHR$(D);: NEXT X
8Ø NEXT Y: LPRINT E$"U1";
9Ø FOR X=1 TO N: LPRINT CHR$(95);:NEXT X: LPRINT
   E$"A"CHR$(6)
1ØØ LPRINT "<";H$;"  >";E$"1"
11Ø LPRINT "< ";: FOR X=1 TO 25: LPRINT "Ø";:NEXT X:
    LPRINT H$" >"
12Ø LPRINT "< Ø"H$"Ø >"
13Ø LPRINT "< Ø";E$"!X";E$"4";"  ";E$"-";
```

Figure Easy-1. Program for the FX-80 ticket.

```
140 LPRINT "TICKET TO SUCCESS!";E$"!@";E$"5";E$"-0";
150 LPRINT H$;"0 >": FOR X=1 TO 2:LPRINT "< 0"H$"0 >":
    NEXT X
160 LPRINT "< 0";: LPRINT CHR$(14);E$"E";" 1 3   5 7";
170 LPRINT CHR$(20)"?@";E$"F";H$;"0 >"
180 LPRINT "9 0";: LPRINT CHR$(14);E$"E";" 2 4 - 6 8";
190 LPRINT CHR$(20);E$"F";H$;"0 :"
200 LPRINT "; 0";H$;"0 =": LPRINT "< 0";H$;"0 >"
210 LPRINT "< 0 PRODUCED "CHR$(15)"BY "E$"!X";"EPSON";
220 LPRINT E$"S0";E$"!@";"(TM)"E$"T";H$;"0 >"
230 LPRINT "< 0";H$;"0 >": LPRINT "< 0"TAB(7)E$"!Q";
240 LPRINT "GENERAL ADMISSION";E$"!@";H$;"0 >":LPRINT
    "< 0"H$"0 >"
250 LPRING "< ";: FOR X=1 TO 25: LPRINT "0";:NEXT X:
    LPRINT H$;" >"
260 LPRINT "< ";H$;" >";E$"A"CHR$(1)
270 FOR X=1 TO N: LPRINT CHR$(95);: NEXT X: LPRINT
280 LPRINT E$"@"
290 DATA 73,0,20,34,0,73,0,34,20,0,73        :'0
300 DATA 64,0,127,0,127,0,64,0,65,0,112      :'1  F
310 DATA 1,0,127,0,127,0,65,0,96,0,0         :'2  F
320 DATA 64,48,72,54,73,6,1,4,72,48,64       :'3  X
330 DATA 1,6,9,48,64,48,73,54,9,6,1          :'4  X
340 DATA 28,34,28,99,0,65,0,99,28,34,28      :'5  8
350 DATA 60,66,60,67,0,1,0,67,60,66,60       :'6  8
360 DATA 31,32,31,96,0,64,0,96,31,32,31      :'7  0
370 DATA 124,2,124,3,0,1,0,3,124,2,124       :'8  0
380 '*** TICKET BORDERS ***
390 DATA 64,0,64,0,64,0,32,0,16,8,7           :'9
400 DATA 7,8,16,0,32,0,64,0,64,0,64          :':
410 DATA 1,0,1,0,1,0,2,0,4,8,112 :';
420 DATA 127,0,0,0,0,0,0,0,0,0,0 :'<
430 DATA 112,8,4,0,2,0,1,0,1,0,1 :'=
440 DATA 0,0,0,0,0,0,0,0,0,0,127 :'>
450 '*** REGISTERED TRADE MARK ***
460 DATA 0,0,0,0,28,34,0,65,62,65,40         :'?
470 DATA 65,44,83,0,34,28,0,0,0,0,0          :'@
```

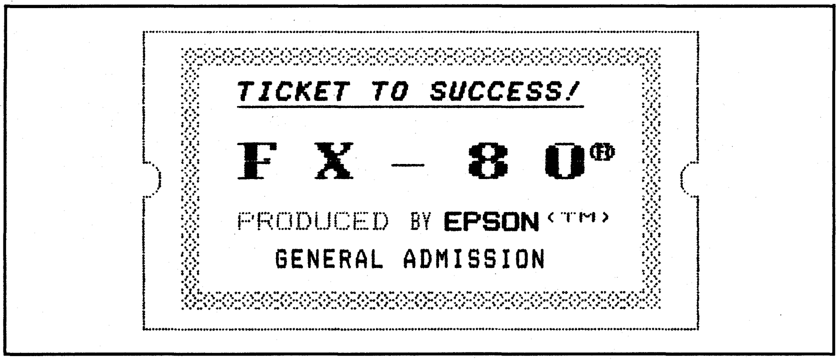


Figure Easy-2. Ticket to success.

Ticket Program Description

We do not attempt a complete explanation of the program here. That's what the rest of the manual is all about. But we do provide a cursory, line-by-line description for those of you who wish to analyze the program.

10 This line stores values in variables for easy access.
20 ⟨ESC⟩"1" sets the line spacing to 7/72 inch. The ⟨ESC⟩"D" sequence sets a horizontal tab stop at column 26.
30 The ⟨ESC⟩":" string copies the entire ROM character set into RAM. See Chapter 17.
40 Designates RAM as the source for the active character set.
50 Prepares the printer to redefine characters 0 through @.
60 Sets a counter for the 17 letters being defined and prints the "attribute byte" of each new character.
70 Reads the data that defines the letters (17 sets of 11).
80 Turns on Unidirectional Mode.
90 Prints the top of the ticket and sets the line feed to 6/72 inch.
100 Prints the newly defined symbol "⟨" (ticket border), tabs to the next stop, prints the other border (⟩), and sets the line spacing back to 7/72 inch.
110 After printing the outside border, this line prints the top of the inside border (which was defined as the zero character).
120 Gives another line of borders.
130 This line prints more borders, then sets the Master Select Mode type style 24 or X. (That's Emphasized, Double-Strike, and Pica combined in one code!) The ESCape "4" turns Italic ON and ⟨ESC⟩"-1" turns ON Underline Mode.

140 Prints TICKET TO SUCCESS! in Master Select 24, Italic, and Underline; then it turns all those modes OFF.
150 Produces two more border lines.
160 Prints the upper half of the "FX-80" letters in CHR$(14) (Expanded) and ⟨ESC⟩"E" (Emphasized) print. Those are more user-defined characters.
170 Turns OFF Expanded Mode and prints the trademark symbol (user-defined as ? and @). Turns OFF Emphasized.
180 Prints the bottom half of the "FX-80" letters.
190 Turns OFF the codes and prints a border.
200 Prints borders.
210 Prints PRODUCED, then switches to Compressed Mode and prints BY. ⟨ESC⟩"!X" sets the Master Select Mode 24 (this time without italic and underline) and prints EPSON. The Master Select Mode automatically turns OFF Compressed.
220 Turns OFF the Master Select 24 and sets Superscript Mode (ESCape "S0"). It prints (TM) and cancels Scripts.
230 Prints another line of borders, then sets the Master Select Mode 17 (Q) (that's Double-Strike and Elite combined).
240 Prints GENERAL ADMISSION and sets everything back to the default printing modes.
250 Prints the bottom of the inside border.
260 Sets line spacing to 1/72 inch.
270 Prints the bottom of the outside border.
280 Returns the printer to the default mode.
290-370 This is the data for the inside border (0) and the "FX-80" letters as user-defined characters 1-8.
390-470 This is the data for the ticket borders and the trade-mark symbol. They are stored as user-defined characters.

Chapter 1
Looking It Over

Once you've removed the packing materials from the FX-80 printer, *the first thing you should do* is make sure you have all of the parts. With the purchase of your FX-80 printer, you should receive the items shown in Figure 1-1:

1) The printer itself.
2) One ribbon cartridge (in box).
3) A smoke-colored plastic paper guide.
4) Two plastic dust covers (1 white, 1 brown).
5) This *FX-80 User's Manual.*

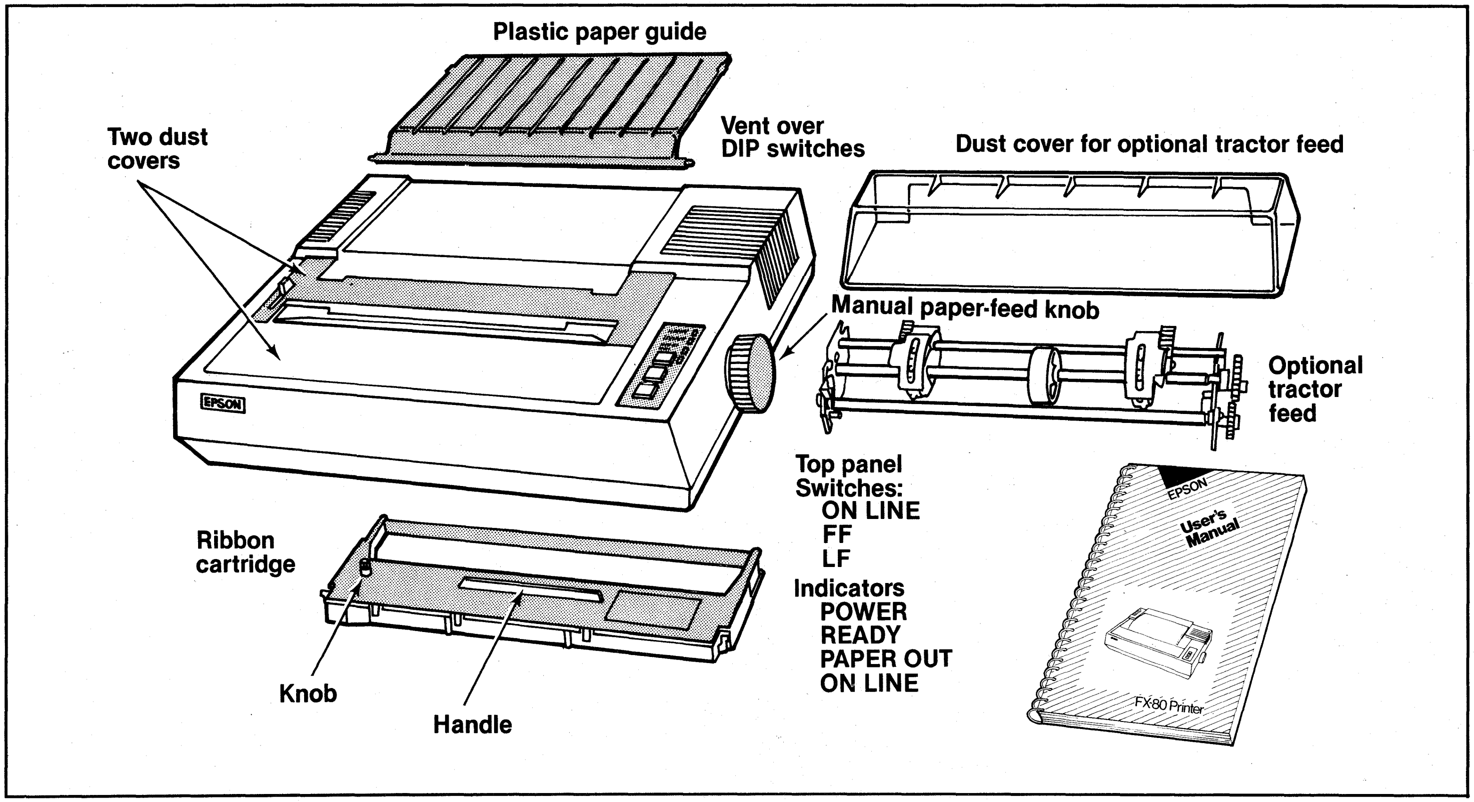


Figure 1-1. Unpacking the FX-80.

Supplies and Accessories

The following items may be purchased separately from your Epson dealer:

Printer cable or interface kit. Each computer system has its own way of connecting to a printer. Some need a cable only, others require a cable and circuit board. The FX-80 printer uses the Centronics™ standard parallel interface scheme. If your system requires a serial printer, you must purchase a serial board for the printer. Your Epson dealer can supply you with a variety of interface boards and cables.

Printer paper. The FX-80 is designed to accommodate several types and sizes of paper. Standard printer paper is 9½ by 11-inch fanfold paper (called continuous-feed) with ½-inch tear-offs on each side for the pin-feed holes; when the printout is torn down to size, or "burst," a standard 8½ by 11-inch page is left. The pin-feed mechanism handles this type of paper. Single sheets of paper (without holes) are moved through the printer by the friction feed mechanism. And continuous-feed papers whose widths range between 4 and 8½ inches (such as mailing labels) are handled with an optional tractor-feed unit which you may purchase from your Epson dealer.

You'll need a stack of paper for the sample programs in this manual. Standard continuous-feed paper is recommended for these application programs.

Tractor feeder (optional). The tractor-feed mechanism is necessary only if you intend to print forms 8½ or less inches wide—such as pin-feed mailing labels or fanfold paper without removable perforations.

Roll paper holder (optional). This option holds continuous-feed paper that is fed by the friction feeder.

Ribbon cartridge replacement. The expected life of a cartridge is three million characters—(roughly 1,000 pages of text.)

Print head replacement. The expected life of a print head is one hundred million characters.

Find Your FX-80 a Safe Home!

Naturally, your printer must be somewhere near the computer (the length of the cable is the limiting factor), but there are other considerations in finding a choice location for your computer/printer operation. For instance, you may want to find an electrical outlet that is not controlled by a switch—since a switch may be accidentally shut off while you have valuable information stored in memory. A quiet spot is always nice, one that's away from other appliances. And your printer needs enough room for the paper to flow freely.

Note: The printer should be turned OFF during all set-up operations.

First Things First

Once you've found a good home for your FX-80, you'll want to get acquainted with it. The first parts that you need to recognize are the pair of covers, the ribbon assembly, and the paper guide.

The covers

The two covers on the FX-80 printer protect it from dust or foreign objects that might damage the printer mechanism. You'll need to remove these lids to gain access to the inside, but they should be replaced when you're through (Figure 1-2). The rear cover comes off with a slight tug upward; it snaps back into place. Move the front lid to its full vertical position, then remove it by lifting straight up (you may need to remove packing tape).

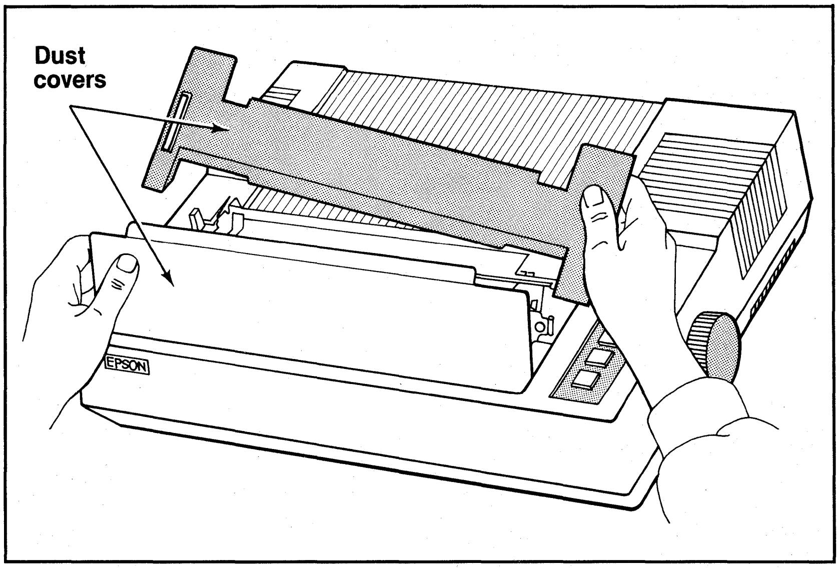


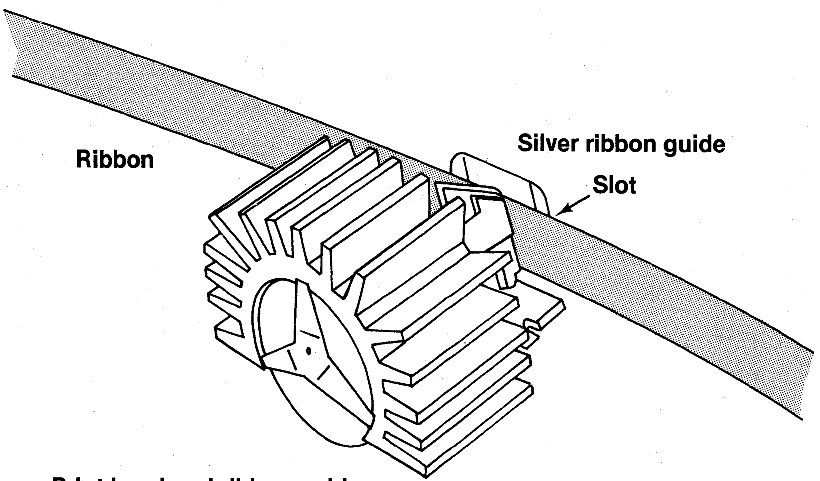
Figure 1-2. Uncovering the FX-80.

The ribbon

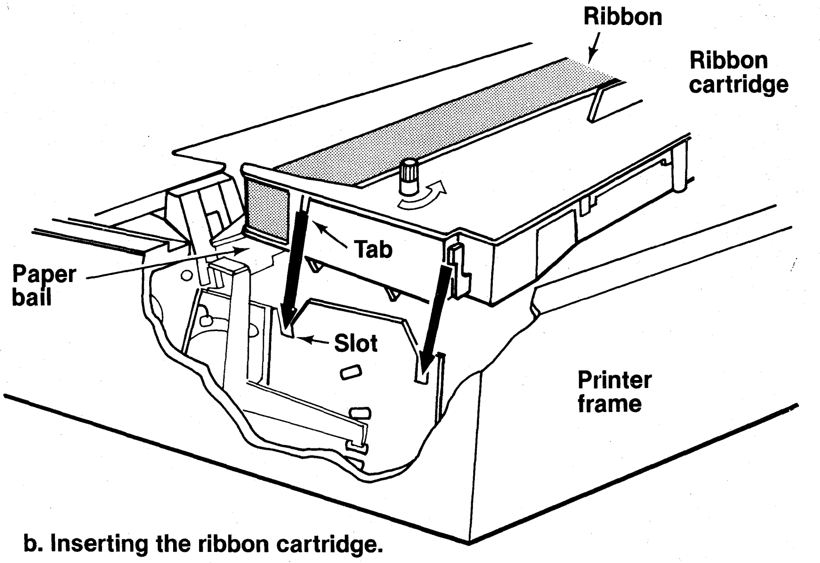
Remove the ribbon cartridge from its packing materials. Holding the cartridge by the plastic fin on the top, lift the paper bail and insert the cartridge. The ribbon should rest directly between the silver ribbon guide and the black print head (Figure 1-3). Note that the paper bail has three positions: resting on the paper, raised straight up, and raised forward (partially up and pulled toward the front of the printer).

The two pairs of tabs at the ends of the ribbon cartridge snap neatly into slots in the printer's frame. The cartridge should fit snugly into place. To remove any slack in the ribbon, turn the ribbon knob in the direction of the arrow.

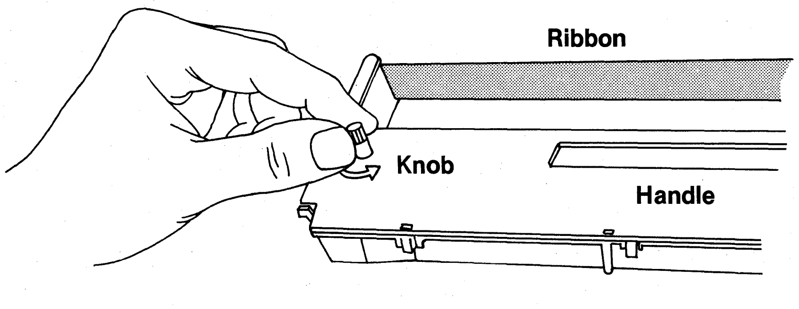
Note: When you replace a ribbon, remember that the print head may be hot from usage; allow it time to cool.



a. Print head and ribbon guide.



b. Inserting the ribbon cartridge.



c. Adjusting the ribbon.

Figure 1-3. The ribbon.

The paper guide

Next, insert the plastic paper guide into the appropriate notches as shown in Figure 1-4.

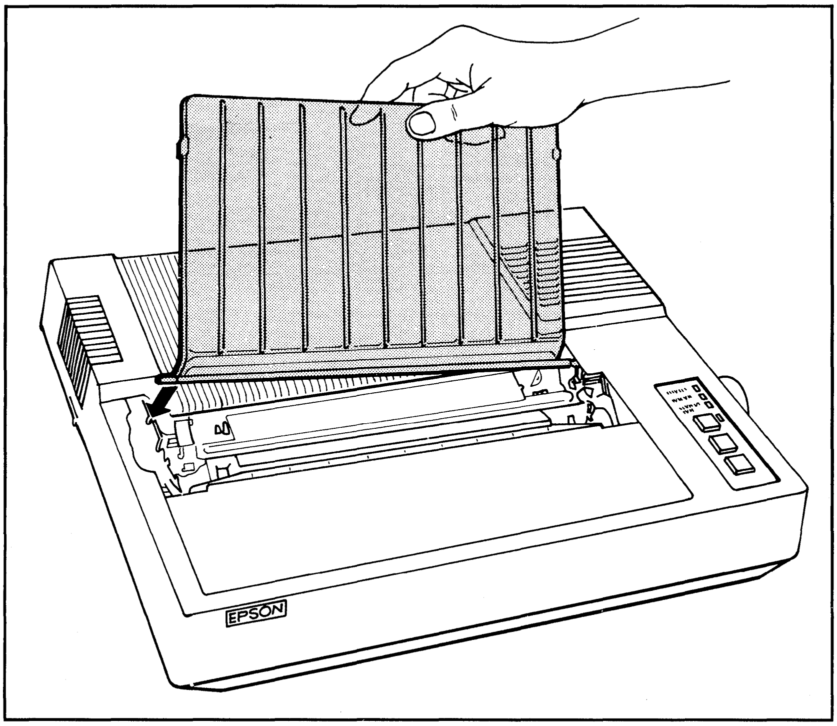


Figure 1-4. Inserting the paper guide.

Paper Feeders

Now it's time to learn how to feed the paper on the FX-80—and for that, one of the following three sections will help.

The pin feeders

The pin-feed mechanism on the FX-80 is a built-in tractor unit. The pin feeder accepts the most common type of fanfold paper, 9½ by 11-inch paper with removable tractor holes. Proper adjustment of the pin feeders is important for smooth paper flow, and the release latches (Figure 1-5) allow you to make slight adjustments in pin feeding. Note that for standard adjustments the arrows on the levers can be lined up with the notches in the silver cover. Set both arrows to 9.5 position.

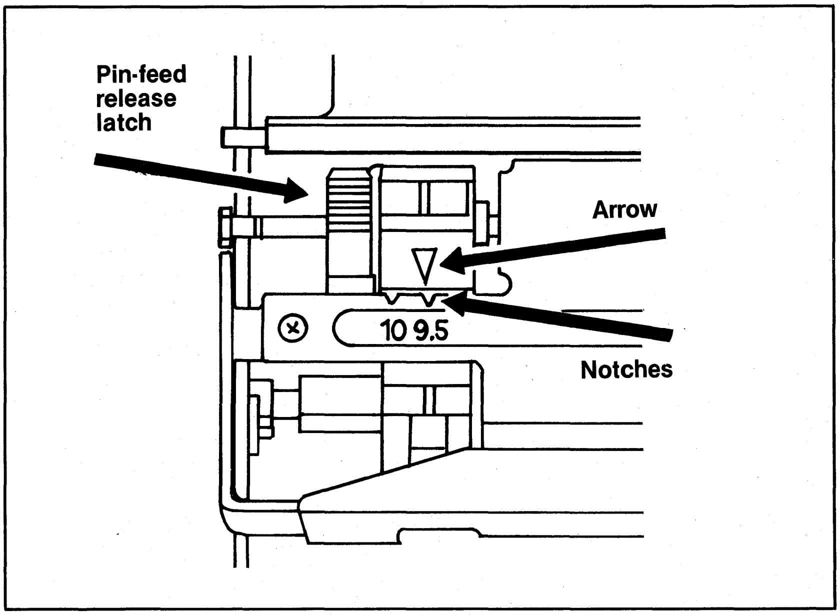


Figure 1-5. Adjusting the pin feeders.

After adjusting the pin feeders, pull the paper-release lever toward the front of the printer (Figure 1-6) and (always with the printer turned OFF) push the printhead to a center position. These two actions are important, since they allow the paper to feed through easily. This puts the pin feeders in charge of getting the paper through.

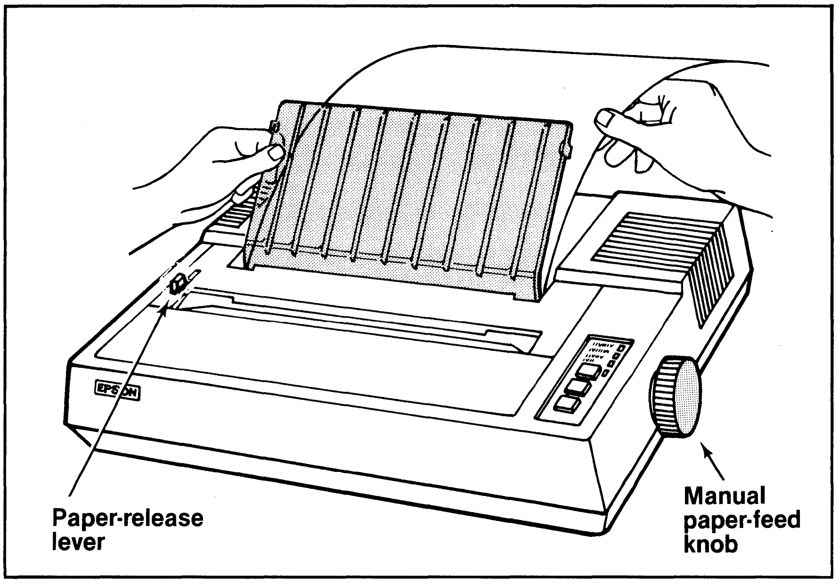


Figure 1-6. Loading the paper.

Set a stack of paper on a flat surface directly behind or under the printer. If the paper is kinked or sitting off to one side, it may not feed through properly. With the paper bail up, run the top sheet under the plastic guide and under the paper roller. Then use the manual-feed knob to run the paper through the rest of the way. The paper should flow over the plastic guide to form a second stack as shown in Figure 1-7.

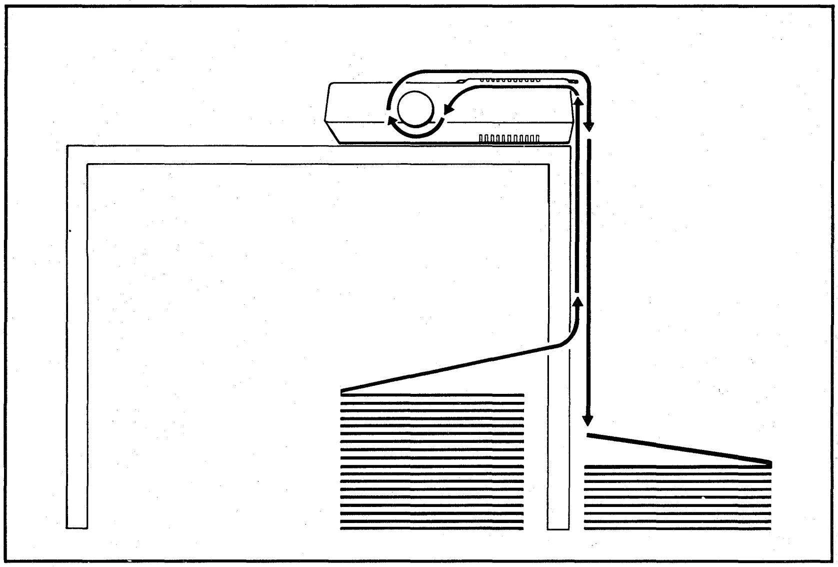


Figure 1-7. The paper path.

The technicians assure us that turning the manual-feed knob with the printer *on* does not damage the gears, but it's easier to turn with the printer off.

Don't get discouraged—it takes a little practice to get the paper loaded right. The pin feeders must be set to just the right width or the paper will jam. Feed the paper through, then, holding the paper firmly, turn the roller knob until the paper holes line up with the pins. If desperation sets in, try folding the first sheet over the second and feeding them through together (thicker paper feeds a little better). With a little practice, you will feed the paper through smoothly every time.

The optional tractor feeder

If you plan to use either 8½-inch paper with permanent pin-feed holes or small labels with holes, you'll need to purchase a tractor-feed unit to accommodate the narrower paper.

To install such a unit, imitate the movement suggested in Figure 1-8. Insert hook A into stud A, and rotate the unit toward the front of the printer (pressing down firmly) until it locks into place. Make sure that the gears are on the right side of the printer (the side with the buttons). You don't need to work any levers to insert the unit, but you may need to adjust the pin feeders if they are in the way of the tractor unit.

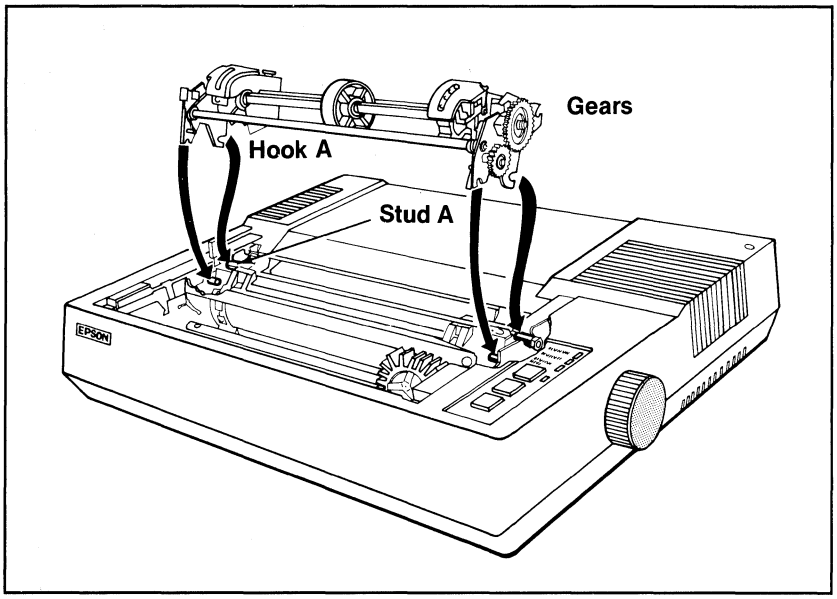


Figure 1-8. Installing the optional tractor feeder.

To adjust the width of the tractor feeders, simply release the tractor's locking levers on each side so that the units move freely (see Figure 1-9). Now adjust the units to match the size of your paper or labels. Lock the units back into place. Pull the paper-release lever toward the front of the printer. It's that easy!

To remove the tractor unit, use the pair of tractor-release levers (one of these is shown in close-up in Figure 1-10). Press both levers forward and rock the unit up and back toward the rear of the printer. The entire tractor unit should come off easily—you should not have to pull or tug on the unit to remove it.

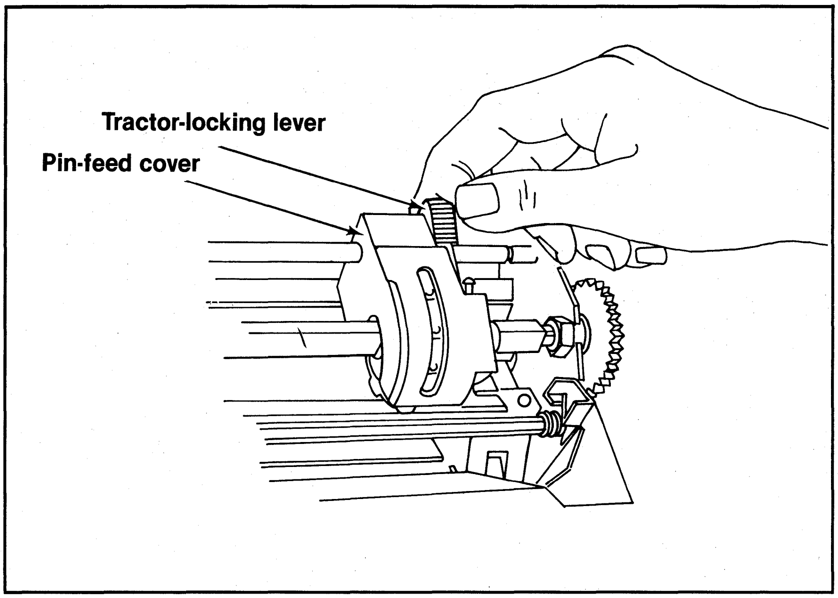


Figure 1-9. Adjusting the tractor width.

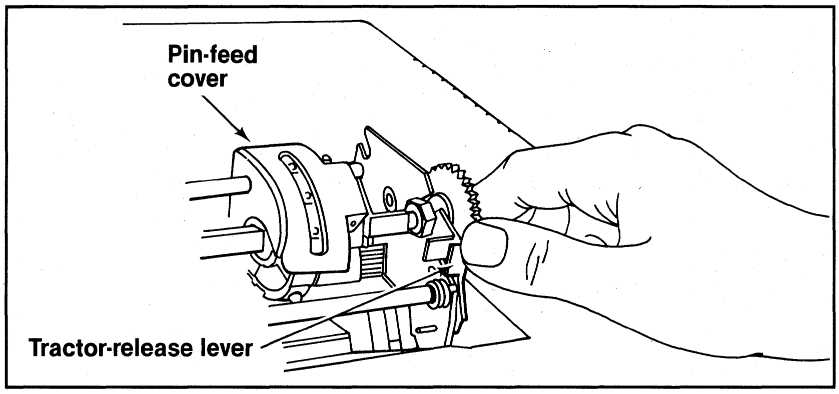


Figure 1-10. Releasing the tractor feeder.

The friction feeder

If you are using single sheets or roll paper, you'll need to remove the tractor unit as shown above and use the friction-feed mechanism.

Simply adjust the pin-feed units so that they are out of the way (as far apart as possible). Pull the paper bail toward the front of the printer, and then push the paper-release lever toward the rear of the printer to engage the friction feed. Use the manual-feed knob to feed the paper through (as you would feed paper through a typewriter). And, finally, lower the paper bail onto the paper to hold it in place.

Parts with Settings

The FX-80 can be adjusted to meet your hardware and software needs. Hardware adjustments that you'll want to know about first involve the paper thickness lever and a set of internal switches; the next two sections fill you in on adjusting these settings.

The paper thickness lever

The paper thickness lever shown in Figure 1-11 adjusts the print head to accommodate various paper thicknesses.

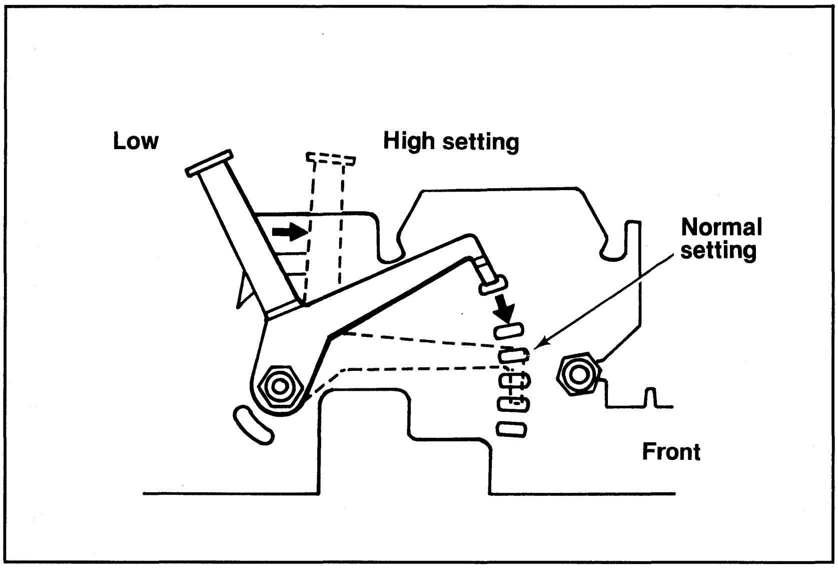


Figure 1-11a. Setting the paper thickness lever (side view).

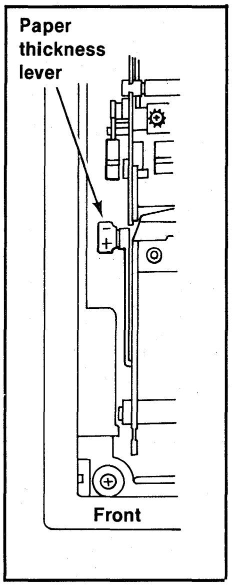


Figure 1-11b. (top view)

The factory sets it for normal paper (starting from the back, that means to the third notch—which gives about 1/500th of an inch), but you can adjust it for printing one original and up to two copies. For thicker paper, move this lever toward the front of the printer. To compensate for faint print, move it toward the rear.

Function switches

We're almost ready to start printing! Just a few more important steps . . . like setting the printer's internal switches. These are tiny switches (inside the printer, of course) that control a number of printer functions (such as line-feed adjustment, the paper-out buzzer, print modes, etc.). In Appendix L, "Customizing the FX-80," we'll show you how to play with the different variations, but for now there are just a few things we should say about them.

The switches are set at the factory and don't need to be touched except for special uses and to solve problems with computer/printer interfacing. You may need to answer one or two interface questions right now. If you are using single-sheet paper, or if your computer doesn't send line feeds to the printer, read on.

Note: Always turn the power *off* (with the switch on the left side of the printer) before touching any internal switch!

The design of the FX-80 printer makes it easy to access the internal switches. They are located under the upper-right vent (Figure 1-12a). To remove the vent, you need a Phillip's-head screwdriver. Once the top screw is removed, take the vent off by pressing down on the top and pulling *sideways.* Insert a coin at the top of the vent and twist to help the vent come off easily (Figure 1-12b).

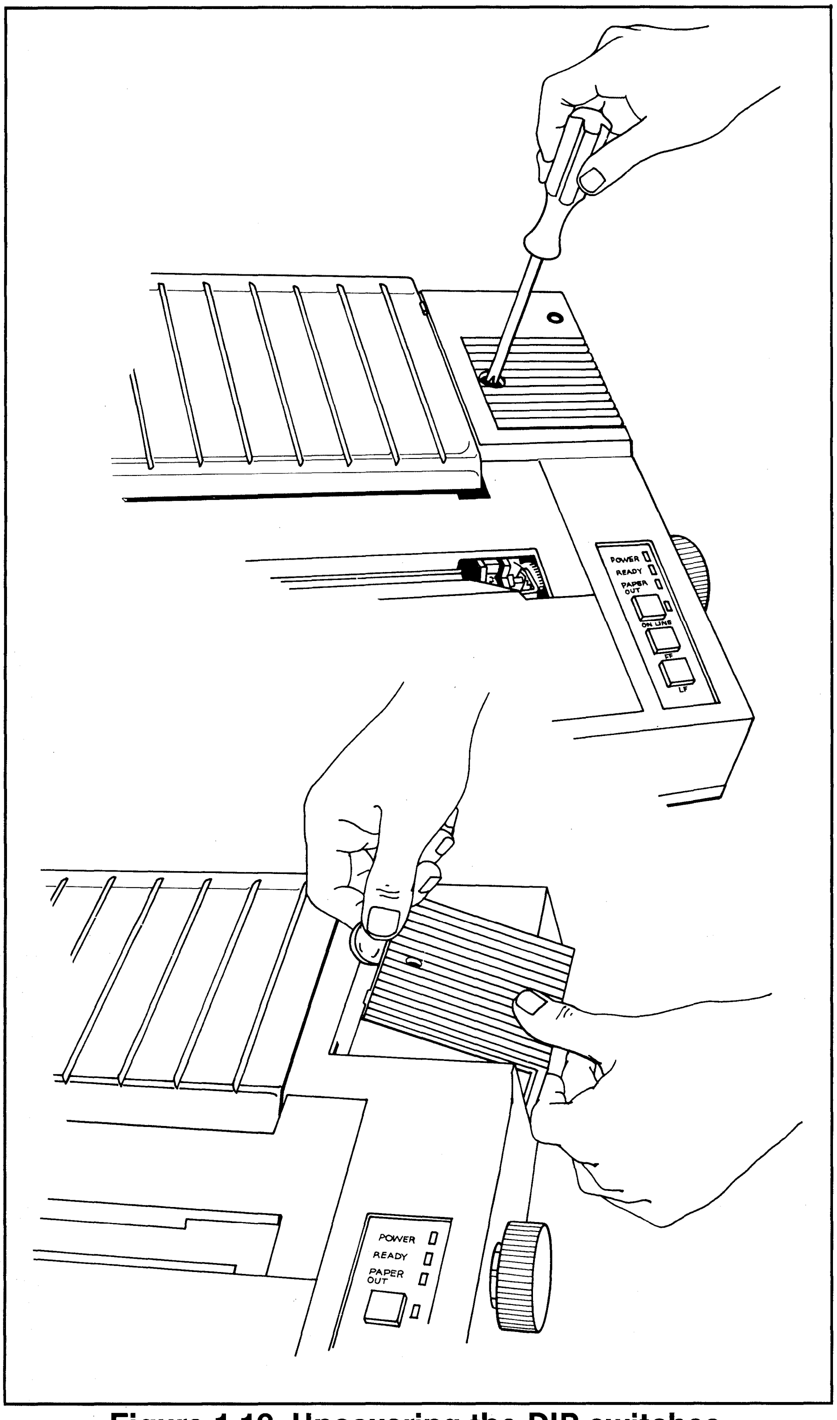


Figure 1-12. Uncovering the DIP switches.

Do not replace the screw because in the course of this manual, we will sometimes suggest that you reset switches. (Keep the screw in a safe spot so that you can replace it if you ever pack up the printer for shipping.)

Table 1-1 shows the factory settings of the internal DIP switches, and Figure 1-13 illustrates them.

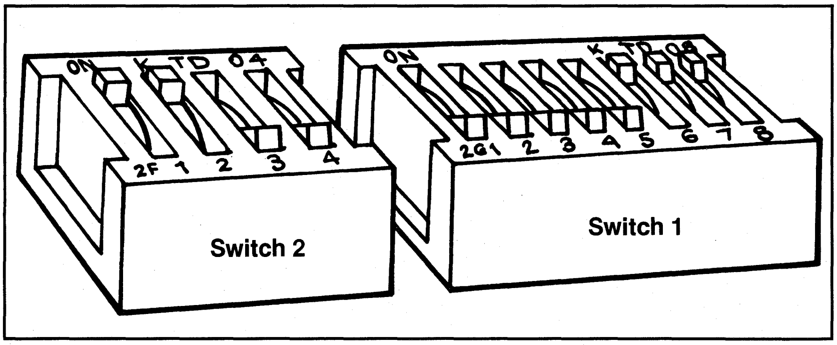


Figure 1-13. Factory setting of the DIP switches.

Table 1-1. The DIP Switches.

Switch 1

1-8	[✓ ON]	International character	OFF
1-7	[✓ ON]	International character	OFF
1-6	[✓ ON]	International character	OFF
1-5	Emphasized	Print mode	[✓ Normal]
1-4	2K buffer	RAM memory	[✓ User-defined Characters]
1-3	Inactive	Paper-out sensor	[✓ Active]
1-2	Ø	Zero font	[✓ 0]
1-1	ON	Compressed Mode	[✓ OFF]

Switch 2

2-4	CR + LF	Automatic line feed	[✓ CR only]
2-3	ON	Skip-over-perforation	[✓ OFF]
2-2	[✓ ON]	Bell	OFF
2-1	[✓ ON]	Printer select	OFF

Note: The shaded boxes show the factory settings.

If you are using single sheet-paper, switch 1-3 should be in the *on* position. (The factory sets this switch to the off position.) Setting this switch to *on* enables the printer to continue printing on the bottom third of the page even after the paper-out sensor detects that there is no more paper in the printer.

Some printer interfaces (like the Apple's) will automatically line-feed the paper as it goes through the printer. Others (like the TRS-80's) rely on the printer to do that; that is what switch 2-4 is for. In the *on* position it will automatically add a line feed to every carriage return. In the *off* position it will not. If you are not sure what your computer requires, leave switch 2-4 the way the factory set it and you can adjust it later. For more on the internal switches, see Appendix K.

(Users of the MX printers will notice that the functions of switches 2-3 and 2-4 are reversed on the FX-80.)

Starting Up

We are finally ready to connect the printer to the computer (remember that some computers need interface kits). First make sure the power switch is off. Connect the printer end of the cable to the connector at the left rear of the printer (as shown in Figure 1-14), where the connector is conveniently located out of the paper path. The other end of the cable plugs into your computer. If your cable includes grounding wires (as does the cable for the Epson QX-10), be sure to fasten the wires to their grounding screws.

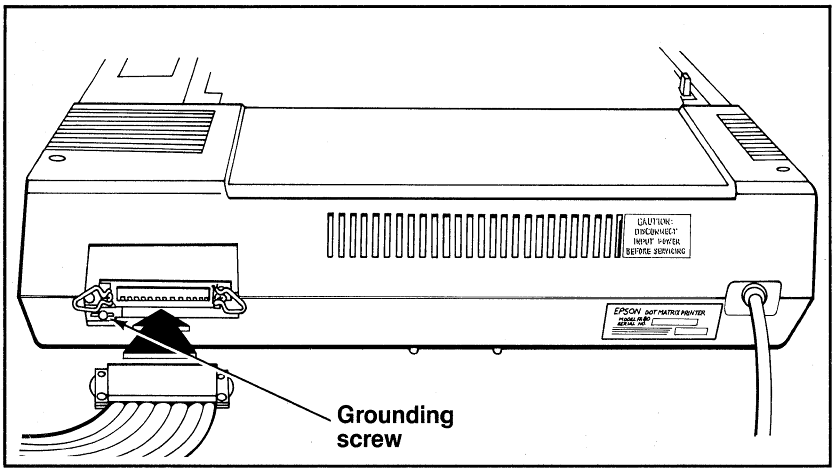


Figure 1-14. Connecting the cable.

Now, turn it on!

You get a little dance from the print head and three lights go on: the POWER light, the READY light, and the ON LINE light. If the ON LINE and READY lights are not on, push the button marked "ON LINE." If the PAPER OUT light is on, then you will have to reload the paper.

Top panel buttons and lights

When the ON LINE light is on, the printer and computer are in direct communication. The FF (form feed) and LF (line feed) buttons have no effect when the printer is on line. Go ahead, try pushing one.

To use the FF and LF buttons, press the ON LINE button to turn its light off. Now you can see what the other buttons do.

Press the LF button once. Okay, that gives one line feed. Now hold the button down for a moment. Line by line advancement!! (We're getting more and more advanced all the time.)

The FF button will advance the paper a complete page (form). If you hold the button down, it will advance several forms.

The "top of form" (or TOF) is set at the paper's current position when the printer is turned on. When you press the FF button, the paper advances to the same relative position on the next page/form. Ideally, you'll line up a form's top perforation with the top of the ribbon before you turn on the printer (see Figure 1-15). That way, your printing can start on the first line of the paper.

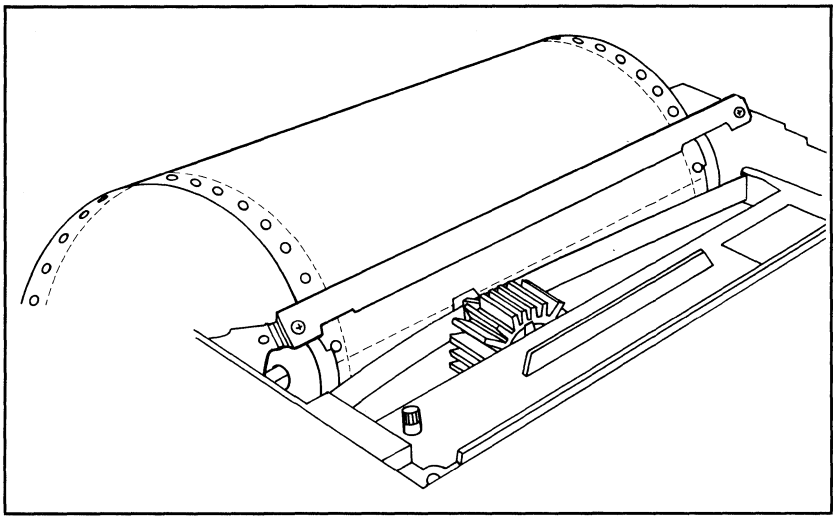


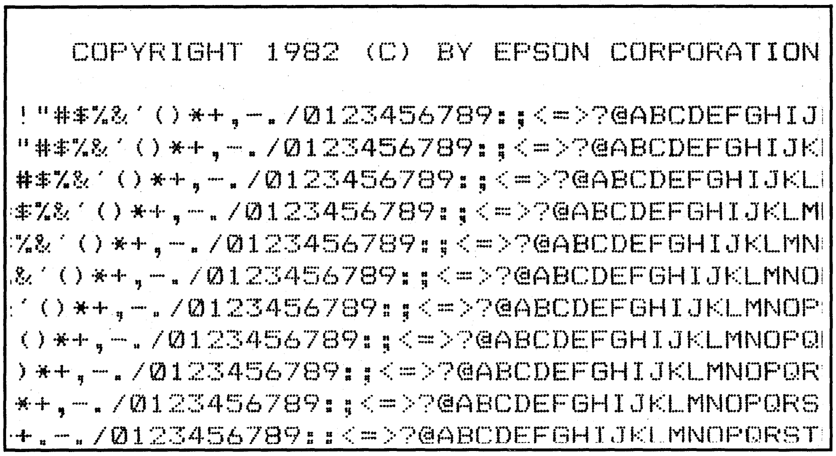
Figure 1-15. Setting the Top Of Form.

If you turn the printer on while the printhead is sitting in the middle of a form, that is precisely where the next form will start when you FORM FEED.

Remember: The ON LINE button has to be in the *off* position (Off Line) for the FF and LF to work.

The FX-80 tests itself

Now it's time to see how your new treasure operates. Turn the printer completely off (with the switch on the left side of the printer), press down the LF button, and turn the printer back on again while still holding the LF button.



```
   COPYRIGHT 1982 (C) BY EPSON CORPORATION

! "#$%&'()*+,-./Ø123456789:;<=>?@ABCDEFGHIJ
"#$%&'()*+,-./Ø123456789:;<=>?@ABCDEFGHIJK
#$%&'()*+,-./Ø123456789:;<=>?@ABCDEFGHIJKL
$%&'()*+,-./Ø123456789:;<=>?@ABCDEFGHIJKLM
%&'()*+,-./Ø123456789:;<=>?@ABCDEFGHIJKLMN
&'()*+,-./Ø123456789:;<=>?@ABCDEFGHIJKLMNO
'()*+,-./Ø123456789:;<=>?@ABCDEFGHIJKLMNOP
()*+,-./Ø123456789:;<=>?@ABCDEFGHIJKLMNOPQ
)*+,-./Ø123456789:;<=>?@ABCDEFGHIJKLMNOPQR
*+,-./Ø123456789:;<=>?@ABCDEFGHIJKLMNOPQRS
+,-./Ø123456789:;<=>?@ABCDEFGHIJKLMNOPQRST
```

Figure 1-16. Sample self test.

Figure 1-16 shows the FX-80's self test, which prints a pattern to display the standard characters that are stored in the printer. The test continues until you turn the printer off. Notice that there are two sets of upper- and lower-case letters (roman and italic), plus lots of other characters.

Impressive, but you've only touched the tip of the iceberg. In the next chapter, you'll find out how to send your first BASIC message to your printer.

Chapter 2
Controlling the Printer from BASIC

In this manual you'll be testing the printer with BASIC programs. So the logical starting point is to learn just how the BASIC in your computer interacts with the FX-80.

One of the simplest things you can do with the FX-80 is print listings of your BASIC programs. You merely load a BASIC program into the computer and send the LISTing output to the printer instead of to the screen.

Unfortunately, different computers access the printer in different ways. For example, the TRS-80 and most other Microsoft BASICs use an L before the usual PRINT or LIST commands (e.g., LPRINT, LLIST, etc.). Some other computers use PRINT# in place of LPRINT. A third group (Apple in particular) uses PR#1 to route information to the printer and PR#0 to restore the flow of information to the screen. Most likely your computer uses one of these three methods. Be sure to consult your computer manual.

We will use the LPRINT and LLIST commands for our examples in this manual. The widespread acceptance of Microsoft BASIC makes these commands as close to a standard as we have in this industry. But remember that you may need to modify some of the programs to match the unique aspects of your system.

Once you have discovered how your computer communicates with the printer, load a BASIC program into memory. Now list it onto the printer, using your computer's version of the LLIST command. Some examples are shown in Table 2-1.

Table 2-1. LISTing on Several Computers.

Command	Computer
`LLIST`	TRS-80 and Microsoft BASIC
`LIST"COMØ:"`	HX-20

`PR#1`
`LIST`
`PR#Ø`
	Apple

If your listing is more than a page long (or if you didn't start the listing at the top of a page), you may notice that it is printed right over the perforation. Don't worry, we'll show you how to correct that in Chapter 8, when we cover forms control.

Meanwhile, printing a program LISTing is a fundamental function of the printer. Be sure you manage this before continuing (if you have trouble, consult your computer's manual for help).

BASIC Communications or . . . The Nonexistent Standard

Part of the difficulty in controlling communications between computer and printer is the lack of a standard coding scheme. When your computer sends out a numeric code for the letter A, the printer had better interpret that code as an A or we're all in trouble. For frequently used characters like the alphabet, numerals, and keyboard symbols, most manufacturers use the American Standard Code for Information Interchange (ASCII, pronounced *ask-key* and shown in full in Appendix A). Use of this coding system assures that computers and printers use the same set of numbers to represent the most commonly used characters. Unfortunately, the original ASCII standard does not allow for the advanced features in today's printers and computers. Manufacturers have therefore adjusted the code to suit their individual needs.

This means that we are faced with compatibility problems between printers and computers. These inconsistencies can usually be overcome by sending the special codes required by the printer in the form of numbers. In the BASIC language, the codes are sent via the CHR$ function.

The CHR$ function

The CHR$ function is designed to print special symbols on the video display. It converts any decimal number from 0 through 255 to a character (or action). What is printed is determined by the particular modified ASCII table that is used by your computer system.

To check what you have, try printing a few characters on your computer's *screen.* The usual format for this is PRINT CHR$(n). The 'n' represents one of the numbers from 0 to 255, each one of which accesses a unique character or action. Try typing this:

```
10 PRINT CHR$(65)
```

and RUN it. Since most computers use the numbers from 32 to 127 to mean the same set of characters, you should see a capital A on the screen.

It's the numbers less than 32 and greater than 127 that produce different results on nearly every brand of computer. Try entering:

```
10 PRINT CHR$(193)
```

and RUNning it. (What you see is what you get.) If you don't see anything on the screen, don't worry. Remember that we are mainly interested in sending that 193 to the printer, and what it prints on the screen is of little consequence.

Printing from BASIC

Well then, how do you send the 193 (or any code) to the printer? To do that, you need to find out how your system activates the printer. The most typical methods are shown in Table 2-2.

Table 2-2. Printer Commands on Several Computers.

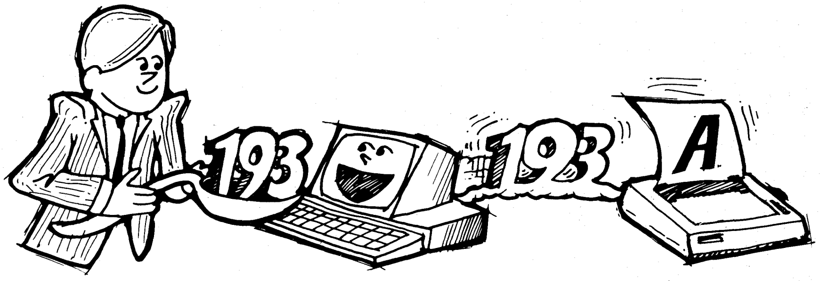
Activating Command	Computer
`1Ø LPRINT CHR$(193)`	TRS-80 and Microsoft BASIC

`5 OPEN "O",#1,"COMØ:"`
`1Ø PRINT#1, CHR$(193)`
`99 CLOSE#1`
	HX-20

`5 PR#1`
`1Ø PRINT CHR$(193)`
`99 PR#0`
	Apple

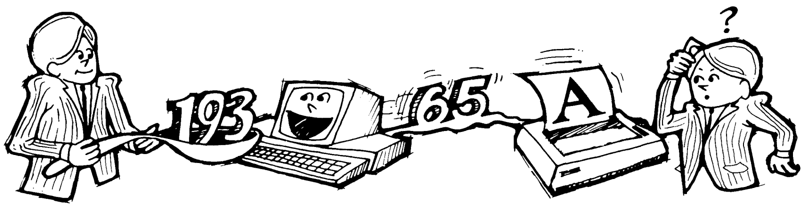
Check your computer's reference manual, and type in the statements appropriate to your computer. Then type RUN.

With any luck, you will get an italic capital A on the printer: *A*



If nothing printed, it's time to double check your computer manual and cable connections. Make sure the printer is ON LINE and the READY light is lit.

If you ended up with a roman A: A



(instead of the one in italics), pay close attention to the following.

The original ASCII code was designed to use the decimal numbers 0 through 127. Computer systems designers soon decided to extend this range (to 0 through 255) in order to make room for more features. Unfortunately, some designers did not anticipate that printers would make use of this extended range. So they designed the BASIC language (actually the printer driver) to intercept any number in the upper half of the range (128 to 255) and automatically convert it to the lower half of the range by subtracting 128. In these systems, the code CHR$(193), for instance, never makes it to the printer. It arrives as a CHR$(65): 193-128 = 65.

For many applications, you don't need the upper half of the ASCII codes. For others, the inability to generate codes greater than 127 is a severe handicap. It is good to be aware of the problem right from the beginning. Apple users and those of you who suspect the worst, see Appendix H.

If you're patting yourself on the back because your printer printed an italic A, postpone your celebration for a bit. Nearly all BASICs intercept codes on their way to the printer and alter them in some way. For example, some popular systems intercept a CHR$(10) (a line feed) and send out a CHR$(13) (carriage return) instead. Typical problem codes include 0 and the numbers 9 to 13. Your computer manual may alert you to these problems. Or experience may have to be your guide.

In order to help systems which can't send a zero, several printers' instruction sequences allow such options as using "0" (quote-zero-quote) in place of CHR$(0). We'll be mentioning some of these problems and solutions throughout this *User's Manual.* Again, Appendix H is intended to help you find and solve these inequities of working with BASIC. Appendix I will help with specific problems on the sample programs.

Control codes

Enough talking about problems. Here's a program line for everyone! Type:

```
    1Ø LPRINT CHR$(7)
```

(Be sure to use the appropriate printer access commands for your system.)

Now RUN it! You should hear a short beep. That's the FX-80's bell or buzzer. Certain codes, like CHR$(7), actually perform printer functions rather than print a specific character. Table 2-3 shows a breakdown of the ASCII codes as interpreted by the FX-80.

Table 2-3. FX-80 Interpretation of ASCII Codes.

Code	FX-80 Interpretation
0 to 31, 127	Printer control codes
32 to 126	Standard (roman) character set
128 to 159, 255	Additional printer control codes
160 to 254	Italic character set

See Appendix A for the entire ASCII chart of CHR$ codes.

This would be a good time to play with these codes on your own. And you may want to take a break before you start the next section.

ESCape!

As more features are added to the printer, even the extended range of codes (0 to 255) is inadequate if only single-code instructions can be used. Because of this, the FX-80's logic is designed to understand special sequences of codes. You use these code sequences to select one or more printing features, or "modes."

Each "ESCape code sequence" consists of an ⟨ESC⟩ code that is followed by one or more CHR$ codes. In BASIC, the ⟨ESC⟩ code itself is sent by using CHR$(27):

```
LPRINT CHR$(27);CHR$(n)
```

or

```
LPRINT CHR$(27);CHR$($n_1$);CHR$($n_2$); . . . ;CHR$($n_k$)
```

(where n_1 represents the first number in a series, n_2 the second, and n_k the conclusion).

In this manual we provide sample programs that allow you to test various features of your FX-80. Quite often we start with a few program lines, make several changes and additions, and end up with a substantial program. You may wish to save some of the programs for later use. We suggest that you periodically save the larger programs in case of power fluctuations or other interruptions.

Start a new program now by entering:

```
NEW
10 LPRINT CHR$(27)CHR$(52)
20 LPRINT "ITALIC CHARACTER SET!"
```

and RUNning it. (Whenever you can RUN a program, we show you the results that you should expect.)

```
ITALIC CHARACTER SET!
```

Note: Some systems require semicolons between all codes as in:

```
LPRINT CHR$(27);CHR$(52)
```

We include semicolons only when needed for clarity.

The FX-80 interprets the CHR$(27)CHR$(52) sequence in line 10 above as a command to switch from roman to italic characters. The LPRINT in line 20 sends a string of characters to the printer to verify that the printer is in Italic Mode.

Now type:

`LLIST` (or your system's version of LLIST)

to check the printer's status. Since all characters are still printed in italics, you can see that this mode stays on until it is turned off.

If your printer is printing one line on top of another or if it is double spacing, you need to change the setting of the FX-80's internal switch 2-4 (see "Function switches" in Chapter 1).

Resetting Commands

There are several ways you can reset a command. You can get back to square one by turning the printer off or using the Master Reset Code; you may CANcel all of the text stored in the printer's memory buffer without changing print modes; you may DELete the last text character sent to the printer; or you may use a command to turn off a specific mode. The next sections cover these various ways of telling the printer that you've changed your mind.

The Master Reset Code

You could turn *off* the Italic Mode by turning the *printer* off, then back on. Turning the printer off resets the printer to its start-up condition (Roman Mode). But cycling the printer off and on may disrupt the computer/printer communications. You will most likely want to find some other way to reset printer modes. The FX-80 has a Master Reset Code for this very purpose: ⟨ESC⟩ CHR$(64).

To see the Master Reset Code work, add these lines to your budding program:

```
30 LPRINT CHR$(27)CHR$(64)
40 LPRINT "BACK TO ROMAN WITH THE MASTER RESET"
```

and RUN it.

```
ITALIC CHARACTER SET!

BACK TO ROMAN WITH THE MASTER RESET
```

Line 30 turns off *all* special print modes and returns the FX-80 to "normal" print (i.e., to its default settings as determined by the internal switch settings). The factory sets this for roman typeface. Line 40 is the proof of the pudding.

Note that line 30 adds an extra line feed between the two rows of text. BASIC automatically provides a line feed after every print line at no extra charge, whether that line prints text or not. A trailing semicolon (;) at the end of this line would eliminate the extra line space. We will use trailing semicolons extensively throughout this manual to link print lines together.

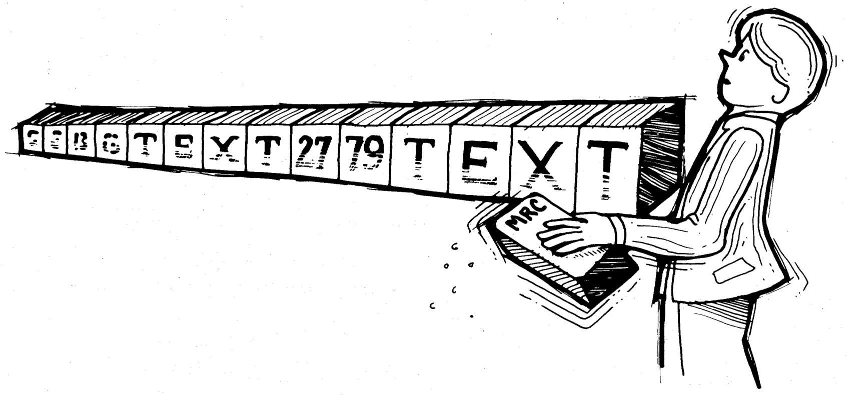
The Master Reset Code is useful when you want to turn off all printer modes. It resets *everything* to its start-up condition, including the user-defined characters which we introduce in Chapter 17. If you have two or three different modes active in the printer at one time, they would all be shut off with the 〈ESC〉CHR$(64).

The print buffer

Every printer control code sent to the printer is stored in the printer's RAM buffer right along with the text. All material goes through this buffer to get to the printed page. This buffer is like a holding tank in which each print line is assembled.

The buffer can hold a full line of text characters (80 characters for normal-width print, more characters for narrower widths) as well as control codes. All information resides in the buffer until the buffer is filled or a control code that empties the buffer is received. Then the line is processed, one character at a time: text characters are printed on the page, and control codes are activated as they are encountered.

When the Master Reset Code is sent to a partially full buffer, all text characters and control codes currently in the buffer get erased. At the same time all printer modes are reset to their default conditions.

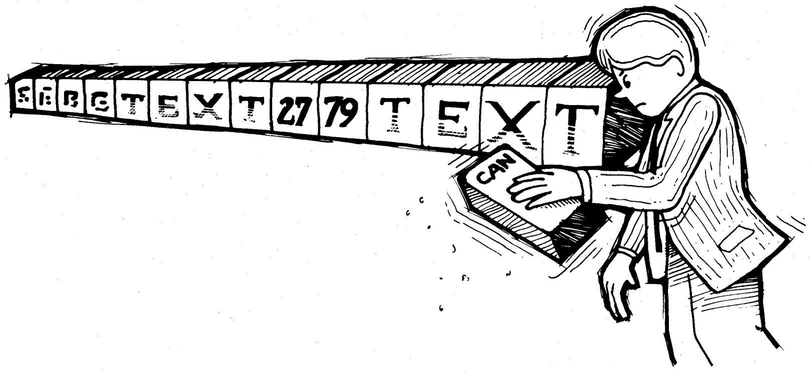


CANcel and DELete

But suppose you don't want that much power. Suppose you only want to erase text in the buffer without affecting any print modes. There is a way! It's called the CANcel code and it looks like this:

```
CHR$(24)
```

The CAN code, like the Master Reset, erases any text currently in the printer buffer. But there the similarity ends. The CAN code does not delete any of the control codes in the buffer, nor does it reset any print modes.

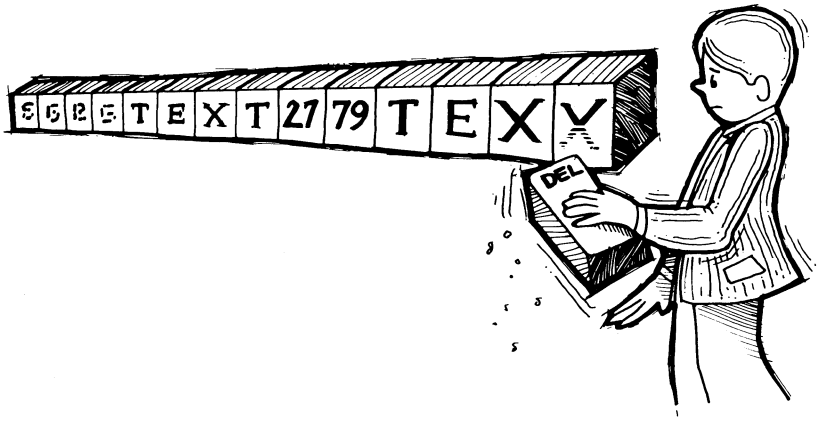


Add the CAN code to the end of line 20 above, print it to see how it acts, and then restore the line.

Another code that deletes characters from the buffer is the DELete code:

```
CHR$(127)
```

When placed in the buffer, DEL deletes only the previous text character. It does not affect control codes.



Specific reset codes

The FX-80 also provides specific codes to turn off each mode separately. For example, an ⟨ESC⟩CHR$(53) will turn off the italic character set and leave everything else untouched. To see how the reset code for italic characters works, change the Master Reset Code in line 30 above to an ⟨ESC⟩CHR$(53):

```
30 LPRINT CHR$(27)CHR$(53)
40 LPRINT "BACK TO ROMAN WITH ITALIC OFF"
```

```
ITALIC CHARACTER SET!

BACK TO ROMAN WITH ITALIC OFF
```

You should get the same results as you did on the last RUN. What the current program doesn't demonstrate is that the ⟨ESC⟩CHR$(53) resets only the Italic Mode, while the Master Reset Code resets all modes. In the following pages, we give you each mode's particular reset code as we cover that mode.

ASCII Characters in ESCape Sequences

Since the FX-80 has so many features, remembering all of them by number is not easy. Fortunately, there is another way to send the required codes to the printer.

In some cases it is convenient to use (inside the quotation marks) the ASCII character that equates with a given code number instead of the CHR$(n) format. A string character is shorter and easier to remember than a number. For example, CHR$(64) can be replaced by "@". (If you don't believe it, trot back to Appendix A and check it out.) So an alternate way to send the Master Reset Code is to send the ESCape code plus "@":

```
    LPRINT CHR$(27)"@"
```

Using the "@" in place of CHR$(64) simplifies the process, and it's also easier to remember. We will use symbols whenever possible to keep the programs short. Change these lines in your current program:

```
    1Ø LPRINT CHR$(27)"4"
    3Ø LPRINT CHR$(27)"@"
```

and RUN it to make sure that both ESCape codes work as before.

In the next two chapters, you'll learn how to control print width and print quality for all kinds of useful applications.

Summary

All printer commands stay active until they are turned off. (Well, there are a few exceptions, but we'll cover them later.) You can turn off all printer commands by turning the printer off or by sending the Master Reset Code. In addition, each mode has its own reset code that will turn off only that particular mode.

See Appendixes D and E for tables of the control codes.

CHR$(n)	ASCII code format. Allows communication between computer and printer:

Codes	**FX-80 Interpretations**
0 to 31, 127	Printer control codes
32 to 126	Roman (standard) character set
128 to 159, 255	Printer control codes (repeat of 0-31, 127)
160 to 254	Italic version of the roman character set

CHR$(24)	CAN code
CHR$(27)	ESCape code
CHR$(127)	DEL code
⟨ESC⟩"@"	Master Reset Code
⟨ESC⟩"4"	Italic Mode ON
⟨ESC⟩"5"	Italic Mode OFF

Notation Used in This Manual

1) A special character (ƀ) will be used to represent blank spaces when several spaces are required in a print string. This will make it easier to count spaces. For example:

```
"ƀƀƀƀƀƀƀƀƀSAMPLE STRING"
```

Means that you type:

```
"         SAMPLE STRING"
```

which allows 9 spaces between the initial quote mark and the first S.

2) Remarks will be included in some program lines, always preceded by an apostrophe (').

Example:

```
1Ø LPRINT CHR$(27)"@" ' Master Reset Code
```

or

```
99 ' Data lines for graphics
```

You may not want to type the remarks into your program; they are for your clarification only.

3) The carat symbol (^) will be used to indicate exponents.
For example:

```
10 X = Y^2
```

means let X equal Y raised to the second power. Some systems use an up-arrow (↑), which prints as a left bracket ([ ) on the FX-80.

Chapter 3
Print Pitches

One of the big advantages the FX-80 printer has over a daisy-wheel printer or a typewriter is the ability it gives you to easily vary the width, or "pitch" of characters. To use this feature well, it's important to understand just how the FX-80 prints. The technique used by the FX-80 printer is called "dot-matrix printing."

Dot-Matrix Printing

To see how dot-matrix printing works, we need to look first at the print head and then at the way characters are stored in the internal memory of the printer.

The print head

A dot matrix is a grid or graph that a dot-matrix printer uses to plot characters (which may be letters of the alphabet, numbers, or symbols). On the FX-80 this matrix is nine rows of dots high and six columns of dots wide. Look at any letter on your printout—it's made up of a series of dots. And, as you can see in Figure 3-1, every letter fits inside this six by nine grid. You may be wondering why the p dips lower than the H. A few lower-case letters must descend below the normal line of print (that's the seventh row from the top of the matrix). All numbers, upper-case letters, and most symbols are formed within the top seven rows of the matrix.

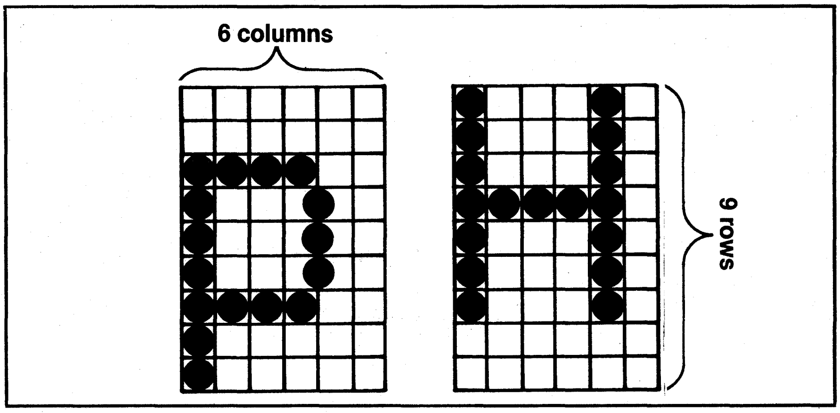
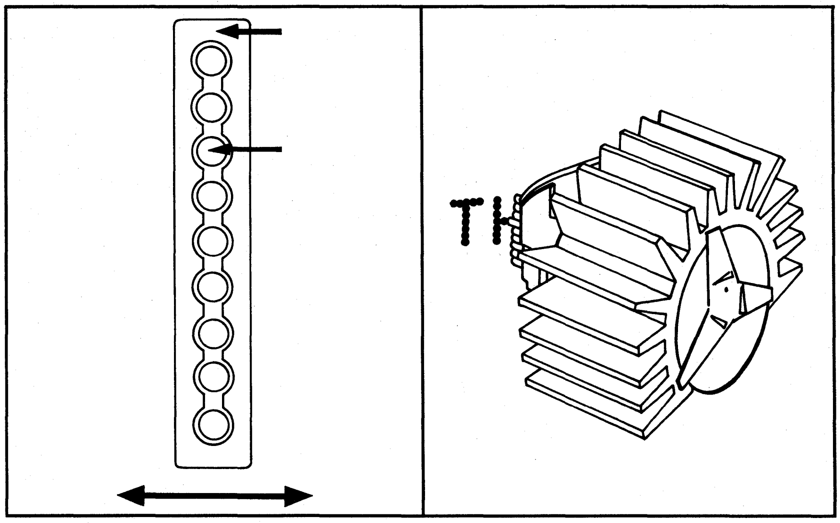


Figure 3-1. Dot-matrix characters.

Main columns

The construction of the print head restricts the maximum height of any character to nine dots. The print head uses a vertical column of nine pins (actually, wires—see Figure 3-2a) that are fired at the paper by electrical impulses. Each pin presses against the ribbon to produce a single dot of the matrix. Because there is only one column of pins, the head must move sequentially to each of the different column positions of the matrix, then fire the appropriate pins. At each position, only the pins necessary to print the current character are fired.



a. The wires **b. In action.**

Figure 3-2. The print head.

To print a capital H as in Figure 3-2b, the print head fires pins 1 through 7 in column 1; pin 3 in columns 2, 3, and 4; and pins 1 through 7 again in column 5.

Intermediate positions

Most characters are designed to be 5 or less columns wide. As we saw in Figure 3-1, leaving the sixth column blank allows for space between each pair of letters. Only one character in the ROM uses all 6 dot positions—the underline character (Figure 3-3).

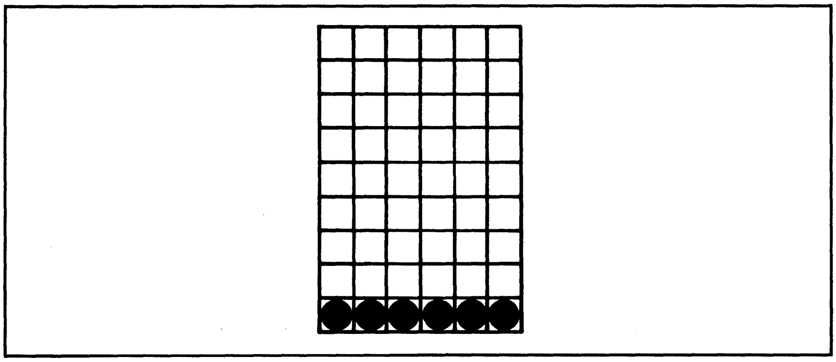


Figure 3-3. All dots in main columns.

All the rest fit neatly into the 5-dot format, freeing the sixth column for spacing. But use of 5 dots does not give quite enough detail for the highest quality characters. So the FX-80 prints some dots halfway between the main columns in the 6-dot-wide matrix. This means that the matrix grid for any character actually contains 11 columns—6 main columns with 5 intermediate columns. You can count the 11 positions on the grid shown in Figure 3-4.

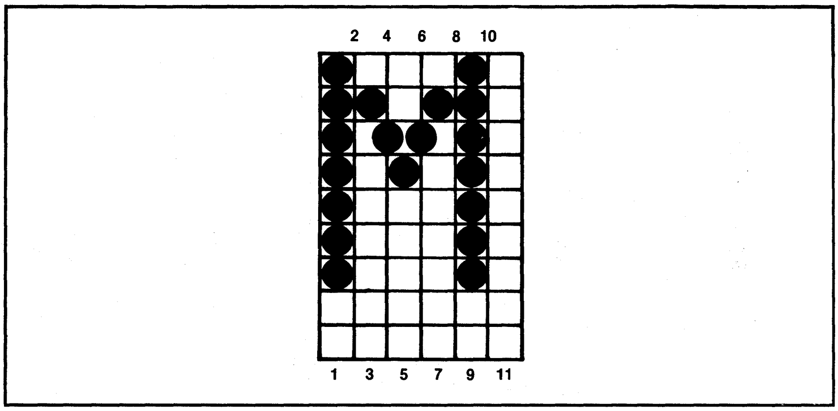


Figure 3-4. Use of intermediate positions 4 and 6.

The dots printed in these intermediate positions actually overlap with those in the main columns. If you look through Appendix B, you'll notice that *none* of the ROM characters use consecutive main and intermediate columns in the same row.

There is a reason for this. Amazing as it seems, the printer is able to reach into memory to recall a character's dot matrix pattern and print the character in 1/160th of a second. At that speed, the print head is simply moving too fast to pull the pins back and forth in time to print an overlapping dot. This fact is crucial in character design, as you will see in Chapter 17.

Modes for Pitches

Three of the FX-80's many modes produce characters in different widths, or "pitches." You may recognize two of these pitches as standard character widths used on typewriters; the third produces a narrower character. Each of these is covered below, followed by a discussion of how the FX-80 handles conflicts between its print modes.

Pica and Elite Modes

At start-up the FX-80 prints 10 characters per inch (cpi). This is the same width, or "pitch" as that of a typewriter's Pica character set. The FX-80 can also print characters in an Elite pitch (12 cpi), which gives 96 characters per line instead of the usual 80.

Changing the print pitch between Pica and Elite does not change the number of *columns* in each character, nor does it change the *pattern* used to create each character. It simply moves the columns closer together or farther apart to change the width of each character. To see how this works, compare the Pica and Elite Ms in Figure 3-5.

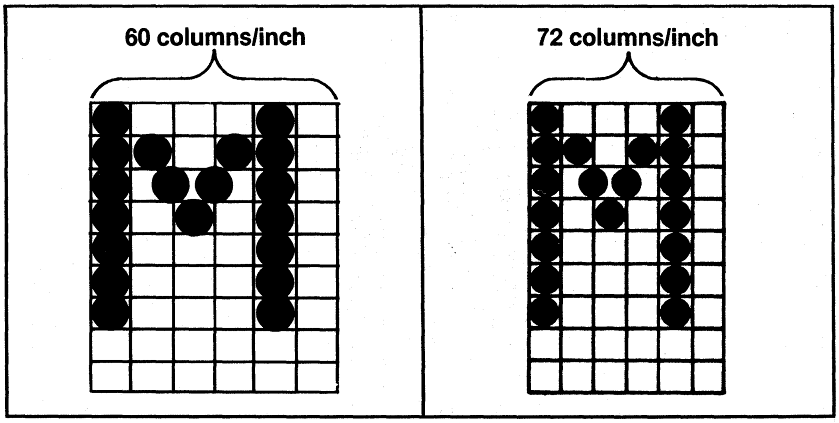


Figure 3-5. Pica and Elite M.

Since the columns used to print Elite characters are packed more closely than those for Pica, more of them fit on one line of print. And notice that Elite pitch compresses spaces as well as characters. Try this new program:

```
NEW
10 LPRINT CHR$(27)"M"
20 LPRINT TAB(10);"COMPARE ELITE PITCH WITH THE PICA
   BELOW!"
30 LPRINT CHR$(27)"P"
40 LPRINT TAB(10);"PICA PITCH IS THE NORMAL PRINT
   WIDTH!"
```

When you RUN it, you should get:

```
          COMPARE ELITE PITCH WITH THE PICA BELOW!

          PICA PITCH IS THE NORMAL PRINT WIDTH!
```

Note: If your BASIC does not have a TAB statement, use 10 blank spaces.

Elite Mode (which produces Elite pitch) is set with an ⟨ESC⟩"M". The TAB function in line 20 above tabs 10 Elite spaces; the TAB in line 40 tabs 10 Pica spaces. The point is that the different width modes apply to tab positions and spaces as well as to characters (more about tabs in Chapter 9).

The ⟨ESC⟩"P" in line 30 exits Elite Mode and returns the printer to Pica pitch. Note that Pica is the factory-set default, so it comes on whenever you turn the printer on.

Compressed Mode

There is a third handy print pitch available with the FX-80 printer. Replace your current program with this one:

```
NEW
20 LPRINT CHR$(15) "COMPRESSED MODE IS SET WITH
   CHR$(15)"
30 LPRINT "IT WILL STAY ON UNTIL YOU CANCEL IT"
40 LPRINT CHR$(18) "PICA AGAIN!"
```

```
COMPRESSED MODE IS SET WITH CHR$(15)
IT WILL STAY ON UNTIL YOU CANCEL IT
PICA AGAIN!
```

Notice that the command to enter Compressed Mode does not include an ESCape code; however, ⟨ESC⟩CHR$(15) can also be used.

Most print modes on the FX-80 stay on until they are turned off; Compressed Mode is no exception. And there is a unique code that turns it off—CHR$(18). (The Master Reset Code would work just as well, but remember that it also resets *all* other current printer modes to the defaults. The FX-80 gives us a choice of resetting codes one at a time or all at once.)

At 17.16 characters per inch, Compressed Mode is the narrowest character type available on the FX-80 printer. We can squeeze 132 Compressed characters into each line (up to 137 with changes in the margin settings; see Chapter 9). Compressed Mode is useful for printing spread sheets or reports that require several columns of information.

If you find yourself using this pitch more often than not, you can change the default pitch to Compressed Mode by setting switch 1-1 on. *(Change all switch settings with the printer off.)* This will make the printer automatically come up in Compressed Mode, after which you can switch to other modes as needed.

Conflicting modes

The three different pitches—10 cpi (Pica), 12 cpi (Elite), and 17.16 cpi (Compressed)—are mutually exclusive. That is, only one can be in use at a given time. When a program activates two conflicting modes at the same time, one of them will take precedence. In the case of Elite and Compressed Modes, for example, Elite has the higher priority. To check this, try RUNning this line in your program:

```
10 LPRINT CHR$(27)"M";"ELITE PITCH"
```

```
ELITE PITCH
COMPRESSED MODE IS SET WITH CHR$(15)
IT WILL STAY ON UNTIL YOU CANCEL IT
PICA AGAIN!
```

The entire printout is in Elite pitch even though Compressed Mode is turned on in line 20. Does that mean that the printer ignores the CHR$(15) when it is in Elite Mode? Let's find out. Change line 30 to read:

```
30 LPRINT CHR$(27)"P";"CANCEL ELITE TO SEE
   COMPRESSED"
```

```
ELITE PITCH
COMPRESSED MODE IS SET WITH CHR$(15)
CANCEL ELITE TO SEE COMPRESSED
PICA AGAIN!
```

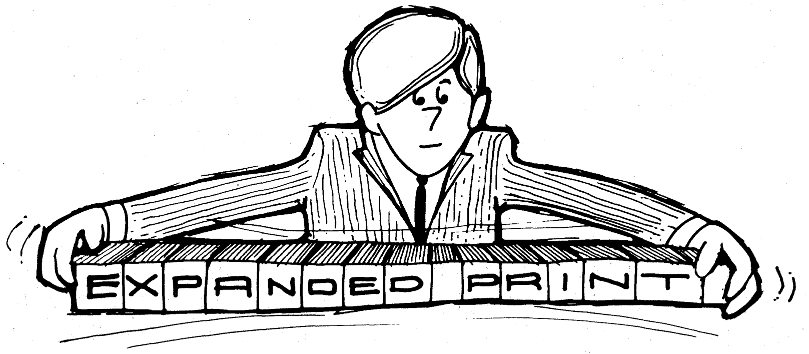
⟨ESC⟩"P" sure enough cancelled Elite, and line 30 printed in Compressed print! So the printer did recognize the CHR$(15) in line 20; it just couldn't show it until the Elite was cancelled.

The lesson just learned should not be taken lightly. It is a good example of how print modes interact on the FX-80.

Mixing Pitch Modes

The previous three modes can't be mixed, but the next mode can be used in combination with any of them. And it can be added for either of two durations: for one line or for a longer passage.

Expanded Mode



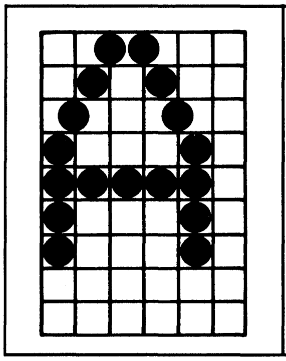
Here is another variation on print width; this one, called Expanded Mode, doubles the width of the current mode. Since it can be combined with all three pitches, Expanded Mode doubles the number of available print pitches. (Note: Expanded Mode is the same as "Double Width print" on the MX printers. A rose by any other name . . .).

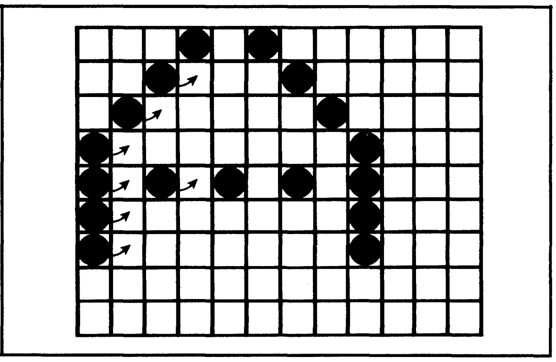
To see Expanded characters, type in:

```
NEW
10 LPRINT CHR$(14)"EXPANDED PRINT"
20 LPRINT "TURNS OFF AFTER EACH LINE WITH CHR$(14)"
30 LPRINT CHR$(27)"W"CHR$(1) "EXPANDED PRINT STAYS
   ON"
40 LPRINT "CONTINUOUSLY WITH <ESC> W"
50 LPRINT CHR$(27)"W"CHR$(0)
```

```
EXPANDED PRINT
TURNS OFF AFTER EACH LINE WITH CHR$(14)
EXPANDED PRINT STAYS ON
CONTINUOUSLY WITH <ESC> W
```

The printer actually extends the dot matrix by spreading the dots horizontally to twice their normal distances apart and adds a duplicate of each dot to the next full dot position (see Figure 3-6).


a. Pica A.


b. Dots spread twice Pica distance.

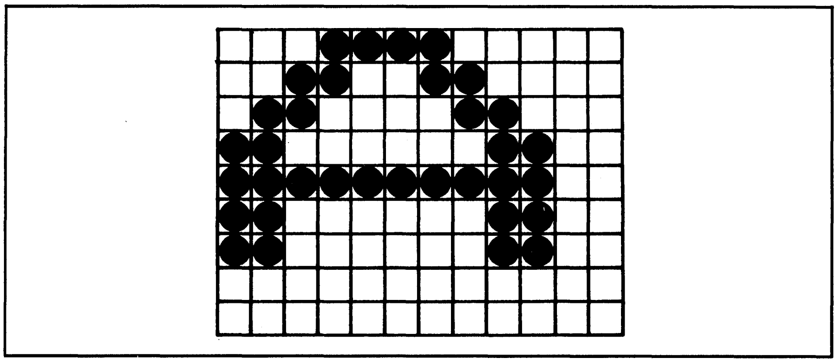

c. Expanded A.

Figure 3-6. Pica and Expanded A.

There are two ways to turn on the single-line Expanded print feature. With CHR$(14) (or optionally with ⟨ESC⟩CHR$(14)), it turns off after each print line. If you want Expanded Mode to stay on line after line, you must use ⟨ESC⟩"W" followed by CHR$(1).

Most modes on the FX-80 stay on continuously, and so your program must turn each mode off when you are finished with it. The few exceptions to this rule will be noted as they come up.

Those whose computers have difficulty sending a CHR$(0) or CHR$(1) should note that the FX-80 will also accept CHR$(27)"W1" and CHR$(27)"W0" to turn Expanded Mode on and off. In fact, all FX-80 modes that use CHR$(1) and CHR$(0) to turn modes on and off will also accept "1" and "0". This shorter form is used in the next program.

Also notice that the ⟨ESC⟩"W"CHR$(1) is turned off with ⟨ESC⟩"W"CHR$(0); but CHR$(14) can be turned off with either CHR$(20) or ⟨ESC⟩"W"CHR$(0). It is *very important* to turn off print modes when you are done with them. If you don't turn Expanded print (⟨ESC⟩"W" variety) off, it will still be active the next time you RUN a program (unless you have turned off the printer, of course!).

Expanded Mode works equally well with any of the three basic pitches. Watch the line numbering carefully as you type and RUN this program:

```
NEW
1Ø LPRINT "YOU CAN PRINT EXPANDED:"
2Ø LPRINT CHR$(27)"W1";"PICA "
4Ø LPRINT CHR$(15);"COMPRESSED "
6Ø LPRINT CHR$(27)"M";"AND ELITE "
8Ø LPRINT CHR$(27)"@";"CHARACTERS!"
```

```
YOU CAN PRINT EXPANDED:
P I C A
COMPRESSED
A N D   E L I T E
CHARACTERS!
```

Mixing print pitches on one line

Suppose that you want to emphasize just one word within a line by printing it in a different pitch. The following program shows how it can be done; it mixes all six print pitches on a single print line. (Notice that this is a modification of the program that you just ran.)

```
1Ø LPRINT "YOU CAN MIX:"
2Ø LPRINT "PICA ";
3Ø LPRINT CHR$(14);"EXPANDED, ";
4Ø LPRINT CHR$(2Ø);CHR$(15);"COMPRESSED ";
5Ø LPRINT CHR$(14);"EXPANDED, ";
6Ø LPRINT CHR$(2Ø);CHR$(27)"M";"AND ELITE ";
7Ø LPRINT CHR$(14);"EXPANDED"
8Ø LPRINT CHR$(27)"@";"CHARACTERS ON THE SAME LINE!"
```

```
YOU CAN MIX:
PICA E X P A N D E D ,   COMPRESSED
CHARACTERS ON THE SAME LINE!
```

The trailing semicolons on *program* lines 20 to 60 keep all the words on one *print* line. Note that your computer may require semicolons between *all* print strings (if so, you will need to add them to later programs).

CHR$(14) and CHR$(20) are used to pop in and out of Expanded Mode.

Compressed Mode is turned on in line 40, but not turned off until line 80. The Compressed Mode is masked in line 60 when Elite is turned on. Since the combined Compressed and Expanded print width is twice as wide as Compressed (17.16 cpi), Expanded Compressed Mode is 8.58 cpi.

Elite width is turned on in line 60 and off in line 80. Expanded Elite characters are 6 cpi. That's right. Half as many cpi as Elite Mode alone (12 cpi).

Doubling the width of the three standard pitches adds more dots to the matrix of each character, so it results in a darker print. There are other ways to get darker print, plus many other ways to change the looks of the type . . . all that is in the next chapter, on print quality.

```
E X P A N D E D ,   AND ELITE E X P A N D E D
```

Summary

Table 3-1. Print Type Summary

Print sample | ← 1 inch → |	Density (in cpi)	Entry code
PICA PRINT	10	Default mode (Roman Pica)
ELITE PRINT	12	⟨ESC⟩"M"
COMPRESSED PRINT	17.16	CHR$(15) or ⟨ESC⟩CHR$(15)
EXPANDED PICA PRINT	5	⟨ESC⟩"W1" or CHR$(14) or ⟨ESC⟩CHR$(14)
EXPANDED ELITE PRINT	6	⟨ESC⟩"W1";⟨ESC⟩"M"
EXPANDED COMPRESSED PRINT	8.58	⟨ESC⟩"W1";CHR$(15)
| ← 1 inch → |		

Codes	FX-80 Interpretations
⟨ESC⟩"M"	Elite Mode ON
⟨ESC⟩"P"	Elite Mode OFF (back to the previous pitch)
CHR$(15)	Compressed Mode ON
CHR$(18)	Compressed Mode OFF
CHR$(14)	Expanded Mode (one line) ON
CHR$(20)	Expanded Mode (one line) OFF
⟨ESC⟩"W1"	Expanded Mode (continuous) ON
⟨ESC⟩"W0"	Expanded Mode (both) OFF

Chapter 4
Print Quality

Modes for Weights

In the last chapter you changed the width of the printed characters to achieve several different print *styles.* The FX printer also offers three modes which improve print *quality:* Double-Strike, Emphasized, and Proportional.

Double-Strike Mode

The first mode which affects the weight and thus the quality (rather than the width) of print is the Double-Strike Mode. It prints each line twice. Try this:

```
NEW
1Ø LPRINT CHR$(27)"G";"DOUBLE-STRIKE PRINT IS
   DARKER";
2Ø LPRINT CHR$(27)"H";" THAN SINGLE-STRIKE"
```

and RUN it:

```
DOUBLE-STRIKE PRINT IS DARKER THAN SINGLE-STRIKE
```

Double-Strike Mode is turned on with ⟨ESC⟩"G". It stays on until turned off with ⟨ESC⟩"H". In this mode, the letters look less like a bunch of dots and more like fully formed characters.

The way Double-Strike gets this result is rather clever: each line is printed once, then the paper is shifted up slightly (about 1/216 of an inch) and the entire line is printed again. In other words, each row has a "shadow" (see Figure 4-1). Double-Strike printing fills in some of the more obvious gaps in the dot matrix of the characters. The end result is better looking print.

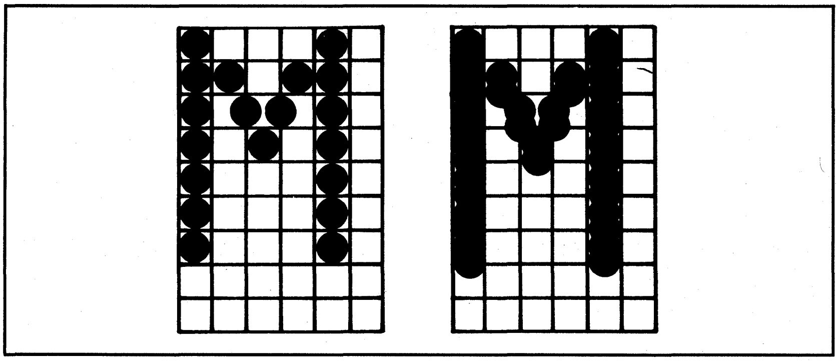


Figure 4-1. Single- and Double-Strike M.

The differences between Double- and Single-Strike printing don't stop with the quality of print. Since each line is printed twice, the throughput of the Double-Strike Mode is half that of normal. It's the old trade-off between speed and print quality. With a normal print speed of 160 characters per second (cps), the FX-80 still steps along pretty lively in the Double-Strike Mode.

Programming note: Notice that the printer always expects another code after the ESCape. This second code tells the printer which mode to turn on or off. In the case of Double-Strike Mode, we use the quoted letter G ("G") to turn it on and "H" to turn it off. The letters, however, don't have to stand alone inside the quotes. Consider:

```
10 LPRINT CHR$(27)"GDOUBLE-STRIKE PRINT IS DARKER";
20 LPRINT CHR$(27)"H THAN SINGLE-STRIKE"
```

Yes, this listing looks peculiar, but these lines give the same output as the earlier version. The G and H are not printed on the paper; instead, they are used as part of the ESCape sequence.

Emphasized Mode

Add these lines to your program to try yet another print mode:

```
30 LPRINT CHR$(27)"E";"EMPHASIZED ADDS A TOUCH OF
   CLASS"
60 LPRINT CHR$(27)"@"
```

```
DOUBLE-STRIKE PRINT IS DARKER THAN SINGLE-STRIKE
EMPHASIZED ADDS A TOUCH OF CLASS
```

⟨ESC⟩"E" produces what we call Emphasized print. In this mode, the dots in each *column* in the matrix definition of the character are added to the dots in the next half column over. That's right, it's very similar to Expanded print, but the dots in Expanded Mode are printed in every other column. Figure 4-2 shows the half-dot spacing of Emphasized Mode.

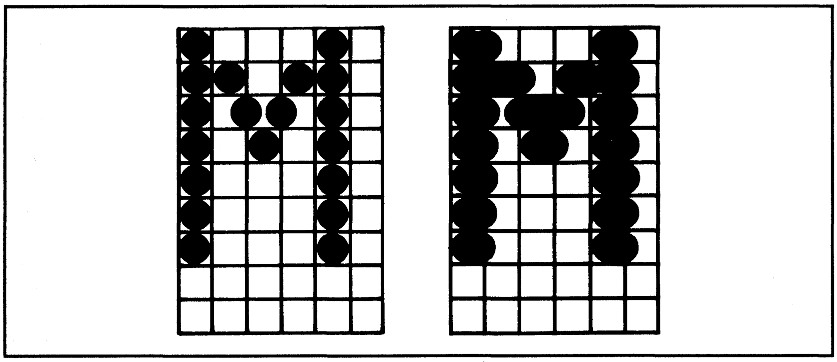


Figure 4-2. Single-Strike and Emphasized M.

Neat trick! Although the print head slows down to half speed in Emphasized Mode, the increase in print quality is well worth it.

<ESC>"F" is the shut-off code for Emphasized Mode.

Emphasized Mode is a variation of Pica; it cannot be combined with Elite or Compressed print. If you like this mode, you can make it the default on power-up by setting internal switch 1-5 on. (See Appendix K for switch settings.)

Emphasized and Double-Strike in concert

For even better quality, you can combine Emphasized with Double-Strike. Add:

```
    4Ø LPRINT CHR$(27)"G";"COMBINED THEY CAN'T BE BEAT!"
```

to your program.

```
DOUBLE-STRIKE PRINT IS DARKER THAN SINGLE-STRIKE
EMPHASIZED ADDS A TOUCH OF CLASS
COMBINED THEY CAN'T BE BEAT!
```

Emphasized Mode (line 30) stays on until it is shut off. Line 40 kicks in Double-Strike before Emphasized is turned off. You see the result above.

The only drawback to all this high-powered printing is the wear and tear on the ribbon. Understandably, these extra dense modes use up the life of a ribbon faster than Single-Strike Mode does. Used sparingly, however, they can give you increased capability for a low overhead.

Proportional Mode

Have you ever wondered why most computer printouts don't look quite as good as typeset books, even when bold characters are used? That's because most dot-matrix printers use a uniform width for each character ("monospacing") whereas typesetting machines set the width for each character "proportional" to its size. That is, narrow characters like i and ! are printed without the excess space that would be used if they were printed in the same width as m and w.

Now the FX-80 offers you a Proportional Mode. In this mode, characters are printed with a uniform amount of blank space between them, which produces much more readable text. For the user, the only drawback to printing everything in Proportional Mode is the difficulty of lining up characters in vertical columns. This makes listings and tables look silly (when not unreadable).

Here's an example of the difference between Monospaced and Proportional Modes. Enter:

```
NEW
1Ø LPRINT "!!!!!!!!!!!!!";
2Ø LPRINT CHR$(27)"p1"
4Ø LPRINT "!!!!!!!!!!!!!PROPORTIONAL ON"
6Ø LPRINT CHR$(27)"pØ";
7Ø LPRINT "!!!!!PROPORTIONAL OFF"
8Ø LPRINT CHR$(27)"@"
```

```
! ! ! ! ! ! ! ! ! ! ! ! !
!!!!!!!!!!!!!PROPORTIONAL ON
! ! ! ! !PROPORTIONAL OFF
```

Lines 10 and 40 print the same number of exclamation marks, but the characters from line 40 are packed more closely. Proportional Mode prints the characters in Emphasized Mode and strips off all unused space between characters.

The 〈ESC〉"p1" turns on the Proportional Mode (just as a "1" or CHR$(1) is used to initiate 〈ESC〉"W"). If your keyboard doesn't have lower-case characters, change line 20 to

```
LPRINT CHR$(27)CHR$(112)"1"
```

The shut-off switch for Proportional print is 〈ESC〉"p0". This puts the printer back into the mode that it was in before it entered Proportional Mode. As an example, if Proportional Mode is entered from Compressed Mode, 〈ESC〉"p0" returns the printer to that mode.

Since all Proportional characters are Emphasized, it makes sense that Proportional characters, like Emphasized, can only be printed in Pica pitch, not Elite or Compressed. In addition, Proportional print cannot be mixed with Double-Strike Mode.

More Mixing of Modes

While we're on the subject of mixing print modes, let's take a closer look at the way the FX-80 handles several print modes at the same time. As you probably noticed, many of the modes introduced so far have a restricted ability to mix with other print modes. In each case there is a good reason why two modes can't be combined.

For example, Emphasized and Proportional (both are Emphasized print) Modes cannot be mixed with Elite or Compressed pitches. The reason is that Emphasized characters already violate the cardinal rule that two overlapping dots cannot be printed in the same row. In Pica Mode at half speed, the printer can make an exception. But the dots in Elite and Compressed characters are already so closely packed that printing them in Emphasized print as well is not possible, even at half speed. The print head simply cannot fire and retract the pins fast enough at this speed.

So what does the printer do when it receives a request for two conflicting modes? Ignore one of them, take a vacation, or beep loudly in disgust? The answer is: none of the above. It turns both modes "on" internally, but—based on a factory-set priority list—selects only one of them for printing text.

As we saw in Chapter 3 when we cancelled Elite to see Compressed print ("Conflicting modes"), the printer remembers that a mode is on even though it doesn't affect the current print line. For example, let's turn on Double-Strike at the same time as we turn on Proportional. Then we can confirm that the printer recognizes the Double-Strike switch when Proportional is turned off.

Add lines 30 and 50, and make some changes to line 70:

```
30 LPRINT CHR$(27)"G";
50 LPRINT "WHEN PROPORTIONAL GOES"
70 LPRINT "!!!!!!!!!!!!OFF, DOUBLE-STRIKE CAN COME
   ON"
```

```
! ! ! ! ! ! ! ! ! ! ! !
!!!!!!!!!!!!PROPORTIONAL ON
WHEN PROPORTIONAL GOES
! ! ! ! ! ! ! ! ! ! ! !OFF, DOUBLE-STRIKE CAN COME ON
```

Even though Proportional Mode will not permit Double-Strike to affect lines 40 and 50 (since Proportional Mode has priority), Double-Strike does take hold as soon as Proportional is shut off. You'll see this same phenomenon as other modes are introduced in this manual.

Mode Priorities

Table 4-1 displays the modes we have covered so far.

Table 4-1. Summary of Modes.

Type of Mode	Mode Name
Typeface	Roman (default) Italic
Pitch	Pica (default) Elite Compressed Expanded
Weight	Single-Strike (default) Double-Strike Emphasized
Spacing	Monospaced (default) Proportional

Note: Pitch and weight together make up the print density.

That's a lot of potential combinations! But they don't all combine successfully, as you have just witnessed. When two modes conflict, one always takes priority. Table 4-2 shows the list of priorities used by the printer:

Table 4-2. Mode Priorities.

$$\begin{tikzcd}
\text{Elite} \arrow[d] \\
\text{Proportional (Emphasized)} \arrow[d] \\
\text{Emphasized} \arrow[d] \\
\text{Compressed}
\end{tikzcd}$$

$$\begin{tikzcd}
& \text{Proportional} \arrow[dl] \arrow[dr] & \\
\text{Script} & & \text{Double-Strike}
\end{tikzcd}$$

Note: Elite Mode takes precedence over the modes beneath it. For example, if both Elite and Compressed are "on," the printing will be in Elite. If Elite is turned off at this point, the printing will change to Compressed.

Elite takes priority over all the modes below it in the chain. The same is true of Proportional and Emphasized Modes. In addition, Proportional has priority over Double-Strike and over the two Script Modes that are covered in Chapter 5.

Summary

Double-Strike and Emphasized Modes can be used to produce bolder characters. The drawbacks are minimal: slower speed and shorter ribbon life. Double-Strike can be combined with all modes except Proportional, while Emphasized cannot be combined with Elite or Compressed.

Proportional spacing strips off any excess spaces between characters. It is only valid in Pica pitch and automatically prints in Emphasized Mode. It cannot be mixed with Double-Strike.

Appendix C gives a complete listing of Proportional Mode's character widths.

Codes	**FX-80 Interpretations**
⟨ESC⟩"E"	Emphasized print ON
⟨ESC⟩"F"	Emphasized print OFF
⟨ESC⟩"G"	Double-Strike ON
⟨ESC⟩"H"	Double-Strike OFF
⟨ESC⟩"p1"	Proportional spacing ON
⟨ESC⟩"p0"	Proportional spacing OFF

Note: Use ⟨ESC⟩CHR$(112)"1" or ⟨ESC⟩CHR$(112)"0" if your computer can't generate a lower-case p.

Chapter 5
Dress-Up Modes and Master Select

Dress-Up Modes

In this chapter we cover four more print modes (Underline, Superscript, Subscript, and Italic); each allows you to add a finishing touch to your printouts. Later we show how these modes can be combined with the modes for print pitch and density that were covered in previous chapters.

Underline Mode

In the old days, dot-matrix printers did not know how to underline words. Even in the not-so-old days, printer users like yourself had to use all kinds of tricks to underline words. The technique usually involved using either the hyphen (—) or underscore (__), along with either a change of line spacing or the use of backspace. When it worked, the right words got underlined, but those methods were tedious and time consuming.

Those days are now gone, thanks to the FX-80. It has a built-in Underline Mode that makes underlining very easy. Here is the format:

```
CHR$(27)"-"CHR$(1)
```

or

```
CHR$(27)"-1"
```

The 〈ESC〉"—1" turns Underline Mode on, and 〈ESC〉"—0" turns it off. Enter and RUN this program to see what it looks like:

```
NEW
2Ø LPRINT CHR$(27)"-1";"UNDERLINING IS SIMPLE!";
4Ø LPRINT CHR$(27)"-Ø";" AND EASY TO TURN OFF"
```

```
UNDERLINING IS SIMPLE! AND EASY TO TURN OFF
```

Notice that the printer takes two passes to print the line, and that it is the second pass which actually does the underlining.

You can underline virtually anything you want—even blank spaces. Try adding this line:

```
3Ø LPRINT "ƀƀƀƀƀƀ";
```

```
UNDERLINING IS SIMPLE!       AND EASY TO TURN OFF
```

to see that the Underline Mode stays on to underscore the blanks.

Being able to underline a blank space really helps when you want to make a form that has lines for signatures or "fill-ins." In Chapters 9 and 10, we'll use Underline Mode in this way. Underline even underlines leading spaces with BASIC TABs—although it doesn't work with the FX-80's internal tabs that are set with ⟨ESC⟩"D" (see Chapter 9).

Super- and Subscript Modes

The three print pitches (the modes discussed in Chapter 3) affect the width of characters: both Elite and Compressed Modes may be seen as squeezing Pica characters *horizontally.* The FX-80 can also compress a character *vertically*—to about half its normal height. These short characters are called *script* characters (not to be confused with fancy "script lettering"). Because the script characters are so short, they can be placed high or low on the line; thus, we have SUPERscripts and SUBscripts.

This is the script command:

```
LPRINT CHR$(27)"S"
```

After the script command, you must specify which script you want, super or sub. Superscript is achieved with a CHR$(0) or "0", Subscript with a CHR$(1) or "1". Whichever code you use, it goes directly after the script command. Try this to see them both:

```
20 LPRINT CHR$(27)"SØ";"SUPERSCRIPT";
3Ø LPRINT CHR$(27)"T AND";
4Ø LPRINT CHR$(27)"S1";" SUBSCRIPT"
5Ø LPRINT CHR$(27)"TCAN EVEN GO ON THE SAME LINE"
```

```
SUPERSCRIPT AND SUBSCRIPT
CAN EVEN GO ON THE SAME LINE
```

Notice that ⟨ESC⟩"T" turns the Script Modes off, and also that both Script Modes are automatically printed with Double-Strike print. (That means that they can't be combined with Proportional print.) If Double-Strike was the current mode before the printer entered a Script Mode, the printer returns to Double-Strike Mode. If not, the printer returns to Single-Strike.

Italic Mode

This is the last of the print modes that affect print quality. Italic characters are printed in a completely different typeface than are the more usual roman characters. Appendix B shows the dot-matrix patterns used to define the italic characters, along with their corresponding ASCII numbers.

Even if your computer cannot send codes greater than 128 to the printer, you can still print italic characters. Add:

```
1Ø LPRINT CHR$(27)"4"
6Ø LPRINT CHR$(27)"@"
```

```
SUPERSCRIPT AND SUBSCRIPT
CAN EVEN GO ON THE SAME LINE
```

Italic Mode mixes just fine with Underline and with Super- and Subscript Modes. In fact, all four of these modes can be mixed and matched at will—with the lone exception that Super- and Subscript cannot be used simultaneously.

Add Underline to your current program with:

```
1Ø LPRINT CHR$(27)"4"CHR$(27)"-1"
```

```
SUPERSCRIPT AND SUBSCRIPT
CAN EVEN GO ON THE SAME LINE
```

Master Print Mode Select

With all the nice looking print types we've covered up to this point, you're probably anxious to start experimenting with different print combinations. Well, here's your chance.

The FX-80 has a way for you to select certain print mode combinations by number. This Master Print Mode Select feature (Master Select for short) makes it easy to use exactly the mode you want. Here is how it works. ⟨ESC⟩"!" activates the Master Select in this format:

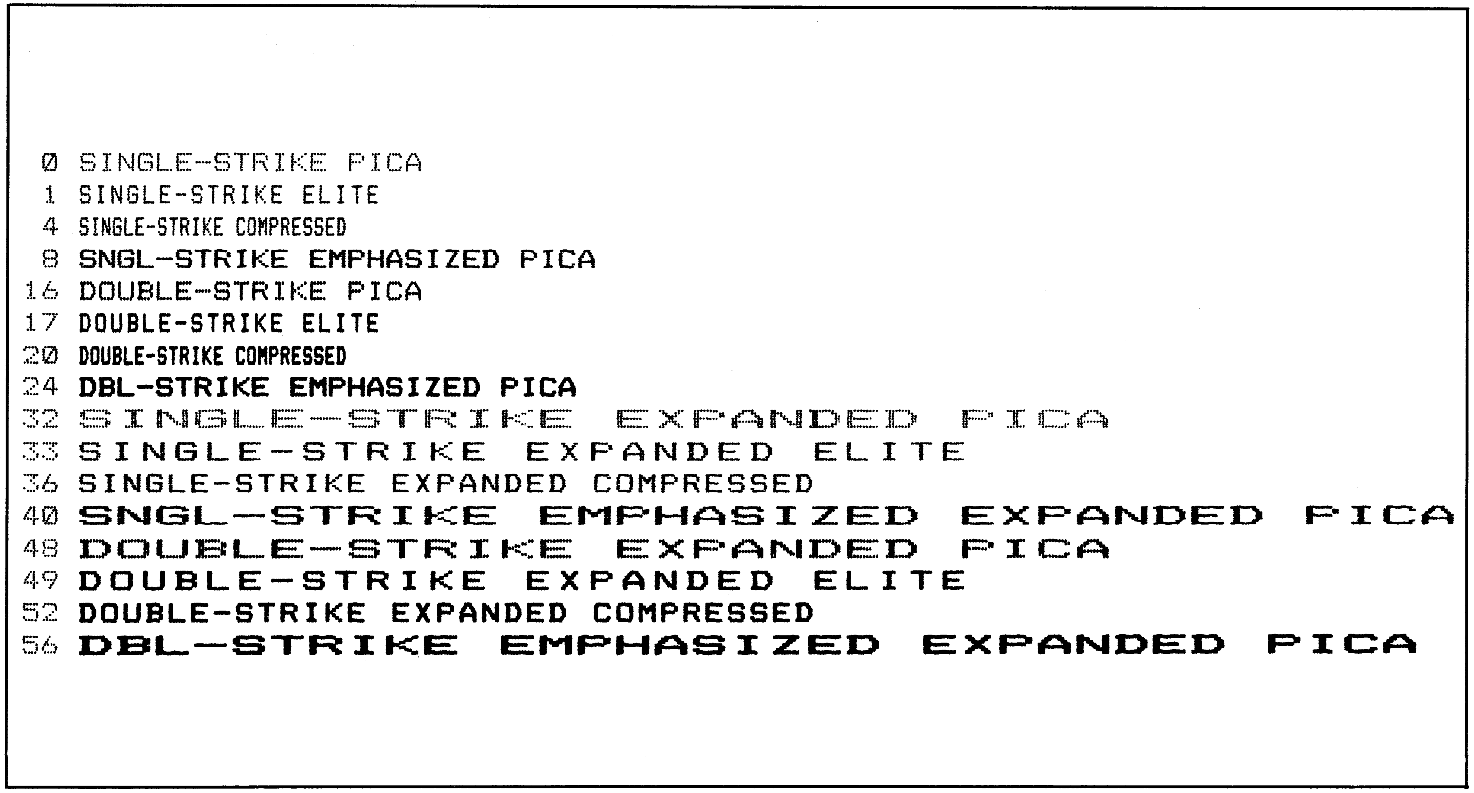
```
    LPRINT CHR$(27)"!"CHR$(n)
```

The n can be any number from 0 through 255, although not every number produces a different combination.

The program shown as Figure 5-1 prints out all of the unique print modes available with the ⟨ESC⟩"!".

```
NEW
2Ø Y$(1)="SINGLE-STRIKE ":Y$(2)="SNGL-STRIKE EMPHASIZED "
3Ø Y$(3)="DOUBLE-STRIKE ":Y$(4)="DBL-STRIKE EMPHASIZED "
4Ø Z$(1)="PICA ":Z$(2)="ELITE ": Z$(3)="COMPRESSED "
5Ø FOR X=1 TO 2
6Ø FOR Y=1 TO 4
7Ø FOR Z=1 TO 3
8Ø READ N: IF N<Ø THEN 13Ø
9Ø LPRINT CHR$(27)"!"CHR$(Ø);:IF N<9 THEN LPRINT " ";
95 ' OK to substitute CHR$(2) for CHR$(Ø)
1ØØ LPRINT N; CHR$(27)"!"CHR$(N);
11Ø LPRINT Y$(Y);:IF X=2 THEN LPRINT "EXPANDED ";
12Ø LPRINT Z$(Z)
13Ø NEXT Z: NEXT Y: NEXT X
14Ø LPRINT CHR$(27)"@"
15Ø DATA Ø,1,4,8,-1,-1,16,17,2Ø,24,-1,-1
16Ø DATA 32,33,36,4Ø,-1,-1,48,49,52,56,-1,-1
```

Figure 5-1. Program for Master Select.



```
 Ø SINGLE-STRIKE PICA
 1 SINGLE-STRIKE ELITE
 4 SINGLE-STRIKE COMPRESSED
 8 SNGL-STRIKE EMPHASIZED PICA
16 DOUBLE-STRIKE PICA
17 DOUBLE-STRIKE ELITE
2Ø DOUBLE-STRIKE COMPRESSED
24 DBL-STRIKE EMPHASIZED PICA
32 S I N G L E - S T R I K E   E X P A N D E D   P I C A
33 S I N G L E - S T R I K E   E X P A N D E D   E L I T E
36 SINGLE-STRIKE EXPANDED COMPRESSED
4Ø S N G L - S T R I K E   E M P H A S I Z E D   E X P A N D E D   P I C A
48 D O U B L E - S T R I K E   E X P A N D E D   P I C A
49 D O U B L E - S T R I K E   E X P A N D E D   E L I T E
52 DOUBLE-STRIKE EXPANDED COMPRESSED
56 D B L - S T R I K E   E M P H A S I Z E D   E X P A N D E D   P I C A
```

Figure 5-2. Unique print modes with ⟨ESC⟩.

How about that for an impressive selection! Give your FX-80 a pat on the back (top?).

Notice that the program stores the desired codes in DATA statements, and that the READ statement in line 10 transfers the codes to the variable N. READ and DATA statements are used again in Chapter 12.

You may use your printout (or the one shown as Figure 5-2) as a handy reference sheet, or you may prefer to see it in chart form. The complete chart for all 256 numbers is shown as Table 5-1.

You can also see from the printout and the chart that the Master Select feature affects only these modes:

Elite
Compressed
Expanded
Double-Strike
Emphasized

Each of the five modes above corresponds to one of the eight data lines (bits) in the computer. Three of the bits are unused. These three bits don't affect the outcome, but they do increase the number of possible codes for each mode combination. For example, the combination of Double-Strike and Elite can be entered with either:

```
    CHR$(27)"!"CHR$(17)
```

or

```
    CHR$(27)"!"CHR$(19)
```

(Et cetera, et cetera, et cetera.) Table 5-1 shows that any of the mode combinations can be selected by more than one number.

Table 5-1. Master Select and the 256 ASCII Codes.

PITCH	WEIGHT			
	Single-Strike	Emphasized	Double-Strike	Double-Strike Emphasized
Pica (10 cpi)	0 2 64 66 128 130 192 194	8 10 12 14 72 74 76 78 136 138 140 142 200 204 206 212	16 18 80 82 144 146 208 210	24 26 28 30 88 90 92 94 152 154 156 158 216 218 220 222
Elite (12 cpi)	1 3 5 8 9 11 13 15 65 67 69 71 73 75 77 79 129 131 133 135 137 139 141 143 193 195 197 199 201 203 205 207	Elite takes precedence over Emphasized.	17 19 21 23 25 27 29 31 81 83 85 87 89 91 93 95 145 147 149 151 153 155 157 159 209 211 213 215 217 219 221 223	Elite takes precedence over Emphasized.
Compressed (17.16 cpi)	4 6 68 70 132 134 196 198	Emphasized takes precedence over Compressed.	20 22 84 86 148 150 212 214	Emphasized takes precedence over Compressed.
Expanded (5 cpi)	32 34 96 98 160 162 224 226	40 42 44 46 104 108 110 166 168 172 174 232 234 236 238	48 50 112 114 176 178 240 242	56 58 60 62 120 122 124 126 136 137 188 190 248 250 252 254
Expanded Elite (6 cpi)	33 35 37 39 41 43 45 47 97 99 101 103 105 107 109 111 161 163 165 167 169 171 173 175 225 227 229 231 233 235 237 239	Elite takes precedence over Emphasized.	49 51 53 55 57 59 61 63 113 115 117 119 121 123 125 127 177 179 181 183 185 187 189 191 241 243 245 247 249 251 253 255	Elite takes precedence over Emphasized.
Expanded Compressed (8.58 cpi)	56 38 100 102 164 166 228 234	Emphasized takes precedence over Compressed.	52 54 116 118 180 182 244 246	Emphasized takes precedence over Compressed.

The inner workings of the Master Select

Some serious users may want to know in detāil how Table 5-1 is derived. This section is for them! Everyone else skip to "Unique modes."

As mentioned above, each mode used by the Master Select command corresponds to one of the computer's bits. These bits have the decimal values shown in Table 5-2.

Table 5-2. Master Select Bits, Codes, and Modes

Bit Number	Decimal Code	Mode Name
1	1	Elite
2	2	—
3	4	Compressed
4	8	Emphasized
5	16	Double-Strike
6	32	Expanded
7	64	—
8	128	—

Each of these five modes is activated when its corresponding bit (shown above) is on, otherwise it is inactive. For example, adding the decimal codes 16 and 1 combines Double-Strike and Elite. Therefore the sequence ⟨ESC⟩"!"CHR$(17) prints that combination. Because the decimal code 2 (bit 2) is not used, adding it to the sequence has no effect. The same holds true with 64 and 128.

The addition of any of the three unused codes to the valid codes simply adds a new number to the mode without changing the mode.

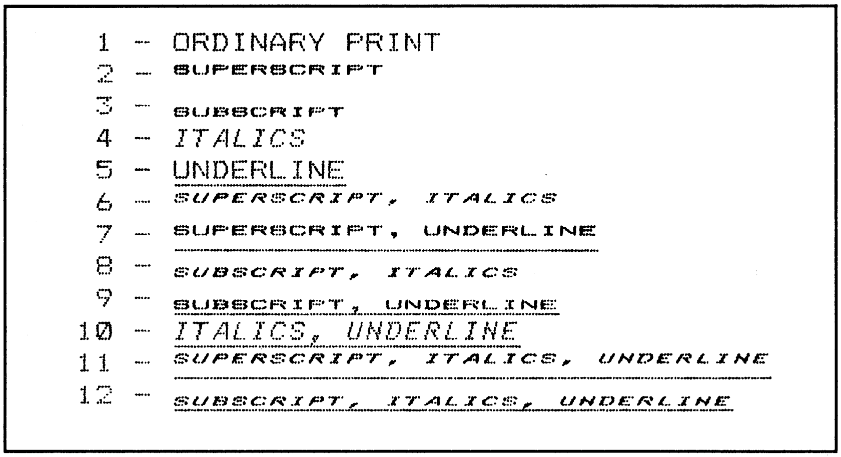
Bits and their corresponding values are covered in Chapter 11.

Unique modes

As you can see from Table 5-1, not every number between 0 and 255 evokes a unique print mode combination. In fact, close inspection of the chart reveals only 16 unique varieties. How? Well, the left column names the 6 available densities (3 pitches and 3 weights). The top row names 4 print qualities. Multiplying 4 times 6 gives us 24 possible combinations. Eight of those, however, involve a conflict of modes—that is, one mode takes precedence over another. That reduces the number of unique modes to 16, half of which are Single-Strike.

If 16 different modes is not enough, then how does 128 "strike" you? By using combinations of the 4 modes from the beginning of this chapter with the 16 Master Select modes, we can drive the number of unique print modes up to 128! (That's right, you get MODE for your money with Epson.)

For you nonbelievers, we'll spell it out in black and white. Start with these four print modes: Superscript, Subscript, Italic, and Underline. Figure 5-3 illustrates the 11 different ways they can be combined:



```
1  - ORDINARY PRINT
2  - SUPERSCRIPT
3  - SUBSCRIPT
4  - ITALICS
5  - UNDERLINE
6  - SUPERSCRIPT, ITALICS
7  - SUPERSCRIPT, UNDERLINE
8  - SUBSCRIPT, ITALICS
9  - SUBSCRIPT, UNDERLINE
1Ø - ITALICS, UNDERLINE
11 - SUPERSCRIPT, ITALICS, UNDERLINE
12 - SUBSCRIPT, ITALICS, UNDERLINE
```

Figure 5-3. Some mode combinations.

If we add the default Pica Mode, there are 12 different combinations.

Now it's time to dig out the calculators. Four of these 12 combinations (the ones that don't involve Super- or Subscript) can be combined with any of the 16 modes from the Master Select. This gives us 4 times 16, or 64 combinations. The other 8 combinations that involve Super- and Subscript Modes can be combined with the 8 Single-Strike Master Select modes. Eight times 8 gives another 64 combinations. Add these to the first 64, and we get a total of 128 unique print modes on the FX-80 printer!!

The 128 modes are displayed in the print sample at the beginning of the manual.

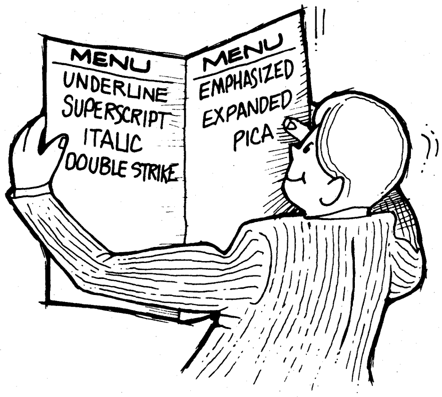
So how would someone print a combination like

UNDERLINED SUPERSCRIPT ITALIC DOUBLE-STRIKE EMPHASIZED EXPANDED PICA?

No problem. With the current program, just manually set one of the 11 combinations of the first 4 modes, and let the Master Print Mode Select take care of the rest. Try adding this line to your program to produce the printout shown as Figure 5-4.

```
10 LPRINT CHR$(27)"-1";CHR$(27)"S0";CHR$(27)"4";
```

(Hmmmm. Nice menu. I'll have the Epson combination special to go, with extra sausage, mushroom, and pepperoni—hold the onions.)



Switching modes with Master Select

Mode selection by number makes it very easy to choose combinations. For example, the combination Double-Strike Emphasized Expanded Pica normally requires:

```
LPRINT
   CHR$(27)"G";CHR$(27)"E";CHR$(27)"W1"
```

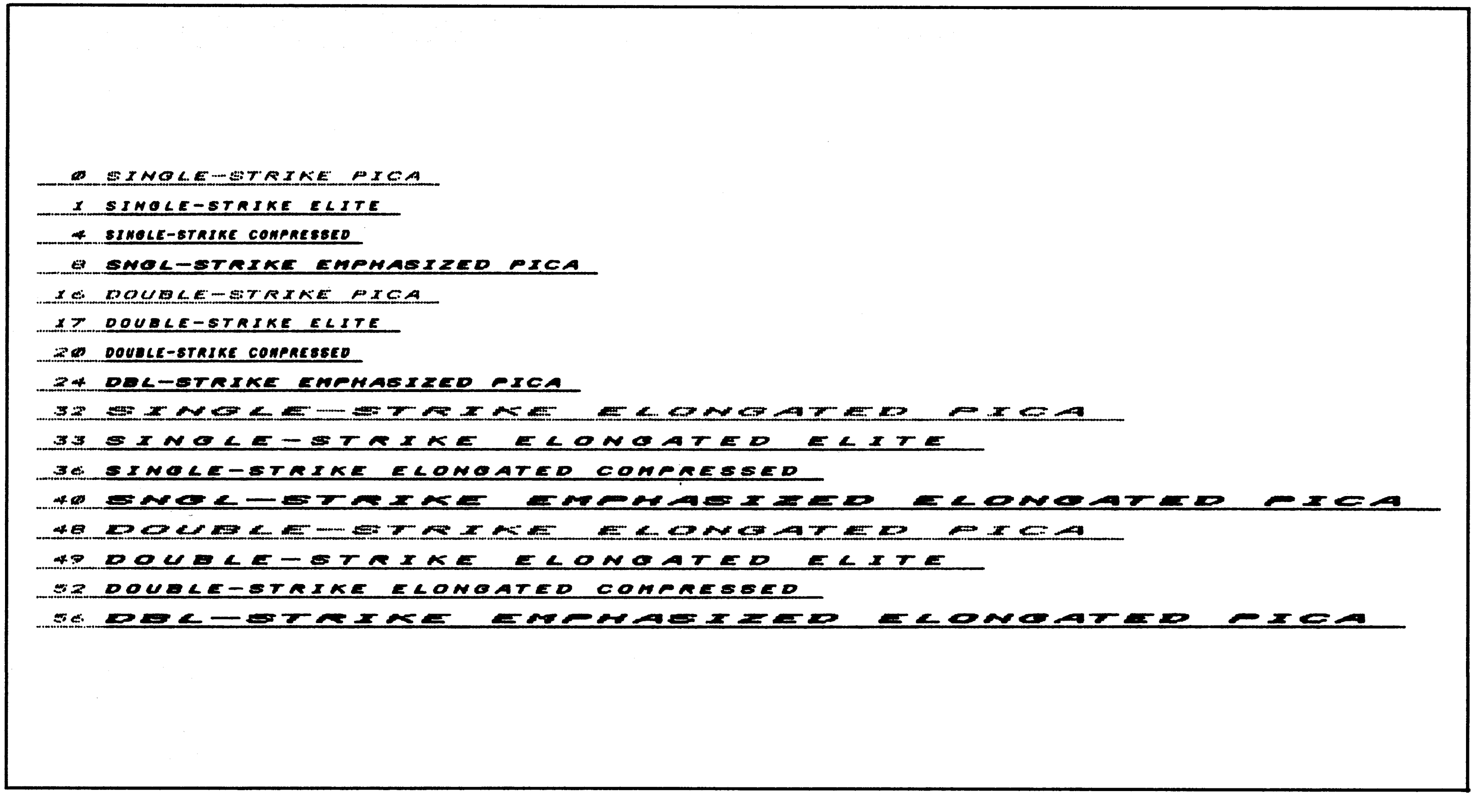
But with Master Select, it is considerably shorter:

```
LPRINT CHR$(27)"!"CHR$(56)
```

or shorter still:

```
LPRINT CHR$(27)"!8" ' Yes, CHR$(56)="8"
```

This latter format is the ultimate in simplicity. In fact, we use this format for Table 5-3, which is a shorter form of Table 5-1. This one shows first a single numeral for each of the unique modes and then a convenient ASCII symbol. You choose which of the pair to use.



```
 Ø SINGLE-STRIKE PICA
 1 SINGLE-STRIKE ELITE
 4 SINGLE-STRIKE COMPRESSED
 8 SNGL-STRIKE EMPHASIZED PICA
16 DOUBLE-STRIKE PICA
17 DOUBLE-STRIKE ELITE
2Ø DOUBLE-STRIKE COMPRESSED
24 DBL-STRIKE EMPHASIZED PICA
32 SINGLE-STRIKE ELONGATED PICA
33 SINGLE-STRIKE ELONGATED ELITE
36 SINGLE-STRIKE ELONGATED COMPRESSED
4Ø SNGL-STRIKE EMPHASIZED ELONGATED PICA
48 DOUBLE-STRIKE ELONGATED PICA
49 DOUBLE-STRIKE ELONGATED ELITE
52 DOUBLE-STRIKE ELONGATED COMPRESSED
56 DBL-STRIKE EMPHASIZED ELONGATED PICA
```

Figure 5-4. Italic print menu.

Table 5-3. Master Select Quick Reference Chart

PITCH	WEIGHT							
	Single-Strike		Emphasized		Double-Strike		Double-Strike Emphasized	
	Num.	ASCII	Num.	ASCII	Num.	ASCII	Num.	ASCII
Pica	0	@	8	H	16	P	24	X
Elite	1	A	←		17	Q	←	
Compressed	4	D	↑		20	T	↑	
Expanded Pica	32	ƀ	40	*	48	0	56	8
Expanded Elite	33	!	←		49	1	←	
Expanded Compressed	36	$	↑		52	4	↑	

Note: Arrows point to the modes which take precedence.

Use either of these formats:

```
LPRINT CHR$(27)"!"CHR$(Ø)
```

or

```
LPRINT CHR$(27)"!@"
```

Another advantage of using the Master Select is that you don't have to turn each different print type off when you want to change to a different combination. Using this method, the current combination is completely reset whenever a new one is selected, no matter how many print modes that involves.

The Master Select as a subroutine

Once you get used to the Master Select, using it will become a habit. Tucking it away as a subroutine may save some typing:

```
NEW
10 N= 4: GOSUB 70: LPRINT
   CHR$(27)"S1THE";CHR$(27)"T"
20 N=17: GOSUB 70: LPRINT CHR$(27)"-1FX-80"
30 N= 8: GOSUB 70: LPRINT CHR$(27)"-0HAS EVER"
40 N=49: GOSUB 70: LPRINT "EXPANDING"
50 N=56: GOSUB 70: LPRINT CHR$(27)"4POSSIBILITIES"
60 LPRINT CHR$(27)"@": STOP
70 LPRINT CHR$(27)"!"CHR$(N);: RETURN
```

```
THE
FX-80
HAS EVER
E X P A N D I N G
P O S S I B I L I T I E S
```

The Script, Underline, and Italic Modes are selected with ESCape codes. All other mode changes are created by sending a number to the subroutine (as shown above in line 70). Just select the desired number from Table 5-3.

Summary

⟨ESC⟩"−1"	Underline Mode ON
⟨ESC⟩"−0"	Underline Mode OFF
⟨ESC⟩"S1"	Subscript Mode ON
⟨ESC⟩"S0"	Superscript Mode ON
⟨ESC⟩"T"	Script Modes OFF
⟨ESC⟩"4"	Italic Mode ON
⟨ESC⟩"5"	Italic Mode OFF
⟨ESC⟩"!"CHR$(n)	Selects a Master Print Mode

Chapter 6
Special Print Features

In this chapter we're going to put the icing on our printout cake. You'll discover several new features that will enhance your control over the printer. So dig right in . . .

Special Characters

Although it is not truly a character, the backspace is discussed here along with international and some other special characters. That is because backspace is often used to create a special character or effect.

The backspace function

The backspace function is handy for making overstrikes. It moves the print head backward one character so that you can print two characters in the same print position. The backspace function can be combined with any of the print modes, which opens some interesting possibilities.

Backspace is activated by typing a CHR$(8) . . . that's it! To see what happens when you mix backspacing with different modes, enter:

```
NEW
10 FOR J=0 TO 1
20 LPRINT CHR$(27)"W"CHR$(J); ' Expanded Mode (J=1)
25 ' TRS Model I see Appendix I
30 LPRINT "BACKSPACE";CHR$(15); ' 15 enters
   Compressed
40 FOR X=1 TO 15: LPRINT CHR$(8);: NEXT X
45 ' Prints 15 backspaces
50 LPRINT CHR$(18);"BACKSPACE" ' 18 cancels
   Compressed
60 NEXT J: LPRINT CHR$(27)"@" ' Master Reset Code
```

```
BACKSPACE
BACKSPACE
```

The 15 backspaces in line 40 are printed in Compressed Mode. The difference in densities makes the second printing of Backspace be offset from the first. You could spend all day mixing densities to get different effects.

In the next program, the offset is not as dramatic. Change what you have to:

```
3Ø LPRINT "BACKSPACES";CHR$(15); ' Adds S
4Ø FOR X=1 TO 17: LPRINT CHR$(8);: NEXT X
45 ' Changes 15 to 17
5Ø LPRINT CHR$(18);"BACKSPACES" ' Adds S
```

```
BACKSPACES
BACKSPACES
```

More typically, backspace is used for single-character overstrikes. Here are a few examples that create some useful mathematical symbols:

```
NEW
1Ø LPRINT "=";CHR$(8);"/" ' (Not equal)
```

```
≠
```

The backspace moves the print head back over the equal sign so that the slash can be printed on top of it.

```
1Ø LPRINT CHR$(27)"SØ";"+";CHR$(8); ' (Plus/minus)
2Ø LPRINT CHR$(27)"T";CHR$(27)"S1-"
3Ø LPRINT CHR$(27)"@"
```

```
±
```

How about that, and only three lines too! Next . . .

```
    1Ø LPRINT CHR$(126);CHR$(8); ' (Approximately equal)
    2Ø LPRINT CHR$(27)"J"CHR$(11);CHR$(126)
    3Ø LPRINT CHR$(27)"@"
```

This program prints CHR$(126) (in English this is a little-used character: it's a Spanish accent mark which is called a tilde). The print head is backspaced by CHR$(8), then a partial line feed is forced by ⟨ESC⟩ "J"CHR$(11) (line feeds are covered in Chapter 7). The second tilde is printed below the first one to form the desired figure.

Some special characters

The tilde character raises an interesting point. There are several characters stored in the FX-80 that can't be generated from many computer keyboards (see Appendix A). In particular, the characters stored for the ASCII codes between 91 and 96, and between 123 and 126, and for their italic counterparts are uncommon. Table 6-1 lists these characters:

Table 6-1. Some Special Characters.

91	[	Left bracket
92	\	Back slash
93	]	Right bracket
94	^	Caret
95	_	Underline
96	`	Accent grave
123	{	Left brace
124	¦	Flat colon
125	}	Right brace
126	~	Tilde

But that's not all. The FX-80 has more special characters hidden in its inner recesses. In fact, it can print characters from countries all over the world.

International characters or . . . Around the world in FX-80 days

Your FX-80 has the makings of quite a world traveler. Hidden in the ROM are letters and special characters used in nine different countries. These international characters can be accessed with:

```
LPRINT CHR$(27)"R"CHR$(n);
```

where n is a number from 0 to 8. The ⟨ESC⟩"R" sequence selects one of the nine countries:

0	USA	3	England	6	Italy
1	France	4	Denmark	7	Spain
2	Germany	5	Sweden	8	Japan

Choosing a new international character set does not give you a completely new set of 256 characters. There are 64 international characters stored in the ROM; 32 in roman and 32 in italic typeface. But they are stored as codes 0 to 31 and 128 to 159, which are not normally accessible (see Chapter 17).

The ⟨ESC⟩"R" command makes these symbols available to us—12 characters at a time—as the following ASCII numbers:

35, 36, 64, 91, 92, 93, 94, 96, 123, 124, 125, 126

The program shown as Figure 6-1 runs through the different international character sets. When you RUN it, the printout (shown here as Table 6-2) displays the symbols used in each set:

```
NEW
10 DIM ARRAY(12): LPRINT CHR$(27)"M"
20 LPRINT CHR$(27)"D";: FOR N = 15 TO 81 STEP 6
30 LPRINT CHR$(N);: NEXT N: LPRINT CHR$(1)
40 FOR X = 1 TO 12: READ ARRAY(X)
50 LPRINT CHR$(9);ARRAY(X);: NEXT X: LPRINT
55      ' Apple see Appendix I
60 DATA 35,36,64,91,92,93,94,96,123,124,125,126
70 FOR Y = 0 TO 8: LPRINT CHR$(27)"R"CHR$(Y);
75      ' Model I see Appendix I
80 READ C$: LPRINT C$;TAB(17);CHR$(14);
90 FOR X = 1 TO 12: LPRINT " "CHR$(ARRAY(X))" ";
100 NEXT X: LPRINT: NEXT Y
110 DATA "U.S.A.","FRANCE","GERMANY","ENGLAND"
120 DATA "DENMARK","SWEDEN","ITALY","SPAIN","JAPAN"
```

Figure 6-1. Program for international characters.

Table 6-2. International Characters in Roman Typeface.

	35	36	64	91	92	93	94	96	123	124	125	126
USA	#	$	@	[	\	]	^	`	{	|	}	~
France	#	$	à	°	ç	§	^	`	é	ù	è	¨
Germany	#	$	§	Ä	Ö	Ü	^	`	ä	ö	ü	ß
England	£	$	@	[	\	]	^	`	{	|	}	~
Denmark	#	$	@	Æ	Ø	Å	^	`	æ	ø	å	~
Sweden	#	¤	É	Ä	Ö	Å	Ü	é	ä	ö	å	ü
Italy	#	$	@	°	\	é	^	ù	à	ò	è	ì
Spain	₧	$	@	¡	Ñ	¿	^	`	¨	ñ	}	~
Japan	#	$	@	[	¥	]	^	`	{	|	}	~

This program provides an easy reference to the international characters; you'll probably want to keep the printout handy.

There are a few things in the program that haven't been covered yet; we'll cover them later. The important thing now is to understand how the ⟨ESC⟩"R" is used to gain access to the international characters.

The international characters can also be printed in italics. Make these changes:

```
80 READ C$: LPRINT C$;TAB(17);CHR$(14);CHR$(27)"4";
85 ' Adds CHR$(27)"4";
100 NEXT X: LPRINT CHR$(27)"5": NEXT Y
105 ' Adds CHR$(27)"5"
```

to get the result shown in Table 6-3:

Table 6-3. International Characters in Italic Typeface.

	35	36	64	91	92	93	94	96	123	124	125	126
USA	#	$	@	[	\	]	^	`	{	|	}	~
France	#	$	à	°	ç	§	^	`	é	ù	è	¨
Germany	#	$	§	Ä	Ö	Ü	^	`	ä	ö	ü	ß
England	£	$	@	[	\	]	^	`	{	|	}	~
Denmark	#	$	@	Æ	Ø	Å	^	`	æ	ø	å	~
Sweden	#	¤	É	Ä	Ö	Å	Ü	é	ä	ö	å	ü
Italy	#	$	@	°	\	é	^	ù	à	ò	è	ì
Spain	₧	$	@	¡	Ñ	¿	^	`	¨	ñ	}	~
Japan	#	$	@	[	¥	]	^	`	{	|	}	~

```
         PRINT IT:

                      ..... en Français,
                      ..... in Italiàno,
                      ..... in English,
                      ..... im Deütsch,
                      ..... pä Svensk,
                      ..... pà Dansk,
                      ..... en Español.
```

International switches

The factory setting of a default international character set—for the USA—is shown in line 1 of Table 6-4. You can change the default country by resetting some of the FX-80's internal switches. Three switches:

Switch 1-6
Switch 1-7
Switch 1-8

generate eight combinations. Turning one or more of the switches to OFF will change the default to one of the other sets. Table 6-4 shows the switch settings for eight of the countries.

Table 6-4. Switch Settings by Country.

Country	Switch 1-6	Switch 1-7	Switch 1-8
USA	On	On	On
France	On	On	Off
Germany	On	Off	On
England	On	Off	Off
Denmark	Off	On	On
Sweden	Off	On	Off
Italy	Off	Off	On
Spain	Off	Off	Off

That leaves one country unaccounted for: Japan. Japan's character set (which differs from the USA's set only in the character for ASCII 92) must be chosen through the use of the 〈ESC〉"R" statement.

The international character sets are pretty easy to use. And that's nice . . . in any language!

Special Speeds

You can control the speed of the FX-80's printing in a couple of ways. You can set it to print at half its usual speed. Or you can cause it to print one character at a time, immediately upon your input. Each of these capabilities is accessed as a mode.

Half-Speed Mode

Epson takes pride in the FX-80's great speed. It can print 160 characters in a second. (That's faster than you can sneeze.) But the FX-80 will go slower if you want it to: the Half-Speed Mode prints 80 characters per second (cps). The command sequence is:

```
LPRINT CHR$(27)"s1" ' Lower-case s
```

"1" turns half speed ON, and, as usual, the "0" version of the ON command turns it OFF. (Use CHR$(115)CHR$(1) if your computer can't send lower-case letters.)

Why would anyone want to make the printer work at half its normal speed?! Good question. The main advantage to half-speed printing is a quieter run (for those late night printing sessions).

But it goes slower yet . . .

Immediate Print Mode

Didn't think you got an electric typewriter along with the printer, did you? Well, the FX-80 has an Immediate Print Mode that does a pretty fair imitation of one. In this mode, all data is printed as soon as it is received by the printer, and the paper feeds back and forth so you can see the line that is currently being printed. This kind of instant feedback can be especially helpful in telecommunications.

Before we unveil the secrets of this feature, let's review the normal operation of the printer buffer. Enter this NEW program:

```
NEW
2Ø A$="": INPUT "TYPE A LETTER";A$
3Ø IF A$="" THEN 5Ø
4Ø LPRINT A$;: GOTO 2Ø
5Ø LPRINT
```

Now type several characters, and after each press the ⟨ENTER⟩ key. True to form, the printer just stuffs the characters into its buffer while it waits for a carriage- return code. To end this version of the program and print the contents of the buffer, press ⟨ENTER⟩ alone. Now add this line:

```
10 LPRINT CHR$(27)"i1" ' Lower-case i
```

And RUN the program as before. (Use CHR$(105)"1" in place of "i1" if necessary). This time the printer prints each string immediately. It also feeds the paper so that you can see what is being printed. Play with this for a while, then, when you are finished playing secretary with your new typewriter, press ⟨ENTER⟩ alone.

The exit code for Immediate Print Mode is:

```
LPRINT CHR$(27)"i0" ' Lower-case i
```

Solving the 7-Bit Dilemma

You'll find lots of references to 7-bit computers in this manual. The term "7-bit" is not an evaluation of the net worth of such computers, but rather a description of the number of data lines active in their printer interfaces. Unfortunately, the inability to turn the eighth data line on and off is a serious drawback when working with a printer. There is no way that 7-bit computers can fully exploit all the features of the printer. But the FX-80 does have a few tricks up its sleeve to give them a helping hand.

First, there *is* a way to permanently turn the high-order (eighth) bit on or off, even with 7-bit computers. There are three instructions that allow a 7-bit computer to control its high-order bit:

⟨ESC⟩">" turns the high-order bit ON
⟨ESC⟩"=" turns it OFF
⟨ESC⟩"#" returns it to normal

Here's an example of how this works:

```
NEW
10 LPRINT CHR$(27)">" ' Sets eighth bit
20 FOR X=65 TO 90: LPRINT CHR$(X);: NEXT X
30 LPRINT CHR$(27)"=" ' Supresses eighth bit
40 FOR X=193 TO 218: LPRINT CHR$(X);: NEXT X
50 LPRINT CHR$(27)"#" ' Returns eighth bit to normal
```

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Setting the eighth bit on in line 10 adds 128 to each of the numbers in line 20; thus it prints italic characters. The ⟨ESC⟩"=" in line 30 effectively subtracts 128 from each of the numbers in line 40, so the second line prints as the characters from 65 through 90. Line 50 returns all to normal. Note that the eighth bit stays either on or off until it is changed by one of the other two codes.

A second aid that the FX-80 offers to users of 7-bit computers involves the ESCape codes. They have been extended to operate with the high-order bit turned on. For example, the FX-80 understands both the normal

```
    ⟨ESC⟩"E"CHR$(27)CHR$(69)
```

and the high-order version,

```
    CHR$(27+128)CHR$(69+128)
```

to mean "enter Emphasized Mode" (adding 128 is the same as setting the high-order bit on).

So what good does that do you 7-bit'ers? Plenty. It means that if you turn the eighth bit on for use in one mode, you can thereafter switch to any mode that can handle numbers from 128 to 255 as parameters.

Summary

CHR$(8)	Causes a backspace
⟨ESC⟩"#"	Cancels ⟨ESC⟩"=" and ⟨ESC⟩">"
⟨ESC⟩"="	Eighth bit OFF
⟨ESC⟩">"	Eighth bit ON
⟨ESC⟩"R"CHR$(n)	Selects an international character set, where n = 0 through 8: 0 = USA (ASCII) 1 = France 2 = Germany 3 = England 4 = Denmark 5 = Sweden 6 = Italy 7 = Spain 8 = Japan

⟨ESC⟩"s1"	Half-Speed Mode ON
⟨ESC⟩"s0"	Half-Speed Mode OFF
⟨ESC⟩"i1"	Immediate Print Mode ON
⟨ESC⟩"i0"	Immediate Print Mode OFF

Chapter 7
Line Spacing and Line Feeds

Up to this point in the manual, there has been no discussion of the way the printer moves a page so that lines of text are not typed on top of each other. In this chapter you will learn how to change the distance that the paper moves; the movement is called a line feed, and the distance is called a line space. The ability to change line spacing is vital to printing graphics, as you will see in later chapters.

Line Spacing

Each line feed must move a specific distance, but that distance need not remain the same for every application (or even every line). In the next three sections, we cover the codes that you can use to change the size of a line space.

Changing line spacing

The standard distance of a line feed is 1/6th of an inch, which gives 6 lines per inch. This distance is equivalent to 12 rows of dots in the standard character matrix (Figure 7-1).

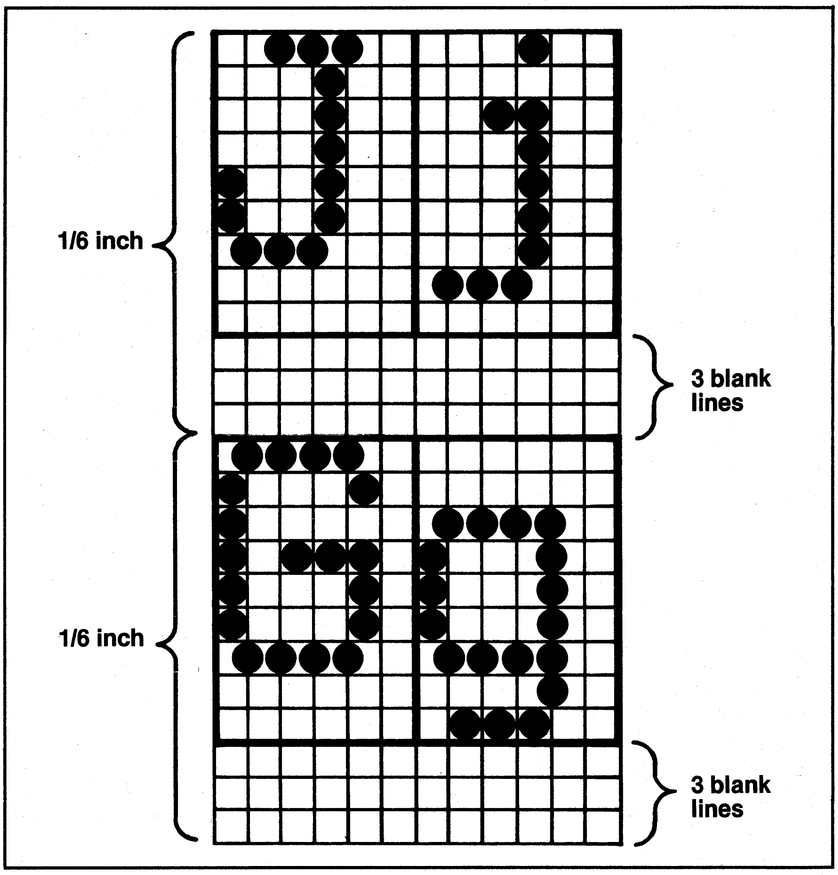


Figure 7-1. Standard character matrix and line feed.

Turn off the printer to clear any previous modes, and enter this NEW program:

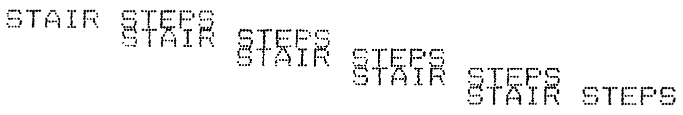
```
NEW
1Ø LPRINT CHR$(27)"Ø"
2Ø FOR X=Ø TO 4
3Ø LPRINT TAB(6*X)"STAIR STEPS"
4Ø NEXT X
5Ø LPRINT CHR$(27)"2"
```

```
STAIR STEPS
      STAIR STEPS
            STAIR STEPS
                  STAIR STEPS
                        STAIR STEPS
```

⟨ESC⟩ "0" changes the usual 12-dot line spacing to a handy variation: 9-dot (1/8″) spacing. (Nine-dot spacing is especially useful in the 9-pin Graphics Mode that we introduce in Chapter 13.) The ⟨ESC⟩ "2" in line 50 restores the normal 12-dot line spacing.

Another convenient line spacing is 7-dot (7/72″). To see this one, change line 10 to:

```
10 LPRINT CHR$(27)"1"
```



```
STAIR STEPS
      STAIR STEPS
            STAIR STEPS
                  STAIR STEPS
                        STAIR STEPS
```

Seven-dot spacing isn't particularly useful for text, but you'll use it extensively in the graphics programs presented later.

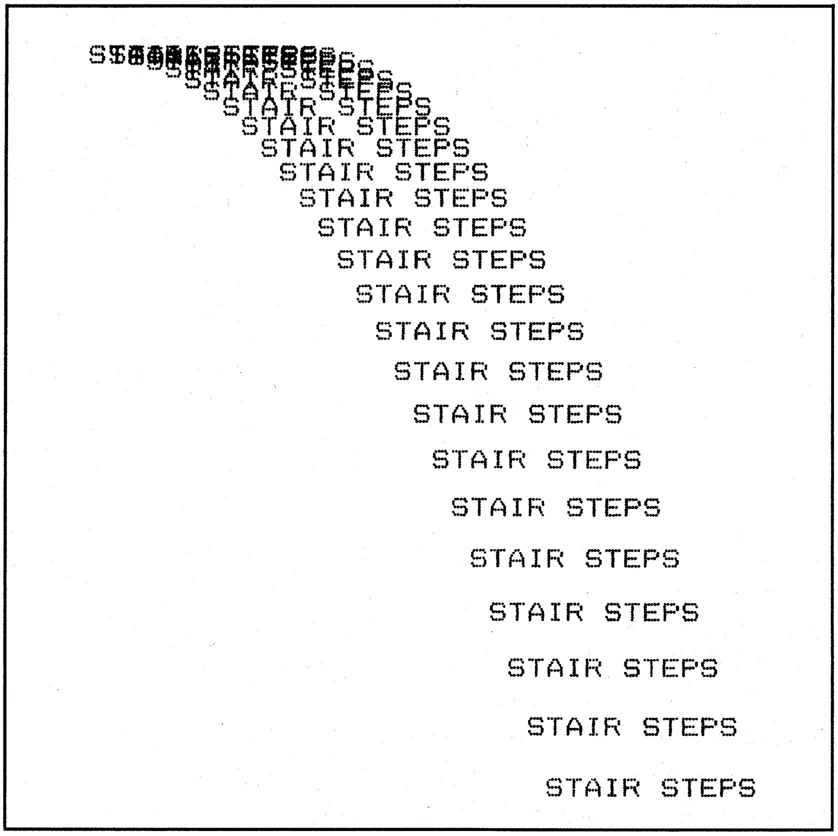
Variable line spacing (n/72″)

The three line spacing commands shown above are included mainly for convenience. The FX-80 allows variable line spacing to range from 0 to 85 dots' worth (0/72″ to 85/72″). Try this program on for size (and check your printout against Figure 7-2).

```
DELETE 10
```

then change lines 20 and 30:

```
20 FOR X=0 TO 24 ' Change 4 to 24
30 LPRINT TAB(X);"STAIR";CHR$(27)"A"CHR$(X);
   "STEPS"
```



```
                STAIR STEPS
                 STAIR STEPS
                  STAIR STEPS
                   STAIR STEPS
                    STAIR STEPS
                     STAIR STEPS
                      STAIR STEPS
                       STAIR STEPS
                        STAIR STEPS
                         STAIR STEPS
                          STAIR STEPS
                           STAIR STEPS
                            STAIR STEPS
                             STAIR STEPS
                              STAIR STEPS
                               STAIR STEPS
                                STAIR STEPS
```

Figure 7-2. Regular STAIR STEPS.

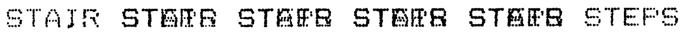
Quite a trick, huh? Figure 7-2 shows what you'll get if your computer does not pull any funny stuff with the control codes. (Using CHR$(X+128) in line 30 may help avoid some potential problems.)

In any case, the point of this program is to show how the ⟨ESC⟩ "A" sequence in line 30 works. ⟨ESC⟩"A"CHR$(n) changes the distance covered by a line feed to n dots (n/72nds of an inch). The ⟨ESC⟩"2" in line 50 restores the spacing to the usual 12 dots. Astute readers will observe that ⟨ESC⟩"2" has the same effect as ⟨ESC⟩ "A"CHR$(12)—which gives you two different ways to skin the same hide.

Notice the positioning of the ⟨ESC⟩"A" sequence in line 30—right between the strings "STAIR" and "STEPS". The printout shows that the line-feed command is not executed until the end of the print line. Otherwise the two strings would be printed on different lines. In fact, the ⟨ESC⟩"A" can be positioned anywhere on the print line with the same result.

Also notice that the lowest valid line spacing is 0 dots (instead of 1 as in the MX printers). This can be terrific for extensive overstrikes on a line. Make these changes to your program:

```
20 FOR X=0 TO 4 ' Change 24 to 4
30 LPRINT
   TAB(6*X);"STAIR";CHR$(27)"A"CHR$(0);"STEPS"
35 ' TRS Model I users see Appendix I
```



Microscopic line spacing (n/216″)

The reason for emphasizing 72nds of an inch is that the distance between the centers of any two pins on the print head is 1/72 of an inch. So these line spacing commands are simply telling the printer just how many dots to space between the lines.

There is also a way to space at smaller intervals than 72nds. Using an ⟨ESC⟩″3″ will set the spacing to increments of 216th of an inch. A 216th of an inch is one-third the distance between the pins of the print head (center to center). So essentially we can print a specific line one-third of a dot lower than the previous line. In fact, that's exactly how the Double-Strike Mode operates. One word of caution. As you can imagine, total accuracy is not guaranteed for such fine settings as 1/216- and 2/216-inch.

This finer line spacing is accessed in much the same way as the variable line spacing that we showed above with ⟨ESC⟩″A″. The format is ⟨ESC⟩″3″CHR$(n), where n can range from 0 to 255. Here's an example using 1/216 of an inch five times. Change lines 10 and 30 to read:

```
10 LPRINT CHR$(27)"3"CHR$(1)
30 LPRINT "ABCDEF"
```

```
ABCDEF
```

It gives very heavy looking letters. To carry this idea to an extreme, increase the loop to 10 or 15. The ability to adjust line spacing in increments of 1/216-inch gives us tremendous control in our vertical formatting.

In fact, you can use this line spacing to fine-tune your graphics printouts. If the 7-dot line spacing leaves gaps in the figures, just change it to 6 2/3 dots (20/216ths) with ⟨ESC⟩"3". We'll use this technique in Chapter 14.

Line Feeds

Besides being able to change the size of line spaces, you can change other aspects of a line feed. You can send it immediately and for one line only, or you can send it as above, as a continuous feature. And you can send it forward or backward.

One-time, immediate line feed (n/216")

The subtitle above may sound like a questionable deal at a used car lot, but the feature is superb. The FX-80 has a special line feed that executes once, then reverts back to the size of the previous line feed. And that's not all—it is executed immediately rather than at the end of the line like the ⟨ESC⟩"A" and ⟨ESC⟩"3". The format is:

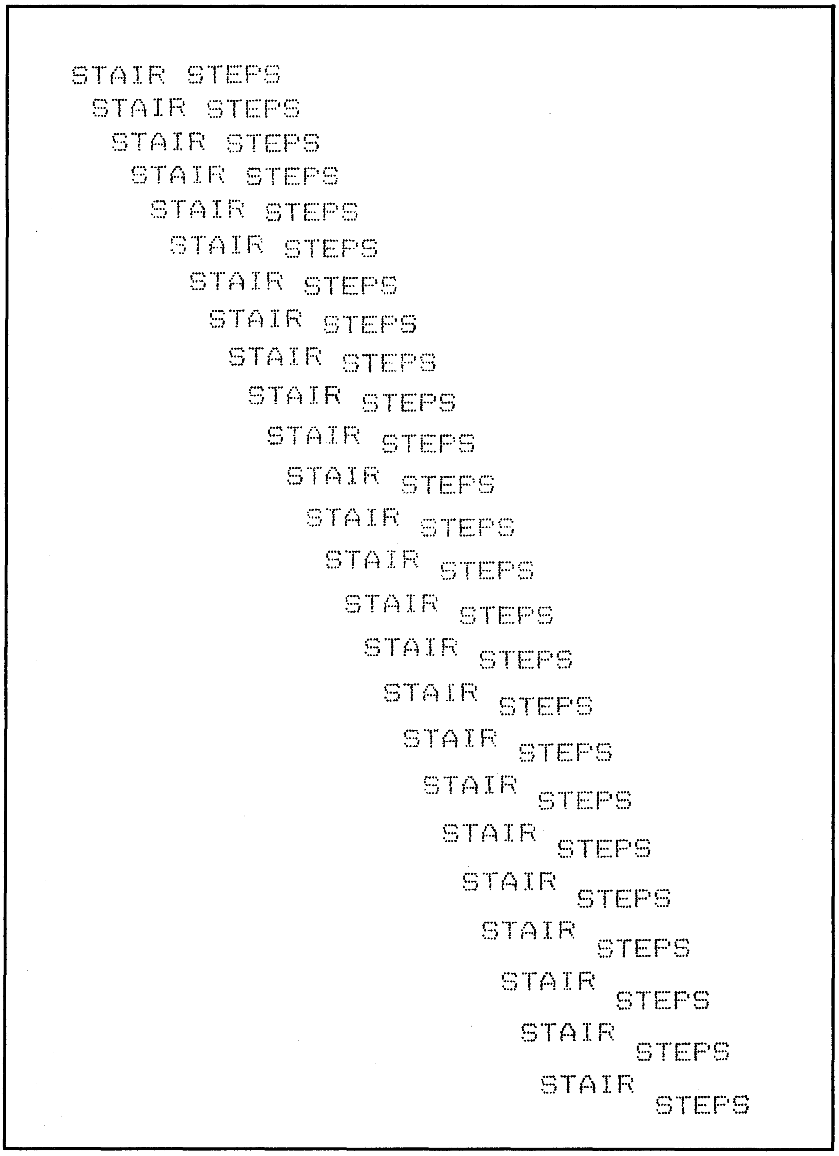
```
LPRINT CHR$(27)"J"CHR$(n)
```

where n represents a distance of between 0 and 255/216ths of an inch. You've got to see this one in action. Put it to the same test you gave the ⟨ESC⟩"A" command:

```
DELETE 1Ø
```

and change:

```
2Ø FOR X=Ø TO 24
3Ø LPRINT TAB(X);"STAIR";CHR$(27)"J"CHR$(X);
   "STEPS"
```



```
STAIR STEPS
 STAIR STEPS
  STAIR STEPS
   STAIR STEPS
    STAIR STEPS
     STAIR STEPS
      STAIR STEPS
       STAIR STEPS
        STAIR STEPS
         STAIR STEPS
          STAIR STEPS
           STAIR STEPS
            STAIR STEPS
             STAIR STEPS
              STAIR STEPS
               STAIR STEPS
                STAIR STEPS
                 STAIR STEPS
                  STAIR STEPS
                   STAIR STEPS
                    STAIR STEPS
                     STAIR STEPS
                      STAIR STEPS
                       STAIR STEPS
                        STAIR STEPS
```

Figure 7-3. Irregular STAIR STEPS.

Don't worry if your computer hiccups on this program.

Compare Figure 7-3 to Figure 7-2. In the ⟨ESC⟩"A" program, STAIR and STEPS are printed on the same line, and the variable line feed is performed at the end of the line. With ⟨ESC⟩"J", a line feed without a carriage return is performed *between* the strings (see how STEPS begins to sag below STAIR?) and a standard 12-dot line feed is made at the end of each line.

So the ⟨ESC⟩"J" does not require a "shut-off" code like the other line-feed control codes. It is executed once, then forgotten. As a bonus, it also does its work without returning the print head to the left margin.

Here's a program that will show off a few of the things you've learned so far (see if you can figure out what the program will do before you print it):

```
NEW
1Ø LPRINT CHR$(14)"X"CHR$(2Ø);
2Ø LPRINT CHR$(15);CHR$(27)"j"CHR$(2Ø)"2 ";CHR$(18);
3Ø LPRINT CHR$(14);CHR$(27)"J"CHR$(2Ø)"Y";CHR$(2Ø);
4Ø LPRINT CHR$(15);CHR$(27)"J"CHR$(2Ø)"3"
5Ø LPRINT CHR$(27)"@";"SPECIAL SUPER AND SUBSCRIPTS"
```

```
   2
X    Y
       3
SPECIAL SUPER AND SUBSCRIPTS
```

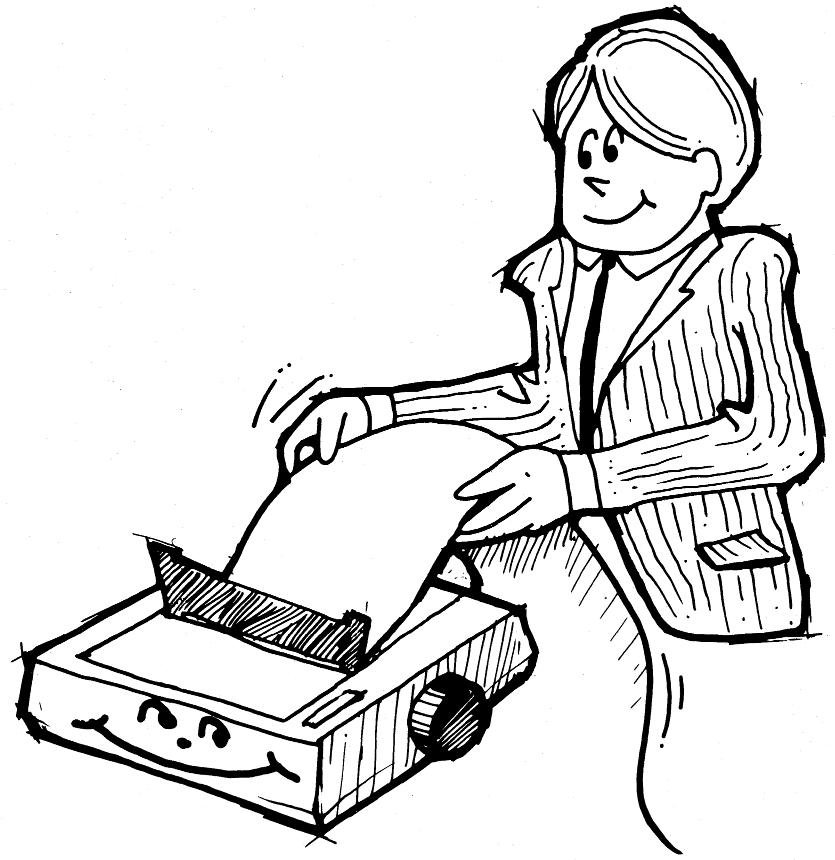
The first line prints an Expanded X. Line 20 (did it stump you?) does a reverse line feed (yes, the paper actually feeds through backward!) of 20/216-inch, then prints a 2 in Compressed pitch. This reverse feed sets the number 2 well above the normal position for superscripted characters. Line 30 prints an Expanded Y (back on the same line as the X) after a 20/216-inch forward paper feed. Line 40 does one more forward feed to print a subscripted 3. And line 50 returns all to normal.

Reverse feed!?! (n/216")

That's right, the FX-80 has reverse feed! ⟨ESC⟩"j" activates the command, and it operates just like the ⟨ESC⟩"J" feed: an immediate, one-time-only line feed without a carriage return. The difference is that the paper feeds in the opposite direction. If your computer cannot send lower-case letters to the printer, you will have to use the numeric value for "j" (CHR$(106)).

Caution: Don't attempt to use the reverse feed with mailing labels. They tend to peel off and get stuck underneath the paper roller. *Always* feed labels forward!

There are two other times to be careful with reverse feed: when the print head rests near the perforation, and when you are using the optional tractor feed.



Well, that's it! And you were probably wondering why anyone would want to change line spacing. You'll see even more uses for variable line spacing when we tackle graphics. Now would be a good time to take a break, before we start on the topic of forms control.

Summary

Table 7-1 sums up the possible line-feed commands and gives a sample of each.

Table 7-1. Line-Spacing Commands.

Code	Line Spacing	Example
〈ESC〉"0"	9/72" (1/8")	8 DOT ——— LINE ——— SPACING ———
〈ESC〉"1"	7/72"	7 DOT ——— LINE ——— SPACING ———
〈ESC〉"2"	12/72" (1/6")	12 DOT ——— LINE ——— SPACING ———
〈ESC〉"A"CHR$(n) n = 1 − 5	n/72"	0—— 1—— 2═══ 3═══ 4═══ 5═══
〈ESC〉"3"CHR$(n)	n/216"	0—— 3—— 6═══ 9═══ 12═══ 15═══
〈ESC〉"J"CHR$(n)	n/216"	0—— 3—— 6═══ 9═══ 12═══ 15═══
〈ESC〉"j"CHR$(n)	n/216"	0—— 3—— 6═══ 9═══ 12═══ 15═══

Chapter 8
Forms Control

Controlling Form Length

The FX-80 has several features that make it easy to print and use custom forms. With the FX-80, you can change margins, set horizontal and vertical tabs, and change the length of a page to match a preprinted form. In this chapter and the next, you will learn about these special forms control features.

Form feed

When the printer is not ON LINE, using the FF button (on the top of the printer) feeds the paper to the top of the next form. But what if you want to advance the paper during a program RUN without stopping everything, pushing the FF button, then starting again? Simple! There is an ASCII code (of course) that causes a form feed whenever you want one.

For the form feed command to be of any use, however, you must first tell the printer where the "Top Of Form" (TOF) is. In most cases you'll want the printer to use the first line below the paper perforation as the TOF line. To get this result, turn the printer off and feed the paper through (using the manual-feed knob) until the next perforation lines up with the top of the ribbon. Or line up the black paper guide with the hole above the perforation (as in Figure 8-1).

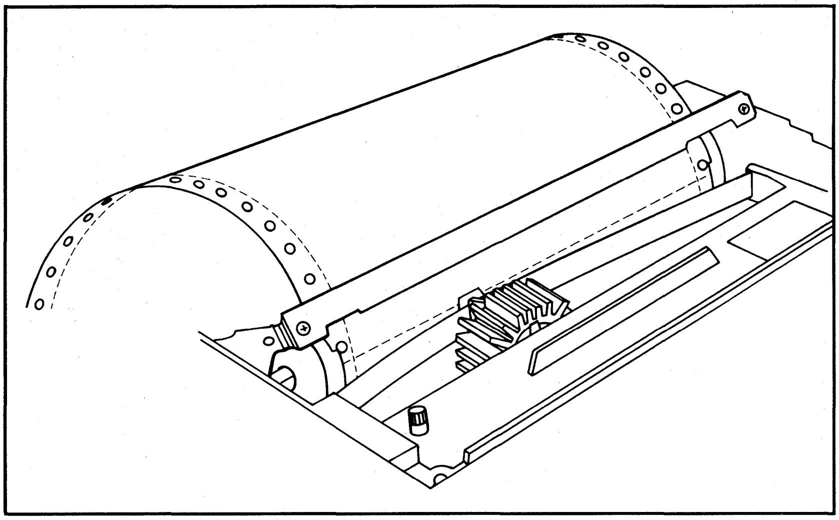


Figure 8-1. Setting the Top Of Form (TOF).

Turn on the printer. The FX-80 will now remember this position on the paper as the TOF. Each form feed will bring the paper back to the corresponding position on the next sheet. Try pressing the FF button (with the printer OFF LINE). Sure enough, right to the top of the form. Now do a few line feeds, then turn the printer back ON LINE, and type:

```
    LPRINT CHR$(12)
```

(ASCII 12 is the FF code.)

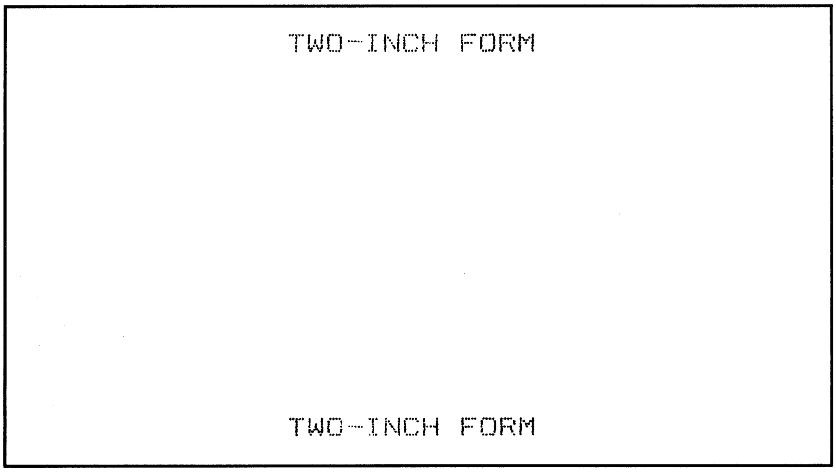
Note: Some users will be better off using CHR$(140).

CHR$(12) sends the paper to the top of the next form. It works just like the FF button with one exception. Because of the automatic line feed at the end of each LPRINT line, sending the FF code in BASIC causes an extra line feed. Don't worry—we can adjust for this extra line feed in our programs if necessary.

Not-so-standard forms

The printer already knows that a form is 11 inches long. But what if you decide to use a shorter form length, say 5 or 7 inches? The printer has no way of "sensing" the length of your paper. You must somehow tell the printer about your shorter (or longer) form. Here's one way to change the form length (Figure 8-2 shows the RUN). Enter:

```
NEW
1Ø LPRINT CHR$(27)"C"CHR$(Ø)CHR$(2)
15 ' TRS-8Ø Model I users see Appendix I
2Ø FOR X=1 TO 4
3Ø LPRINT CHR$(12)"TWO-INCH FORM"
35 ' TRS uses CHR$(14Ø)
4Ø NEXT X
5Ø LPRINT CHR$(27)"@"
```



```
                    TWO-INCH FORM




                    TWO-INCH FORM
```

Figure 8-2. Example of 2-inch spacing.

The ⟨ESC⟩"C" is the key. It changes the form length in the printer's memory and also resets the TOF to the current paper position (just as if you'd turned the printer off and on).

The printer gives us two ways to set the form length—by inches or by number of lines. The two formats are:

⟨ESC⟩"C"CHR$(0)CHR$(n)	Sets the form length to n *inches*
⟨ESC⟩"C"CHR$(n)	Sets the form length to n *lines*

CHR$(0) makes the difference between the two commands.

In the program above, line 10 uses the first format to set the form length to 2 inches; the allowable range is from 1 to 22 inches. The CHR$(12) in line 30 activates the form feed to send the form to the top of the next 2-inch form. (If your computer can't send a 12, try 140. Or see Appendix H.)

The second format (without CHR$(0)) sets the form length by counting the number of lines. The printer accepts form lengths from 1 to 127 lines. Check it by making the changes shown below and RUNning the program again; see if your printout matches Figure 8-3.

```
1Ø LPRINT CHR$(27)"C"CHR$(2)
3Ø LPRINT CHR$(12)"TWO-LINE FORM"
```

```
                    TWO-LINE FORM

                    TWO-LINE FORM

                    TWO-LINE FORM

                    TWO-LINE FORM
```

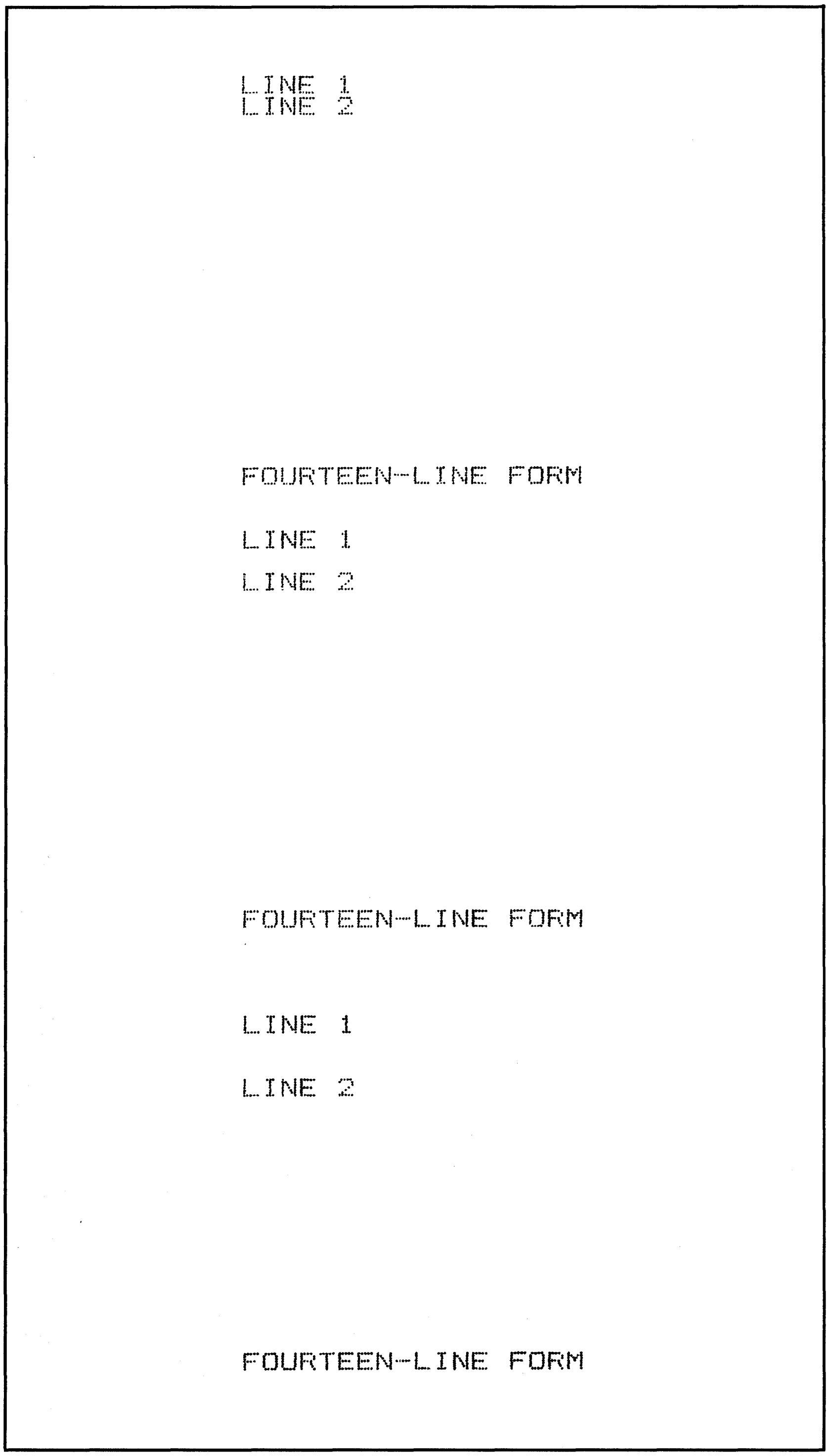
Figure 8-3. Example of 2-line spacing.

So what's the big difference when we can just convert lines to inches (or vice-versa)? Mainly it's a matter of convenience, but setting the form length by number of lines does give you a greater potential range. You want proof? Okay. Suppose the line spacing is set to its maximum 85/72nds of an inch. If the form length is also set to its maximum of 127 lines, the result is nearly 150 inches, a lot more than the maximum of 22 inches. Except for this difference, which form setting command you use is a matter of personal preference.

Absolute form length

In both cases the form length is set absolutely when the ⟨ESC⟩"C" is issued. That is, subsequent changes in line spacing have no effect on form length. Test this out with the following changes to the current program:

```
1Ø LPRINT CHR$(27)"C"CHR$(14)
22 LPRINT CHR$(27)"A"CHR$(8*X)
24 LPRINT "LINE 1"
26 LPRINT "LINE 2"
3Ø LPRINT CHR$(12)"FOURTEEN-LINE FORM"
```



```
LINE 1
LINE 2




FOURTEEN-LINE FORM

LINE 1

LINE 2




FOURTEEN-LINE FORM


LINE 1

LINE 2




FOURTEEN-LINE FORM
```

Figure 8-4. Example of 14-line spacing.

Once the form length is set (line 10), the line spacing changes have no effect on it (see Figure 8-4). The ⟨ESC⟩"@" in line 50 resets the form length to 11 inches and also sets the TOF to the current line.

Avoiding the Paper Perforation

The only sure way to avoid printing on the paper perforation is to set the top of form properly and make sure the printer is informed of the correct form length. Then appropriately placed form feeds will keep the printing off the perforation.

But there are some situations in which sending form feeds on every page is not convenient, and others in which we can't control the process with a BASIC program. A good example of the latter is LLISTing a long BASIC program. If your computer's LLIST command doesn't automatically skip perforations, you are pretty much out of luck with some printers. Fortunately, the FX-80 has a way to output a form feed on *every* page to clean up those listings. You might think of it as an "automatic Skip-Over-Perforation" (SOP) command.

The SOP solution

The Skip-Over-Perforation command can automatically skip lines at the bottom of each page to avoid printing right on the perforation, but you must first inform the printer of two things—1) where the correct top of form is, and 2) how long the current forms are. For standard 11-inch forms, just position the paper correctly before turning on the printer; for other form lengths, use the ⟨ESC⟩"C" command.

Okay, it's your turn to try this out. Make sure the perforation is even with the top of the ribbon (as in Figure 8-1), then type:

```
NEW
1Ø LPRINT CHR$(27)"N"CHR$(6);
2Ø FOR X=1 TO 7Ø
3Ø LPRINT "SIX-LINE SKIP...COMING UP";X
4Ø NEXT X
```

The ⟨ESC⟩"N"CHR$(6) in line 10 tells the printer to skip six lines (one inch) above the perforation to the top of the next page. The number of lines can be set to any value from 1 to 127 as long as it is less than the current form length (set by ⟨ESC⟩"C").

Note: No matter what your SOP setting is, the printer skips that many blank lines above the perforation to the new TOF. This means you will have all of your blank space at the bottom of each form. To get the same amount of blank space on the top and bottom of each page, you must set the TOF below the perforation. For the standard 6-line (1-inch) gap, setting the top of form to half an inch below the perforation will provide a blank half inch at the bottom of the page and another at the top of the next page (Figure 8-5).

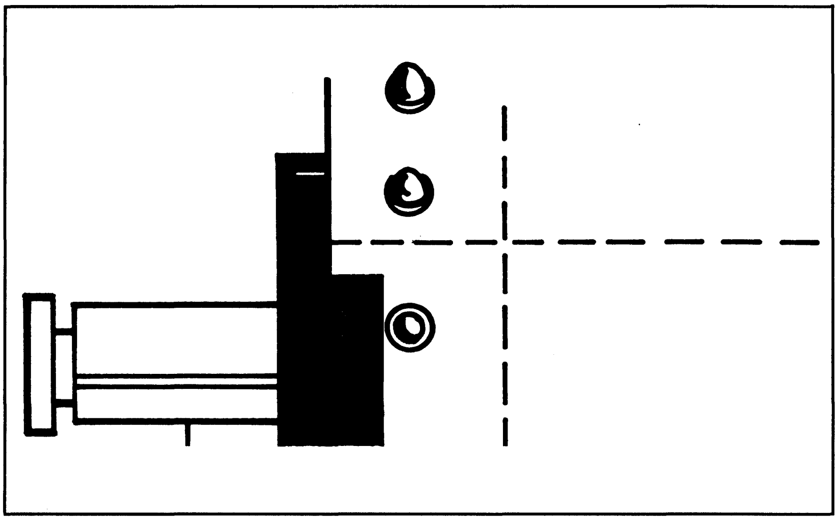


Figure 8-5. Centering the SOP gap.

The SOP feature can be turned off with ⟨ESC⟩"O" (that's the letter "oh"). It is also cancelled by the Master Reset Code or by setting a new length with ⟨ESC⟩"C".

Setting the SOP with a switch

The Skip-Over-Perforation feature can be very habit-forming. In fact, you may find the SOP so useful that you want it to become a default feature on your printer. Nothing could be easier.

Turn the printer *off* and locate the internal switches under the upper right vent (Chapter 1 has the details). Turn switch 2-3 to the *on* position.

Set the top of a new form about half an inch below the perforation (see Figure 8-5) and turn the printer back on. Then

```
    DELETE 1Ø
```

then change line 30:

```
30 LPRINT "AUTO SKIP COMING UP";X
```

and RUN the program. An automatic 6-line SOP will now be active every time you turn the printer on. And you can still turn it off with the ⟨ESC⟩"O" or change the setting with ⟨ESC⟩"N"CHR$(n).

Reset switch 2-3 before proceeding so you don't get any "surprises" with your future programs.

The Single-Sheet Trick

If you use single-sheet paper and run to the end of the form, the Paper-Out Sensor will automatically shut down the printing until you load another sheet and continue. To inform you, a bell beeps. The feature is designed to prevent the printer from accidentally printing on the roller. However, it also prevents you from printing on the last quarter of a page when you're using single-sheet paper. If you need to print on this part of the page, you can permanently override the Paper-Out Sensor by setting switch 1-3 to on.

The Sensor can also be temporarily disabled with software codes. Let's see how this works.

With the Paper-Out Sensor active (switch 1-3 off) and the bell alive (switch 2-2 on), load a single sheet of paper and RUN this program:

```
10 LPRINT
20 INPUT "HIT <ENTER> WHEN READY"; A
30 FOR X = 1 TO 60
40 LPRINT "PRINTING WILL STOP BEFORE THE PAPER RUNS
   OUT"
50 NEXT X
```

The printer quits about 3/4 of the way down the page. Good, the sensor is doing its job. Now we'll throw it a curve. Load in a new sheet of paper and change line 10 to:

```
10 LPRINT CHR$(27)"8"
```

For this RUN, the printer prints all 60 lines, ignoring the paper-out condition.

You'll want to use this code with caution, but it's nice to have when needed. To restore the Sensor to full power, either use LPRINT CHR$(27)"9" or turn the printer off.

Summary

The printer automatically considers each page to be 11 inches or 66 normal lines long. You can change the length of each form (in inches or lines) with an ESCape code. The Skip-Over-Perforation feature and the Paper-Out alarm can be turned on or off, either by the internal switches or by ASCII codes.

Note: Better reset any switches before going on to Chapter 9.

CHR$(12)	Produces a form feed
⟨ESC⟩"C"CHR$(n)	Sets form length in lines
⟨ESC⟩"C"CHR$(0)CHR$(n)	Sets form length in inches
⟨ESC⟩"N"CHR$(n)	Skip-Over-Perforation ON, set to n lines
⟨ESC⟩"O"	Automatic SOP OFF
⟨ESC⟩"8"	Paper-Out Sensor OFF
⟨ESC⟩"9"	Paper-Out Sensor ON

Chapter 9
Formatting
or . . . EPSON picks up the TAB

In this chapter we'll take a look at the way to set left and right margins, and to set and use both horizontal and vertical tabs.

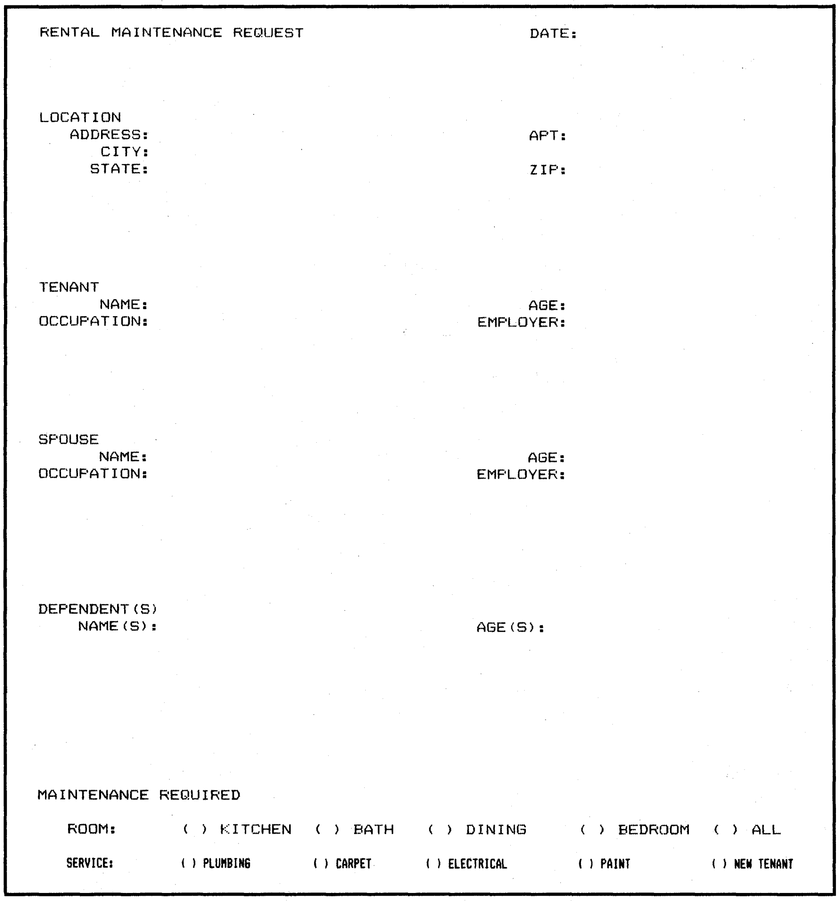
You're right if you think these sound like features on a typewriter. Most of them are. Both printer and typewriter have the ability to set margins and remember tab stops across the page. The major differences lie in the ease of changing settings on the printer and the printer's ability to store and use vertical tabs. In fact, that's where we start.

Vertical Tabs (VTabs)

The vertical tab feature of the FX-80 makes it easy to skip several lines on a page. It can be used to skip lines of text to make room for a diagram or chart, or to print out a form like the one shown as Figure 9-1. The sections of the form are separated by series of blank lines. A series of lines could be skipped by using LPRINTs in a loop, but that is an inefficient way to program when you have an FX-80 at your disposal. The FX-80 allows you to skip directly to the place where you want to start your next section by printing a single code.

First you have to tell the printer the line numbers that it is going to skip to (vertical tab stops). Enter:

```
NEW
2Ø LPRINTCHR$(27)"B"CHR$(3)CHR$(8)CHR$(18)
   CHR$(27)CHR$(37)CHR$(48)CHR$(1);
```



```
RENTAL MAINTENANCE REQUEST                        DATE:



LOCATION
   ADDRESS:                                       APT:
      CITY:
     STATE:                                       ZIP:





TENANT
      NAME:                                        AGE:
OCCUPATION:                                   EMPLOYER:






SPOUSE
      NAME:                                        AGE:
OCCUPATION:                                   EMPLOYER:






DEPENDENT(S)
    NAME(S):                                  AGE(S):







MAINTENANCE REQUIRED

   ROOM:        ( ) KITCHEN   ( ) BATH    ( ) DINING     ( ) BEDROOM   ( ) ALL

   SERVICE:     ( ) PLUMBING  ( ) CARPET  ( ) ELECTRICAL ( ) PAINT     ( ) NEW TENANT
```

Figure 9-1. Rental maintenance form.

⟨ESC⟩ "B" starts the vertical-tab setting sequence. The codes following the ⟨ESC⟩"B" set tab stops at lines 3, 8, 18, 27, 37, and 48. The trailing CHR$(1) terminates the tab setting process.

Since you can set from 1 to 16 vertical tabs, the printer needs a way to know when you are done. To signal the end, you can always use a CHR$(0) to terminate the tab setting. Alternatively, you may enter any code that is less than the last code entered—this will also stop the process.

As long as the printer stays on, the tabs set in line 20 will remain set—unless they are changed or replaced with another set of tabs.

Activating vertical tabs

Vertical tabs are activated by CHR$(11), which is the ASCII code for VT. (No, it's not an extraterrestrial—but it *is* super.) Set the printer to the top of a form and try it out with:

```
1Ø V$=CHR$(11)
4Ø LPRINT V$;"VT PHONE HOME"
```

```
                    VT PHONE HOME
```

Sure enough, the CHR$(11) (stored in V$) tabs down to line 3 and prints the string: VT PHONE HOME.

But what about the rest of the VTabs set in line 20? Each one requires a CHR$(11). To test this, press the FF button to advance to the top of the next form, then enter these lines to modify your program:

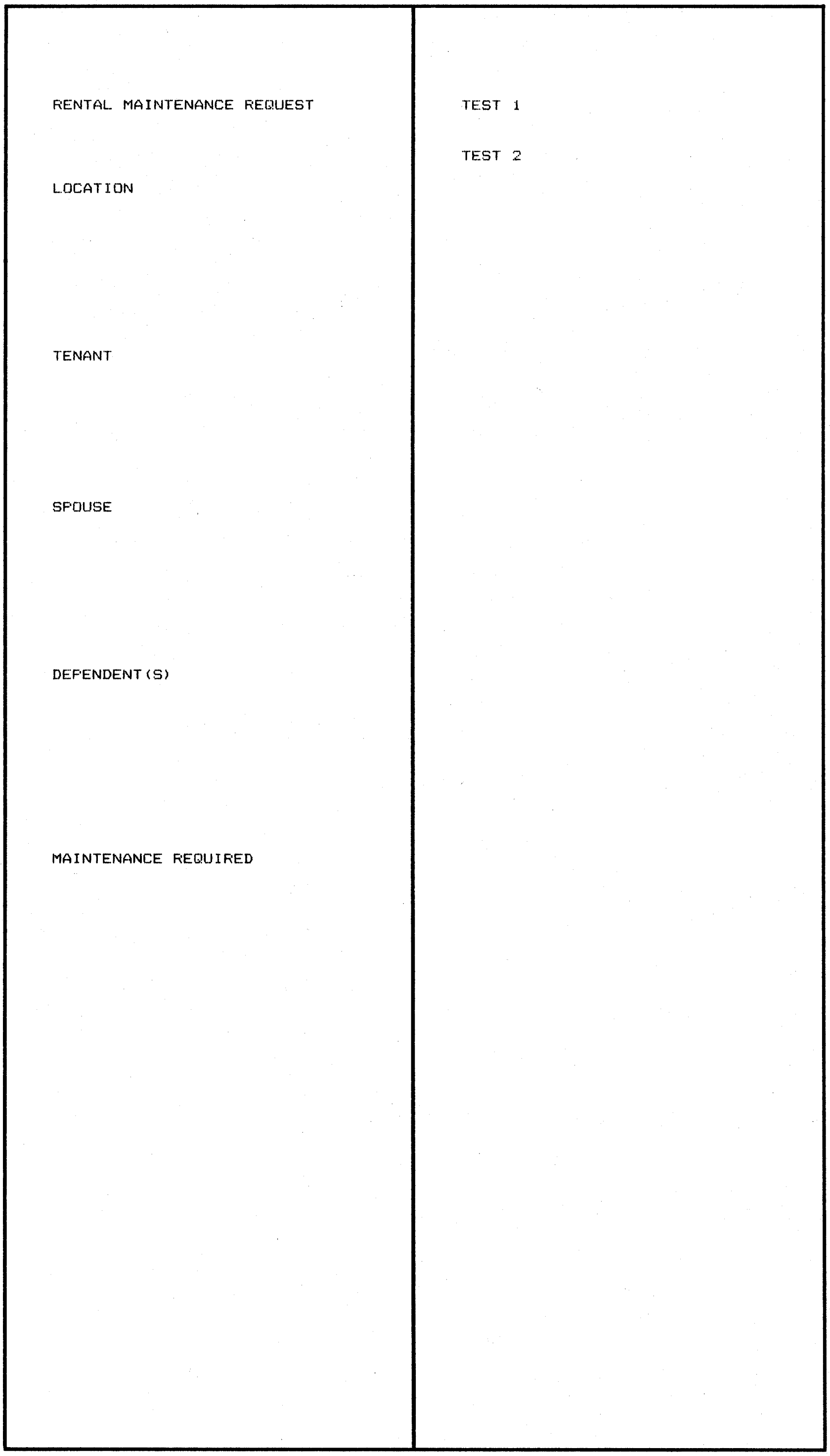
```
4Ø LPRINT V$;"RENTAL MAINTENANCE REQUEST";
6Ø LPRINT V$;"LOCATION"
1ØØ LPRINT V$;"TENANT"
13Ø LPRINT V$;"SPOUSE"
16Ø LPRINT V$;"DEPENDENT(S)"
18Ø LPRINT V$;"MAINTENANCE REQUIRED": LPRINT
19Ø LPRINT V$;"TEST 1"
2ØØ LPRINT V$;"TEST 2"
```

As you can see in Figure 9-2, the first six LPRINT V$ items use the tab stops that were set in line 20. The next LPRINT V$ (line 190) feeds to the top of the next form and prints TEST 1 on line 0. From then on, tabbing continues as usual, and the second form really starts with the TEST 2 on line 3.

FF to the next page, then add the following lines to your program. Count the spaces carefully.

```
7Ø LPRINT "b̸b̸b̸ADDRESS:"
8Ø LPRINT "b̸b̸b̸b̸b̸b̸CITY:"
9Ø LPRINT "b̸b̸b̸b̸b̸STATE:"
11Ø LPRINT "b̸b̸b̸b̸b̸b̸NAME:"
12Ø LPRINT "OCCUPATION:"
14Ø LPRINT "b̸b̸b̸b̸b̸b̸NAME:"
15Ø LPRINT "OCCUPATION:"
17Ø LPRINT "b̸b̸b̸b̸NAME(S):"
25Ø LPRINT V$;
26Ø LPRINT CHR$(27)"@"

DELETE 19Ø-2ØØ
```



RENTAL MAINTENANCE REQUEST

LOCATION

TENANT

SPOUSE

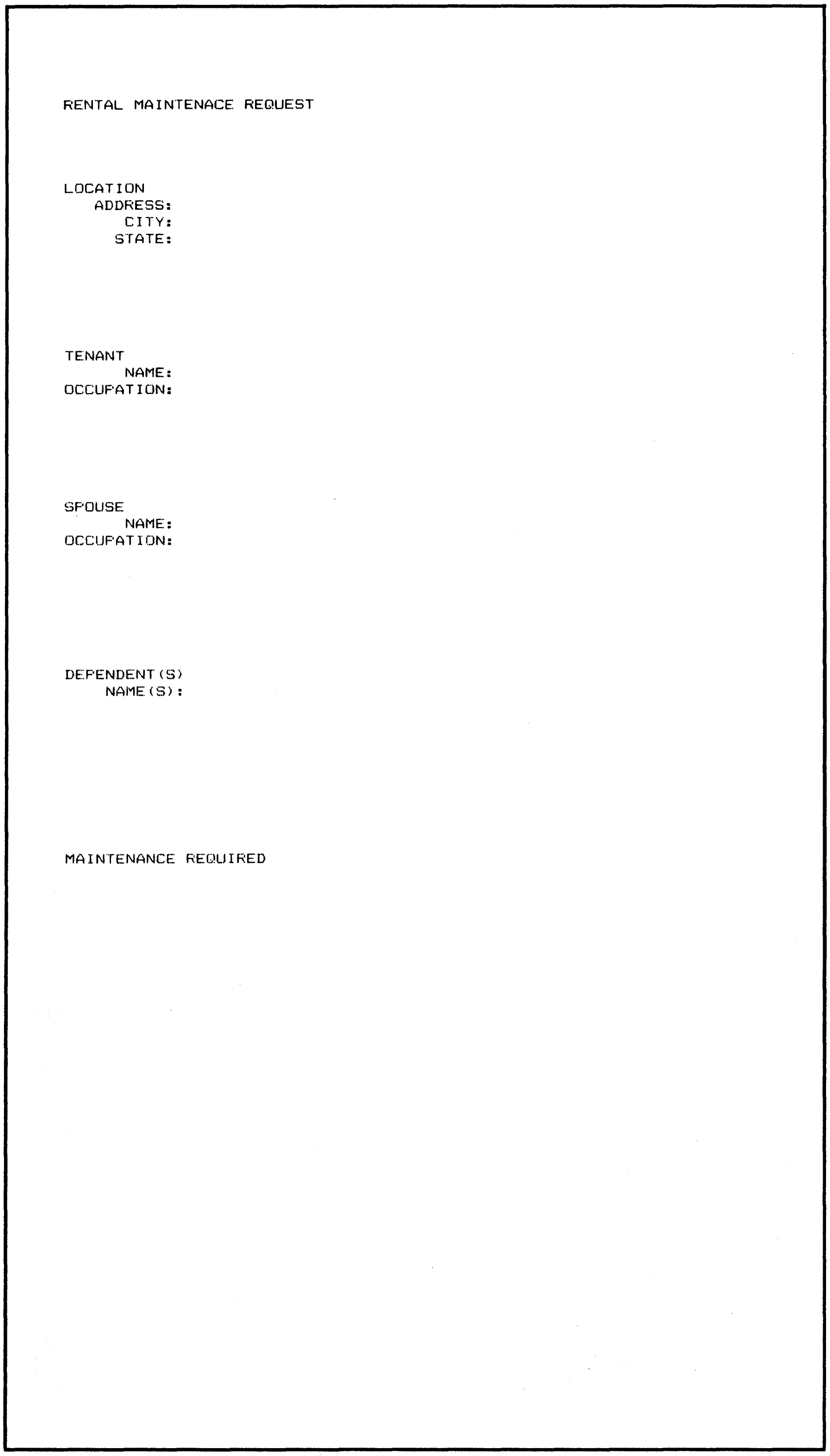
DEPENDENT(S)

MAINTENANCE REQUIRED

TEST 1

TEST 2

Figure 9-2. Form using two pages.



```
RENTAL MAINTENACE REQUEST



LOCATION
   ADDRESS:
      CITY:
     STATE:




TENANT
      NAME:
OCCUPATION:




SPOUSE
      NAME:
OCCUPATION:




DEPENDENT(S)
    NAME(S):




MAINTENANCE REQUIRED
```

Figure 9-3. Setting up the form.

Notice (Figure 9-3) that the automatic line feed at the end of line 260 moves the paper one line past the top of form. Since the first printed line is line 3, the line feed doesn't disrupt the printing of the form.

The ability to set vertical tabs is a nice feature. Heavy users can also store several different settings of vertical tabs in memory, to be recalled as necessary. We tell that tale in the next chapter.

Fixed VTabs

The numbers in the ⟨ESC⟩"B" sequence let you set vertical tabs in multiples of the current line spacing. And once the tabs are set in a program, any subsequent changes in line spacing do not affect the tab positions on the paper. This terrific feature will keep us all from accidentally messing up carefully planned tab settings. Test this out by changing the line spacing in the current program from nine to seven dots (after the VTabs are set!):

```
30 LPRINT CHR$(27)"1"
```

Sure enough, the line spacing in Figure 9-4 is a mess, but the tab stops stay in the same places.

Delete line 30 before you move on:

```
DELETE 30
```

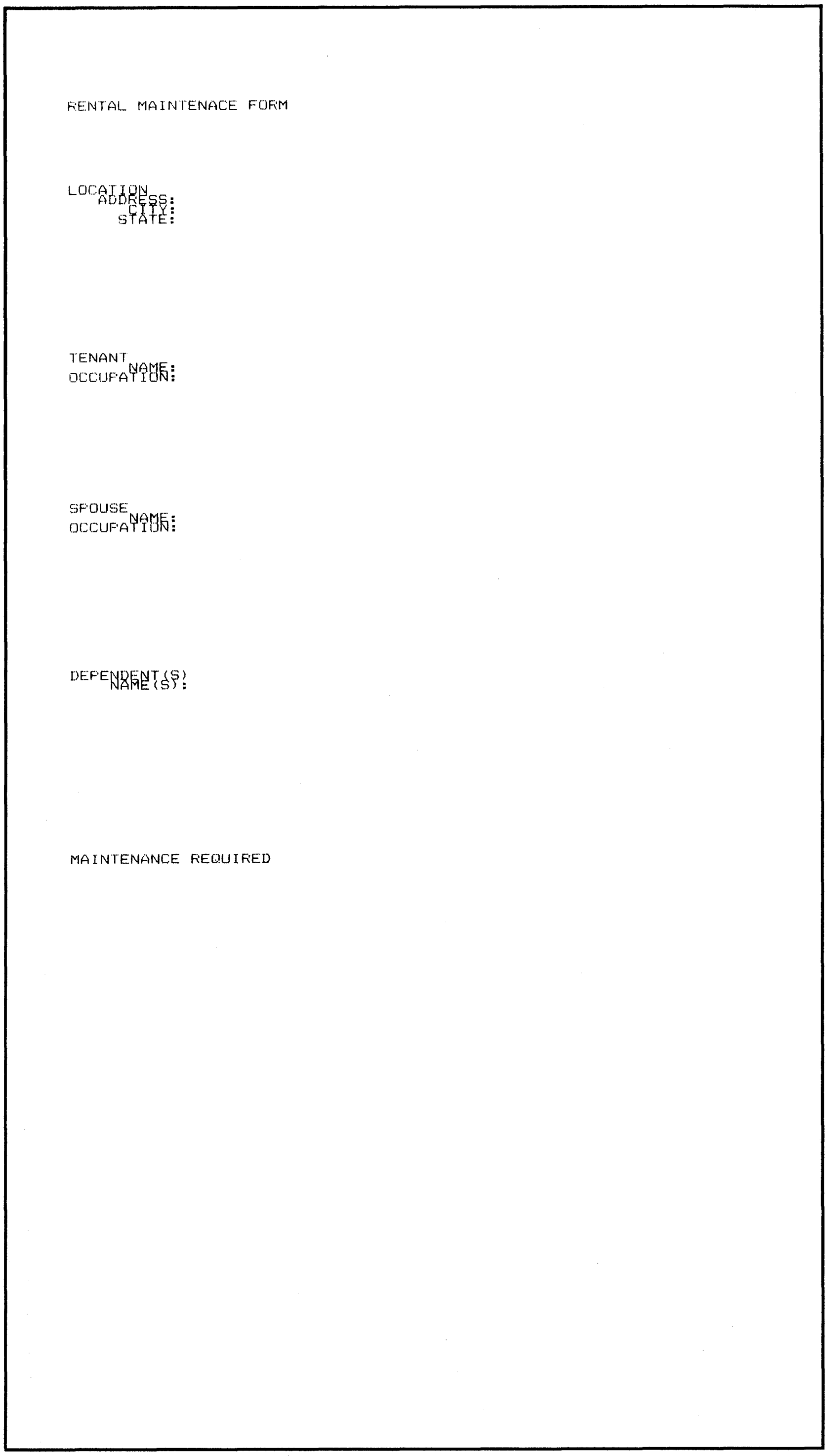
Range limitations

The vertical tab counter actually starts at row 0. So the 66 lines on a normal page are numbered as tab stops 0 through 65. But the allowable range of tab settings in the ⟨ESC⟩"B" sequence is from 1 to 254, which means that you could tab right off the end of a page if you wanted.

Actually, the first tab must be set to less than the last line on a page; the rest can be set to anything up to 254 lines. For most forms, you will want the maximum vertical tab to be 65, the last line on an 11-inch form. This number will change if you change the form length or line spacing. For example, the maximum first tab on an 11-inch form with 7-dot line spacing (⟨ESC⟩"1") would be 113.

Horizontal Tabs (HTabs)

The FX-80 can also remember up to 32 horizontal tab stops. You can use them for one or more indentations or to set up columns.



```
RENTAL MAINTENACE FORM


LOCATION
    ADDRESS:
       CITY:
      STATE:


TENANT
       NAME:
OCCUPATION:


SPOUSE
       NAME:
OCCUPATION:


DEPENDENT(S)
     NAME(S):


MAINTENANCE REQUIRED
```

Figure 9-4. A test of changing line spacing.

Horizontal tabs are set with the ⟨ESC⟩"D" sequence. To set two for your form, add the following line to your current program:

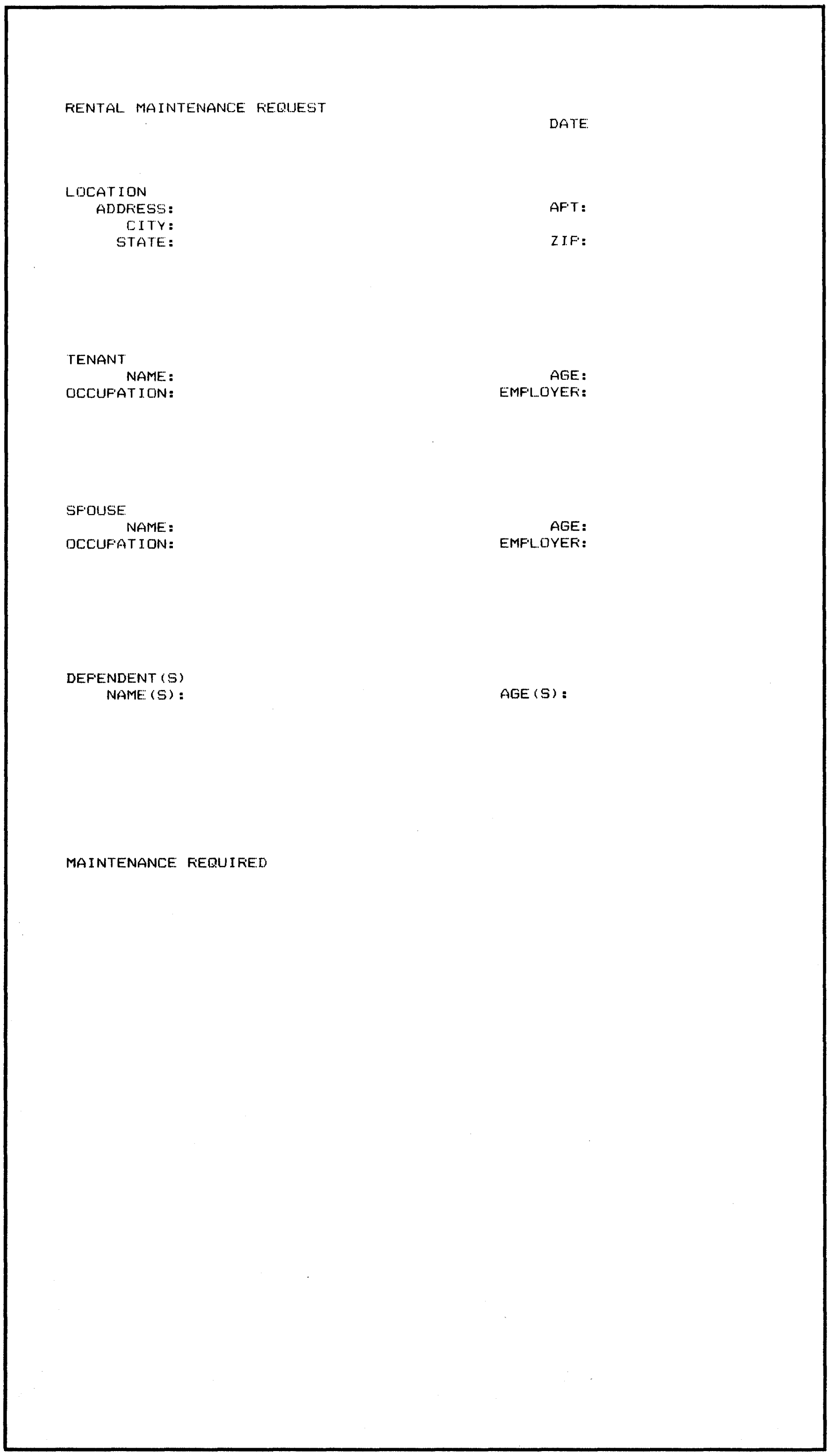
```
    3Ø LPRINT CHR$(27)"D"CHR$(43)CHR$(48)CHR$(1);
```

The numbers 43 and 48 represent tab stops at columns 43 and 48, and the CHR$(1) terminates the definition process.

Just as with vertical tabs, any number less than the last tab setting (including 0) will terminate the tab sequence. And the printer also remembers horizontal tab stops until you turn it off or define a new sequence of tabs.

Horizontal tabs are activated by CHR$(9), which is the ASCII code for HT. To see how this code works, try RUNning your program with these changes:

```
    1Ø V$=CHR$(11): H$=CHR$(9)
    15 ' Apple users see Appendix I
    5Ø LPRINT H$;H$;"DATE"
    7Ø LPRINT "ƀƀƀADDRESS:";H$;H$;"APT:"
    9Ø LPRINT "ƀƀƀƀƀSTATE:";H$;H$;"ZIP:"
    11Ø LPRINT "ƀƀƀƀƀƀNAME:";H$;H$;"AGE:"
    12Ø LPRINT "OCCUPATION:";H$;"EMPLOYER:"
    14Ø LPRINT "ƀƀƀƀƀƀNAME:";H$;H$;"AGE:"
    15Ø LPRINT "OCCUPATION:";H$;"EMPLOYER:"
    17Ø LPRINT "ƀƀƀƀNAME(S):";H$;"AGE(S):"
```



```
RENTAL MAINTENANCE REQUEST
                                                DATE


LOCATION
   ADDRESS:                                     APT:
      CITY:
     STATE:                                     ZIP:






TENANT
      NAME:                                     AGE:
OCCUPATION:                                EMPLOYER:






SPOUSE
      NAME:                                     AGE:
OCCUPATION:                                EMPLOYER:







DEPENDENT(S)
    NAME(S):                               AGE(S):








MAINTENANCE REQUIRED
```

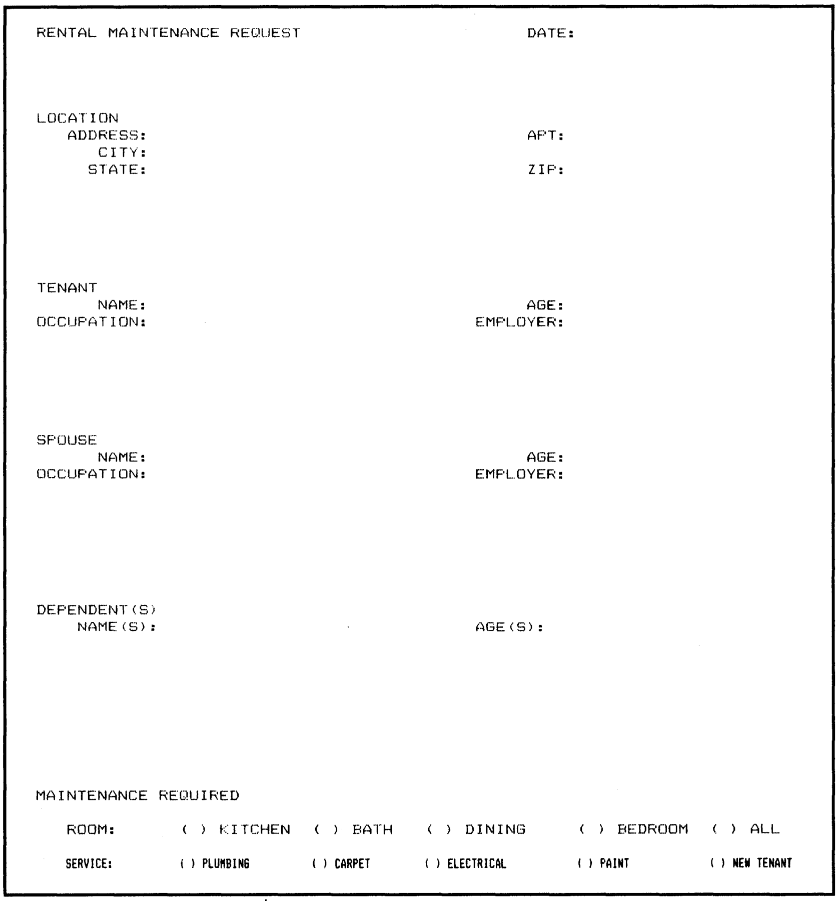
Figure 9-5. Adding horizontal tabs.

CHR$(9) is stored in H$ for convenience. Notice (in Figure 9-5) that many of the strings are printed at the second horizontal tab stop, not the first. By entering a number of H$s, each without a string to print, you cause the FX-80 to skip that number of unneeded tab stops.

Note that no tab settings can ever be larger numbers than the one that represents the last column of the page: column 79 in Pica, 95 in Elite, and 131 in Compressed.

Also note that tabs are set in the currently active pitch. Subsequent changes in pitch do not affect the tab positions on the paper. Here's proof. Add:

```
190 LPRINT CHR$(27)"D"CHR$(14)CHR$(27)CHR$(38)
    CHR$(53)CHR$(66)CHR$(1);
200 LPRINT "ØØØROOM:";H$;"( ) KITCHEN";H$;"( )
    BATH";
    H$;"( ) DINING";
210 LPRINT H$;"( ) BEDROOM";H$;"( ) ALL"
220 LPRINT CHR$(15)
230 LPRINT "ØØØØØSERVICE:";H$;"( ) PLUMBING";
    H$;"( ) CARPET";H$;
240 LPRINT "( ) ELECTRICAL";H$;"( ) PAINT";
    H$;"( ) NEW TENANT"
```



```
RENTAL MAINTENANCE REQUEST                              DATE:



LOCATION
   ADDRESS:                                             APT:
      CITY:
     STATE:                                             ZIP:




TENANT
      NAME:                                             AGE:
OCCUPATION:                                        EMPLOYER:




SPOUSE
      NAME:                                             AGE:
OCCUPATION:                                        EMPLOYER:




DEPENDENT(S)
    NAME(S):                                       AGE(S):






MAINTENANCE REQUIRED

   ROOM:         ( ) KITCHEN   ( ) BATH    ( ) DINING     ( ) BEDROOM   ( ) ALL

   SERVICE:      ( ) PLUMBING      ( ) CARPET     ( ) ELECTRICAL      ( ) PAINT      ( ) NEW TENANT
```

Figure 9-6. A test of changing pitch.

Line 190 sets new HTabs (as in Figure 9-6). The two lines of maintenance options which use these tabs are printed in different pitches to show that the tab stops are not affected by such changes. The only exception to this rule is discussed below.

The final listing for this program makes up Figure 9-7. Name this program FORM, and save it to use in Chapter 10.

```
1Ø V$=CHR$(11): H$=CHR$(9)
2Ø LPRINT CHR$(27)"B"CHR$(3)CHR$(8)CHR$(18)CHR$(27)
   CHR$(37)CHR$(48)CHR$( 1);
3Ø LPRINT CHR$(27)"D"CHR$(43)CHR$(48)CHR$(1);
4Ø LPRINT V$;"RENTAL MAINTENANCE REQUEST";
5Ø LPRINT H$;H$;"DATE"
6Ø LPRINT V$;"LOCATION"
7Ø LPRINT "ƀƀƀADDRESS:";H$;H$;"APT:"
8Ø LPRINT "ƀƀƀƀƀƀCITY:"
9Ø LPRINT "ƀƀƀƀƀSTATE:";H$;H$;"ZIP:"
1ØØ LPRINT V$;"TENANT"
11Ø LPRINT "ƀƀƀƀƀƀNAME:";H$;H$;"AGE:"
12Ø LPRINT "OCCUPATION:";H$;"EMPLOYER:"
13Ø LPRINT V$;"SPOUSE"
14Ø LPRINT "ƀƀƀƀƀƀNAME:";H$;H$;"AGE:"
15Ø LPRINT "OCCUPATION:";H$;"EMPLOYER:"
16Ø LPRINT V$;"DEPENDENT(S)"
17Ø LPRINT "ƀƀƀƀNAME(S):";H$;"AGE(S):"
18Ø LPRINT V$;"MAINTENANCE REQUIRED": LPRINT
19Ø LPRINT CHR$(27)"D"CHR$(14)CHR$(27)CHR$(38)
    CHR$(53)CHR$(66)CHR$(1);
2ØØ LPRINT "ƀƀƀROOM:";H$;"( ) KITCHEN";H$;"( ) BATH";
    H$;"( ) DINING";
21Ø LPRINT H$;"( ) BEDROOM";H$;"( ) ALL"
22Ø LPRINT CHR$(15)
23Ø LPRINT "ƀƀƀƀƀSERVICE:";H$;"( ) PLUMBING";
    H$;"( ) CARPET";H$;
24Ø LPRINT "( ) ELECTRICAL";H$;"( ) PAINT";
    H$;"( ) NEW TENANT"
25Ø LPRINT V$;
26Ø LPRINT CHR$(27)"@"
```

Figure 9-7. Program for FORM.

Default horizontal tabs

The FX-80 has a set of built-in tab stops for quick and easy horizontal tabbing. You don't need to type an HTab setting at all—simply type a CHR$(9) and the printer will tab over eight spaces. The built-in settings are (counting from 0) set at columns 8, 16, 24, etc. To give these tabs a workout, enter:

```
NEW
10 FOR X=1 TO 9
20 LPRINT CHR$(9);"T";
30 NEXT X
```

```
T       T       T       T       T       T
```

These default tabs are similar to the "quick and easy" tabs you get when you use commas in a PRINT statement. You'll find them useful for applications that don't require a great deal of horizontal control. For those special "custom" settings, use the ⟨ESC⟩"D".

One word of caution in using the default tabs. In Expanded Mode, the default tabs change from 8, 16, 24, 32, 40 . . . to 8, 24, 40, 56. . . . In other words, every other tab setting is skipped when Expanded Mode is used with default tabs. This happens only with the default tabs, not when tabs are set with the ⟨ESC⟩"D" sequence.

BASIC TABs

Why all this fuss about printer tabs when the BASIC program in your computer has a perfectly good TAB function? Simple. BASIC TAB functions are designed to be used on the screen. Your system's TAB may not be able to reach past the 64 or 80 columns used for your screen. Or worse, on some systems the TAB works fine to the limit of the screen, but for larger numbers the TAB function acts like SPACE$(n): it prints n spaces between tabs. With all this variation, using the reliable printer tabs can save hours of frustration.

Setting the Margins

Most word processing programs have codes for setting left and right margins. If yours doesn't, or if you want to change margins from BASIC, the FX-80 has just the thing—margin control. For normal characters, the default margins are set for a full 80 characters. You can change each margin to suit your needs.

Left margin

Enter this one-line command without a line number:

```
LPRINT CHR$(27)"l"CHR$(20)
```

If your system has trouble with lower-case characters, use CHR$(108) in place of lower-case l.

This command sets the left margin to column 20 (counting from column 0). Invalid settings, such as those greater than the current page width (78 for Pica, 93 for Elite, and 133 for Compressed) are ignored. All margin settings go into effect when a carriage return is received, and they stay in effect until changed. Now type:

```
LLIST
```

```
                    10 FOR X=1 TO 9
                    20 LPRINT CHR$(9);"T";
                    30 NEXT X
```

A listing of your current program should be 20 spaces to the right of the usual left margin.

Like the VTab and HTab features, the margin settings are not affected by changes in the width of the print after they are set. So if you set the left margin in Pica Mode, then switch to Compressed, the left margin stays where you set it. For example, type:

```
LPRINT CHR$(15)
LLIST
```

```
                              10 FOR X=1 TO 9
                              20 LPRINT CHR$(9);"T";
                              30 NEXT X
```

There are still 20 Pica spaces of margin.

Right margin

The right margin is set with ⟨ESC⟩"Q". Enter this command:

```
LPRINT CHR$(18);CHR$(27)"Q"CHR$(19)
```

Now type LLIST.

What happened? Well, the CHR$(18) turned off Compressed Mode, but there's no change in the margin. (Because the right margin setting would be on the wrong side of the current left margin which is still set at 20—Oops!) The printer simply ignores incorrect settings. This time try it with:

```
    LPRINT CHR$(27)"Q"CHR$(3Ø)
```

and LLIST:

```
10 FOR X=1
 TO 9
20 LPRINT
CHR$(9);"T
";
30 NEXT X
```

Much better. All the characters are printed between columns 20 and 30. The limit parameters for the right margin command are 2 and 80 in Pica, 3 and 96 in Elite, and 4 and 137 in Compressed.

Using the margin commands together

The two margin commands use different numbering systems. While the left margin command counts from 0 to 79 in Pica Mode, the right margin command counts from 1 to 80. So in theory, you could set the left margin to 39 and the right margin to 40 to limit printing to a single column in the middle of the page. In practice, however, you can't print less than two columns in Pica mode, three in Elite, and four in Compressed (the printer requires room to print at least one Expanded Pica character).

Margin Settings and Other Modes

The left and right margin settings cancel any horizontal tabs set by 〈ESC〉"D", so be sure to set the margins first.

Because margin settings also clear the buffer in the same way that 〈ESC〉"@" does, don't issue these commands at the end of a print line.

Another interesting twist is the way that the margin settings affect Compressed Mode. The default right margin for Compressed Mode is normally 132 characters; this uses the first 7 11/16 inches of the print line. By setting the right margin to 137, you can surpass the normal margin of 132 and control the full 8-inch print line.

Unidirectional Print

The FX-80 provides a high-quality print in its normal bidirectional print mode. There may, however, be situations in which vertical columns printed in Compressed Mode are slightly misaligned. To alleviate any such potential problems, the printer has a Unidirectional Mode.

To see how it works, enter:

```
LPRINT CHR$(27)"@":NEW
```

and then RUN this program:

```
20 LPRINT CHR$(27)" l"CHR$(40)     ' lower-case el
30 LPRINT CHR$(27)" 1";CHR$(15)     ' number one
40 FOR X=0 TO 10
50 LPRINT "#"
60 NEXT X
70 LPRINT CHR$(27)"@"
```

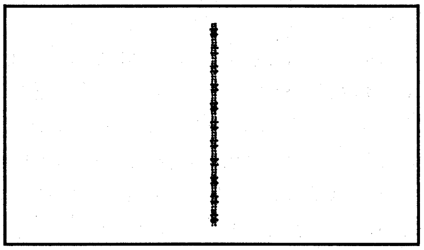


Figure 9-8. Uneven columns.

Look at the printout (Figure 9-8) ever so carefully . . . notice how the columns are slightly uneven? Yes, you have to look very closely. If nothing seems amiss, fine. But turn on the Unidirectional print to see if it makes any difference. Add line 10 and RUN it again:

```
10 LPRINT CHR$(27)"U1"
```

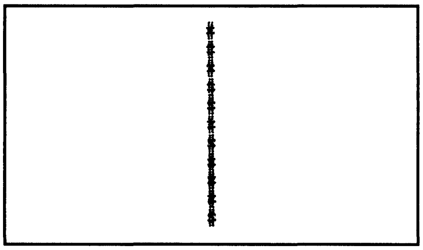


Figure 9-9. Even columns with Unidirectional Mode.

The difference is very subtle, but Figure 9-9 does look more uniform. Did you watch the print head as it printed? It printed in only one direction instead of two. That helps to line up the columns for finely tuned printouts. We used this mode in the TICKET program of the "One Easy Lesson," and we'll use it again in Chapter 19 to help line up our user-defined graphics characters.

⟨ESC⟩"U1" turns on Unidirectional Mode and ⟨ESC⟩"U0" turns it off. There is also a Unidirectional Mode that turns off at the end of one print line—you turn it on with ⟨ESC⟩"⟨".

Summary

CHR\$(9)	Activates horizontal tabbing. Default tabs at columns 8, 16, 24, . . .
CHR\$(11)	Activates vertical tabbing.
⟨ESC⟩"B"CHR\$(n_1)CHR\$(n_2) . . . CHR\$(n_k)CHR\$(0)	Sets vertical tabs at lines n_1, n_2 . . . n_k. Terminates with CHR\$(0) or any number less than n_k
⟨ESC⟩"D"CHR\$(n_1)CHR\$(n_2) . . . CHR\$(n_k)CHR\$(0)	Sets horizontal tabs at columns n_1, n_2 . . . n_k. Terminates with CHR\$(0) or any number less than n_k
⟨ESC⟩"Q"CHR\$(n)	Sets the right margin at n. Ranges are 2 to 80 in Pica, 3 to 96 in Elite, and 4 to 137 in Compressed

⟨ESC⟩"l"CHR$(n)	Sets the left margin at n. CHR$(108) may be substituted for "l". Ranges are 0 to 78 in Pica, 0 to 93 in Elite, and 0 to 133 in Compressed
⟨ESC⟩"⟨"	One-line Unidirectional print ON
⟨ESC⟩"U1"	Continuous Unidirectional print ON
⟨ESC⟩"U0"	Unidirectional print OFF

Note: For all tabs, counting starts at zero.

Chapter 10
Saving Vertical Tabs in Channels

This chapter completes our coverage of formatting. It describes the FX-80's ability to store several vertical tab settings in memory at once, and it demonstrates possible applications.

Do not feel constrained to type in the programs in this chapter. Since the programs are rather long and the topic covered may not be of interest to everyone, consider this chapter optional.

Setting the Numbers

The format for vertical tab settings "by the numbers" is:

```
LPRINT CHR$(27)"b"CHR$(N)CHR$(n1)CHR$(n2) . . .
   CHR$(nk)CHR$(1)
```

where N is the reference number with which this tab setting will be stored. (Use CHR$(98) if your system can't send lower-case b.) N may range from 0 to 7. Note that ⟨ESC⟩ "b"CHR$(0) is the same as ⟨ESC⟩ "B".

A particular set of tabs, which is collectively called a channel, is loaded with:

```
LPRINT CHR$(27)"/"CHR$(n)
```

where n selects channel 0 through 7.

There are two primary reasons to use several channels in one program: 1) to fill out a preprinted form that uses different tab stops for each type of respondent; and 2) to create a multipage report or form that uses different tab settings on each page. Figure 10-1 shows the whole program for the former, and Figure 10-3 for the latter.

```
NEW
10 LPRINT CHR$(27)"D"CHR$(14)CHR$(54)CHR$(1);
20 LPRINT
   CHR$(27)"b"CHR$(0)CHR$(3)CHR$(09)CHR$(50)CHR$(1);
25  ' Model I and Apple users see Appendix I
30 LPRINT CHR$(27)"b"CHR$(1)CHR$(3)CHR$(09)
   CHR$(19)CHR$(50)CHR$(1);
40 LPRINT CHR$(27)"b"CHR$(2)CHR$(3)CHR$(09)CHR$(19)
   CHR$(38)CHR$(50)CHR$( 1);
50 LPRINT CHR$(27)"b"CHR$(3)CHR$(3)CHR$(09)CHR$(19)
   CHR$(28)CHR$(50)CHR$( 1);
60 LPRINT CHR$(27)"b"CHR$(4)CHR$(3)CHR$(09)CHR$(19)
   CHR$(28)CHR$(38)CHR$( 50)CHR$(1);
70 DATA 000,100,101,110,111
80 DIM A(5),A$(40),B$(20),C$(10)
90 FOR I=0 TO 4: READ A(I): NEXT I
100 B = 0
110 PRINT "IS THERE A CURRENT TENANT (Y/N)";:INPUT A$
120 IF LEFT$(A$,1)="N" THEN 200
130 B = 100
140 PRINT "IS THE TENANT MARRIED (Y/N)";:INPUT A$
150 IF LEFT$(A$,1)="N" THEN 170
160 B = 110
170 PRINT "DOES THE TENANT HAVE ANY CHILDREN (Y/
    N)";:INPUT A$
180 IF LEFT$(A$,1)="N" THEN 200
190 B = B + 1
200 FOR K=0 TO 4
210 IF A(K)<>B THEN NEXT K
220 LPRINT CHR$(27)"/"CHR$(K);
230 PRINT: INPUT "DATE (MM/DD/YY)";A$
240 LPRINT CHR$(11)CHR$(9)CHR$(9)A$CHR$(11);
250 INPUT "STREET ADDRESS ";A$:INPUT "APARTMENT
    NUMBER";B$
260 LPRINT CHR$(9)A$CHR$(9)B$
270 INPUT "CITY ";A$:INPUT "STATE ";B$: INPUT "ZIP CODE
    ";C$
280 LPRINT CHR$(9)A$:LPRINT CHR$(9)B$CHR$(9)C$CHR$(11);
290 IF B < 100 THEN 460
300 PRINT:INPUT "TENANT'S NAME ";A$:INPUT "AGE ";B$
310 LPRINT CHR$(9)A$CHR$(9)B$
320 INPUT "OCCUPATION ";A$: INPUT "EMPLOYER ";B$
330 LPRINT CHR$(9)A$CHR$(9)B$CHR$(11);
340 B = B - 100
```

Figure 10-1. Program for using FORM.

```
350 IF B < 10 THEN 410
360 PRINT:INPUT "SPOUSE'S NAME ";A$: INPUT "AGE ";B$
370 LPRINT CHR$(9)A$CHR$(9)B$
    380 INPUT "OCCUPATION ";A$: INPUT "EMPLOYER ";B$
390 LPRINT CHR$(9)A$CHR$(9)B$CHR$(11);
400 B = B - 10
410 IF B = 0 THEN 460
420 PRINT
430 INPUT "HOW MANY CHILDREN ";NC:FOR I=1TONC
440 PRINT:PRINT "CHILD #";I:INPUT "NAME ";A$:INPUT "AGE
    ";B$
450 LPRINT CHR$(9)A$CHR$(9)B$:NEXT I:LPRINT CHR$(11);
460 LPRINT CHR$(27)"D"CHR$(14)CHR$(27)CHR$(38)CHR$(53)
    CHR$(66)CHR$(1);
470 PRINT "PROBLEM AREA:  1 - KITCHEN   2 - BATH   3 -
    DINING"
480 PRINT "       4 - BEDROOM    5 - ALL";:INPUT N
490 FOR K = 1 TO N:LPRINT CHR$(9);:NEXT K
500 LPRINT " x":LPRINT
510 PRINT "SERVICE NEEDED:  1 - PLUMBING   2 - CARPET  3
    - ELECTRICAL"
520 PRINT "  4 - PAINT    5 - NEW TENANTS";:INPUT N
530 FOR K = 1 TO N:LPRINT CHR$(9);:NEXT K
540 LPRINT CHR$(15);" x";CHR$(18);
550 LPRINT CHR$(11)CHR$(27)"@";
560 END
```

```
RUN 1:IS THERE A CURRENT TENANT (Y/N)? Y
IS THE TENANT MARRIED (Y/N)? N
DOES THE TENANT HAVE ANY CHILDREN (Y/N)? N
DATE (MM/DD/YY)? 11/13/85
STREET ADDRESS ? 1234 HESS STREET
APARTMENT NUMBER? <ENTER>
CITY ? LOST ANGELS
STATE ? CA
ZIP CODE ? 11111
TENANT'S NAME ? JOHN HOPKINS
AGE ? 24
OCCUPATION ? NONE
EMPLOYER ? NONE
PROBLEM AREA:  1 - KITCHEN  2 - BATH    3 - DINING
   4 - BEDROOM    5 - ALL? 2
SERVICE NEEDED:  1 - PLUMBING   2 - CARPET   3 -
   ELECTRICAL  4 - PAINT     5 - NEW TENANTS? 1
```

```
RENTAL MAINTENANCE REQUEST                          DATE: 11/13/85


LOCATION
   ADDRESS:   1234 HESS STREET                      APT:
      CITY:   LOST ANGELS
     STATE:   CA                                    ZIP:  00000



TENANT
      NAME:   JOHN HOPKINS                          AGE:  24
OCCUPATION:   NONE                             EMPLOYER:  NONE



SPOUSE
      NAME:                                         AGE:
OCCUPATION:                                    EMPLOYER:



DEPENDENT(S)
    NAME(S):                                   AGE(S):



MAINTENANCE REQUIRED

   ROOM:      ( ) KITCHEN  (x) BATH   ( ) DINING     ( ) BEDROOM  ( ) ALL

   SERVICE:   (x) PLUMBING   ( ) CARPET   ( ) ELECTRICAL   ( ) PAINT   ( ) NEW TENANT
```

```
RUN 2:IS THERE A CURRENT TENANT (Y/N)? N
DATE (MM/DD/YY)? 11/13/85
STREET ADDRESS ? 475 HART LANE
APARTMENT NUMBER? 43
CITY ? HAMPTON
STATE ? FL
ZIP CODE ? 11111
PROBLEM AREA:  1 - KITCHEN  2 - BATH    3 - DINING
   4 - BEDROOM    5 - ALL? 5
SERVICE NEEDED:  1 - PLUMBING   2 - CARPET   3 -
   ELECTRICAL  4 - PAINT     5 - NEW TENANTS? 5
```

```
RENTAL MAINTENANCE REQUEST                          DATE: 11/13/85


LOCATION
   ADDRESS:   475 HART LANE                         APT:  43
      CITY:   HAMPTON
     STATE:   FL                                    ZIP:  00000



TENANT
      NAME:                                         AGE:
OCCUPATION:                                    EMPLOYER:



SPOUSE
      NAME:                                         AGE:
OCCUPATION:                                    EMPLOYER:



DEPENDENT(S)
    NAME(S):                                   AGE(S):



MAINTENANCE REQUIRED

   ROOM:      ( ) KITCHEN  ( ) BATH   ( ) DINING     ( ) BEDROOM  (x) ALL

   SERVICE:   ( ) PLUMBING   ( ) CARPET   ( ) ELECTRICAL   ( ) PAINT   (x) NEW TENANT
```

Figure 10-2. Four sets of sample answers and printouts.

```
RUN 3:IS THERE A CURRENT TENANT (Y/N)? Y
IS THE TENANT MARRIED (Y/N)? Y
DOES THE TENANT HAVE ANY CHILDREN (Y/N)? N
DATE (MM/DD/YY)? 11/13/85
STREET ADDRESS ? 387 LANG ST
APARTMENT NUMBER? 88
CITY ? SAN DOJO
STATE ? TX
ZIP CODE ? 33333
TENANT'S NAME ? DON WALKER
AGE ? 55
OCCUPATION ? TEACHER
EMPLOYER ? SD SCHOOLS
SPOUSE'S NAME ? JANE WALKER
AGE ? 43
OCCUPATION ? PILOT
EMPLOYER ? ASP AIRLINES
PROBLEM AREA:  1 - KITCHEN  2 - BATH    3 - DINING
   4 - BEDROOM    5 - ALL? 1
SERVICE NEEDED:  1 - PLUMBING   2 - CARPET   3 -
   ELECTRICAL  4 - PAINT    5 - NEW TENANTS? 4
```

```
RENTAL MAINTENANCE REQUEST                      DATE: 11/13/85


LOCATION
   ADDRESS:    387 LANG ST                      APT:   88
      CITY:    SAN DOJO
     STATE:    TX                               ZIP:   00000



TENANT
      NAME:    DON WALKER                       AGE:   55
OCCUPATION:    TEACHER                     EMPLOYER:   SD SCHOOLS



SPOUSE
      NAME:    JANE WALKER                      AGE:   43
OCCUPATION:    PILOT                       EMPLOYER:   ASP AIRLINES



DEPENDENT(S)
    NAME(S):                               AGE(S):



MAINTENANCE REQUIRED

   ROOM:       (x) KITCHEN  ( ) BATH   ( ) DINING    ( ) BEDROOM  ( ) ALL

   SERVICE:    ( ) PLUMBING    ( ) CARPET    ( ) ELECTRICAL    (x) PAINT    ( ) NEW TENANT
```

```
RUN 4:IS THERE A CURRENT TENANT (Y/N)? Y
IS THE TENANT MARRIED (Y/N)? Y
DOES THE TENANT HAVE ANY CHILDREN (Y/N)? Y
DATE (MM/DD/YY)? 11/13/85
STREET ADDRESS ? 980 WEST FULTON
APARTMENT NUMBER? 2
CITY ? EASTWOOD
STATE ? NY
ZIP CODE ? 44444
TENANT'S NAME ? SANDRA MOORE
AGE ? 23
OCCUPATION ? COMPUTER PROGRAMMER
EMPLOYER ? NOSPE INC.
SPOUSE'S NAME ? DON MOORE
AGE ? 23
OCCUPATION ? CIRCUIT DESIGN
EMPLOYER ? IT
HOW MANY CHILDREN ? 2
CHILD # 1
NAME ? TONYA
AGE ? 2
CHILD # 2
NAME ? DANA
AGE ? 1
PROBLEM AREA:  1 - KITCHEN  2 - BATH    3 - DINING
   4 - BEDROOM    5 - ALL? 5
SERVICE NEEDED:  1 - PLUMBING   2 - CARPET   3 -
   ELECTRICAL  4 - PAINT    5 - NEW TENANTS? 5
```

```
RENTAL MAINTENANCE REQUEST                      DATE: 11/13/85


LOCATION
   ADDRESS:    980 WEST FULTON                  APT:   2
      CITY:    EASTWOOD
     STATE:    NY                               ZIP:   00000



TENANT
      NAME:    SANDRA MOORE                     AGE:   23
OCCUPATION:    COMPUTER PROGRAMMER         EMPLOYER:   NOSPE INC.



SPOUSE
      NAME:    DON MOORE                        AGE:   23
OCCUPATION:    CIRCUIT DESIGN              EMPLOYER:   IT



DEPENDENT(S)
    NAME(S):   TONYA                       AGE(S):     2
               DANA                                    1



MAINTENANCE REQUIRED

   ROOM:       ( ) KITCHEN  ( ) BATH   ( ) DINING    ( ) BEDROOM  (x) ALL

   SERVICE:    ( ) PLUMBING    ( ) CARPET    ( ) ELECTRICAL    ( ) PAINT    (x) NEW TENANT
```

Filling in Preprinted Forms

To demonstrate the first case, you need a preprinted form. As luck would have it, you created one in Chapter 9, and we suggested that you save it as FORM, remember? Well, LOAD it back into memory so that you can make a series of four preprinted forms. Now follow:

1. Set your fanfold paper to the Top Of Form (with one blank leading sheet) by turning the printer off, rolling the paper to TOF (see Figure 8-1), and powering the printer back up.
2. Run the program four times in succession. *Do not tear off the printout!*
3. Turn the printer off, and carefully feed the paper backward until it reaches the top of the first form. Keep the bottom sheet taut as it feeds. Turn your FX-80 on.
4. Type in and RUN the program shown as Figure 10-1, which is designed to be run by a rental manager when a unit is in need of maintenance. This program asks three questions and then, based on the answers, selects the appropriate vertical tab settings for a second printout of the form. If you use the answers that are suggested in the "RUN" columns of Figure 10-2, you will get the results shown there.

Yes, this is a long program, but it certainly demonstrates the idea.

Notice in the program listing that the proper vertical settings for using the form are chosen in lines 200 to 220, based on the answers to three questions.

Multiple-Page Forms

A second application for channels of vertical tab settings is creating multiple-page forms. The program of Figure 10-3 shows how a different channel can be used on each page. Set the paper to the Top Of Form, and type in the program for the results shown in Figure 10-4.

```
NEW
10 LPRINT CHR$(27)"b"CHR$(1)CHR$(15)CHR$(40)CHR$(1);
20     ' SET TABS AT 15,40 FOR CHANNEL #1
30 LPRINT CHR$(27)"b"CHR$(2)CHR$(25)CHR$(30)CHR$(1);
40     ' SET TABS FOR CHANNEL #2 AT 25,30
50 LPRINT CHR$(27)"b"CHR$(5)CHR$(50)CHR$(55)CHR$(1);
60     ' SET TABS FOR CHANNEL #5 AT 50,55
100 ' ****** START OF MAIN PROGRAM *******:
110 FOR Z=1 TO 3 : READ X
120 LPRINT "TOP OF PAGE"
130 LPRINT CHR$(27) "/" CHR$(X);
140    ' SET CURRENT CHANNEL "PAGE"
150 LPRINT CHR$(11) "TAB #1 FOR CHANNEL"X
160 LPRINT CHR$(11) "TAB #2 FOR CHANNEL"X
170 LPRINT CHR$(140);
180 NEXT Z
190 DATA 1,2,5
200 LPRINT CHR$(27)"@"
10000 END
```

Figure 10-3. Program for multiple-page channels.

These example applications of vertical tabbing channels should get you rolling.

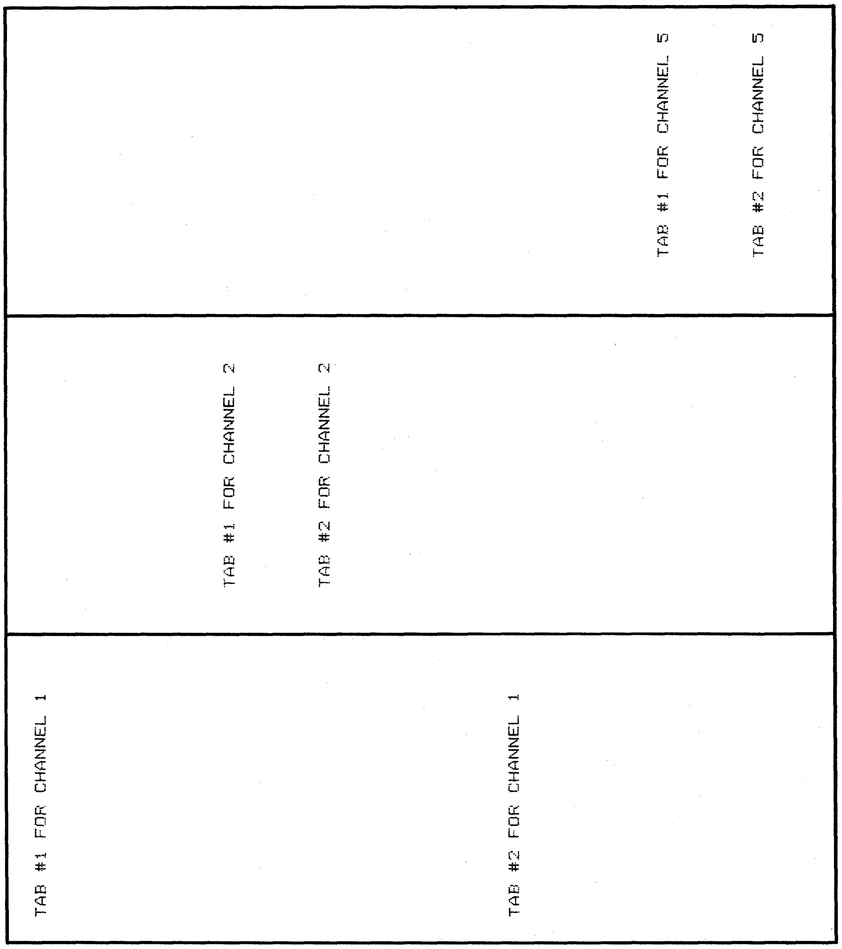
Next Up . . . Graphics

We've covered 10 whole chapters so far—that's about half the manual. It's time to look back over our shoulders at where we've been and ahead to where we are going. So far, you've learned about a substantial number of printer features. In fact, you've covered nearly all the features the FX-80 has to offer. But the best is yet to come.

The rest of the chapters are devoted to user-defined characters and graphics. If you have an ounce of creativity in your bones, you'll find these topics fascinating. They open the door to a whole new way of controlling the printer. Before you walk in, why not take a break?

Summary

CHR$(27)"b"CHR$(N)CHR$($n_1$)CHR$(n_2) . . . CHR $($n_k$)CHR$(1)	N sets the vertical channel number. 0<=N<=7; 0 is the same as <ESC>"B"
CHR$(27)"/"CHR$(n)	Selects channel n



```
TAB #1 FOR CHANNEL 1

TAB #2 FOR CHANNEL 1
```

```
TAB #1 FOR CHANNEL 2

TAB #2 FOR CHANNEL 2
```

```
TAB #1 FOR CHANNEL 5

TAB #2 FOR CHANNEL 5
```

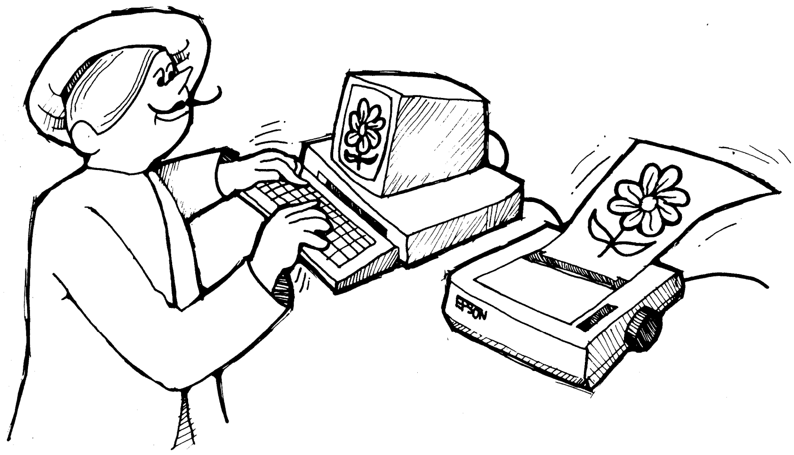
Figure 10-4. Three pages of channels printed out.

Chapter 11
Introduction to Dot Graphics

Welcome to the world of Epson graphics. If you like the printer so far, wait until you see what it can do with high-resolution dot graphics. We have some exciting things in store, but don't look too far ahead because that would spoil the surprise.

The Big Dot Matrix

Imagine the whole page as a huge matrix of dots . . . You can use this matrix like an artist's canvas to create your own graphic images. Where the painter uses brush and paint, you will use computer and printer to breathe life into your ideas.



So, how many dots can you squeeze onto a page? Let's see, each character is 6 dots wide and there are 80 characters per horizontal line—that's 480 dots per row. And there are 12 dots per vertical line, with 66 rows—that makes a total of 792 dots extending down the page. Calculators ready? That gives a grand total potential for 380,160 dots on a single sheet of paper. Whew! And that doesn't take into account the printer's ability to compress the print, both horizontally and vertically. Let's face it, we'll need a lot of dots and data to fill the entire page with graphics!

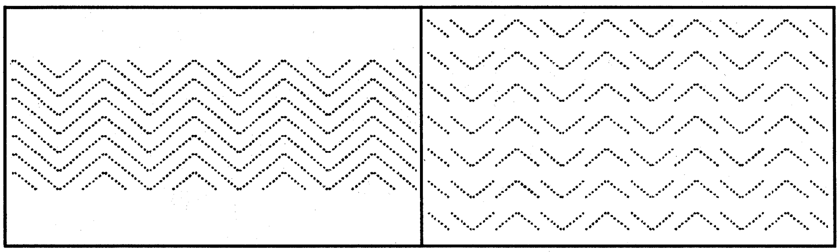
Graphics Mode

Printing high-resolution graphics on the FX-80 requires an entirely new kind of mode. In any of the several versions of Graphics Mode, none of the *predefined* characters or symbols in the printer's memory are used. Instead, you create the patterns of dots that are printed. Yes, you control where and when each and every dot is printed. That's power!

And how do these modes operate? Essentially, a single pattern of dots is printed on each pass of the print head. Even though there are 9 pins on the print head, the printer is designed to print only 8 at a time—the top 8. (There is an optional 9-dot mode which we will get to in due time.) The limiting factor is that printer/computer communications are based on 8 data lines, and each pin of the print head corresponds to 1 of these 8. So each sweep of the head prints up to 8 dots.

Seven-bit computers are further limited since they control only 7 of the 9 pins (more on this in a moment).

To print figures taller than 7 or 8 dots, just adjust the line spacing to 8 dots (7 dots for 7-bit computers), and print several lines until the figure is complete. See Figure 11-1, below.



a. 7-dot line spacing. b. 12-dot line spacing.
Figure 11-1. Comparison of 7- and 12-dot line spacing.

Each pass of the print head contains one piece of the total pattern, which can be as tall as you desire. You don't have to use the whole page or even an entire line for your graphics figures. In fact, you can reserve as little or as much space on the page as you like for a figure—and position it anywhere on the page!

Entering a Graphics Mode or . . . Reservations required!

For each row of a graphics figure, you must first enter a Graphics Mode and then tell the printer the number of columns you wish to print on that line. In ordinary Graphics Mode, you can print up to 480 columns of dots across an 8-inch page; higher density modes use more columns. The height of a figure depends on the number of passes used; it can can be pages long—or only a few dots tall.

So while you are telling the printer that you're going to print graphics, you also tell it how many columns to print in each horizontal row. Here is the format for making this reservation:

```
    LPRINT CHR$(27)"K"CHR$(n1)CHR$(n2);
```

The ⟨ESC⟩"K" selects Graphics Mode (which uses what we'll call "normal" density for now). The two numbers (n_1 and n_2) determine the number of columns reserved for graphics.

Why two numbers instead of one? As usual, greed is at the root. Wouldn't you like to be able to print a figure across the entire width of the page if you chose? Of course you would! And an 8-inch page can hold up to 480 normal-density graphics dots. Unfortunately, since the BASIC CHR$ function is limited to numbers from 0 to 255, we can't send a number as large as 480 to the printer. Carumba!

That's where the second number fits in. The two numbers work in concert to send *very* large numbers to the printer. The first number (n_1) indicates a number of columns (from 0 to 255), as you'd expect. A 255 in that position means "reserve 255 columns for graphics." Thus any figure less than half a page wide can be handled easily by the first number alone. In this case, we pacify the printer, which still expects a value for the second number (n_2), with a 0.

Often a figure needs more than half a page. To reserve more than 255 columns for graphics, the second number (n_2) must be greater than 0. But n_2 does *not* represent single dots; it represents bunches of 256 dots. Using a 1 in the second position means "reserve one bunch of 256 dots plus whatever is in the first number." A 2 in that spot means "reserve 512 dots plus . . ." and so on—up to 7 times 256 (or 1792) dots of reserved width for the n_2 portion.

Note: The FX-80 will accept numbers larger than 7 for n_2, but it is pointless. If you reserve room for more graphics data than the printer can use in a single print line, the printer will print to the end of the line, then wait patiently for the rest of the data. If you don't send enough, it will hang. If you do send the amount that you specified, the printer graciously accepts the data, tosses it into the circular file, then moves on to print the next line as if nothing strange had happened.

The maximum number of dots that can be reserved, then, is:

```
CHR$(27)"K"CHR$(255)CHR$(7);
```

which is 255 dots plus 7 times 256 dots . . . for a grand total of 2047!

But the total width of a line in normal density is only 480 dots. For now, we'll stick with that density, which means that we won't use numbers over 480. Later we'll see that the FX-80 does have Graphics Modes of greater density, and that these modes can require numbers as large as 1920.

The 7-Bit Limitation

There always seems to be a catch. In Chapter 6, we mentioned that some computers have problems sending the upper half of the ASCII codes (128 to 255). And we said that these computers would have problems with dot graphics . . . well, this is probably a good time to explain why. (If you have an Apple II, you might want to sit down and take a very deep breath. If you are not sure if your computer fits into this category, see Appendix H.)

It all starts with the computer/printer interface (that figures!). You see, to send numbers greater than 127 requires 8 active data lines between the computer and the printer. That's OK except that some computers (Apples in particular) snuff out that eighth bit and allow only 7 lines of data to pass through to the printer. Such a computer automatically subtracts 128 from all numbers in the range 128 to 255, making the effective range 0 to 127. The number 227, for example, would arrive at the printer as a 99. Arrrgh!

This means that a whole lot of information cannot be transferred to the printer even though the printer is ready and willing to accept it. (Computers sure do have a hard time keeping up with printers, don't they?) You users with this problem will have to find ways around it. We'll help whenever possible.

The 7-bit limitation affects graphics in two ways. First, when you enter a Graphics Mode, the first number (n_1) you send can only go up to 127. By changing n_2 from 0 to 1, 2, 3, etc., computers with this limitation get figure widths with gaps: 0 to 127, 256 to 383, 512 to 639, etc.

Let's look at an example. The usual way to reserve the entire width of the page for graphics is:

```
    LPRINT CHR$(27)"K"CHR$(225)CHR$(1);
```

This gives 225 dots plus 1 times 256 dots, or 480 total. But a 7-bit computer cannot send the 225; the largest number it can send is 127. So the maximum number attainable with $n_2=1$ is $127 + 1 \times 256 = 383$, which is less than 480. If you change n_1 to a 0 and n_2 to a 2, you get $0 + 2 \times 256 = 512$. Now it's too high! See the problem?

So, what can these users do? Well, mainly try not to get discouraged. A lot can be done in 383 columns of dots, and if you *must* have more than 383 columns of graphics, there is a solution. Just enter normal-density Graphics Mode twice on the same line (see the sample at the end of this chapter), once for 383 columns and a second time for 97 columns. This gives coverage of the full 480 columns. It's not elegant, but it averts any crisis and gets the job done.

The second limitation is that 7-bit computers can't fire the top pin of the print head (Figure 11-2). Since each of the top 8 pins corresponds to 1 bit in Graphics Mode, 7-bit computers can only fire the middle 7 pins of the 9-pin print head. You'll see how that works shortly.

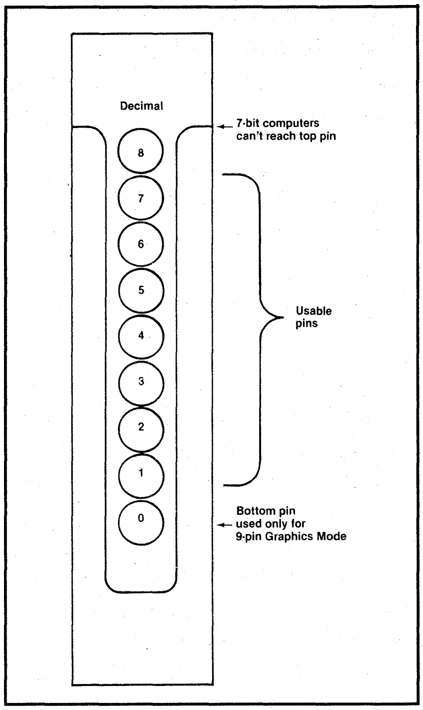

Decimal
7-bit computers can't reach top pin
8
7
6
5
4
3
2
1
0
Usable pins
Bottom pin used only for 9-pin Graphics Mode

Figure 11-2. Usable pins labelled sequentially.

In deference to users who have this problem, sample graphics programs in this manual will be limited to 7 pins whenever possible. Even so, there may be times when 7-bit computer users have to make adjustments. See Appendix H for ways of handling this dilemma.

Firing the pins

So much for problems. Once the printer is in Graphics Mode, the next step is to tell the print head which pins to fire at each new position. This is done by sending numbers via the CHR$ function. Each number sent represents a unique combination of pins.

The eight pins are not numbered 1 through 8 as one might expect. Since computers use the the binary number system (1s and 0s), each pin corresponds to one bit in a binary number. In the decimal system, this means that each pin of the print head must be expressed as a power of two. That's right: 1, 2, 4, 8, 16, 32, 64, and 128.

Why powers of two? Wouldn't it be easier to label the pins 1 to 8? Well, that works fine if you only want to fire one pin at a time, but a 1 to 8 labelling system leads to total chaos if you need to fire multiple pins.

For example, suppose that you want to fire the bottom 3 pins (1, 2, and 3 with a 1 to 8 labelling system). You'd send the printer a 6 (1 + 2 + 3 = 6). But the poor printer has no way of knowing whether you want to fire pin 6 alone or pins 1 and 5 together (1 + 5 = 6), or any other combination that adds up to 6. See the problem?

By assigning powers of 2 to each pin (1, 2, 4, 8, 16, 32, 64, and 128), each pattern of pins corresponds to a unique decimal number from 0 through 255. That's right, no duplicates. With this numbering system, the number 6 represents pins 2 and 4 (since 2 + 4 = 6). There is no other way to get 6 by adding powers of 2. (Sorry, Horace, subtraction is not allowed.) Figure 11-3 shows the pins with their unique labelling numbers.

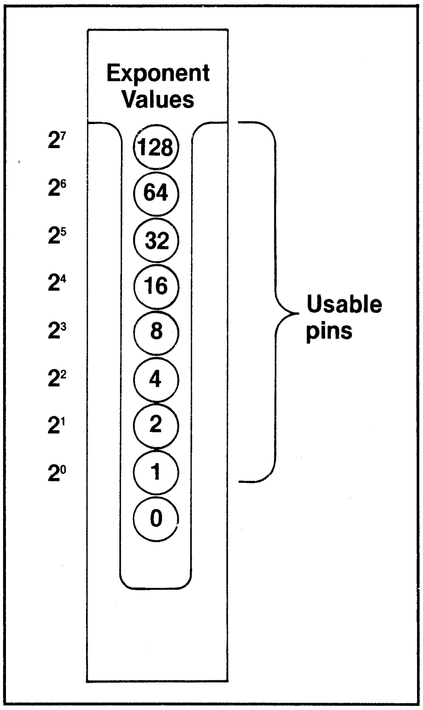


Figure 11-3. Usable pins labelled uniquely.

Testing, testing

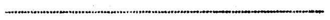
Now that you know the values for the pins, how would you fire the top pin? Why, with LPRINT CHR$(128), of course! And how about the bottom graphics pin? That's right, LPRINT CHR$(1). If you wanted to fire only the top and bottom pins, you'd simply add 128 and 1, then LPRINT CHR$(129). By adding the appropriate numbers together, you can fire any combination of pins you want!

Now you can see why not being able to send an ASCII code above 127 is the same as not being able to fire the top pin.

First Graphics Program

Our first testing of the FX-80's graphics potential will consist of firing the bottom graphics pin. We chose this pin because . . . the only way to go from there is up. Enter and RUN this NEW program (be careful to include the trailing semicolons):

```
NEW
1Ø LPRINT CHR$(27)"K"CHR$(1ØØ)CHR$(Ø);
15 ' Users who can't send CHR$(Ø) see Appendix I.
2Ø FOR X=1 TO 1ØØ
3Ø LPRINT CHR$(1);
4Ø NEXT X
5Ø LPRINT
```



This program deserves a full discussion:

Line 10 prepares the printer to accept 100 columns of graphics data. (Remember n_1 and n_2?)

Line 20 starts a loop for the LPRINT statement. Note that the loop matches the number of columns specified in line 10. The printer is expecting 100 bytes of data; it interprets everything as graphics data until this quota is filled.

Line 30 sends a 1 to fire the bottom pin. The trailing semicolon (;) is necessary to delay the carriage return (ASCII 13) that is sent automatically at the end of each LPRINT line. Otherwise the printer would receive the sequence 1,13,1,13 . . . instead of 1,1,1,1.

Line 40 completes the loop.

Line 50 doesn't print anything—it just provides a carriage return to force the return at the end of the print line to override the trailing semicolon in line 30. Forcing the carriage return is not really necessary since the line is the last one of this program. It's just a good habit to develop.

Graphics Width Limitations

By adjusting n_1 and n_2, users can print up to 480 columns with a single ⟨ESC⟩"K" sequence. The 7-bit limitation of some systems requires a little more effort to get the job done. Everyone change lines 10 and 20 to:

```
10 LPRINT CHR$(27)"K"CHR$(224)CHR$(1);
20 FOR X=1 TO 480
```

If you are an 8-bit user, RUNning this program should give you a line that stretches clear across the page. If you are a 7-bit user, you will get much less. However, you can achieve a full-page line with this program:

```
10 LPRINT CHR$(27)"K"CHR$(127)CHR$(1);
20 FOR X=1 TO 383: LPRINT CHR$(1);: NEXT X
30 LPRINT CHR$(27)"K"CHR$(97)CHR$(0);
40 FOR X=1 TO 97: LPRINT CHR$(1);: NEXT X
50 LPRINT
```

This program simply breaks the 48 columns into two smaller chunks. You know the old saying: if your barn door isn't wide enough, raise smaller cows.

Meditate on this stuff for a while. Experiment with different pin patterns, then move on to the next chapter for some graphics programming techniques.

Summary

⟨ESC⟩"K"CHR$($n_1$)CHR$(n_2);	Enters graphics mode and specifies width setting. Width = $n_1+256*n_2$ where: $0<=n_1<=255$ and $0<=n_2<=7$.

Note: Graphics dots are printed 60 per inch horizontally and 72 per inch vertically.

Chapter 12
Printer Graphics: Firing Single Pins

The key to understanding high-resolution graphics on the FX-80 printer is learning how to fire the pins of the print head. Toward this end, we've devoted an entire chapter to exercising these pins one at a time.

Controlling the Pins

The following powers-of-two fire the pins on the print head from the top down: 128, 64, 32, 16, 8, 4, 2, 1. Firing individual pins is simply a matter of sending these numbers (one at a time) to the printer. And, to make things easier, you can use the form 2^X and let the exponent X vary between 0 and 7. Here is the relationship between the normal and the exponential forms of these numbers:

```
2^7 = 128
2^6 = 64
2^5 = 32
2^4 = 16
2^3 =   8
2^2 =   4
2^1 =   2
2^0 =   1
```

Note: We use the carat (^) to represent exponentiation. For example, 2^6 means raise 2 to the 6th power. Some other systems use the up-arrow (↑), which prints as a left bracket ([ ) on the FX-80.

Using the form 2^X, you can fire individual pins by letting X vary between 0 and 7. Here's how it works. To exercise the pins in a pattern (a slash) that shows off their placement, enter:

```
NEW
4Ø LPRINT CHR$(27)"K"CHR$(7)CHR$(Ø);
45 ' Model I users see Appendix I
8Ø FOR X=Ø TO 6
11Ø LPRINT CHR$(2^X);
12Ø NEXT X
```

When X equals 0, 2^X is 1—so the bottom graphics pin is fired. When X equals 1, 2^X is 2—so the second pin is fired. This pattern continues right up through X equals 6, which fires the seventh pin. We purposely omit X equals 7 to accommodate systems that are limited to seven bits.

Changing direction

The next step is to change the direction of the slash. Can you guess how it's done? Sure, just reverse the order of the exponents, and the same routine can be used. In fact, we'll turn it into a subroutine:

```
1Ø F=Ø
4Ø LPRINT CHR$(27)"K"CHR$(14)CHR$(Ø); ' Change 7 to
   14
5Ø GOSUB 8Ø: F=1: GOSUB 8Ø
7Ø LPRINT CHR$(27)"@": STOP
8Ø FOR X=Ø TO 6 ' No change
9Ø N=X: IF F=1 THEN N=6-X
11Ø LPRINT CHR$(2^N); ' Change X to N
12Ø NEXT X: RETURN ' Add RETURN
```

The routine is initially called with N equal to X, so the exponents increase in order from 0 to 6. The second time the routine is called, N is equal to 6 minus X, which reverses the order (from 6 down to 0). F is used as a flag to activate the change of direction in subroutine 80. Line 90 selects the correct value for the exponent N.

This two-directional slash routine can be repeated indefinitely. For an interesting variation, we can alternate the direction of the slashes each time a pair is printed by changing the flag F in line 50. Make this change:

```
20 FOR L=1 TO 2
30 FOR J=0 TO 9
50 GOSUB 80: F=1-F: GOSUB 80 ' Change F=1 to F=1-F
60 NEXT J: LPRINT: NEXT L
```

Line 50 makes F alternate between 0 and 1. The J loop repeats pairs of diagonals on one line. The L loop adds a second line.

Graphics Line Spacing

For most graphics programs, you'll want to change from the normal 12-dot line spacing to 7-dot (or 8-dot spacing if you use the eighth bit). Change the line spacing in line 10 of your current program, which will then look like this:

```
10 F=0: LPRINT CHR$(27)"1"
20 FOR L=1 TO 2
30 FOR J=0 TO 9
40 LPRINT CHR$(27)"K"CHR$(14)CHR$(0);
50 GOSUB 80: F=1-F: GOSUB 80
60 NEXT J: LPRINT: NEXT L
70 LPRINT CHR$(27)"@": STOP
80 FOR X=0 TO 6
90 N=X: IF F=1 THEN N=6-X
110 LPRINT CHR$(2^N);
120 NEXT X: RETURN
```

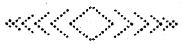
See what a big difference the line spacing makes? All of the multiple-line graphics programs in this manual use this line spacing.

Variable Control

This next and final version of the program exercises even more control over the slashes. It varies not only their direction, but also their sizes (length and height) on the print line. Although the program still uses only one subroutine, it prints a different pattern each time. The differences in the patterns are achieved by the IF-THEN tests in the subroutine.

To get the final listing, add line 100 and change lines 10, 30, 40, 50, 80, 90, and 120:

```
1Ø LPRINT CHR$(27)"1" ' Delete F=Ø:
2Ø FOR L=1 TO 2
3Ø FOR J=Ø TO 1 ' Change 9 to 1
4Ø LPRINT CHR$(27)"K"CHR$(27)CHR$(Ø); ' Change 14 to
  27
5Ø GOSUB 8Ø
6Ø NEXT J: LPRINT: NEXT L
7Ø LPRINT CHR$(27)"@": STOP
8Ø FOR X=1 TO 6: Y=X: IF J=1 THEN Y=7-X
9Ø FOR Z=Ø TO Y: N=Z: IF J=1 THEN N=Y-Z
1ØØ IF L=2 THEN N=7-N
11Ø LPRINT CHR$(2^N);
12Ø NEXT Z: NEXT X: RETURN ' Add NEXT Z:
```



It may not be obvious from the printout that there are two lines of print—but there are. The first line prints slashes by moving from bottom to top and then from top to bottom, whereas the second line prints them first from top to bottom and then from bottom to top.

And how do the IF-THEN statements fit into the picture? The one in line 80 changes the length of the slashes. The one in line 90 controls the direction and height. The IF-THEN in line 100 makes the bottom row a mirror image of the top.

Although there were no new commands presented in this chapter, the control gained is a very important step in mastering bit-image graphics. In Chapter 13 you will learn about several different graphics modes that are available on the FX-80 and also the way to use READ and DATA statements to define unique pin patterns.

Chapter 13
Graphics Pin Patterns and New Graphics Modes

So far, our graphics applications have been limited to working with a single pin per column. As you'll see in later chapters, adding the pin numbers to get pin combinations makes for some interesting effects.

Here's a program that fires the bottom four graphics pins in each column. Since these pins are labelled 1, 2, 4, and 8, and since the sum of these four label numbers is 15, we'll send a CHR$(15) to the printer. Type:

```
NEW
2Ø A$=CHR$(27)+"K"+CHR$(12Ø)+CHR$(Ø)
25 ' TRS Model I users see Appendix I
3Ø B$=CHR$(15)
4Ø LPRINT A$;: FOR X=1 TO 12Ø: LPRINT B$;: NEXT X
8Ø LPRINT CHR$(27)"@"
```

Sure enough, CHR$(15) fires the bottom four pins.

This is basically the same program as the one we used to start the last chapter, but we've added a few new wrinkles. We've stored the string used to enter Graphics Mode in the variable A$ and the string for data in B$. This makes it easy to enter a Graphics Mode several times in a program.

Software Notes

Astute observers will notice the trailing semicolon after the B$ in line 40. If this line were text rather than graphics data, the Master Reset Code in line 80 would be placed in the same text buffer and consequently would wipe the text out. This doesn't happen with the graphics command and data because they are gone by the time the Master Reset Code makes its appearance. The ⟨ESC⟩"K" command has cleared the buffer.

To see this in action, add a line to your program:

```
    5Ø INPUT A
```

The INPUT causes the program run to pause so that you can see that the graphics design is printed before the program reaches the Master Reset Code in line 80. To let the program finish, press ⟨ENTER⟩.

There's an important correlation to note. Anything that was in the buffer prior to the graphics string remains unscathed, and anything left behind is erased. To verify this, make these changes in your program:

```
    35 LPRINT "PROTECTED TEXT";
    5Ø LPRINT "THIS WILL NOT PRINT";
```

and RUN it once.

In general, you need to remember that the Master Reset Code poses no threat to graphics data. But also remember to be careful about using trailing semicolons after text when the Master Reset Code is lurking in the shadows.

Before continuing, delete lines 35 and 50.

More Software Notes

Sending codes from BASIC can be a problem in printing graphics just as it is with printing text. The problem again is that the computer handles some of the control codes—the codes between 0 and 31—in a special way instead of delivering them to the printer as requested.

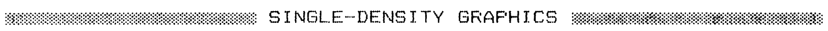
For example, one computer tries to handle form feeds by itself. It counts lines to keep track of the Top Of Form. If a CHR$(12) is sent, the computer intercepts it and sends instead a series of line feeds (CHR$(10)). You can see how this would upset the printing of graphics. TRS-80 users should watch out for numbers 0, 10, and 12; Apple users, for 9 and 13; and those of you with other systems may have your own sets of surprise codes.

Does this mean that you can't use these pin patterns in your graphics programs? Well . . . yes, it does. At least, not via the usual CHR$ function. But you can often design around these problems or POKE the trouble codes directly to the printer. Learning how to cope with these problems is part of learning how to use the printer with your computer. See Appendix H for a discussion of alternatives.

Changing Densities

There is a way to compress the density of dots used for graphics that is comparable to changing from Pica to Compressed Mode for text. It's just a matter of entering a different one of the several Graphics Modes. Before you try any of them out, make these changes in your current program:

```
3Ø B$=CHR$(85)+CHR$(42)
4Ø LPRINT A$;: FOR X=1 TO 6Ø: LPRINT B$;: NEXT X
5Ø LPRINT " SINGLE-DENSITY GRAPHICS ";
6Ø LPRINT A$;: FOR X=1 TO 6Ø: LPRINT B$;: NEXT X
```



```
SINGLE-DENSITY GRAPHICS
```

The printer fires pins 1, 3, 5, and 7 in the first column and pins 2, 4, and 6 in the second. And it alternates that sequence for 120 columns—120 columns in the normal density.

The program also mixes graphics and text on the same line. It does that by using trailing semicolons to keep both kinds of output on the same print line.

High-Speed Double-Density Graphics Mode

Now let's print the same pattern in twice the normal (Single-Density Mode) density. Make these changes:

```
2Ø A$=CHR$(27)+"Y"+CHR$(12Ø)+CHR$(Ø) ' Change K to Y
5Ø LPRINT " HIGH-SPEED DOUBLE-DENSITY GRAPHICS ";
```

There are still 120 columns, only they are pressed together and printed in half the space. They are also printed at the same speed as they are in Single-Density Graphics.

But this High-Speed Double-Density Graphics Mode has one drawback. Because the dots are so closely packed, dots in the same row cannot appear in two consecutive columns. In the above program we've avoided the problem by never printing the same dot twice in a row. If you try to print two consecutive dots, the printer simply ignores the second one (see Figure 13-1).

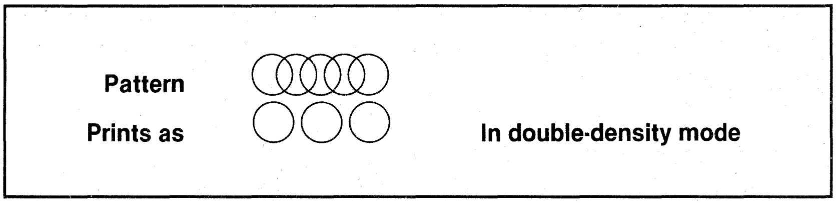


Figure 13-1. Double-Density handling of dots in consecutive columns.

To check this out, change the pin patterns in line 30 to:

```
30 B$=CHR$(127)+CHR$(42) ' Change 85 to 127
```

As Figure 13-2 shows, the dots from the CHR$(42) are not printed at all. The print head is simply moving too fast to retract the pins and then instantly fire them again.

Low-Speed Double-Density Graphics Mode

Ah, but the FX-80 has a special print mode for this very problem. It's called the Low-Speed Double-Density Graphics Mode. Try these lines:

```
20 A$=CHR$(27)+"L"+CHR$(120)+CHR$(0) ' Change Y to L
50 LPRINT " LOW-SPEED DOUBLE-DENSITY GRAPHICS ";
```

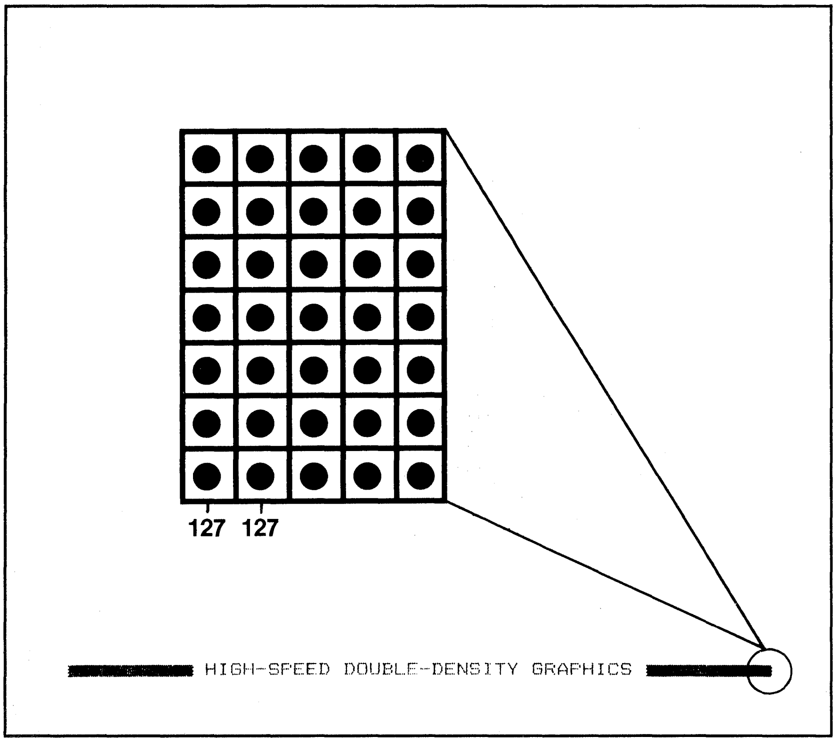


Figure 13-2. No overlapping dots.

Take note of the print speed when you RUN your program this time. It's the same density as in the previous mode, but printed at half the speed. A close look at Figure 13-3 reveals that this time the CHR$(42) columns are printed as requested.

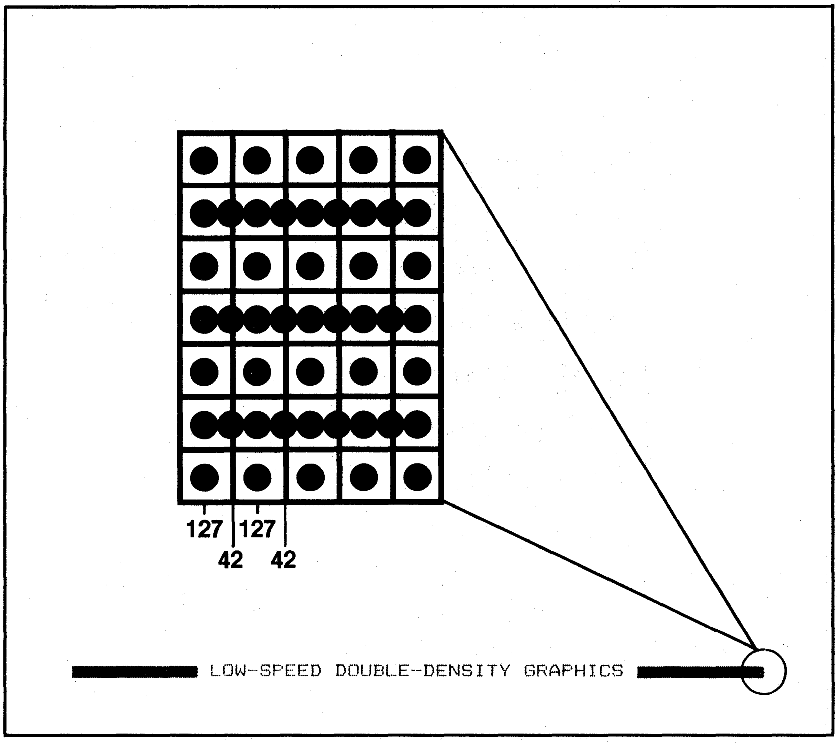
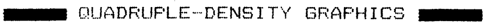


Figure 13-3. Dots do overlap.

Quadruple-Density Graphics Mode

That's right, the FX-80 gives you a mode to print dots four times the normal density. Change these lines to read:

```
20 A$=CHR$(27)+"Z"+CHR$(120)+CHR$(0) ' Change L to Z
30 B$=CHR$(85)+CHR$(42) ' Change 127 to 85
50 LPRINT " QUADRUPLE-DENSITY GRAPHICS ";
```



Very impressive! In Quadruple-Density Graphics Mode, the printer prints 480 times 4 (= 1920) columns of dots on a single line. The setting required to fill the entire line in this mode is n_1 equals 128 and n_2 equals 7, since 7 times 256 plus 128 equals 1920.

Of course, at this density the same limitation as for High-Speed Double-Density Graphics Mode applies—you can't print two adjacent dots in the same row.

More densities

Yes, there are more densities. The FX-80 provides a single ESCape sequence with which you can enter any of seven unique graphics modes with ease—including the modes just covered. The format of this command is:

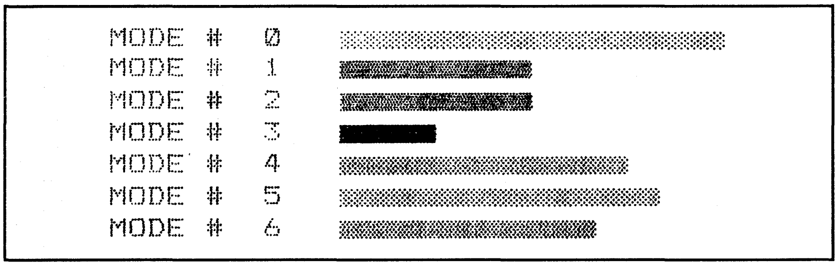
```
LPRINT CHR$(27)"*"CHR$(m)CHR$(n1)CHR(n2);
```

where m indicates the number (0 to 6) of the desired Graphics Mode. There are six special densities available (Double-Density is part of two modes, one for each print speed). The settings n_1 and n_2 are the usual graphics width settings.

You can print all the modes by making these changes to your listing:

```
DELETE 4Ø

1Ø FOR M=Ø TO 6
2Ø A$=CHR$(27)+"*"+CHR$(M)+CHR$(12Ø)+CHR$(Ø)
5Ø LPRINT "MODE # ";X;TAB(12);
7Ø LPRINT: NEXT M
```



```
MODE #  Ø
MODE #  1
MODE #  2
MODE #  3
MODE #  4
MODE #  5
MODE #  6
```

Figure 13-4. The seven modes for densities.

Figure 13-4 displays all seven of the FX-80 modes that affect density. Table 13-1 describes them.

Table 13-1. Description of the Seven Density Modes.

Mode	Density	Alternate Code	Description	Head Speed (In./Sec.)
0	Single	⟨ESC⟩"K"	60 dots per inch; 480 dots per 8" line	16
1	Low-Speed Double	⟨ESC⟩"L"	120 dots per inch; 960 dots per 8" line	8
2	High-Speed Double	⟨ESC⟩"Y"	Same density as Mode 1, but faster. The printer does not print consecutive dots in any one row.	16
3	Quadruple	⟨ESC⟩"Z"	240 dots per inch; 1920 dots per 8" line. The printer does not print consecutive dots in any one row.	8
4	QX-10		Matches the screen density of Epson's QX-10: 80 dots per inch; 640 dots per 8" line. (This makes it easy to do screen dumps.)	8
5	One-to-One (Plotter)		72 dots per inch 576 dots per 8" line. Produces the same density horizontally as vertically, which makes circles look round.	12
6	CRT Screens		90 dots per inch; 720 dots per 8" line. Matches the Corvus CONCEPT and the DEC screens.	8

All seven of these modes can be accessed with the format:

⟨ESC⟩"*"CHR$(n)

where n represents the mode number. The first four modes also can be accessed with their alternate codes.

9-Pin Graphics Mode

Recall that in the last chapter we said that the bottom (ninth) pin of the print head is not normally used in the Graphics Modes. That's because most microcomputers communicate with parallel type peripheral devices using eight data lines (even if they have 16-bit processors). With a printer, each data line corresponds to one pin on the print head. Thus each byte sent will fire up to eight pins.

But the printer has nine pins available. So how do we fire the ninth pin with only eight data lines? In fact, do we really want to bother with just one extra pin? Well, for such graphics-intensive applications as screen dumps, printing nine pins at a time can speed up the process considerably. For this purpose, the FX-80 has a special 9-Pin Graphics Mode (but not for 7-bit computers). In this mode the printer takes two bytes to fire all nine pins—as shown in Figure 13-5.

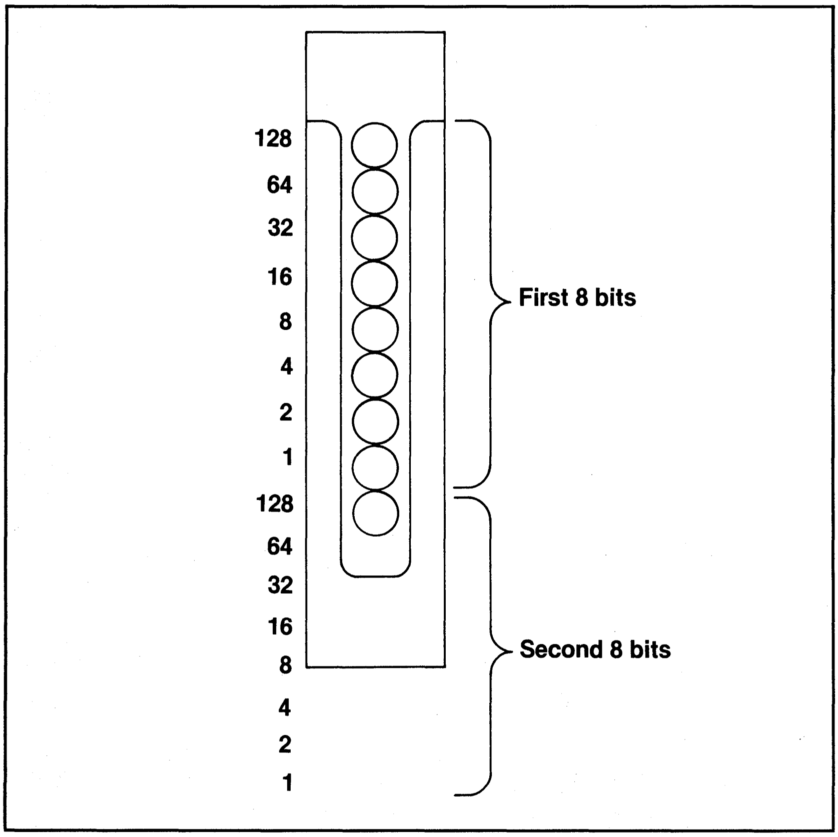


Figure 13-5. 9-pin graphics.

Since computers are faster than printers, there is no significant time loss in printing a single line of graphics with nine pins. You get nine dots per line in about the same time as you get eight dots in the other Graphics Modes. Not a bad deal!

The format for entering 9-Pin Graphics Mode is:

```
LPRINT CHR$(27)"^"CHR$(d)CHR$(n1)CHR$(n2);
```

(Use CHR$(94) if you can't generate the caret symbol (^) from your keyboard.) The d determines the density of the graphics: d set to 0 produces single density; d set to 1 produces double density.

The CHR$($n_1$) and CHR$(n_2) represent the usual width settings, but each print pattern requires two bytes (instead of one). For example, to print 60 columns of graphics, you must send 120 data bytes.

Firing nine pins with eight data lines is just a shade more difficult than firing seven or eight pins. It takes two bytes to define each 9-dot pin pattern: the first byte determines the pattern of the top eight pins in the usual way, and only the top bit of the second byte is used. Thus any second byte of 128 or greater fires the bottom pin of the print head; anything less does not. Try this sample program:

```
DELETE 1Ø
DELETE 5Ø
DELETE 7Ø
2Ø A$=CHR$(27)+CHR$(94)+CHR$(Ø)+CHR$(6Ø)+CHR$(Ø)
3Ø B$=CHR$(85)+CHR$(Ø)+CHR$(42)+CHR$(128)
6Ø LPRINT A$;: FOR X=1 TO 3Ø: LPRINT B$;: NEXT X
```

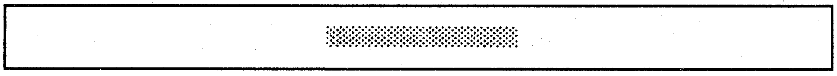


Figure 13-6. Printout to check bottom pin.

Compare this with the densities in Figure 13-4 (this one is single density). Look closely at Figure 13-6; you'll see that the bottom pin is printed in every other column.

For fans of 9-Pin Graphics, the ⟨ESC⟩"0" line spacing is ideal: it sets the line spacing to 9/72nds of an inch (9 dots).

Pattern Design

The next phase in printing graphics is to arrange pin firing sequences into meaningful designs. Figure 13-7 shows how a design can be sketched on graph paper.

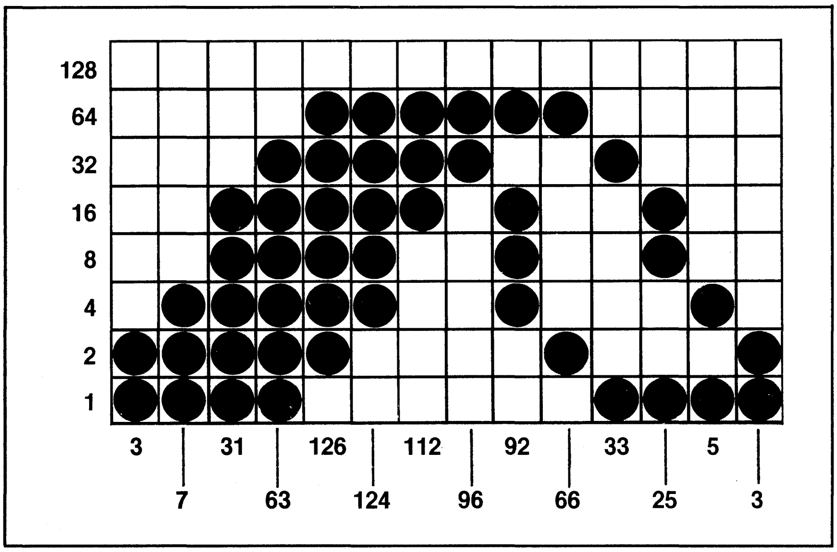


Figure 13-7. Sketch of a graphic design.

In Figure 13-7, the dots correspond to the pin numbers shown on the side of the figure. The number for each pin pattern is shown at the bottom of each column.

Once the numbers for the pin patterns are calculated, they can be stored in DATA statements. Items in a DATA statement are separated by commas. These items are read from the DATA statement into variables with a READ statement. Add these lines to your program:

```
50 READ N
90 DATA 3,7,31,63,126,124,112,96,92,66,33,25,5,3
```

Line 50 reads the first data number into the variable N. To read the rest of the numbers, line 50 must be executed in a loop. Make these changes to the program:

```
2Ø A$=CHR$(27)+"K"+CHR$(14)+CHR$(Ø)
3Ø LPRINT A$;
4Ø FOR X=1 TO 14
6Ø LPRINT CHR$(N);
7Ø NEXT X
```

Perfect! Just like the plan.

Repeating a pattern

Now how would you go about repeating this same pattern—add more data? Here you don't need to. For simple programs like this one, a RESTORE statement works quite well. Change two lines:

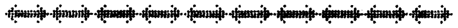
```
3Ø FOR Y=1 TO 1Ø: RESTORE: LPRINT A$;
7Ø NEXT X: NEXT Y
```

The new loop in line 30 repeats the pattern 10 times. But you don't need 10 repetitions of the DATA statements. The RESTORE statement in line 30 tells the program to read the same data again. Used with restraint, the RESTORE statement can reduce the amount of data needed.

Repeating DATA numbers

Sometimes it is necessary to repeat the same DATA number several times in succession. Try this change:

```
9Ø DATA 8,28,62,93,28,28,28,28,28,28,93,62,28,8
```



This program fires the middle three pins (CHR$(28)) 6 times in succession. But can you imagine typing the data for 20 or 50 repetitions of the same number? There has to be a better way—and there is.

You can enter the number of repetitions right into the DATA lines, coded as a negative number. Then change the READ routine to test for a negative number. When a negative number is read, the program will transfer control to a special subroutine that does the repeating (or use a STRING$ statement). Try this change:

```
50 READ N: IF N<0 THEN 100
90 DATA 8,28,62,93,-6,28,93,62,28,8
100 READ R: FOR J=1 TO -N
110 LPRINT CHR$(R);: NEXT J
120 X=X-N-1: GOTO 70
```

And RUN it again. Same thing, right? And with less data. The number of repetitions (6) is entered into the DATA line as a negative number that is followed by the pattern (28) to be repeated. Yet even with this short cut, graphic designs require lots of data.

Summary

⟨ESC⟩"K"	Single-Density Graphics Mode
⟨ESC⟩"Y"	High-Speed Double-Density Graphics Mode
⟨ESC⟩"L"	Low-Speed Double-Density Graphics Mode
⟨ESC⟩"Z"	Quadruple-Density Graphics Mode
⟨ESC⟩"*"CHR$(n)	Variable Graphics Density Setting: 0<=n<=6
⟨ESC⟩"^"CHR$(0)	Single-Density 9-Pin Graphics Mode
⟨ESC⟩"^"CHR$(1)	Double-Density 9-Pin Graphics Mode

Chapter 14
Two Types of Graphics Programs

In this chapter we take you through the development of two graphics programs, from design to implementation. In the process, you'll have a chance to explore the way data are used to produce more complex designs.

The two programs use entirely different techniques. The first program uses a method of storing and recalling data similar to that used in the ribbon program in the last chapter; pin patterns and their repetition factors are stored in DATA statements. The second program does not read pin patterns from DATA statements. Instead, the few pin patterns needed are coded into the program as constants, while the DATA lines contain only the number of repetitions for the patterns.

These examples show how easy it is to create high-resolution dot graphics on your FX-80. We hope they inspire you to include graphics in your own programs.

The Planning Process

It should be apparent by now that printing high-resolution images requires careful planning and lots of data. There are programs available for some computers that enable users to draw figures on the screen, then dump them to the printer. Without such a program, there really is no quick and easy way to calculate the data required for pin patterns.

The usual routine goes like this:

1) Plot the designs, dot by dot, on graph paper.
2) Translate the dots into their appropriate pin numbers, seven or eight rows at a time.
3) Figure out the easiest way to send those numbers to the printer.

It's not as hard as it sounds! It just requires some patience. And, where symmetrical designs are involved, the computer can do most of the work. Follow the development of the programs in this chapter, then get yourself some graph paper and a pencil, and have a good time.

Logo Design

The first program creates a sample graphics logo intended for use as a letterhead for business forms. Since we use it that way in a later chapter, enter the line numbers exactly as shown.

The first step is to design the logo on graph paper; this layout is shown in Figure 14-1. The figure is designed to be printed in Double-Density Graphics Mode.

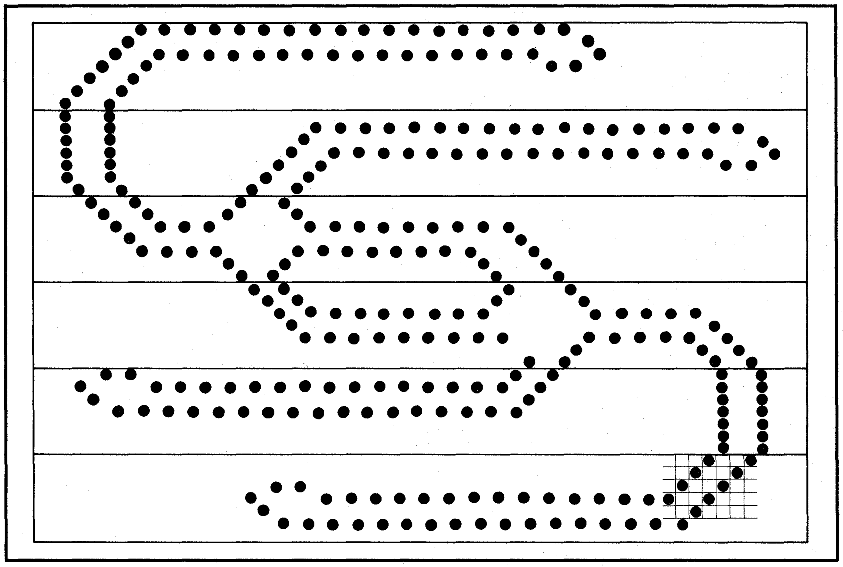


Figure 14-1. Logo layout.

Once the figure is conceived and drawn, it is enlarged and split into six rows of seven dots each, and then the pin patterns are calculated and recorded.

Here is the DATA line for the first row. Type:

```
NEW
1200 DATA 0,1,2,4,9,18,36,72,-16,16,
   64,8,64,8,32,16,0,-7,0,0,128
```

Handling of the data is fairly straightforward. In most cases, the program merely reads a number and sends it to the printer:

```
710 READ N: IF N=128 THEN 770
720 IF N>=0 THEN LPRINT CHR$(N);: GOTO 710
725 ' Apple and Model I users see Appendix I
```

If N happens to be negative, control passes through line 720 to 730. Negative numbers in the DATA lines represent repeat factors as they did in the last chapter, but the repeat routine is slightly different. This program repeats numbers in pairs:

```
730 READ P,R: FOR J=1 TO -N:
    LPRINT CHR$(P)CHR$(R);: NEXT J
740 GOTO 710
```

The -16 in line 1200 is read into N. Since it is negative, line 730 reads the next two numbers (16 and 64) into P and R and prints them in a loop. Control then returns to line 710, which READs the next number.

Line 710 also tests for the number 128, a "stopper" number. Since 127 is the largest number that can fire 7 pins, the 128 can be placed at the end of each DATA line to signal the end of the print line. When the 128 is read, control passes to line 770, which produces the necessary line feed. Now add:

```
770 LPRINT
```

Note that the only exit from the continuous loop is through line 770.

There's only one thing left to do before printing the first line—enter a Graphics Mode:

```
700 LPRINT CHR$(27)"L"CHR$(60)CHR$(0);
```

Now RUN the program.

That's a good start. To complete the program, add 7-dot line spacing and a loop to process the last five lines of DATA:

```
100 LPRINT CHR$(27)"1"
690 FOR K=1 TO 6
770 LPRINT: NEXT K: LPRINT CHR$(27)"@": STOP
1199 ' <<< LOGO DATA >>>
1210 DATA Ø,126,1,Ø,126,1,-5,Ø,Ø,1,2,4,9,
     18,36,-16,8,32,4,32,4,16,8,Ø,128
1220 DATA Ø,Ø,Ø,64,32,16,72,36,-3,16,4,34,
     65,Ø,Ø,65,34,-8,16,4,18,9,4,2,1,Ø,-9,Ø,Ø,128
1230 DATA -8,Ø,Ø,64,32,16,72,36,16,-7,
     4,16,36,65,Ø,Ø,1
     66,36,16,-3,4,16,4,18,9,4,2,1,-2,Ø,Ø,128
1240 DATA Ø,32,16,64,8,64,-15,8,32,72,16,
     32,64,-6,Ø,Ø,Ø,127,Ø,Ø,127,Ø,Ø,Ø,128
1250 DATA -7,Ø,Ø,Ø,8,4,16,2,16,-15,2,8,18,
     36,72,16,32,64,-2,Ø,Ø,128
```

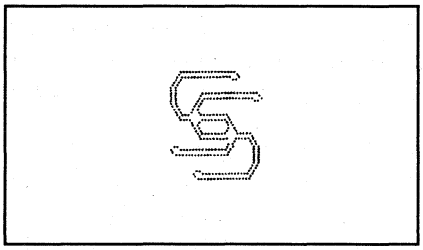


Figure 14-2. Double-S logo.

And there you have it . . . Figure 14-2, a genuine logo which could be used on all types of correspondence. SAVE this program as LOGO.

3-Dimensional Design

With a little imagination and creativity, a lot can be done with dot graphics. Take a look at the 3-dimensional design of Figure 14-3, which spells out the name of your favorite printer.

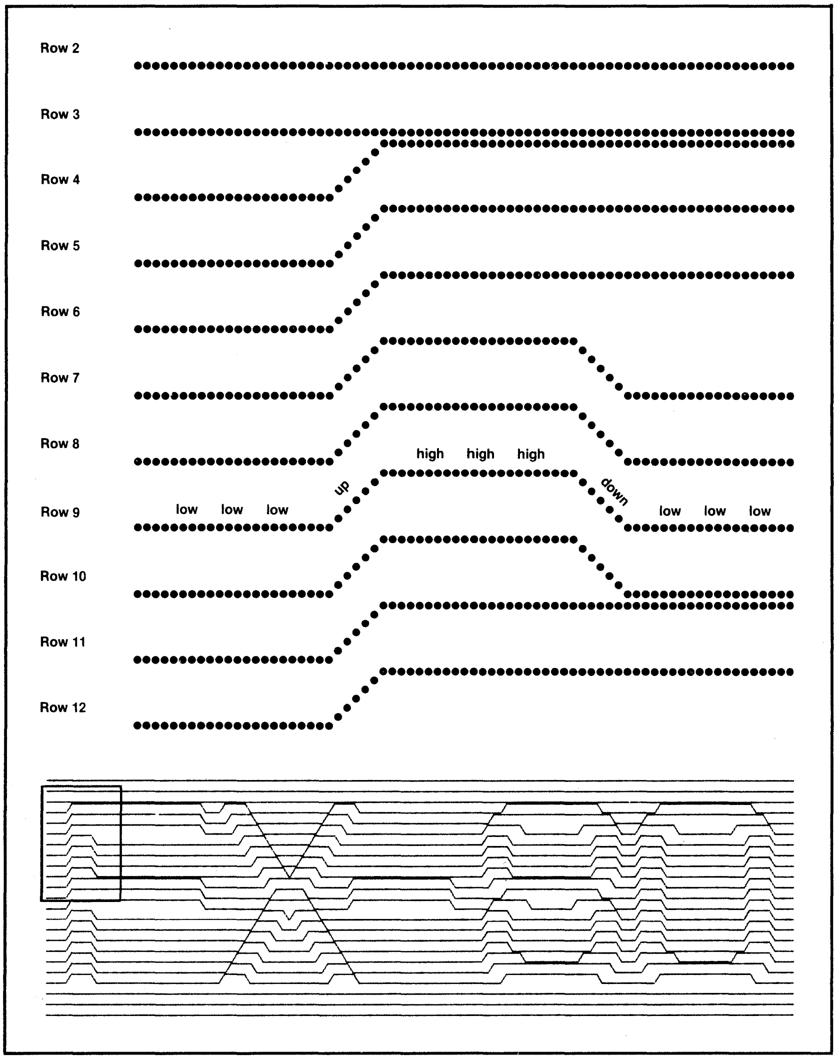


Figure 14-3. Analysis of a corner of FX-80 design.

A very few pin patterns are needed for this program. In fact, each "pattern" consists of only one pin, which means the numbers are easy to calculate:

1	for the low pin
64	for the high pin
1, 2, 4, 8, 16, 32, 64	for the diagonal rise
64, 32, 16, 8, 4, 2, 1	for the diagonal fall

These pin patterns are coded right into the program. Only the number of repetitions of the low and high sections are stored as data.

A close look at Figure 14-3 reveals that most of the lines are made by repeating a 4-step pattern:

1) Fire the bottom pin (pin 1), repeat L times.

2) Draw a diagonal rise (pins 1 - 64).

3) Fire the top pin (pin 64), repeat H times.

4) Draw a diagonal fall (pins 64 - 1).

This pattern is repeated several times. Printing the figure is mainly a matter of reading the length of the low and high sections, then printing the 4-part cycle.

Entering the program

This program will be entered in chunks, so don't RUN it until we give the word.

Might as well start off with the easy stuff. First, set the line spacing for 7-pin graphics:

```
NEW ' SAVE the LOGO program first
10 LPRINT CHR$(27)"1"
```

Note: Use 〈ESC〉 "3" line spacing if necessary.

Next up are the three straight lines at the start of the design. There's no need for anything fancy—just a single dot printed across the page. Add these lines:

```
20 G$=CHR$(27)+"L"+CHR$(51)+CHR$(3): GOSUB 160
150 LPRINT CHR$(27)"@": STOP
160 FOR X=1 TO 3: LPRINT G$;
170 FOR Y=1 TO 819: LPRINT CHR$(1);: NEXT Y
180 LPRINT: NEXT X: RETURN
```

Line 20 stores the graphics entry string into G$. This string produces Low-Speed Double-Density graphics of 819 columns (51 + 3 x 256 = 819). Line 170 fires the bottom graphics pin 819 times. The X loop repeats the routine to print the line 3 times.

Because this line-printing routine (lines 160 to 180) will be used again to print the bottom three lines of the figure, it is set aside as a subroutine. It is called by the GOSUB in line 20 and separated from the rest of the program by the STOP in line 150.

So far, so good! Now for the rest of the figure. The following lines control the 4-step process:

```
80 FOR X=1 TO L: LPRINT CHR$(1);: NEXT X
100 LPRINT CHR$(1)CHR$(2)CHR$(4)CHR$(8)
    CHR$(16)CHR$(32)CHR$(64);
110 FOR X=1 TO H: LPRINT CHR$(64);: NEXT X
120 LPRINT CHR$(64)CHR$(32)CHR$(16)
    CHR$(8)CHR$(4)CHR$(2)CHR$(1);
```

The 7-dot rise and fall are printed by lines 100 and 120. The lengths of the low and high sections are stored in DATA statements, then read into the variables L and H. Line 80 prints the bottom pin L times; line 110 prints the top pin H times.

The next step is to add the READ portion of the program:

```
50 READ L,H
60 L=L*7: H=H*7
70 IF L=0 THEN 90
90 IF H<0 THEN LPRINT: GOTO 150
130 GOTO 50
```

Line 50 READs numbers from the DATA statements in pairs: the first is stored into L, the second into H. L and H are then each multiplied by seven; this extends the width of the figure without increasing the size of the DATA numbers. This enlargement factor must be the same as the number of dots in the rise and fall, or else the design will not line up properly.

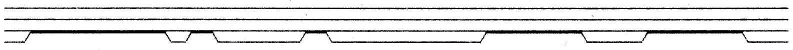
If L is READ as 0, line 70 skips line 80. This enables us to print the center portion of the X, where no low section is required.

Line 90 serves two purposes. It forces a line feed each time a negative number is read, and it skips the last three steps of the 4-step cycle so that each print line can end on a low.

To print the program with one line of data, add:

```
40 LPRINT G$;
190 DATA 3,20,2,3,12,3,22,14,8,14,6,-1
```

and RUN it.



The negative number at the end of line 190 signals the end of the print line.

The last program change is to add a loop to print 17 lines and to enter the DATA statements shown in Figure 14-4:

```
30 FOR D=1 TO 17: PRINT "ROW ";D
90 IF H<0 THEN LPRINT: GOTO 140   ' Change 150 to 140
140 NEXT D: GOSUB 160
200 DATA 3,20,3,3,10,3,21,18,4,18,4,-1
210 DATA 3,20,4,3,8,3,21,5,8,5,2,5,8,5,3,-1
220 DATA 3,3,22,3,6,3,22,3,12,3,2,3,12,3,3,-1
230 DATA 3,3,23,3,4,3,23,3,12,3,2,3,12,3,3,-1
240 DATA 3,3,24,3,2,3,24,3,12,3,2,3,12,3,3,-1
250 DATA 3,3,25,3,0,3,25,3,12,3,2,3,12,3,3,-1
260 DATA 3,20,9,6,5,15,5,18,3,3,12,3,3,-1
270 DATA 3,20,10,4,6,15,7,14,5,3,12,3,3,-1
280 DATA 3,20,9,6,5,15,5,5,6,5,3,3,12,3,3,-1
290 DATA 3,3,25,3,0,3,25,3,12,3,2,3,12,3,3,-1
300 DATA 3,3,24,3,2,3,24,3,12,3,2,3,12,3,3,-1
310 DATA 3,3,23,3,4,3,23,3,12,3,2,3,12,3,3,-1
320 DATA 3,3,22,3,6,3,22,3,12,3,2,3,12,3,3,-1
330 DATA 3,3,21,3,8,3,21,5,8,5,2,5,8,5,3,-1
340 DATA 3,3,20,3,10,3,21,18,4,18,4,-1
350 DATA 3,3,19,3,12,3,22,14,8,14,6,-1
```

Figure 14-4. DATA for the program.

Yes, indeed, high-resolution graphics does require a large amount of data! Okay, now RUN the program:

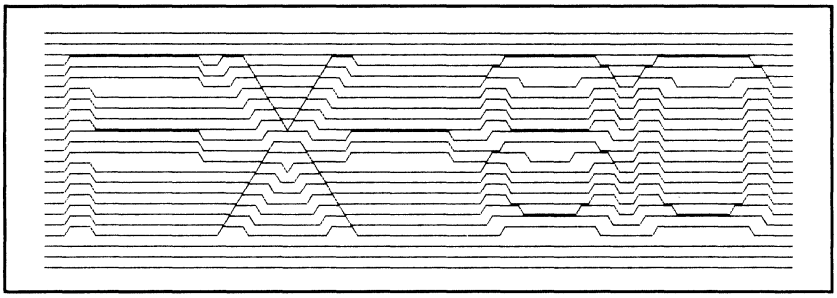


Figure 14-5. FX-80 design.

Success! In your printout (which should look like Figure 14-5), can you see the 3D effect of the letters? Well, the modifications to follow will make the letters easier to read.

Type LLIST to see what you've got up to this point. It should match the listing of Figure 14-6.

Flexing your graphics muscles

Now that you've typed in all that data, it's time to have some fun. Changing a few of the pin patterns can have a dramatic effect on results.

These changes fill in the diagonals as in Figure 14-7:

```
100 LPRINT CHR$(1)CHR$(3)CHR$(7)CHR$(15)
    CHR$(31)CHR$(63)CHR$(127);
120 LPRINT CHR$(127)CHR$(63)CHR$(31)CHR$(15)
    CHR$(7)CHR$(3)CHR$(1);
```

```
10 LPRINT CHR$(27)"1"
20 G$=CHR$(27)+"L"+CHR$(51)+CHR$(3): GOSUB 160
30 FOR D=1 TO 17: PRINT "ROW ";D
40 LPRINT G$;
50 READ L,H
60 L=L*7: H=H*7
70 IF L=0 THEN 90
80 FOR X=1 TO L: LPRINT CHR$(1);: NEXT X
90 IF H<0 LPRINT: GOTO 140
100 LPRINT CHR$(1)CHR$(2)CHR$(4)CHR$(8)CHR$(16)CHR$(32)
    CHR$(64);
110 FOR X=1 TO H: LPRINT CHR$(64);: NEXT X
120 LPRINT CHR$(64)CHR$(32)CHR$(16)CHR$(8)CHR$(4)CHR$(2)
    CHR$(1);
130 GOTO 50
140 NEXT D: GOSUB 160
150 LPRINT CHR$(27)"@": STOP
160 FOR X=1 TO 3: LPRINT G$;
170 FOR Y=1 TO 819: LPRINT CHR$(1);: NEXT Y
180 LPRINT: NEXT X: RETURN
190 DATA 3,20,2,3,12,3,22,14,8,14,6,-1
200 DATA 3,20,3,3,10,3,21,18,4,18,4,-1
210 DATA 3,20,4,3,8,3,21,5,8,5,2,5,8,5,3,-1
220 DATA 3,3,22,3,6,3,22,3,12,3,2,3,12,3,3,-1
230 DATA 3,3,23,3,4,3,23,3,12,3,2,3,12,3,3,-1
240 DATA 3,3,24,3,2,3,24,3,12,3,2,3,12,3,3,-1
250 DATA 3,3,25,3,0,3,25,3,12,3,2,3,12,3,3,-1
260 DATA 3,20,9,6,5,15,5,18,3,3,12,3,3,-1
270 DATA 3,20,10,4,6,15,7,14,5,3,12,3,3,-1
280 DATA 3,20,9,6,5,15,5,5,6,5,3,3,12,3,3,-1
290 DATA 3,3,25,3,0,3,25,3,12,3,2,3,12,3,3,-1
300 DATA 3,3,24,3,2,3,24,3,12,3,2,3,12,3,3,-1
310 DATA 3,3,23,3,4,3,23,3,12,3,2,3,12,3,3,-1
320 DATA 3,3,22,3,6,3,22,3,12,3,2,3,12,3,3,-1
330 DATA 3,3,21,3,8,3,21,5,8,5,2,5,8,5,3,-1
340 DATA 3,3,20,3,10,3,21,18,4,18,4,-1
350 DATA 3,3,19,3,12,3,22,14,8,14,6,-1
```

Figure 14-6. Program for FX-80 design.

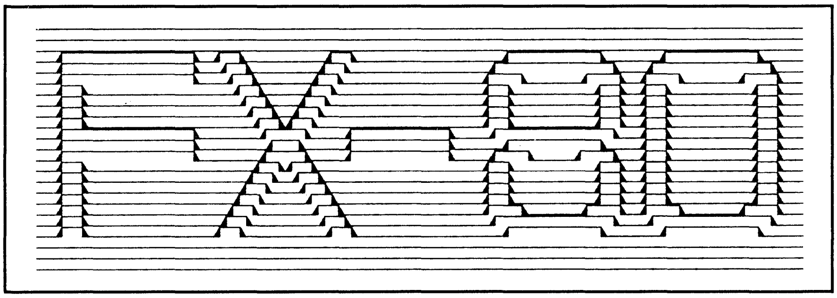


Figure 14-7. More distinct FX-80.

And one additional change fills in the entire FX-80 (Figure 14-8):

```
110 FOR X=1 TO H: LPRINT CHR$(127);: NEXT X
```

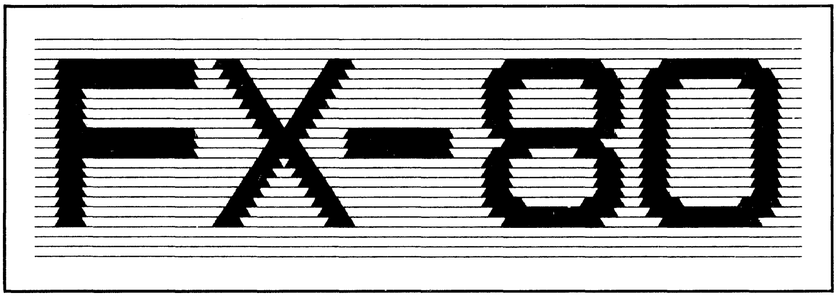


Figure 14-8. Most distinct FX-80.

And now, class, your homework assignment. See if you can do a complete black/white reverse like the one in Figure 14-9.

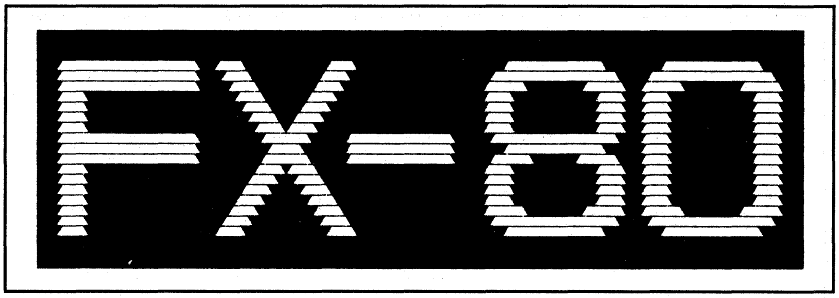


Figure 14-9. Reversed FX-80.

Stay tuned for more exciting graphics techniques in the next chapter.

Chapter 15
Plotter Graphics

As you explore the world of dot graphics, you may run into printer limitations. For instance, dot-matrix printers are designed for fast printing of text. They can also print high-resolution graphics, as you saw in the logo program. But the side-to-side motion of a dot-matrix printer makes it difficult to place the print head in the middle of a page and trace out a lazy spiral or even a circle.

Does that mean the FX-80 can't create the same type of figures as a plotter? Certainly not! You just have to approach the problem a little differently than you would in working with a plotter. In fact, with a little advanced planning, you can use the dot graphics modes to simulate the activity of a full fledged plotter.

The secret to bringing out the plotter in your FX-80 is to apply the talents of your computer. After all, your printer and computer are getting along famously. The computer has a large amount of memory that can be used as a sketch pad. Using mathematical formulas, you can sketch all sorts of crazy patterns in any direction you choose. When the picture is complete in memory, it can be sent line by line to the printer. When you use this approach, the printer doesn't have to act like a plotter to produce like one.

The figures plotted (er, printed) in this chapter show how easy it is to simulate "plotter graphics" with the FX-80. We start by using the computer's memory as a sketch pad for our plotting. To do this, we set up a correspondence between the desired pattern of dots on paper and the computer's memory, as in Figure 15-1.

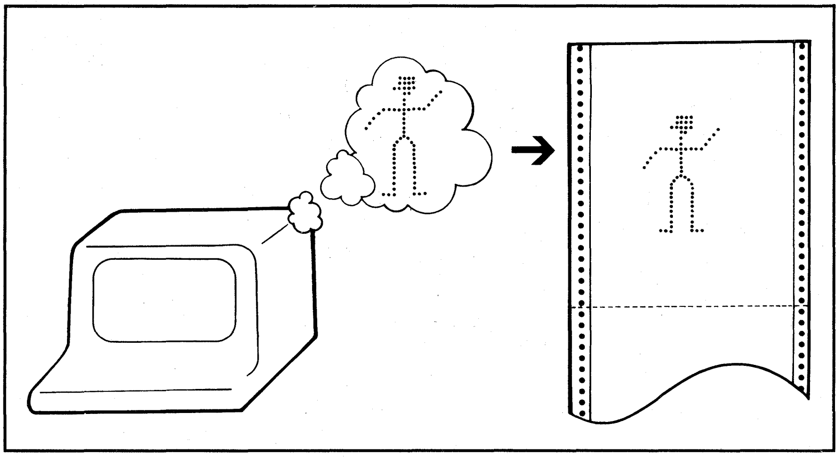


Figure 15-1. Using the computer as a sketch pad.

Arrays

The area of computer memory that stores your pattern of dots is called an array. Think of an array as an ordered set of cubby holes or cells arranged in rows and columns (like Post Office boxes). Each cell of the array corresponds to a dot position on the paper (Figure 15-2).

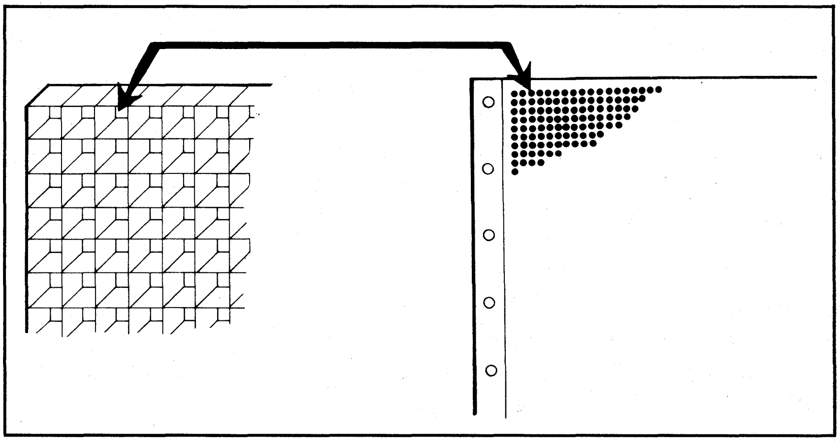


Figure 15-2. An array in memory and on paper.

Although the cells in a numeric array can hold any numeric constant, we only need to concern ourselves with the two numbers one and zero. Figure 15-3 demonstrates using a one to represent a dot and a zero to represent no dot.

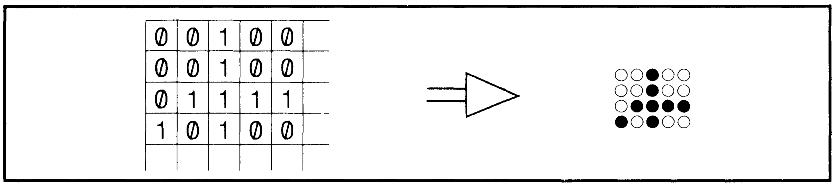


Figure 15-3. Ones and zeros become dots and no-dots.

Why all this fuss and stew about arrays? Our purpose here is to simulate a plotter, remember? And once the correspondence between array cells and dot positions is firmly established, we can easily plot in any direction in the array.

Let's look at the way each cell is named. The cells are arranged in rows and columns, so each cell can be easily pinpointed by its row and column position.

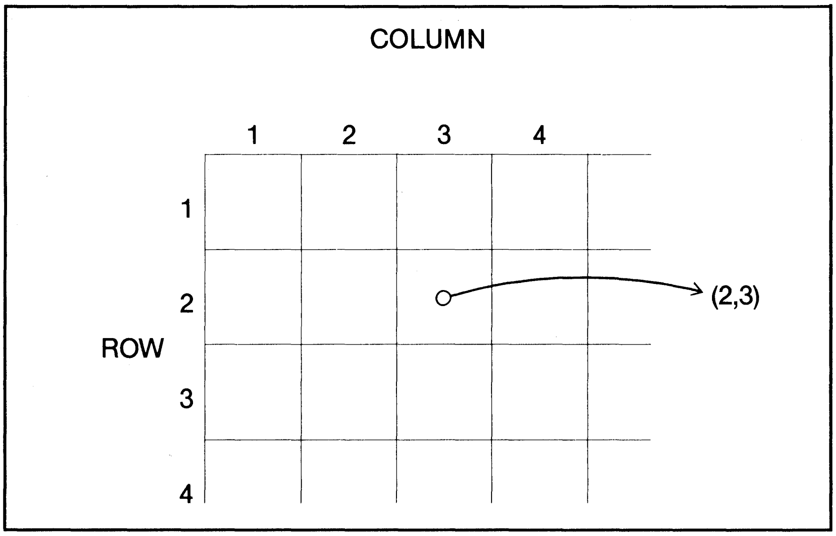


Figure 15-4. Finding a cell's label.

The example cell of Figure 15-4 is at the intersection of row 2 and column 3, so it is labelled cell (2,3). In BASIC, you give the entire array a name, then append the address to the name. If the array of Figure 15-4 is array A, the cell name is A(2,3).

DIMensioning an array

Most BASICs allow the use of up to 10 rows and 10 columns in an array without any special preparation of the computer's memory. However, arrays use up lots of memory, so the system must be informed if you intend to use a larger array. In BASIC, this is done with the DIMension statement. Start a NEW program:

```
NEW
10 DEFINT A: N=21: DIM A(N,N)
15 ' Use DIM A(21,21) if your
   system rebels at line 10
170 LPRINT CHR$(27)"@": STOP
```

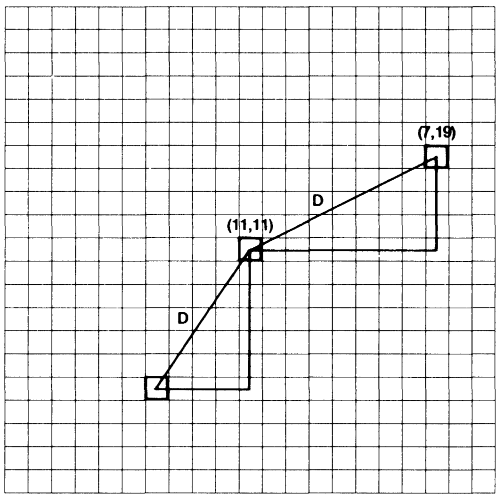
The DIM statement in line 10 reserves enough memory for 21 rows and 21 columns of numbers. That's 441 total cells, and each cell takes up 2, 4, or 8 bytes, depending on the precision of the variables being used. The DEFINT restricts all variables starting with the letter A to be of the integer type (2 bytes), thus saving memory.

Filling an array

To fill the array with ones and zeros, you can use simple LET statements. For example, A(2,3)=1 sets location (2,3) equal to 1, while A(4,1)=0 sets location (4,1) equal to 0. Actually, most computer systems set all numeric variables, including arrays, to 0 each time a program is run. If that is true on your system, then plotting a figure in memory is simply a matter of depositing 1s in the correct positions.

For example, suppose you want to plot a circle of radius 10 in the 21 row by 21 column array—definitely a job for a plotter! You could use the distance formula from high school math (as in Figure 15-5) to calculate the distance from the center cell (11,11) to each of the surrounding cells. If this distance is equal to 10, the cell content is changed to one; otherwise, the cell value remains zero.

Note: If your system does not initialize numeric variables, you'll have to set each cell equal to zero.



$$\leftarrow D = \sqrt{(19-11)^2 + (7-11)^2}$$
$$= \sqrt{8^2 + (-4)^2}$$
$$= \sqrt{64 + 16}$$
$$= \sqrt{80}$$
$$\approx 8.94 \neq 10$$

Figure 15-5. Plotting a circle.

Plotting a Circle

The cells of the array can be examined in any order; the following program uses a row by row scan with two loops:

```
20 FOR R=1 TO N: FOR C=1 TO N
```

At each cell, the distance is calculated with:

```
30 D=SQR( (R-11)^2 + (C-11)^2 )
```

Next we compare this distance with the desired radius of the circle (10). If the distance to the current cell A(R,C) is close to 10, we set its contents equal to one; otherwise, we leave it alone.

```
40 IF INT(D+.5)=10 THEN A(R,C)=1
```

Line 40 sets any cell whose distance is between 9.5 and 10.5 equal to one.

The final step to plotting a circle in the array is to close the loops and display the contents of the array. Add two lines to your program:

```
50 LPRINT A(R,C);: NEXT C: LPRINT
60 PRINT "ROW ";R: NEXT R
```

and RUN it.

```
5 ' FIGURE 15-6
1Ø DEFINT A: N=21: DIM A(N,N)
2Ø FOR R=1 TO N: FOR C=1 TO N
3Ø D=SQR((R-11)^2 + (C-11)^2)
4Ø IF INT(D+.5)=10 THEN A(R,C)=1
5Ø LPRINT A(R,C);: NEXT C: LPRINT
6Ø PRINT "ROW ";R: NEXT R
17Ø LPRINT CHR$(27)"@": STOP
```

```
Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  1  1  1  1  1  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø
Ø  Ø  Ø  Ø  Ø  1  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  1  Ø  Ø  Ø  Ø  Ø
Ø  Ø  Ø  Ø  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  Ø  Ø  Ø  Ø
Ø  Ø  Ø  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  Ø  Ø  Ø
Ø  Ø  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  Ø  Ø
Ø  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  Ø
Ø  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  Ø
1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1
1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1
1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1
1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1
1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1
1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1
1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1
Ø  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  Ø
Ø  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  Ø
Ø  Ø  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  Ø  Ø
Ø  Ø  Ø  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  Ø  Ø  Ø
Ø  Ø  Ø  Ø  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  Ø  Ø  Ø  Ø
Ø  Ø  Ø  Ø  Ø  1  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  1  Ø  Ø  Ø  Ø  Ø
Ø  Ø  Ø  Ø  Ø  Ø  Ø  1  1  1  1  1  1  1  Ø  Ø  Ø  Ø  Ø  Ø  Ø
```

Figure 15-6. Program and RUN for displaying an array.

Figure 15-6 shows the results of our scheming in terms of ones and zeros.

Translating the ones to dots

The next step is to translate the contents of the array to dots on the paper. First, modify line 50 so that it doesn't print the array:

```
5Ø NEXT C
```

Then fill the usual graphics prescription for 7-dot line spacing:

```
    7Ø LPRINT CHR$(27)"1";CHR$(7);
```

When the array is filled, the CHR$(7) will ring the bell to alert you.

The next line loads the beginning (B), ending (E), and step (S) values for the loop that will read and print the array. Variables are used because they can be changed later so that your program can read the array in a variety of directions.

```
    1ØØ B=1: E=N-6: S=1
```

Using seven pins of the print head on each pass, the program will take three passes to print a 21-row array.

If you change the array size, remember to use a multiple of seven.

Let's continue. Add:

```
    11Ø FOR P=B TO E STEP 7*S
    12Ø PRINT "LOADING ROWS";P;" TO ";P+6*S
    13Ø LPRINT CHR$(27)"*"CHR$(5)CHR$(N)CHR$(Ø);
    15Ø FOR C=1 TO N: GOSUB 18Ø: NEXT C
    16Ø LPRINT: NEXT P
```

Line 110 loads the array rows from beginning (B) to end (E) in sets of seven.

Line 120 prints an update of the computer's progress to the screen.

Line 130 enters the one-to-one Graphics Mode and reserves N columns for graphics (N is the width of the array).

Line 150 accesses the subroutine that actually calculates the pin patterns for each column. (That's coming up next.)

And line 160 closes the loop for each pass (P) of the print head.

One last step

The last step before printing the figure is to convert those ones and zeros to pin firing sequences. Add this subroutine:

```
    18Ø F=Ø: FOR R=P TO P+6*S STEP S
    19Ø IF A(R,C)=1 THEN F=F+2^ABS(P+6*S-R)
    2ØØ NEXT R
    22Ø LPRINT CHR$(F);: RETURN
    225 ' TRS users see Appendix I
```

Understanding the subroutine is easy if you take it one step at a time. This subroutine calculates the pin firing pattern (F) for each column of seven dots. The array is examined vertically, one cell at a time. When a one is encountered, the appropriate power of 2 is added to F in line 190. The exponent is the difference between the current row (R) and the last row in this pass of the print head (P + 6 * S). In line 220, F is sent to the printer as a graphics pin pattern.

Check your listing against Figure 15-7 to make sure you have it all. If everything checks out, type RUN.

```
10 DEFINT A: N=21: DIM A(N,N)
20 FOR R=1 TO N: FOR C=1 TO N
30 D=SQR( (R-11)^2 + (C-11)^2 )
40 IF INT(D+.5)=10 THEN A(R,C)=1
50 NEXT C
60 PRINT "ROW ";R: NEXT R
70 LPRINT CHR$(27)"1";CHR$(7);
100 B=1: E=N-6: S=1
110 FOR P=B TO E STEP 7*S
120 PRINT "LOADING ROWS ";P;" TO ";P+6*S
130 LPRINT CHR$(27)*"CHR$(5)CHR$(N)CHR$(0);
150 FOR C=1 TO N: GOSUB 180: NEXT C
160 LPRINT: NEXT P
170 LPRINT CHR$(27)"@": STOP
180 F=0: FOR R=P TO P+6*S STEP S
190 IF A(R,C)=1 THEN F=F+2^ABS(P+6*S-R)
200 NEXT R
220 LPRINT CHR$(F);: RETURN
```

Figure 15-7. Program for a circle.

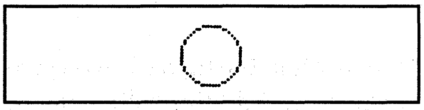


Figure 15-8. The circle.

Aha! So the array looks like Figure 15-8 when it's translated into dots. At least it does if you didn't run into any trouble codes.

Problem?

If all went well, skip to the "Higher Resolution" section, below. If not, let's take a time out for our sponsor . . . Solutions Galore.

If your printout doesn't look much like Figure 15-8, it's likely the problem involves the codes from 9 to 13. Remember that the dot patterns sent to the printer are determined by the figure stored in the array. If you want the figure to print correctly, you'll have to either POKE the codes directly or use a printer driver that allows the codes to pass through as sent. If this is not possible, there is a way to avoid these codes during printing without doing too much damage to the figure. The test below picks off any potential problem codes and changes them to less dangerous numbers.

```
210 IF F>8 AND F<14 THEN F=F-5
```

This line takes any number between 8 and 14 and subtracts 5 from it, putting it out of the trouble range. Adjust this test to fit your system.

Careful observers may note another problem with the figure. The standard 7-dot line spacing may be off just enough to add a slight gap every seven rows. An easy fix for this is to adjust the line spacing as needed with the <ESC>"3" command. This gives you the ability to make adjustments as fine as 1/3 of a dot. For example, a 6 2/3 dot line spacing can be set with:

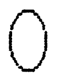
```
70 LPRINT CHR$(27)"3"CHR$(20);CHR$(7);
   ' Don't forget the bell
```

Now back to the regularly scheduled program . . .

Higher Resolution

If everything went according to plan, your printed figure resembles a circle of radius 10 dots, but the resolution isn't all that you might hope for. For one thing, the individual dots are clearly visible. To make a more continuous figure, you need a graphics mode of higher density. Change line 130 to:

```
130 LPRINT CHR$(27)"L"CHR$(N)CHR$(0);
```



The density is better, but this time the figure is distorted. Mathematically inclined folks can adjust for this distortion by creating an ellipse in the array (the horizontal compression creates a circle). If you pursue this course, keep in mind that the FX has several graphics densities available.

With figures this small, it is difficult to obtain a smooth curve. Figure 15-8, for instance, looks more like an octagon than a circle. The solution is to draw larger circles.

Unfortunately, larger arrays gobble up memory. For example, an array wide enough to stretch clear across the page (in Single Density) would contain over 200,000 cells (480 x 480). Considering that each cell takes up at least two bytes, there is not much hope of fitting the entire thing in memory all at once.

So how can we squeeze more print out of your arrays? For non-symmetric designs, you are pretty well stuck with the memory limitation of two bytes per cell unless you are willing to resort to more drastic measures. One such measure would be to let each bit of the numbers stored in the array cells represent one graphics dot. This would increase the storage ability but tremendously complicate the programming.

For symmetric designs such as the circle, you can take advantage of the symmetry to increase our output four-fold without increasing the size of the array one iota. How? By using the array to plot one fourth of the circle in memory, then copying the array to generate the other three parts (Figure 15-9).

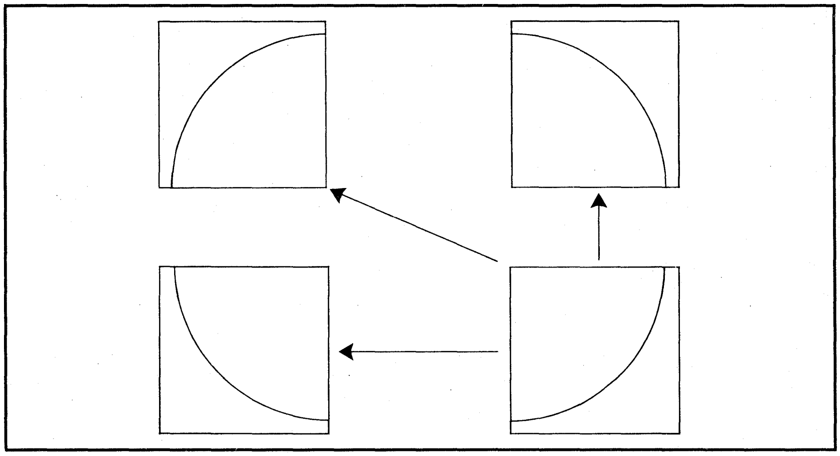
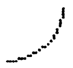


Figure 15-9. Divide and conquer.

Let's try this out with the current program. Here are the changes needed to plot the lower right corner:

```
30 D=SQR(R^2 + C^2)
40 IF INT(D+.5)=20 THEN A(R,C)=1
130 LPRINT CHR$(27)"*"CHR$(5)CHR$(N)CHR$(0);
```



The new figure is created by moving the center of the circle from (11,11) to (0,0) and increasing the radius from 10 to 20.

Reflections

Once the desired image is stored in the array, it can be rotated and reflected in several different directions. It's all done with mirrors; at least, it looks like mirror reflections when you are done. The mirror effect is created by reading the array in different directions.

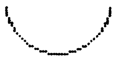
The array is currently read from left to right, seven rows at a time, but it is just as easy to read it in the reverse order. Add this line:

```
140 FOR C=N TO 1 STEP -1: GOSUB 180: NEXT C
```

And double the graphics width setting in line 130:

```
130 LPRINT CHR$(27)"*"CHR$(5)CHR$(2*N)CHR$(0);
     ' Change N to 2*N
```

Now RUN it.



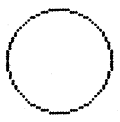
The left half of the figure is the mirror image of the right half. With a few more changes, the program can read the array upside down and double the output again. Add these lines:

```
80 B=N: E=7: S=-1
90 FOR Z=1 TO 2
```

and change these:

```
60 PRINT "T MINUS ";N-R: NEXT R
100 IF Z=2 THEN B=1: E=N-6: S=1
    ' Add the IF-THEN phrase
160 LPRINT: NEXT P: NEXT Z
    ' Add the Z phrase
```

Lines 80, 90, and 100 change the order in which the array is read. First (Z=1) it is read upside down. Then (Z=2) it is read right side up as before. Go ahead and RUN it to see how it works (have a cup of coffee while you're waiting for it to print).



There are two important points worthy of meditation. One, instead of tracing the circle like a plotter, the program assembles the pattern in the computer's memory, then prints it line by line. Two, the program takes advantage of symmetry to print a figure four times the size of the original array.

The exploding universe

With a few more program changes and lots of imagination, you can turn this mundane circle into an exploding universe. Yes, you read correctly. First, change the size so you can see the full impact of the figure:

```
10 DEFINT A: N=105: DIM A(N,N)
   ' Note: 105 is a multiple of 7
```

Yes, this one will take even longer to print out, but that's the price you pay for greatness.

Next, the distance formula is modified slightly. Type:

```
30 D=(R^2 + C^2)/N^2
```

This adjustment makes it easier to compare the distance value with the value of the RND function (line 40 below).

Once the computer knows the distance of each cell from the upper left corner, it can use the following test to determine which cells receive ones and which cells remain zero.

```
    4Ø IF D>RND(Ø) THEN A(R,C) = 1
```

Line 40 compares the modified distance (D) of each cell to a random number between zero and one. If D is greater than the random number, a one goes in that cell.

Program note: If your computer does not automatically set all variables to zero, add: 35 A(R,C)=0. Use your computer's version of RND(0), line 40 (some computers use RND(X) or RND).

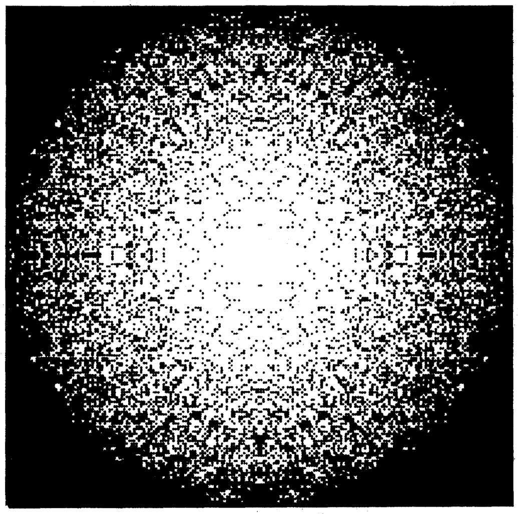
The use of a random number in line 40 adds a measure of uncertainty to the placement of the dots. Cells close to the upper left corner of the square array have a high probability of containing a zero, those far away have a high probability of containing a one. This use of randomness means that each program run will yield a different pattern.

The big bang

Are you ready to see what the program does? Change:

```
    6Ø PRINT "COUNTDOWN TO BIG BANG. T MINUS ";
       N-R: NEXT R
```

Okay, now RUN the program (this time you can fly to Columbia for coffee while you're waiting . . . but it's worth it!)



Can you see the white stars against the blackness of outer space?

This design also shows how the use of symmetry can increase your output. Unfortunately, this figure is about as large as you can print, given the current set-up and the memory limitaions of most small microcomputers.

But isn't there a way to print this kind of figure without such a heavy drain on computer memory? Sure enough, there is, but you must be willing to forgo the mirror-image effect of the above program and be prepared for an agonizingly slow program. We don't mean to be less than enthusiastic; we just want to be sure you understand the trade-offs.

With that as an introduction, Figure 15-10 lists a program which will allow you to print patterns that fill up an entire page, even in Quadruple Density!

```
NEW
1Ø N=476: M=INT((N+1)/2)
2Ø LPRINT CHR$(27)"1";CHR$(7);
3Ø  P=1 TO N-6 STEP 7
4Ø PRINT "PRINTING ROWS ";P;" TO ";P+6
5Ø N2=INT(N/256): N1=N-256*N2
6Ø LPRINT CHR$(27)"*"CHR$(5)CHR$(N1)CHR$(N2);
7Ø FOR C=1 TO N
8Ø   F=Ø: PRINT C;
9Ø   FOR R=P TO P+6
1ØØ    D=((R-M)^2+(C-M)^2)/M^2
11Ø    IF D>RND(Ø) THEN F=F+2^ABS(P+6-R)
12Ø   NEXT R: LPRINT CHR$(F);
13Ø  NEXT C: LPRINT: NEXT P
14Ø  LPRINT CHR$(27)"@"
```

Figure 15-10. Program for full-page pattern.

By changing the value of N to different multiples of seven, you can generate this pattern in different sizes. With N set to 476, be prepared to let your computer cook for about 10 to 15 hours (no, we aren't kidding)! See the results in Figure 15-11.

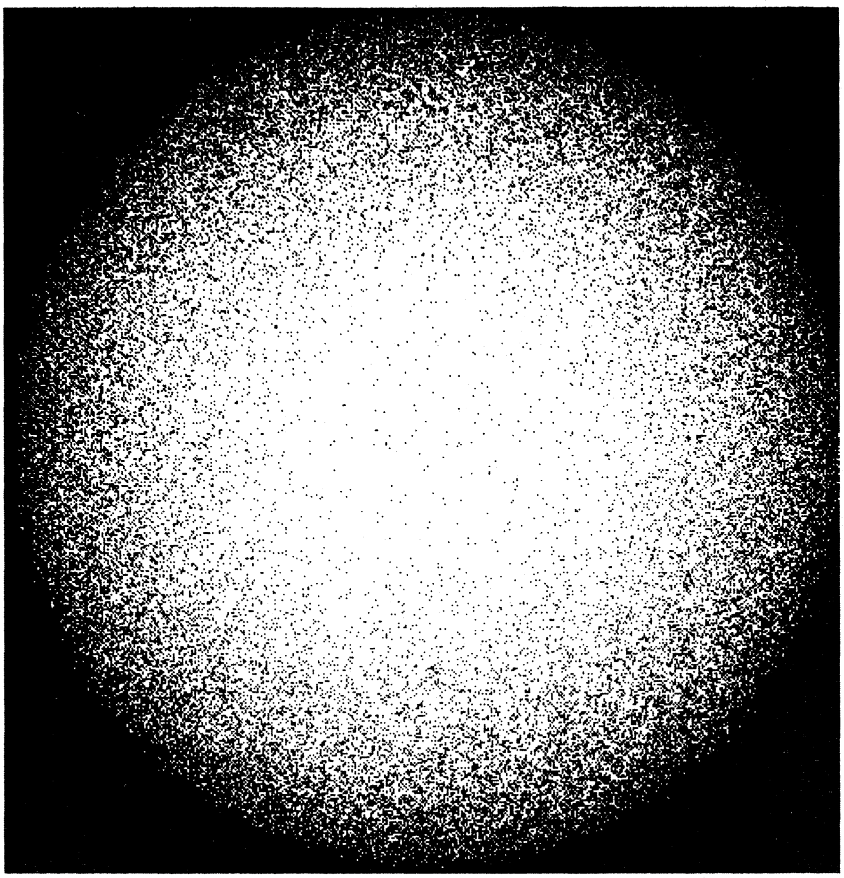


Figure 15-11. Full-page explosion.

Chapter 16
Advanced Graphics in Memory

In this chapter we continue our exploration of generating graphics patterns in memory. Specifically, a one-dimensional array will be used to create a two-dimensional printed figure.

The sequence of ones and zeros stored in this array will be used to generate all the pin patterns to be printed. As shown in Figure 16-1, a one in the array will indicate a graphics dot.

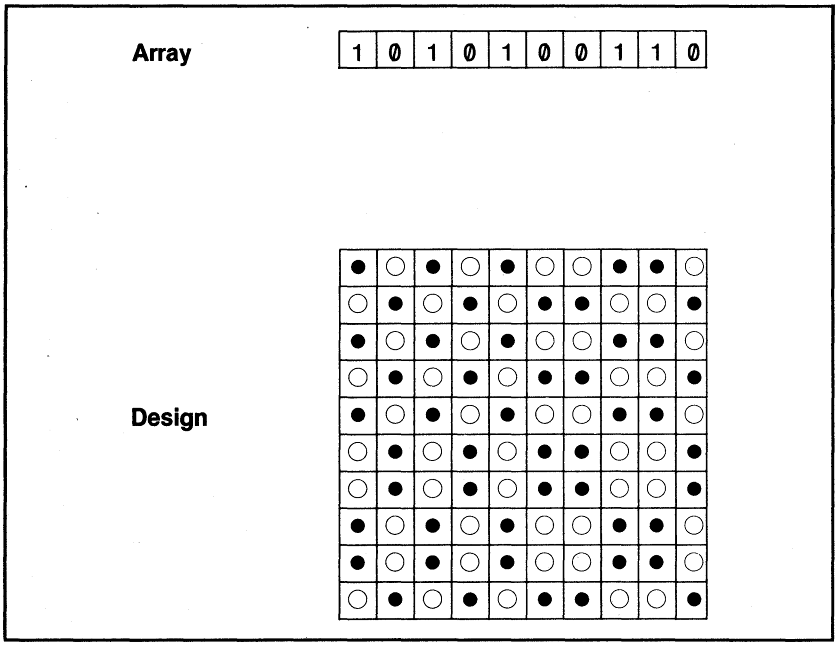


Figure 16-1. Array into design.

Ongoing generation of graphics in memory can easily be thought of as a process with five "phases": generating the array; calculating the pin patterns; setting the width; printing the figure; and working out variations. Our discussion follows these phases.

Generating the Array (Phase One)

Since this program is fairly long, we only ask you to run it periodically, and we'll let you know when. Begin by defining variables.

```
NEW
1Ø DIM A(48Ø): X=1: C=Ø
2Ø MAX=5: MIN=1: RE=4: N=Ø
```

The array A, which is DIMensioned in line 10, will store the pattern. The variables in line 20 are used in program loops to control the size and shape of the figure. You can change these values later to create your own variations on the pattern.

Table 16-1 lists the variables for easy reference.

Table 16-1. Variables for SYMMETRY Program.

Variable	Purpose
A	Array
C	Counter of array elements
DOT	Counter of dots; used to calculate P
H	Highest number used in calculating P
J	Loop counter
K	Loop counter
L	Loop counter
LAST	Last pass of the print head
MAX	Maximum number for the pattern
MIN	Minimum number for the pattern
N	Number of pins in the current pattern
N1	Length of the graphics line
N2	Length of the graphics line
P	Pin firing pattern
P0	Reverse pattern of P
PASS	Number of the current pass
R	Remainder
RE	Number of repeats of the pattern
X	0 or 1 to fill the array

Here are the loops:

```
30 FOR J=1 TO RE
40 N=N+1
50 GOSUB 300
60 IF N<MAX THEN 40
70 N=N-1
80 GOSUB 300
90 IF N>MIN THEN 70
100 NEXT J: PRINT
```

The J loop will REpeat four times (RE = 4). It has two sub-loops that depend on the value of N. Each time through the first loop (lines 40 to 60), N increases by one—to the value of MAX; each second loop (lines 70 to 90), N decreases by one—to the value of MIN. For each value of N, the program calls subroutine 300. This routine adds more ones and zeros into the array each time it is called. Type:

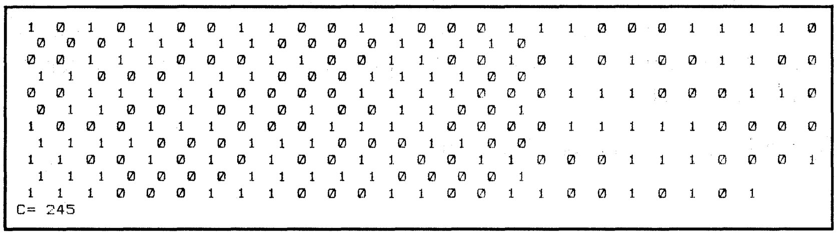
```
290 STOP
300 FOR K=0 TO MAX-N
310 FOR L=1 TO N
320 C=C+1: A(C)=X
330 NEXT L: X=1-X
340 NEXT K: PRINT N;: RETURN
```

Line 320 in the L loop stores the ones and zeros into the array. The end of line 330 makes X alternate between zero and one.

To see what the array contains at this point, type:

```
110 FOR K=1 TO C: LPRINT A(K);:
    NEXT K: LPRINT: LPRINT"C="C
```

and RUN your growing program for the results shown in Figure 16-2:



```
 1  Ø  1  Ø  1  Ø  Ø  1  1  Ø  Ø  1  1  Ø  Ø  Ø  1  1  1  Ø  Ø  Ø  1  1  1  1  Ø
  Ø  Ø  Ø  1  1  1  1  1  Ø  Ø  Ø  Ø  1  1  1  1  Ø
 Ø  Ø  1  1  1  Ø  Ø  Ø  1  1  Ø  Ø  1  1  Ø  Ø  1  Ø  1  Ø  1  Ø  Ø  1  1  Ø  Ø
  1  1  Ø  Ø  Ø  1  1  1  Ø  Ø  Ø  1  1  1  1  Ø  Ø
 Ø  Ø  1  1  1  1  1  Ø  Ø  Ø  Ø  1  1  1  1  Ø  Ø  Ø  1  1  1  Ø  Ø  Ø  1  1  Ø
  Ø  1  1  Ø  Ø  1  Ø  1  Ø  1  Ø  Ø  1  1  Ø  Ø  1
 1  Ø  Ø  Ø  1  1  1  Ø  Ø  Ø  1  1  1  1  Ø  Ø  Ø  Ø  1  1  1  1  1  Ø  Ø  Ø  Ø
  1  1  1  1  Ø  Ø  Ø  1  1  1  Ø  Ø  Ø  1  1  Ø  Ø
 1  1  Ø  Ø  1  Ø  1  Ø  1  Ø  Ø  1  1  Ø  Ø  1  1  Ø  Ø  Ø  1  1  1  Ø  Ø  Ø  1
  1  1  1  Ø  Ø  Ø  Ø  1  1  1  1  1  Ø  Ø  Ø  Ø  1
 1  1  1  Ø  Ø  Ø  1  1  1  Ø  Ø  Ø  1  1  Ø  Ø  1  1  Ø  Ø  1  Ø  1  Ø  1
C= 245
```

Figure 16-2. Printing the contents of the array.

All that is a one-line array; it just wraps around when printed. It shows how the FOR-NEXT loops use variables to create patterns. The overall pattern gets made up of five sets of pattern 1, four sets of 2, three sets of 3, . . . as shown in Figure 16-3.

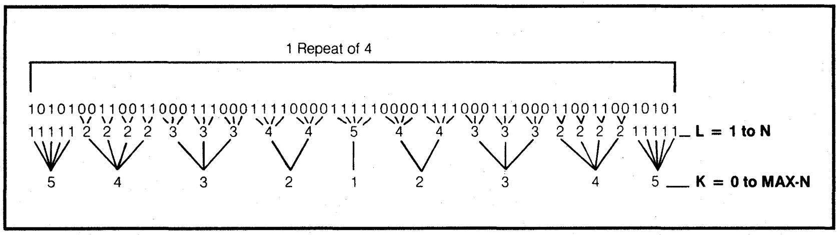


Figure 16-3. Pattern sets.

Before proceeding, modify line 110 so that it prints to the screen instead of to the printer:

```
110 FOR K=1 TO C: PRINT A(K);:
    NEXT K: PRINT: PRINT "C="C
```

Describing the 2-Dimensional Matrix (Pause)

The 1-line array just created will be used to generate a 2-dimensional pattern on the paper. This represents a significant saving of memory compared to the method of generating an array that we used in the last chapter.

To do this, we simulate (without actually creating) a 2-dimensional array in a 3-step mental process:

1) Convert ones and zeros to dots and blank spaces.

2) Duplicate the array vertically.

3) Define a "times table"—where the intersection of a row and column is the "product" of the two values. The product of two dots of the same color is a black dot; the product of two dots of different colors is a white dot.

Figure 16-4 depicts this "times table."

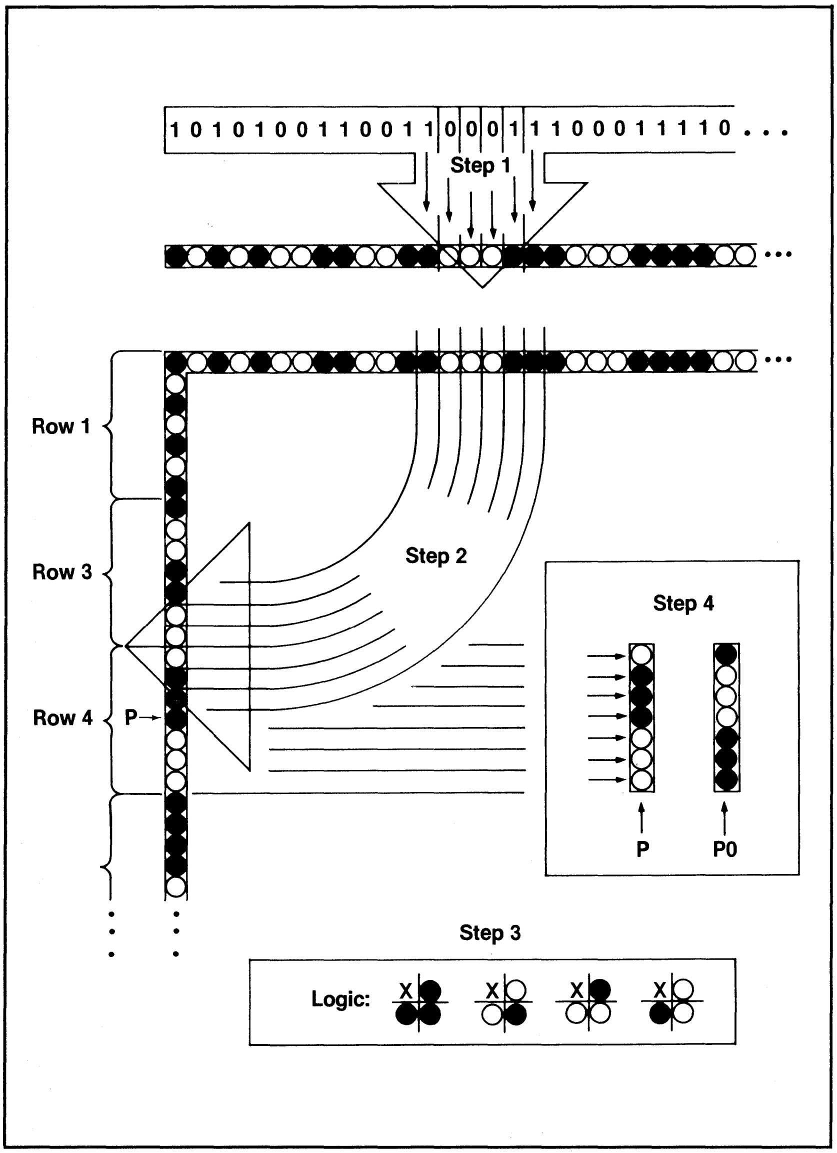


Figure 16-4. Translation of the times table.

The next phase consists of calculating the pin patterns; here's a job for the computer.

Calculating Pin Patterns (Phase Two)

Two patterns are used in each pass of the print head. P is the same pattern as the one formed by the seven vertical dots at the start of each print row, and P0 is its black/white reverse image. The times table shown in Figure 16-4 produces the pattern P in each column that is headed by a black dot. Pattern P0 is printed in the other columns. The pin patterns are created this way:

```
140 FOR PASS=0 TO LAST: P=0:
    PRINT "PASS " PASS " OF " LAST
160 FOR DOT=0 TO H:
170 IF A(7*PASS+DOT+1)=1 THEN P=P+2^(6-DOT)
180 NEXT DOT
185 ' See Appendix I re problem codes and P
190 P0=127-P: IF PASS=LAST THEN P0=P0+1-2^(7-R)
```

For each pass of the print head (0 to LAST), the program calculates the pin patterns, seven dots at a time. Line 170 calculates P, and line 190 calculates its complement, P0.

Adjust the line spacing to match the seven-dot passes:

```
120 LPRINT CHR$(27)"1";
```

If this produces slight gaps between rows, adjust the spacing accordingly (e.g., CHR$(27)"3"CHR$(20)).

The length of the array depends entirely on the variables in line 20. If the array length is not a multiple of seven, the pin patterns (P and P0) for the last pass of the print head must be adjusted accordingly. Add:

```
130 LAST=INT(C/7): R=C-7*LAST
150 H=6: IF PASS=LAST THEN H=R-1
```

Setting the Graphics Width (Phase Three)

The required graphics width is C, the size of the array. If, however, C is greater than 255, the value n_2 in the graphics entry string must change from zero to one. With this in mind, add:

```
200 N1=C: N2=0
205 ' See Appendix I re problem codes and P0
210 IF C>255 THEN N1=C-256: N2=1
220 LPRINT CHR$(27)"*"CHR$(5)CHR$(N1)CHR$(N2);
```

⟨ESC⟩"*"CHR$(5) is the one-to-one graphics density setting from Chapter 13. Using it ensures that the figure prints a square image on the paper.

Printing the Figure (Phase Four)

The groundwork is just about complete. Type:

```
230 FOR K=1 TO C
240 IF A(K)=1 THEN LPRINT CHR$(P);
250 IF A(K)<>1 THEN LPRINT CHR$(P0);
260 NEXT K
270 LPRINT
280 NEXT PASS
```

This part of the program checks the array for ones and zeros. If it finds a one, the pattern P is printed. If it finds anything else, P0 is printed (see Figure 16-4, above). Line 280 completes the PASS loop.

Check your listing against the listing of Figure 16-5 to make sure it's all there:

```
10 DIM A(48Ø): X=1: C=Ø                          INITIALIZE VARIABLES
2Ø MAX=5: MIN=1: RE=4: N=Ø
3Ø FOR J=1 TO RE
4Ø   N=N+1
5Ø   GOSUB 3ØØ
6Ø   IF N<MAX THEN 4Ø                             GENERATE ARRAY
7Ø   N=N-1
8Ø   GOSUB 3ØØ
9Ø   IF N>MIN THEN 7Ø
1ØØ NEXT J: PRINT
11Ø FOR K=1 TO C: PRINT A(K);: NEXT K: PRINT: PRINT"C="C
12Ø LPRINT CHR$(27)"1";
13Ø LAST=INT(C/7): R=C-7*LAST
14Ø FOR PASS=Ø TO LAST: P=Ø: PRINT "PASS " PASS" OF "LAST
15Ø H=6: IF PASS=LAST THEN H=R-1
16Ø FOR DOT=Ø TO H                                          CALCULATES
17Ø   IF A(7*PASS+DOT+1)=1 THEN P=P+2[(6-DOT)               PIN PATTERN
18Ø NEXT DOT: IF P=12 THEN P=4                                P+PO
19Ø PØ=127-P: IF PASS=LAST THEN PØ=PØ+1-2[(7-R)
2ØØ N1=C: N2=Ø: IF PØ=12 THEN PØ=4                          ENTERS
21Ø IF C>255 THEN N1=C-256: N2=1                            GRAPHICS
22Ø LPRINT CHR$(27)"*"CHR$(5)CHR$(N1)CHR$(N2);               MODES
23Ø FOR K-1 TO C
24Ø   IF A(K)=1 THEN LPRINT CHR$(P);
25Ø   IF A(K)<>1 THEN LPRINT CHR$(PØ);                 PRINTS PIN
26Ø NEXT K                                             PATTERNS
27Ø LPRINT
28Ø NEXT PASS
29Ø STOP
3ØØ FOR K=0 TO MAX-N
31Ø   FOR L=1 TO N
32Ø     C=C+1: A(C)=X                               SUBROUTINE TO
33Ø   NEXT L: X=1-X                                   FILL ARRAY.
34Ø NEXT K: PRINT N;: RETURN
```

Figure 16-5. Program for SYMMETRY.

then RUN it to see if it looks like Figure 16-6.

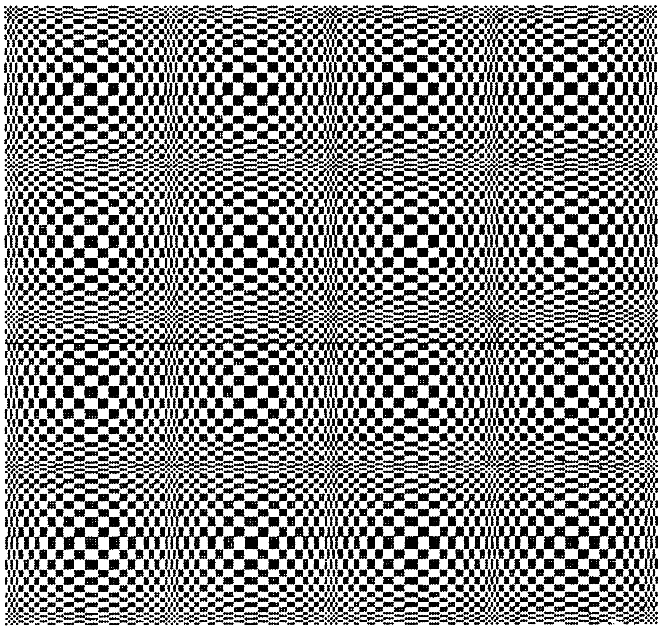


Figure 16-6. Symmetric pattern 1.

That's enough to knock your eyes right out of their sockets! And all that from a one-line array!

Variations on a Theme (Phase Five)

The symmetric pattern in this program is completely controlled by the computer, dot for dot. Small changes in the program can affect the pattern in a big way. For example, try this simple change in line 300:

```
300 FOR K=0 TO 0
```

And RUN it again:

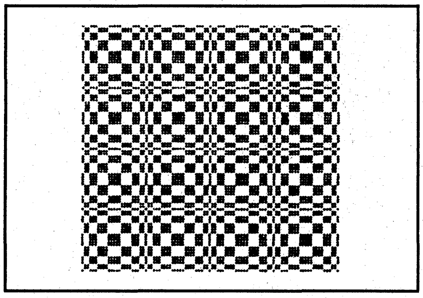


Figure 16-7.
Symmetric pattern 2.

Notice in your printout (or in Figure 16-7) that each string of ones and zeros in the array prints only once. The K loop in line 300 controls the repetitions of these strings.

Here's another interesting variation:

```
20 MAX=64: MIN=1: RE=1: N=1/2
40 N=N*2
70 N=N/2
300 FOR K=0 TO 0
```

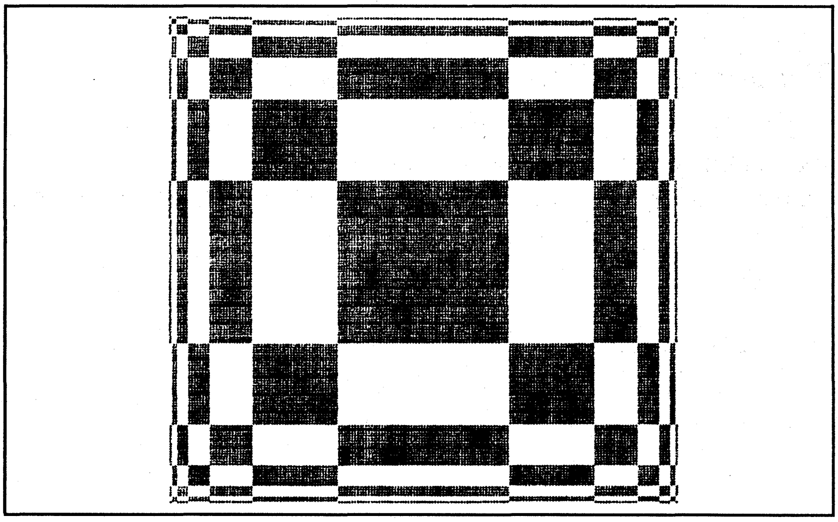


Figure 16-8. Symmetric pattern 3.

Quite a difference! The program doubles and halves N instead of adding and subtracting one. This geometric progression creates a very different pattern.

Also notice that the pattern only repeats once because the variable RE is set to one.

Now's the time to experiment with some of your own changes to the variables and loops.

The next chapter introduces one of the most exciting features of the FX-80: user-defined characters. That's right, you can redefine the entire character set if you want to!

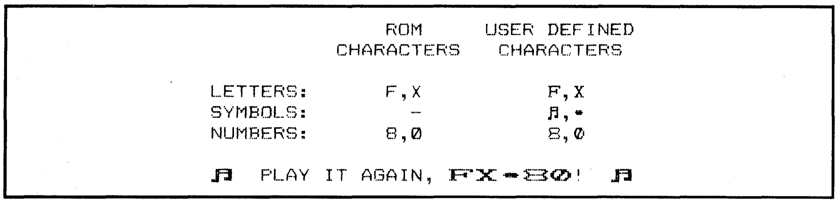
Chapter 17
User-Defined Characters

To quote tennis guru Vic Braden:

"If you read between these covers, you are going to be famous by Friday!"

Famous or not, if you've studied all the program examples of this manual, you should be quite adept at printing both graphics and text with the FX-80. In this chapter we're going to share the secrets of the ultimate in printer control—defining your own characters.

With the FX-80, you can create any number of new characters, including special symbols, graphics patterns to serve as building blocks for larger designs, or even whole type fonts. Your characters can be used for any purpose as long as they fit into the same dot matrix as the ROM characters do—9 dots tall by 11 dots wide (6 columns plus 5 intermediate columns). Figure 17-1 shows a comparison of a few sample characters and their ROM equivalents.



```
                    ROM         USER DEFINED
                 CHARACTERS      CHARACTERS

    LETTERS:        F,X             F,X
    SYMBOLS:         -              ♫,•
    NUMBERS:        8,0             8,0

    ♫  PLAY IT AGAIN, FX-80!  ♫
```

Figure 17-1. Sample characters and ROM equivalents.

Once you define your own characters, they can be used over and over, just like the FX-80's ROM characters. In fact, user-defined characters are printed just like any other ASCII characters.

But before you can take advantage of this fantastic feature, you must first make a little preparation.

Preparation

Internal switch 1-4 controls the use of the FX-80's 2K RAM buffer. This RAM memory can be used as a large text buffer to smooth printer/computer communications, or it can be programmed with a set of user-defined characters. Unfortunately, it can't serve both ways at the same time. In this and succeeding chapters, we'll use this RAM area for user-defined characters. So set switch 1-4 *off* before proceeding.

There's one more bit of preparation before you begin. LOAD the program that you saved as LOGO in Chapter 14—we will add to it in this chapter.

Defining Characters

Characters are defined with the ⟨ESC⟩"&" command sequence. The format is:

```
LPRINT CHR$(27)"&"CHR$(r)CHR$(c₁)CHR$(c₂);
```

The r tells the printer in which RAM area the characters are to be stored. With a stock printer, there is only one area available: RAM area 0. (And it is only available if switch 1-4 is *off*!)

The notations c_1 and c_2 specify the range of characters to be defined. The entire range of ASCII numbers from 0 to 255 (for which the ROM characters are shown in Appendix A) can be used, except for those areas where control codes reside (0 to 31, 127, 128 to 159, 255). Some of the control codes can also be used, but only after special ESCape codes are issued. We'll get to that a bit later.

Here's how c_1 and c_2 work. Suppose you want to redefine the letters from A to E. The associated ASCII numbers are 65 and 69, so you simply let c_1 be 65 (A) and c_2 be 69 (E). Any of the keyboard characters can be redefined in a similar manner. For example, $c_1 = 97$; $c_2 = 101$ would select lower-case letters a through e, and $c_1 = 33$; $c_2 = 43$ would select the symbols ! through +.

To simplify things a bit, the ASCII symbols can be used in place of CHR\$(c_1) and CHR\$(c_2). For example, either:

```
    LPRINT CHR$(27)"&"CHR$(Ø)CHR$(65)CHR$(69);
```

or:

```
    LPRINT CHR$(27)"&"CHR$(Ø)"AE";
```

selects characters A through E.

On some occasion you may wish to define only one character. That's okay, too. Just use the same number for c_1 and c_2. In fact, let's do that now. Add:

```
    13Ø LPRINT CHR$(27)"&"CHR$(Ø)"EE";
         ' Model I users see Appendix I
```

The trailing semicolon is very important. The ESC"&" sequence expects more data to follow (just as Graphics Mode does). The trailing semicolon prevents an unwanted carriage-return code from disrupting the data.

For each character to be defined (determined by c_1 and c_2), the printer expects 12 data numbers to follow. The first of these numbers is called the "attribute byte." It determines some special attributes or characteristics of the character being defined. The next 11 numbers contain the dot patterns of the symbol being defined—nothing fancy, just 11 standard graphics pin patterns.

Designing characters

The first step in defining a new character is to lay out the dot pattern. Check Appendix B to see how the ROM characters are designed. User-defined characters share the same limitations as those found in the ROM. Characters can be a maximum of 8 dots tall (even though the matrix is 9 dots) and 11 dots wide. Most characters, as shown in the Appendix, use only the top 7 pins of the print head; lower-case characters with descenders use the bottom 7. Also note that all characters except for the underline (decimal 95) leave the last two columns unused; this provides space between the letters when they are printed.

Figure 17-2 shows the design of a letter E in an 8 by 11 matrix.

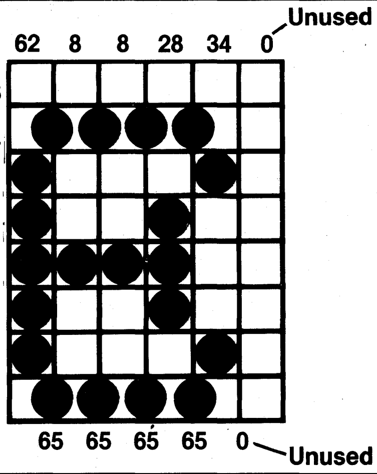


Figure 17-2. User-defined E.

To be consistent with the ROM characters, we use only 7 rows. The character would normally go in the top 8 rows, but we shift all the dots down one row so that 7-bit machines can stay with the program.

Also note that two adjacent dots cannot be printed in the same row. Even in Half-Speed Mode, the printer simply refuses to print two overlapping dots. Figure 17-3 illustrates an E which is incorrectly designed because it uses overlapping dots:

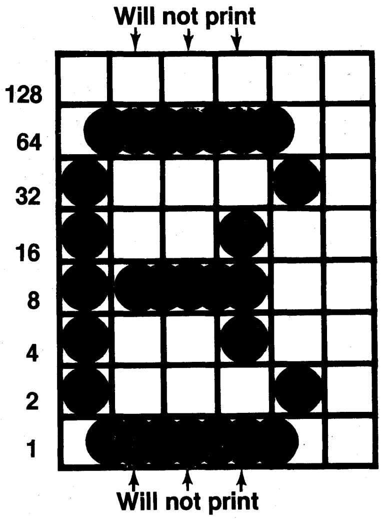


Figure 17-3. Incorrectly designed E.

Translating dots to DATA

The data numbers for each column of Figure 17-2 are calculated just like those in Graphics Mode. And the appropriate numbers can just as easily be stored in DATA statements. Type in the READ routine and data for the character in Figure 17-2:

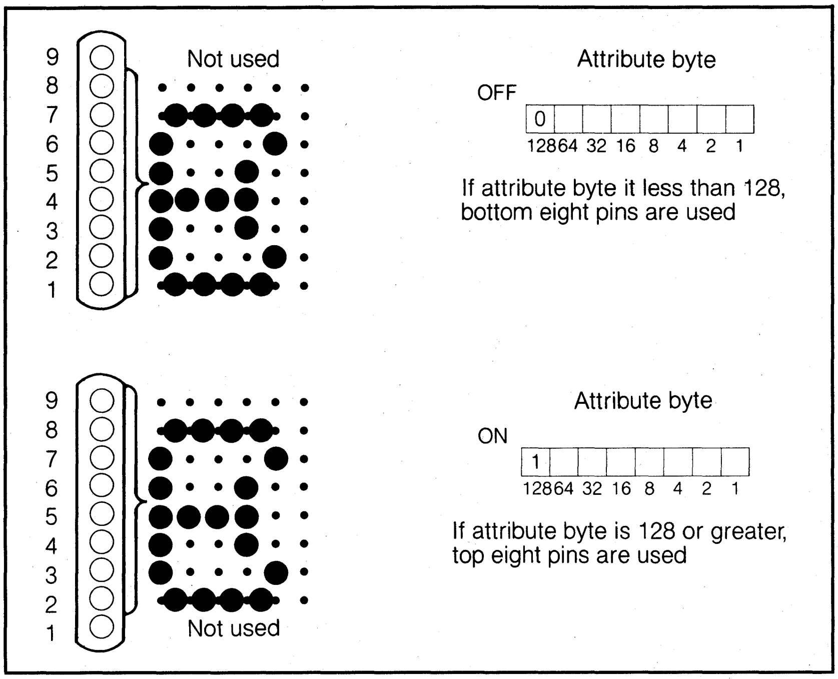
```
150 FOR X=1 TO 11: READ C:
    LPRINT CHR$(C);: NEXT X
1170 DATA 62,65,8,65,8,65,28,65,34,0,0: ' My E
```

Notice that the DATA statement contains 11 numbers even though the design uses only 9 of the 11 columns. Unused columns must be coded as 0.

The attribute byte

The attribute byte is the first of the 12 data numbers required to define any character. At print time it controls two aspects of the way the character is printed. First, it determines which 8 pins of the print head are used to print the character. For most characters, the top 8 pins are used, but for lower-case characters with descenders (like g and p), the bottom 8 pins can be used.

So how does the attribute byte determine which 8 pins are used? Good question. At print time, the printer checks the attribute byte before each character is printed. If the high-order bit is on, the top 8 pins of the print head are used; if the high-order bit is off, the bottom 8 are used. To put it another way, if the attribute byte for a given character is 128 or greater, the top 8 pins are used; if it is 127 or less, the bottom 8 are used. Figure 17-4 demonstrates these choices.



9
8
7
6
5
4
3
2
1
Not used
Attribute byte
OFF
0
128 64 32 16 8 4 2 1
If attribute byte it less than 128, bottom eight pins are used
9
8
7
6
5
4
3
2
1
Not used
Attribute byte
ON
1
128 64 32 16 8 4 2 1
If attribute byte is 128 or greater, top eight pins are used

Figure 17-4. Use of pins chosen by attribute byte.

Proportional print information

The attribute byte also contains information used to print a character in Proportional Mode. It tells the printer in which columns to start and stop printing for each character. If we label the 11 columns determined by the data numbers as columns 0 to 10, then in Proportional Mode the minimum and maximum starting and stopping columns will be 0 and 11. Why 11 instead of 10? Column 11 is the maximum value because Proportional characters are always Emphasized; this makes each character wider by one "intermediate dot column" (see Chapter 4 on the Emphasized Mode). So when defining your own characters for proportional printing, always reserve one extra column.

Let's create a sample attribute byte together. Suppose you want a character to start in column 1 and end in column 10. How do you put this information into the attribute byte? Easy, just follow along. The starting column number (1) is converted to a 3-bit binary number (001) and stored in bits 4, 5, and 6 of the attribute byte. The ending column number (10) is converted to a 4-bit binary number (1010) and stored in bits 0 to 3. The conversions are shown in Figure 17-5.

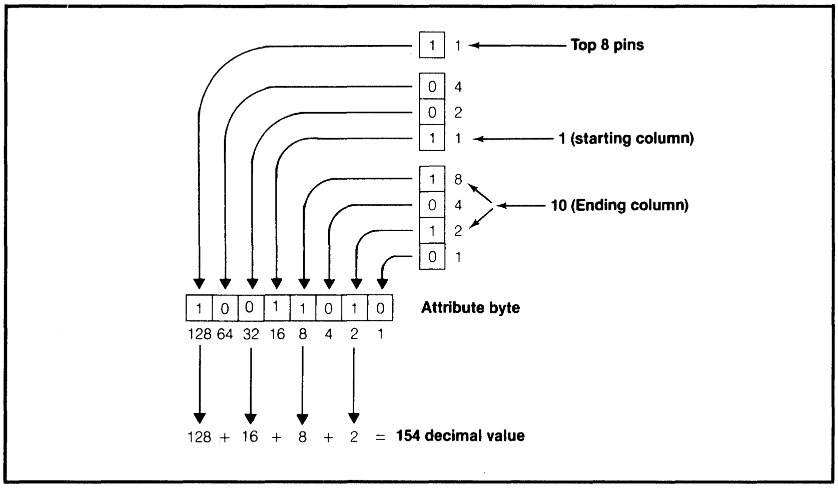


Figure 17-5. Attribute byte conversions.

The full 8-bit attribute byte, then, is composed of three parts:

1) Bit 7 determines which pins are used to print the character.

2) Bits 4, 5, and 6 determine the starting column number.

3) Bits 0, 1, 2, and 3 determine the ending column number.

So the attribute byte constructed in Figure 17-5 (CHR$(154)) would use the top eight pins, start printing in column 1, and end in column 10.

Note that the proportional print information is used only when the character is printed in Proportional Mode. Otherwise the full range of columns 0 to 11 is used. Also note that if 7-bit users set the high-order bit (with ⟨ESC⟩">") before the ⟨ESC⟩"&" sequence, it stays on for the attribute and character data bytes.

One final note. Even if you choose not to print the columns from 0 through 11, you must send the printer 11 data numbers plus the attribute byte. The printer expects 12 data numbers for each character, no matter what!

So much for the example. In the current program, we'll set the attribute byte to 139. This value starts at column 0, ends at column 11, and uses the top 8 pins of the print head. Seven-bit computers can use 139 or 11. Either way, the printer will use the bottom 8 pins for 7-bit systems. Add:

```
14Ø LPRINT CHR$(139);
```

Printing User-Defined Characters

If you RUN the program at this point, it will define the character E in RAM area 0 (assuming switch 1-4 is off), but only the ROM version of the E will print. Try it. Add:

```
180 LPRINT "EEEEE"
200 LPRINT CHR$(27)"@": STOP
```

```
                    EEEEE
```

Sure enough, the standard (Roman Pica) E is printed. To print your newly defined character, you must tell the printer to ignore the ROM and print only RAM characters. The format for this instruction is:

```
LPRINT CHR$(27)"%"CHR$(n1)CHR$(n2);
```

The ⟨ESC⟩"%" sequence determines the currently active character set. The n_1 selects either ROM (0) or RAM (1), while n_2 selects the area (0 is the only area available). The command to activate the RAM area is:

```
120 LPRINT CHR$(27)"%"CHR$(1)CHR$(0);
```

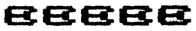
but before you print the user-defined E, make it more visible by adding:

```
170 LPRINT CHR$(27)"!8";
190 LPRINT CHR$(27)"!@"
```

Line 170 uses the Master Select code to print Double-Strike, Expanded, Emphasized Pica characters. Line 190 uses the Master Select to return to Pica Mode.

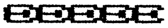
Here are the new lines you've typed so far:

```
120 LPRINT CHR$(27)"%"CHR$(1)CHR$(0);
130 LPRINT CHR$(27)"&"CHR$(0)"EE";
140 LPRINT CHR$(139);
150 FOR X=1 TO 11: READ C: LPRINT CHR$(C);: NEXT X
170 LPRINT CHR$(27)"!8";
180 LPRINT "EEEEE";
190 LPRINT CHR$(27)"!@"
200 LPRINT CHR$(27)"@": STOP
1170 DATA 62,65,8,65,8,65,28,65,34,0,0: ' My E
```



Let's see what it looks like in Proportional Mode (without the last two columns printed). Add:

```
140 LPRINT CHR$(137);
175 LPRINT CHR$(27)"p1";
```



See how the Es are packed closely together? Fine. Before proceeding, change back to monospacing with:

```
140 LPRINT CHR$(139);
```

and

```
DELETE 175
```

While you are in the neighborhood, take a look at some of the other characters in RAM with:

```
180 LPRINT "EPSON"
```

Oops! What happened? Where is the rest of EPSON? All right, we confess! The only characters in the user-defined RAM are those you put there yourself. Characters that haven't been defined print as blank spaces. So the RAM area is like a big blank chalk board waiting for you to fill it up. At this point, because you have only defined an E, that's all you get from RAM.

Download Commands

Wouldn't it be nice if you could magically transport some of the ROM characters over to the RAM area so you wouldn't have to switch back and forth or define an entire character set each time you use the RAM area? Sure it would. There are plenty of applications where you only need to define a few special characters to be used with the standard alphabet and numbers. That's why the FX-80 provides the option of copying (sometimes called downloading) the entire ROM set into the user-defined RAM area.

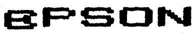
The download command has the format:

```
LPRINT CHR$(27)":"CHR$(n1)CHR$(n2)CHR$(n3);
```

This command is designed with possible future expansion in mind. For now, set all three numbers to 0:

```
11Ø LPRINT CHR$(27)":"CHR$(Ø)CHR$(Ø)CHR$(Ø);
```

Now RUN the program:



That's more like it! You get the custom designed E and the normal characters copied over from the ROM. Notice that the E is lower on the page than the other characters even though the high-order bit of the attribute byte is on. What gives? Well, in order to save 7-bit users from total frustration, we designed the character to use the bottom seven rows (as in Figure 17-4a). Normally, the top seven rows are used for all but lower-case characters with descenders.

Caution: Be very careful about using the Master Reset Code after defining your own characters in RAM. This code wipes out the entire contents of RAM . . . goodbye user-defined characters!

Defining Strings of Characters

Once the ESCape sequences are in place, adding more characters is a breeze. To see how much of a breeze, simply add more data:

```
1150 DATA 7,8,16,36,64,36,16,8,7,0,0: ' My A
1160 DATA 127,0,72,0,72,0,76,2,121,0,0: ' My R
```

and make these changes:

```
130 LPRINT CHR$(27)"&"CHR$(0)"rt";
140 FOR Y=1 TO 3: LPRINT CHR$(139);
160 NEXT Y
180 LPRINT "rst"
```

Line 130 controls the reading of the data. It expects data for three characters: r, s, and t. This example uses lower-case characters. If necessary, you can use CHR$(114) and CHR$(116) in place of the "rt".

The attribute byte for each character is sent in line 140 and the other 11 bytes are read from DATA lines. This method is nice for quick and easy character definition.

If you intend to print your characters in Proportional Mode, you'll want to add a different attribute byte to the start of each DATA line and adjust the READ routine for 12 numbers. Don't forget the attribute byte or you'll end up with some "interesting" results.

Defining Control Codes

For some of you dedicated users, the range from 32 to 126 and 160 to 254 may not be large enough to accommodate all the characters you want. How can that be? Well, perhaps you have a passion for Egyptian hieroglyphics, or maybe you need a complete set of mathematics symbols. And what about the entire Japanese character set (it has some 4000 symbols)?

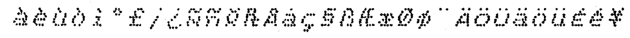
If you get carried away with user-defined characters, you may end up searching for more storage. Anticipating this need, Epson provides commands that will allow you to define and print certain control codes in the same way that you treat other characters. (Remember that the low-order control codes are the ASCII codes 0 through 31 plus 127, and the high-order control codes are 128 through 159 plus 255.)

Control codes are a strange kettle of fish. These codes do not normally print symbols on paper, rather they cause the printer to change modes. To make them print as normal symbols requires an extra command. For example, the command to "normalize" the high-order control codes is:

```
LPRINT CHR$(27)"6"
```

Let's use this command to see how the ROM control codes print when they are stripped of their high and mighty ways. Add:

```
2 LPRINT CHR$(27)"6"
4 FOR X=128 TO 159: LPRINT CHR$(X);: NEXT X
5 ' 7-bit'ers see Appendix I
6 LPRINT CHR$(27)"7"
8 STOP
```



Aha! That's where the international characters (italic version) are hiding in the ROM. So the ⟨ESC⟩"6" command without the ⟨ESC⟩"R" gives you access to the international characters. The ⟨ESC⟩"7" turns these characters back into control codes. Let's see how the ⟨ESC⟩"6" and ⟨ESC⟩"7" commands work with user-defined characters in RAM.

Delete lines 2 through 8, and type:

```
13Ø LPRINT CHR$(27)"&"CHR$(Ø)CHR$(128)CHR$(131);
14Ø FOR Y=1 TO 4: LPRINT CHR$(139);
164 ' 7-bit'ers can't run this-see Appendix I
165 LPRINT CHR$(27)"6"
18Ø LPRINT CHR$(128)CHR$(129)CHR$(13Ø)CHR$(131)
114Ø DATA Ø,126,1,2,4,8,4,2,1,126,Ø: ' My W
```

And let 'er go:

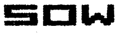
The lower-order control codes can also be defined, but not all of them can be printed with ease. 〈ESC〉"I1" makes them printable, and 〈ESC〉"I0" returns them to normal.

Just as the higher control codes hide the italic international characters, the lower control codes hide (you guessed it!) the roman international characters. But let's see how the 〈ESC〉"I1" and 〈ESC〉"I0" codes work with user-defined characters. Change:

```
130 LPRINT CHR$(27)"&" CHR$(0)CHR$(1)CHR$(3);
140 FOR Y=1 TO 3: LPRINT CHR$(139);
165 LPRINT CHR$(27)"I1"
180 LPRINT CHR$(1)CHR$(2)CHR$(3)
```

And add:

```
1100 DATA 0,121,0,73,0,73,0,73,0,79,0: ' My S
1110 DATA 0,127,0,65,0,65,0,65,0,127,0: ' My Oh
```



The program now contains six DATA lines, but it uses only the first three. The three characters are stored in ASCII codes 1, 2, and 3 in RAM; they are printed from line 180.

Printing low-order control codes

Not all of the low-order (0 to 31) control codes can be changed to print as normal characters—nor would you want them to. Imagine, if you changed code 27 to print as a normal character . . . no more ESCape codes! You would have a hard time getting anything done.

Codes that currently activate special modes or actions by the printer cannot be printed as normal characters. These include 7 to 15, 17 to 20, 24, and 27. However, it is possible to print the characters stored in these locations with the 〈ESC〉"R" command.

Here's how it works. Suppose you choose to define the ASCII code 8 (normally a backspace). The 〈ESC〉"&" command will work fine, but printing CHR$(8) still produces a backspace, even after an 〈ESC〉"I1". 〈ESC〉"R" to the rescue. 〈ESC〉"R" lets you print the character stored in location 8 with another ASCII code. The 〈ESC〉"R" sort of magically transports the character to an easily printable location. To find out what is stored where, use Table 17-1.

Table 17-1. International Character Internal Codes.

Dec. Code	USA	France	Germ.	Eng.	Denm.	Sweden	Italy	Spain	Japan
35				6				12	
36						11			
64		0	16			29			
91		5	23		18	23	5	7	
92		15	24		20	24		9	31
93		16	25		13	13	30	8	
94						25			
96						30	2		
123		30	26		19	26	0	22	
124		2	27		21	27	3	10	
125		1	28		14	14	1		
126		22	17			28	4		

Find 8 in the table; it is in the CHR$(93) row under the Spain heading. To print the character stored in 8, use ⟨ESC⟩"R" to activate the Spanish character set, and print CHR$(93). Ole! This same technique can be used to access any of the normally unprintable control codes.

Here's a word or two to the wise: using an international character set while defining characters can be a two-edged sword. If you are currently in one of the international modes (other than USA), then defining any of the codes 35, 36, 64, 91 to 94, 96, or 123 to 126 gets a bit tricky. These codes are merely pointers to the control-code areas in which the international characters are really stored.

To define any of these characters while in an international mode, the "true" location of the character must be used. For instance, if the printer is in the Spanish set and you wish to define character 93, you must use CHR$(8) in the ⟨ESC⟩"&" sequence to define the character, but CHR$(93) to print it.

To make sure you have this concept down cold, try answering this one: How would you redefine the ESCape code? (Not to worry, we won't disable your precious ESCape code.) First find 27 on the chart. It occurs in two places, one of which is in the column labelled Sweden and the row labelled 124. So a user-defined character stored at 27 could be printed in the Swedish set as character 124. Whew!

Mode Strings

For some applications, you may wish to use all 256 RAM locations for your own special symbols. In that case, there is no need to download the ROM into RAM. But you will need a quick and easy way to switch back and forth between the two character sets. One easy way to do this is to define two character strings:

```
8Ø RAM$=CHR$(27)+"%"+CHR$(1)+CHR$(Ø)
9Ø ROM$=CHR$(27)+"%"+CHR$(Ø)+CHR$(Ø)
```

Add these lines to the current program. To demonstrate their effect, try:

```
13Ø LPRINT CHR$(27)"&"CHR$(Ø)"18";
14Ø FOR Y=1 TO 8: LPRINT CHR$(139);
18Ø LPRINT ROM$;"12345678";RAM$;"12345678"
112Ø DATA Ø,63,64,8,64,8,64,28,64,32,Ø: ' My F
113Ø DATA Ø,32,64,Ø,64,63,64,Ø,64,32,Ø: ' My T
```

```
12345678SOFTWARE
```

If you find yourself defining characters in small groups, the same technique can be used to store part of the ⟨ESC⟩"&" command:

```
Z$=CHR$(27)+"&"+CHR$(Ø)
```

Z$ can be used to define each new string of characters with a simple command like:

```
LPRINT Z$;"AZ";
```

or

```
LPRINT Z$;CHR$(128)CHR$(159);
```

Strata

Our current program uses eight user-defined characters as well as the LOGO graphics design that you loaded at the beginning of the chapter. We are grooming this program to be used again in later chapters. Make these few final changes for this chapter:

```
DELETE 8Ø
DELETE 9Ø
DELETE 165
18Ø LPRINT " 147646 12345678"
```

Now RUN it:

```
STRATA SOFTWARE
```

Save the current program as: STRATA

We will continue to add to this program. In the next chapter, you'll discover how user-defined characters can be used to create some striking effects.

Summary

〈ESC〉"%"CHR\$(n_1)CHR\$(n_2)	Activates a given character set, where n_1 indicates ROM (0) or RAM (1) and n_2 is 0.
〈ESC〉"&"CHR\$(n_1)CHR\$(n_2)CHR\$(n_3);	Defines characters, where n_1 selects the RAM buffer (0), n_2 is the starting character, and n_3 is the ending character.
〈ESC〉"6"	Enables printing of codes 128 to 159.
〈ESC〉"7"	Disables printing of codes 128 to 159.
〈ESC〉":"CHR\$(n_1)CHR\$(n_2)CHR\$(n_3)	Downloads ROM characters into RAM. All three numbers are 0.
〈ESC〉"I1"	Enables printing of the codes 0 to 31 except those used as control codes. The control codes can be printed with 〈ESC〉"R".
〈ESC〉"I0"	Disables printing of codes 0 to 31.

For each character in the 〈ESC〉"&" sequence from n_2 to n_3, the printer expects 12 data numbers. The first number, called the attribute byte, determines the height and width characteristics of the character at print time. The other 11 numbers determine the pin patterns used to print the character.

Chapter 18
Combining User-Defined Characters

In this chapter we'll explore the technique of combining user-defined characters to make large letters and symbols.

Large Letters: Double Wide

We'll start by placing two characters next to each other to form a double-width letter. Enter this new program, being careful to enter the line numbers as written (and SAVE the STRATA program from the last chapter first!):

```
NEW
20 LPRINT CHR$(27)":"CHR$(0)CHR$(0)CHR$(0);
25 ' Copies ROM to RAM;
   Model I users see Appendix I
30 LPRINT CHR$(27)"%"CHR$(1)CHR$(0); ' Activates RAM
60 LPRINT CHR$(27)"&"CHR$(0)"AB";
65 ' Defines characters A & B
70 FOR Y=1 TO 2: LPRINT CHR$(139); ' Attribute byte
80 FOR X = 1 TO 11: READ N: LPRINT CHR$(N);: NEXT X
90 NEXT Y
180 LPRINT CHR$(27)"@";: END
```

The ESCape sequences in lines 20, 30, and 60 are the standard fare from the last chapter. This program prepares the printer to define the two characters A and B. Enter the DATA lines:

```
200 DATA 0,2,5,0,9,16,3,32,70,0,84
210 DATA 68,32,66,49,8,21,10,5,2,0,0
215 ' See Appendix I for trouble codes
```

And how do they look side by side? Enter:

```
100 LPRINT "AB"
```

Very nice. Using two characters side by side provides a larger matrix and therefore gives more flexibility in character design. But there is one problem. When two user- defined characters are placed side by side, there is one intermediate column that contains no dots (unless the first character is printed in Emphasized Mode).

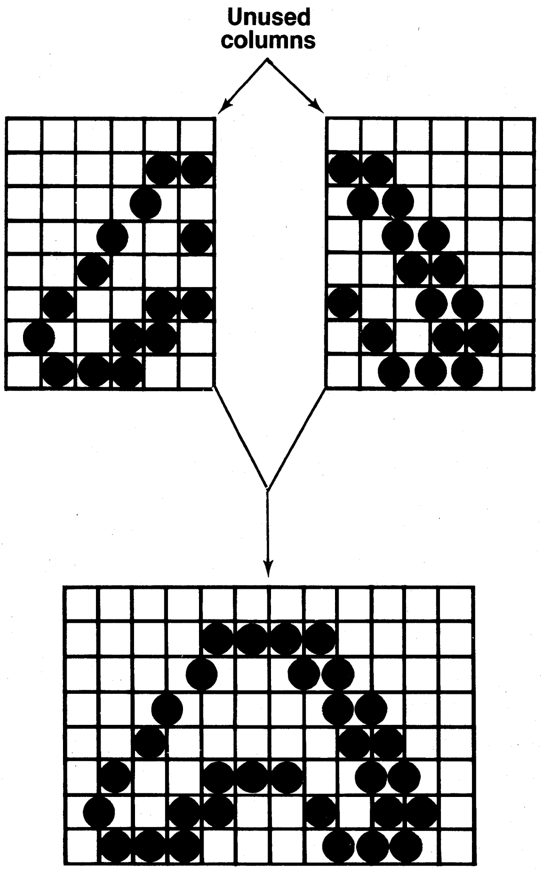


Figure 18-1. Side-by-side user-defined characters.

Figure 18-1 illustrates this problem. You'll have to design around the missing link.

Large Letters: Double High

As long as we're playing building blocks, let's stack two characters, one on top of the other, with these changes:

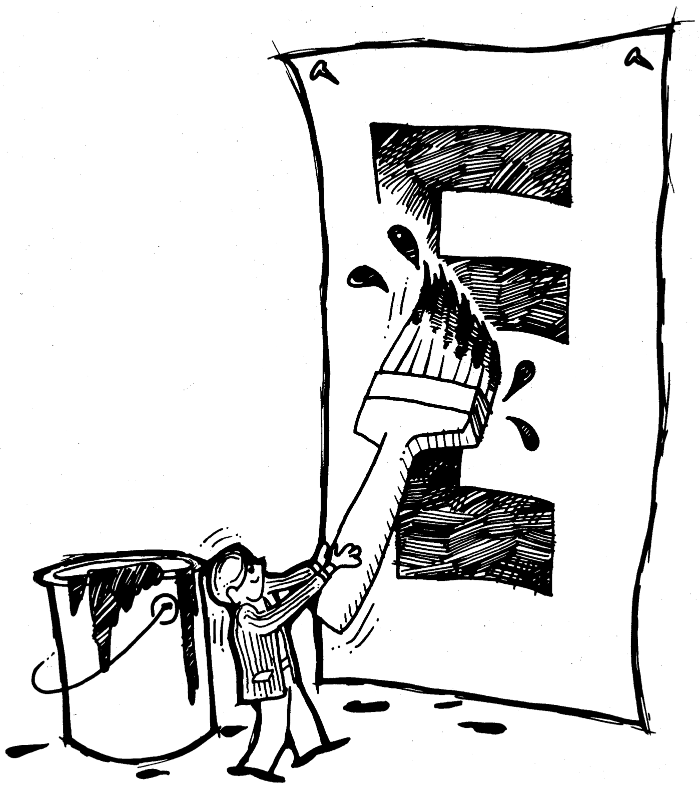
```
10 LPRINT CHR$(27)"1";
100 LPRINT "A"
110 LPRINT "B"
200 DATA 16,32,95,0,64,0,127,0,63,0,0
210 DATA 14,0,123,0,3,0,123,0,127,0,15
```

Line 10 changes the line spacing to seven dots. If there are slight gaps between rows, adjust the line spacing accordingly (e.g., CHR$(27)"3"CHR$(20)).

With a little imagination, you can do some dynamite things by combining characters.

Large Letters: Double High and Double Wide

For even more exciting type styles, you can design letters that are both two characters tall and two characters wide. This gives an 18 by 22 matrix in which your creativity can run rampant.



But which ASCII numbers can be used to store the four characters that will make up each letter . . . hmmmm? A quick glance at the ASCII chart (Appendix A) shows that there are four symbols that readily relate to each letter of the alphabet. They are the upper- and lower-case versions of each letter in its roman and italic typefaces. For example, the letter G could be designed using the following four ASCII codes:

CHR$(71)	Upper-case roman G
CHR$(103)	Lower-case roman g
CHR$(199)	Upper-case italic *G*
CHR$(231)	Lower-case italic *g*

Such usage is shown in Figure 18-2.

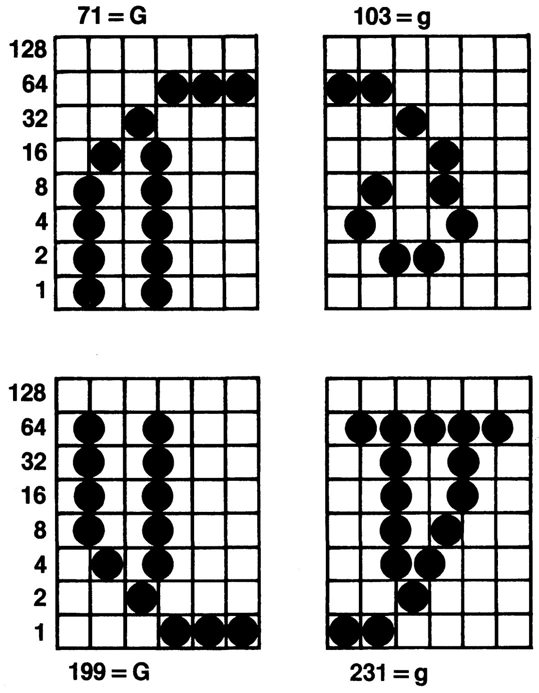


Figure 18-2. Double high and wide character.

Sorry, but you 7-bit users will have to skip to "Building Symbols from a Small Core Set," below.

The list above makes it easy to remember the four characters that will make up the "font" for each letter, but in order to define letters in groups of four, you'll have to modify the definition process. Fortunately, the ASCII numbers that represent the four different versions of each character have a consistent pattern. That pattern is shown in Table 18-1.

Table 18-1. ASCII Pattern for Character Versions.

Pattern	Example
letter = L	G = 72
lower-case letter = L + 32	g = 72 + 32 = 110
italic letter = L + 128	*G* = 72 + 128 = 200
italic lower-case letter = L + 160	*g* = 72 + 160 = 232

With this in mind, add these lines:

```
40 READ L: PRINT CHR$(L) ' Print to screen
50 FOR Y=0 TO 1: FOR Z=0 TO 1:
   A = L + 128*Y + 32*Z
```

and make these changes:

```
60 LPRINT CHR$(27)"&"CHR$(0)CHR$(A)CHR$(A);
70 LPRINT CHR$(139);
90 NEXT Z: NEXT Y
```

Line 50 calculates the code (A) to be defined in line 60 by adding the appropriate amount to the base letter L. Line 60 is the ⟨ESC⟩"&" defining sequence, and line 70 sets the attribute byte to 139.

The code for the letter to be defined and the data for its four components are stored in DATA statements. Type:

```
250 ' My G
260 DATA 71
270 DATA 0,15,16,0,32,31,64,0,64,0,64
280 DATA 64,4,72,2,32,2,24,4,0,0,0
290 DATA 0,120,4,0,2,124,1,0,1,0,1
300 DATA 1,64,0,124,2,68,8,120,0,64,0
```

Let's Print

Here's the printing routine:

```
100 A$="": INPUT "ENTER A STRING"; A$:
    IF A$="" THEN 180
110 INPUT "ENTER A MASTER PRINT MODE NUMBER";M
120 LPRINT CHR$(27)"!"CHR$(M);
130 FOR Y=0 TO 1: FOR X=1 TO LEN(A$)
140 A = ASC(MID$(A$,X,1)) + 128*Y
150 LPRINT CHR$(A)CHR$(A+32);
160 NEXT X: LPRINT: NEXT Y
170 LPRINT: GOTO 100
```

This program automatically prints all four parts of each letter. You type just a single letter in the text; it does the rest. Before you RUN, check it against Figure 18-3.

```
10 LPRINT CHR$(27)"1";
20 LPRINT CHR$(27)":"CHR$(0)CHR$(0)CHR$(0)CHR$(0);
30 LPRINT CHR$(27)"%"CHR$(1)CHR$(0);
40 READ L: PRINT CHR$(L)
50 FOR Y=0 TO 1: FOR Z=0 TO 1: A = L + 128*Y + 32*Z
60 LPRINT CHR$(27)"&"CHR$(0)CHR$(A)CHR$(A);
70 LPRINT CHR$(139);
80 FOR X=1 TO 11: READ N: LPRINT CHR$(N);: NEXT X
90 NEXT Z: NEXT Y
100 A$="": INPUT "ENTER A STRING";A$: IF A$="" THEN 180
110 INPUT "ENTER A MASTER PRINT MODE NUMBER";M
120 LPRINT CHR$(27)"!"CHR$(M);
130 FOR Y=0 TO 1: FOR X=1 TO LEN(A$)
140 A = ASC(MID$(A$,X,1)) + 128*Y
150 LPRINT CHR$(A)CHR$(A+32);
160 NEXT X: LPRINT: NEXT Y
170 LPRINT: GOTO 100
180 LPRINT CHR$(27)"@": STOP
250 ' G
260 DATA 71
270 DATA 0,15,16,0,32,31,64,0,64,0,64
280 DATA 64,4,72,2,32,2,24,4,0,0,0-
290 DATA 0,120,4,0,2,124,1,0,1,0,1
300 DATA 1,64,0,124,2,68,8,120,0,64,0
```

Figure 18-3. Program for large G.

When you RUN it, you should see the prompt:

ENTER A STRING?

That's from line 100, of course. You can respond with any string of letters, but for now type:

`GO` 〈ENTER〉 (don't type a space after the O)

The next prompt on the screen is:

ENTER A MASTER PRINT MODE NUMBER?

For now, enter a 24. Remember, all codes from 0 to 255 produce a combination of print modes, but there are only 16 unique combinations. You may want to refer to one of the Master Select charts from Chapter 5.

Press 〈ENTER〉 to stop the program.

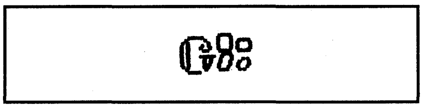


Figure 18-4. Large G.

Gee, that's nice! The four characters defined in lines 270 to 300 combine to print a large G. The O is printed as four small characters. Figure 18-4 illustrates the way the program arranges the four versions of each character to make one oversized letter.

Line 140 examines the A$ string, character by character, and determines its ASCII value. Line 150 prints both upper- and lower-case versions of each character on the first pass of the print head. On the second pass, Y is set to 1 and 128 is added to A in line 140. Thus line 150 prints the italic versions of the character.

To get a better idea of what this program can do, you'll need to add some more data. The DATA lines below supply data for the letters A-M-E-S and for the space character. (It is necessary to redefine the space character since two of its four components print as @ signs.)

Be careful. Although the space character is user- definable, you should avoid using it as anything other than a space character. No matter how it is defined, the space character will never print at the beginning or end of a line. That's fine for a space but if it's an important letter or something . . . see the problem?

Enter the data lines as shown in Figure 18-5.

```
190 ' SPACE
200 DATA 32
210 DATA 0,0,0,0,0,0,0,0,0,0,0
220 DATA 0,0,0,0,0,0,0,0,0,0,0
230 DATA 0,0,0,0,0,0,0,0,0,0,0
240 DATA 0,0,0,0,0,0,0,0,0,0,0
310 ' A
320 DATA 65
330 DATA 0,0,1,0,1,0,6,24,32,92
340 DATA 67,32,34,4,3,0,0,0,0,0,0
350 DATA 0,65,32,7,24,33,64,32,16,0,8
360 DATA 8,97,24,7,0,97,24,7,0,1,0
370 ' M
380 DATA 77
390 DATA 0,64,0,127,0,32,16,7,8,0,4
400 DATA 4,0,8,7,16,32,0,127,0,64,0
410 DATA 0,1,0,127,0,1,0,127,0,1,0
420 DATA 0,1,0,127,0,1,0,127,0,1,0
430 ' E
440 DATA 69
450 DATA 0,64,0,127,0,64,0,62,65,0,65
460 DATA 64,0,67,0,64,0,64,32,0,0,0
470 DATA 0,1,0,127,0,1,0,126,1,0,1
480 DATA 1,0,69,0,9,0,9,0,6,0,0
490 ' S
500 DATA 83
510 DATA 0,0,0,24,36,0,66,16,105,0,68
520 DATA 74,0,69,0,68,0,40,16,0,0,0
530 DATA 0,8,22,0,33,0,33,16,1,0,65
540 DATA 33,0,17,0,75,4,33,0,22,8,0
```

Figure 18-5 Data for AMES.

Now add a loop to READ the new data by changing these lines:

```
40 FOR W=1 TO 6: READ L: PRINT CHR$(L)
90 NEXT Z: NEXT Y: NEXT W
```

And RUN. This time type:

```
GAMES SEEM SAME
49
```

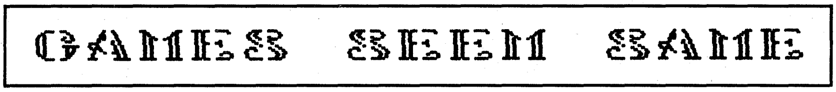


Figure 18-6. Games seem same.

How about that! The letters that make up Figure 18-6 are pretty fancy, right? But look at those gaps down the middle of each letter. We can do without those . . . or can we?

Those gaps appear when characters are printed side by side in any mode that compresses the matrix columns. That includes Elite and Compressed Modes. For a comparison of the three print pitches, RUN the program again and enter:

MESSAGES, 48
MESSAGES, 49
MESSAGES, 52

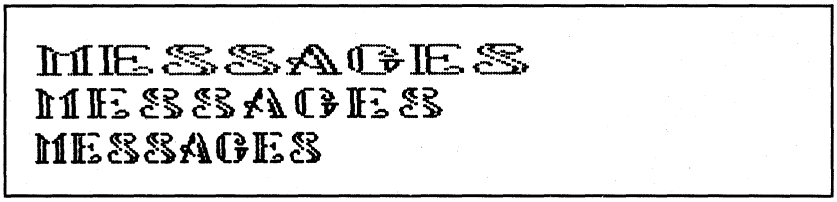


Figure 18-7. Messages in three pitches.

When you look at Figure 18-7, do you get a message? All three mode combinations include Double-Strike and Expanded print; the only difference between them is the pitch. The first pitch is Pica, the second is Elite, the third is Compressed.

As you can see, the letters printed in the Elite and Compressed Modes have gaps between their component characters. Unless you like this "feature," you'll have to avoid Elite and Compressed Modes when printing user-defined characters side by side.

Despite this limitation, you should have a good time adding the rest of the alphabet or defining your own character set. By the way, the Introduction at the beginning of this manual shows a few more of these Double Wide and Double High letters.

You may want to SAVE the current program before proceeding.

Building Symbols from a Small Core Set

Combining user-defined characters is a great way to create frequently used logos or fancy headings. But as you saw, defining an entire alphabet of oversized letters uses up ASCII codes rather quickly.

Fortunately, there is an alternative. In some cases, you may be able to define a handful of "core" characters that could be combined to make any letter in the alphabet. This requires a bit of imagination; we present an example here to lubricate those creative gears.

Prepare for the program changes by deleting lines 20, 40, 50, and 100 to 540. Now change:

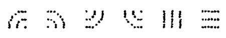
```
6Ø LPRINT CHR$(27)"&"CHR$(Ø)"16";
7Ø FOR Y=1 TO 6: LPRINT CHR$(139);
9Ø NEXT Y
```

Deleting line 20 ensures that the printer does not download the ROM characters. That makes your defined characters the only ones around—no funny stuff on the printer. Here is the data:

```
1ØØ ' Six Characters
11Ø DATA 7,8,16,Ø,32,3,68,Ø,72,Ø,73
12Ø DATA 73,Ø,72,Ø,68,3,32,Ø,16,8,7
13Ø DATA 73,Ø,9,Ø,17,96,2,Ø,4,8,112
14Ø DATA 112,8,4,Ø,2,96,17,Ø,9,Ø,73
15Ø DATA 127,Ø,Ø,Ø,Ø,127,Ø,Ø,Ø,Ø,127
16Ø DATA 73,73,73,73,73,73,73,73,73,73,73
```

That's right, there are only six characters, but it is a very powerful set of characters. With them, you can print an entire alphabet and more! To see the "Magnificent Six," type:

```
18Ø LPRINT "1 2 3 4 5 6"
2ØØ LPRINT CHR$(27)"@": STOP
```



You want more excitement? Okay, try printing them in a different order. Type:

```
170 FOR Y=1 TO 5
180 READ P$: LPRINT P$
190 NEXT Y
210 ' Tracks
220 DATA "626626201620162620501 66"
230 DATA "05005050505050505050500"
240 DATA "05005630565050005630462"
250 DATA "05005050505050505050005"
260 DATA "05005046304636305046663"
```

Give it a RUN to get the pattern shown in Figure 18-8.

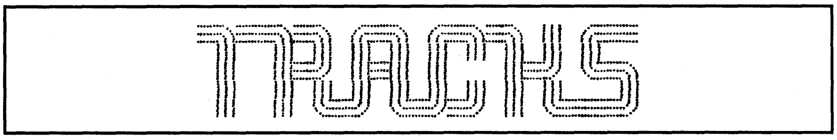


Figure 18-8. Tracks.

That's one powerful character set! But it doesn't stop there. Those little characters can do some amazing tricks. Try this on for size:

```
DELETE 210-260

210 ' Pattern
220 DATA "00012000","01665620"
230 DATA "05055050","16634652"
240 DATA "45621663","05055050"
250 DATA "04656630","00043000"
```

And change this line before you RUN it for Figure 18-9:

```
170 FOR Y=1 TO 8
```

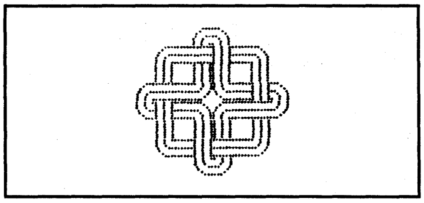


Figure 18-9. Intricate pattern.

Have fun creating your own designs with these characters. You may wish to SAVE the program before proceeding.

Line Graphics

The later MX series printers have a set of "line graphics" characters stored in ROM. In the following program, we define a similar set of characters.

What are line graphics characters, you ask? They are a set of characters that fit together to make borders and outlines for all kinds of forms. Since they are so useful, we use them again in the next chapter; be sure to save the next program when you are through.

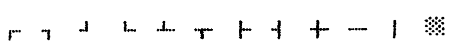
Once again we will add to the program which you built and saved (as STRATA) in the last chapter. LOAD in the STRATA program now, and type:

```
DELETE 18Ø-19Ø

13Ø LPRINT CHR$(27)"&" CHR$(Ø)"al";
14Ø FOR Y=1 TO 12: LPRINT CHR$(139);
17Ø LPRINT "a b c d e f g h i j k l"
```

Remember, if your computer cannot send lower-case letters, use the ASCII values for the letters you need (see Appendix A). Add:

```
899 ' <<< LINE GRAPHICS AND SHADING >>>
9ØØ DATA Ø,Ø,Ø,Ø,15,Ø,8,Ø,8,Ø,8: ' a
91Ø DATA 8,Ø,8,Ø,15,Ø,Ø,Ø,Ø,Ø,Ø: ' b
92Ø DATA 8,Ø,8,Ø,12Ø,Ø,Ø,Ø,Ø,Ø,Ø: ' c
93Ø DATA Ø,Ø,Ø,Ø,12Ø,Ø,8,Ø,8,Ø,8: ' d
94Ø DATA 8,Ø,8,Ø,12Ø,Ø,8,Ø,8,Ø,8: ' e
95Ø DATA 8,Ø,8,Ø,15,Ø,8,Ø,8,Ø,8: ' f
96Ø DATA Ø,Ø,Ø,Ø,127,Ø,8,Ø,8,Ø,8: ' g
97Ø DATA 8,Ø,8,Ø,127,Ø,Ø,Ø,Ø,Ø,Ø: ' h
98Ø DATA 8,Ø,8,Ø,127,Ø,8,Ø,8,Ø,8: ' i
99Ø DATA 8,Ø,8,Ø,8,Ø,8,Ø,8,Ø,8: ' j
1ØØØ DATA Ø,Ø,Ø,Ø,127,Ø,Ø,Ø,Ø,Ø,Ø:' k
1Ø1Ø DATA 84,Ø,17Ø,Ø,84,Ø,17Ø,Ø,84,Ø,17Ø: ' l
```



That's what the line graphics characters look like on their off hours. You can put them to work like this:

```
170 LPRINT "ajjjjjjfjjjjjjjb"
175 LPRINT "k NAME k PHONE k"
180 LPRINT "gjjjjjjijjjjjjjh"
185 LPRINT "kØØØØØØkØØØØØØØk"
190 LPRINT "djjjjjjejjjjjjjc"
```

and then RUN the program for Figure 18-10:

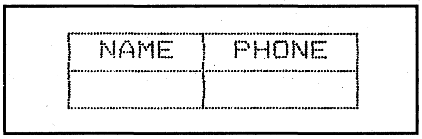


Figure 18-10.
Phone list.

Note that the text lines use upper-case letters (the lower-case letters are user-defined).

Make sure you SAVE this program as LINE—but before you do,

```
DELETE 170-190
```

Don't miss the next chapter—we show you how to apply many of the techniques you have learned so far.

Chapter 19
Printing Applications

You've come a long way with the FX-80 printer. By now you and your FX-80 should be great pals and able to print just about anything together. Now you both get a chance to show off.

In this chapter we present two programs that really put the FX-80 through its paces. The programs pull together many of the programming techniques that you've learned throughout the course of this manual.

The programs are considerably longer than those in previous chapters. If your typing fingers are sore, go in peace and save this chapter for another day. But we do encourage you to try these programs for two reasons. First, you'll have a chance to really test your understanding of the FX-80 printer. Second, these longer programs show you how truly versatile your printer can be.

Examine the programs at your leisure, and by all means enjoy using them on your printer!

Preparing for Take-off

Load in the LINE program you saved at the end of Chapter 18. It contains data for line graphics characters, several user-defined letters, and a graphics logo. The first program in this chapter uses the line graphics characters to print a sales chart. The second uses all three sets of data to print a statement form.

Take a moment to look over the LINE program before we modify it. Lines 100 through 160 handle the details of defining characters. Lines 900 to 1010 contain the data for the line graphics characters, and lines 1100 to 1170 hold the data for the user-defined letters. Lines 690 to 770 draw the logo from Chapter 12; lines 1200 to 1250 hold the data for the logo.

Barchart

This program creates the barchart shown in Figure 19-1. It uses the line graphics characters from the LINE program as well as three new user-defined characters. The following lines must be altered:

```
100 LPRINT CHR$(27)"1";CHR$(27)"U1";
130 LPRINT CHR$(27)"&"CHR$(0)CHR$(94)CHR$(107);
135 ' Model I users see Appendix I
140 FOR Y=1 TO 14: LPRINT CHR$(139);
160 NEXT Y: LPRINT CHR$(27)"C"CHR$(66);
```

Note: You may have to change line spacing to ⟨ESC⟩"3"CHR$(20) to avoid gaps in vertical spacing.

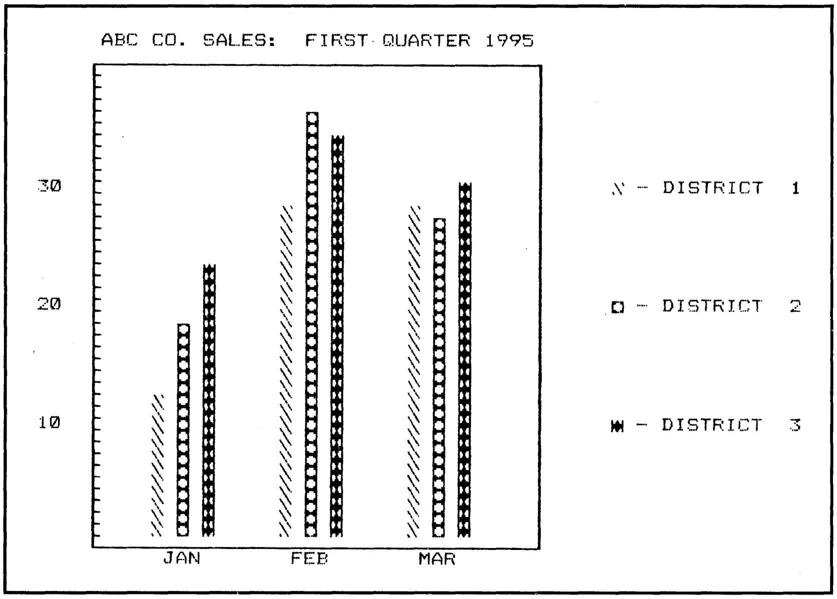


Figure 19-1. Barchart.

The ⟨ESC⟩"U1" activates unidirectional print so that the line graphics characters will line up properly. Line 130 selects 14 characters (^ through k) to be defined. Line 140 starts the READ loop and sends the attribute byte. Line 160 closes the Y loop and sets the form length to 66 lines. It also sets the Top Of Form to the current position of the print head.

The other program changes are additions of new lines. Enter:

```
10 FOR J=1 TO 3: FOR K=1 TO 3
20 READ MAX(J,K)
30 NEXT K: NEXT J
40 DATA 12,18,23,28,36,34,28,27,30
```

The maximum heights for the vertical bars in the chart are READ from line 40 and stored in the array, MAX. The next data lines define three new characters:

```
50 DATA 2,1,64,32,16,8,4,2,1,64,32
60 DATA 127,0,99,0,65,0,65,0,99,0,127
70 DATA 127,0,28,0,62,65,62,0,28,0,127
```

Reverse and forward line feeds as well as emphasized print need to be switched on and off throughout the program—storing the appropriate commands as strings makes them relatively easy to use:

```
80 U$=CHR$(27)+"j"+CHR$(11): B$=CHR$(27)+"E"
90 D$=CHR$(27)+"J"+CHR$(11); C$=CHR$(27)+"F"
95 ' Use 10 or 12 if CHR$(11) doesn't work
```

The next 18 lines print the barchart, starting with:

```
190 LPRINT CHR$(27)"D"CHR$(14)CHR$(19)
    CHR$(24)CHR$(34)CHR$(44)CHR$(54);
200 LPRINT CHR$(60)CHR$(1);: H$=CHR$(9): Z=1
```

Lines 190 and 200 set horizontal tab stops and store the horizontal tab character in H$. There's more:

```
210 LPRINT H$;H$;
    " ABC CO. SALES: FIRST QUARTER 1995": LPRINT
220 LPRINT H$;H$;B$;"a";: N=34: A$="j":
    GOSUB 800: LPRINT "b"
799 ' *** STRING$ ROUTINE ***
800 FOR J=1 TO N: LPRINT A$: NEXT J: RETURN
```

Lines 210 and 220 start the printing, using horizontal tabs and special characters. For systems without a STRING$ function, the GOSUB in line 220 can print a string of characters. Add:

```
230 LPRINT H$;H$;U$;"k";H$;H$;H$;H$;"k"
240 FOR R=39 TO 1 STEP -1: LPRINT H$;: F=0
250 IF R/10=INT(R/10) THEN LPRINT R;: F=1
260 LPRINT B$;H$;U$;"g";D$;C$;
```

Line 250 prints the value of R every tenth line and sets a flag (F) to print the "DISTRICTS."

```
270 FOR M=1 TO 3: LPRINT H$;
280 FOR P=1 TO 3
290 IF R>MAX(M,P) THEN LPRINT " ";
    ELSE LPRINT CHR$(93+P);" ";
```

Line 290 compares the current row (R) with the array (MAX) to determine whether to print a character or a blank space. And finally, add:

```
300 NEXT P: NEXT M: LPRINT B$;H$;"k"; C$;:
    IF F=0 THEN LPRINT: GOTO 320
310 LPRINT H$;CHR$(93+Z)" - DISTRICT ";Z: Z=Z+1
320 NEXT R: LPRINT H$;H$;B$;U$;"g";
    H$;H$;H$;H$;"k"
330 LPRINT H$;H$;"d";: A$="j":
    GOSUB 800: LPRINT "c"
340 LPRINT C$;H$;H$;H$;" JAN";
    H$;" FEB";H$;" MAR": LPRINT
390 LPRINT CHR$(27)"@": STOP
```

Line 300 closes the P and M loops, prints the right-hand border, and sends control to either 310 or 320. Line 310 prints the districts. Lines 320 through 340 print the bottom portion of the chart.

Now it's time to RUN your program and see if it looks like Figure 19-1.

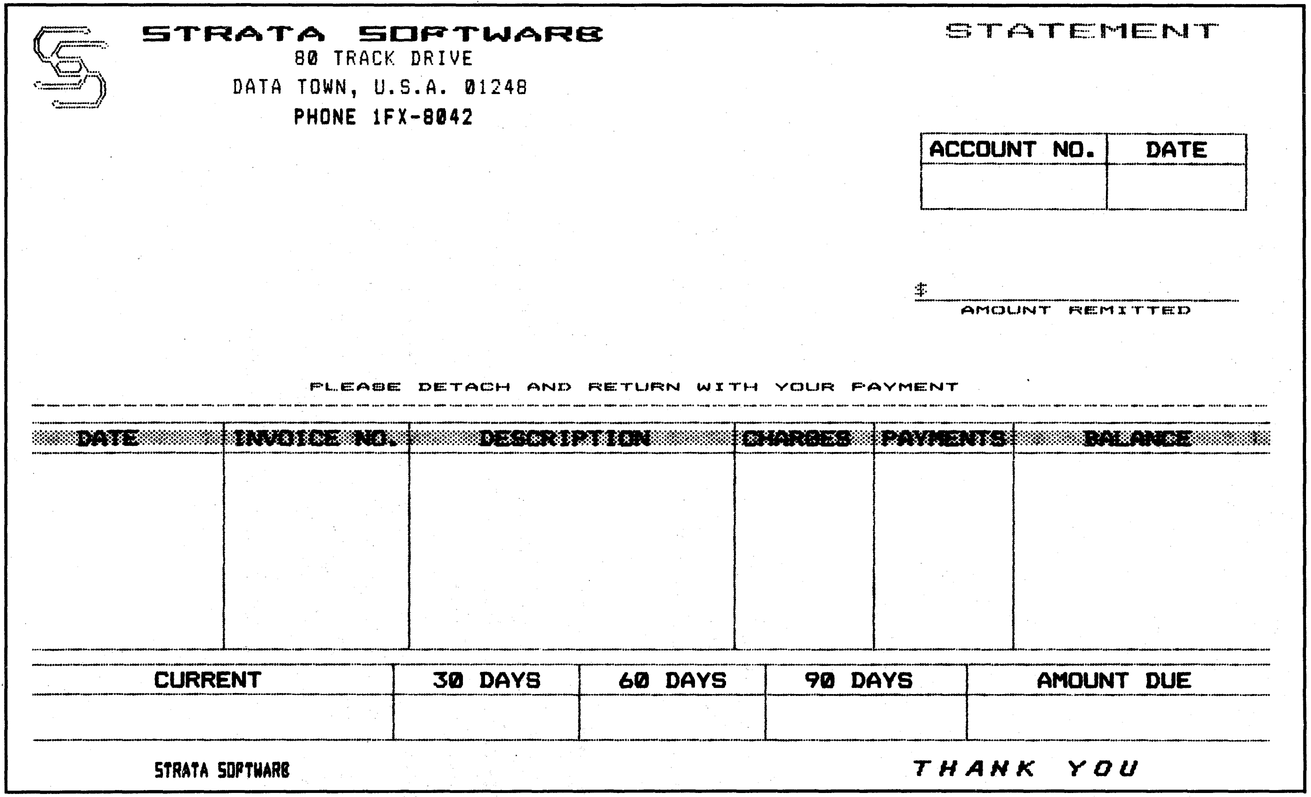
The heights of the bars can be changed by adjusting the data in line 40. Go ahead and SAVE the program as BARCHART if you like. The next program starts where this one ends.

Statement Form

The final program produces the stunning statement form that appears as Figure 19-2. It's amazing what you can do with a full-feature printer like the FX-80!

Do you recognize parts of the figure from previous chapters? There is the logo from Chapter 12, user-defined letters (SOFTWARE) from Chapter 17, and line graphics characters from Chapter 18. This program combines all those routines into one neat package.

Figure 19-3 gives the complete listing for the statement form program:



STRATA SOFTWARE
80 TRACK DRIVE
DATA TOWN, U.S.A. 01248
PHONE 1FX-8042

STATEMENT

ACCOUNT NO.	DATE

$ ____________
AMOUNT REMITTED

PLEASE DETACH AND RETURN WITH YOUR PAYMENT

DATE	INVOICE NO.	DESCRIPTION	CHARGES	PAYMENTS	BALANCE

CURRENT	30 DAYS	60 DAYS	90 DAYS	AMOUNT DUE

STRATA SOFTWARE

THANK YOU

Figure 19-2. Statement form.

```
100 DIM A(18): LPRINT CHR$(27)"1";CHR$(27)"U1";
105 ' may need to change line spacing to CHR$(27)"3"CHR$(20);
110 LPRINT CHR$(27)":"CHR$(0)CHR$(0)CHR$(0);
120 LPRINT CHR$(27)"%"CHR$(1)CHR$(0);
130 LPRINT CHR$(27)"&"CHR$(0)"at";
140 FOR Y=1 TO 20: LPRINT CHR$(139);
150 FOR X=1 TO 11: READ C: LPRINT CHR$(C);: NEXT X
160 NEXT Y: LPRINT CHR$(27)"C"CHR$(66);: GOSUB 700
170 LPRINT CHR$(27)"!8";" mpsrpr mnopqrst";CHR$(27)"1@";
180 LPRINT CHR$(27)"B"CHR$(18)CHR$(25)CHR$(1);
190 LPRINT
    CHR$(27)"D"CHR$(13)CHR$(17)CHR$(57)CHR$(69)CHR$(1);
200 H$=CHR$(9): LPRINT H$;CHR$(14)" STATEMENT"
210 GOSUB 700: LPRINT: GOSUB 700
220 LPRINT H$;H$;CHR$(27)"!A";"80 TRACK DRIVE": GOSUB 700
230 LPRINT: GOSUB 700: LPRINT H$;"DATA TOWN, U.S.A. 01248":
    GOSUB 700
240 LPRINT: LPRINT H$;H$;CHR$(27)"!Q";"PHONE 1FX-8042"
250 LPRINT CHR$(27)"!@";
260 LPRINT CHR$(27)"D"CHR$(57)CHR$(72)CHR$(1);
270 C=2: H=2: F=0: FT=1: GOSUB 500: LPRINT
280 LPRINT CHR$(11);H$;"$"CHR$(8);: A$=CHR$(95): N=21:
    GOSUB 800
290 LPRINT: LPRINT H$;CHR$(27)"S1";  AMOUNT REMITTED"
300 LPRINT CHR$(27)"D"CHR$(3)CHR$(13)CHR$(29)CHR$(46)
    CHR$(55)CHR$(68)CHR$(1);
310 LPRINT CHR$(11);H$;H$;CHR$(27)"S0";  "  PLEASE DETACH AN
    RETURN WITH YOUR PAYMENT"
320 LPRINT CHR$(27)"T";: N=80: A$="-": GOSUB 800: LPRINT
330 C=6: H=12: F=1: FT=0: GOSUB 500
340 LPRINT CHR$(27)"D"CHR$(8)CHR$(26)CHR$(38)CHR$(50)
    CHR$(65)CHR$(1);
350 C=5: H=2: F=0: GOSUB 500: LPRINT
360 LPRINT CHR$(27)"D"CHR&(8)CHR$(57)CHR$(0)
370 LPRINT CHR$(27)"!T";H$;"mpsrpr mnopqrst"H$;
380 LPRINT CHR$(27)"!1"; CHR$(27)"4";"THANK YOU"
390 LPRINT CHR$(27)"@": STOP
499 ' *** BOX SUBROUTINE ***
500 FOR K=1 TO 5: READ L$(K),M$(K),N$(K),R$(K): NEXT K
510 FOR K=1 TO C: READ W(K): NEXT K
520 FOR L=1 TO 5: IF L=4 THEN FOR G=1 TO H
530   IF FT=1 THEN LPRINT H$;
540   LPRINT L$(L);: FOR K=1 TO C-1
550   FOR J=1 TO W(K): LPRINT M$(L);: NEXT J
```

Figure 19-3. Program for STATEMENT.

```
560  LPRINT N$(L);: NEXT K
570  N=W(C): A$=M$(L): GOSUB 8ØØ: LPRINT R$(L);
58Ø  IF L<>2 THEN 64Ø
59Ø  LPRINT CHR$(27)"!X";CHR$(27)"A"CHR$(1)
6ØØ  FOR Q=1 TO C: READ T$: LPRINT H$;T$;: NEXT Q
61Ø  LPRINT CHR$(27)"!@";
62Ø  IF F=1 THEN LPRINT CHR$(27)"A"CHR$(Ø):  N=8Ø: A$="1":
    GOSUB 8ØØ
63Ø  LPRINT CHR$(27)"j"CHR$(3);
64Ø  IF L<5 THEN LPRINT CHR$(27)"1"
65Ø  IF L=4 THEN NEXT G
66Ø NEXT L: RETURN
699 ' *** DRAW LOGO ***
7ØØ LPRINT CHR$(27)"L"CHR(6Ø)CHR$(Ø);
71Ø READ N: IF N=128 THEN 77Ø
72Ø IF N>=Ø THEN LPRINT CHR$(N);: GOTO 71Ø
73Ø READ P,R: FOR J=1 TO -N: LPRINT CHR$(P)  CHR$(R);: NEXT J
74Ø GOTO 71Ø
77Ø RETURN
799 ' *** STRING$ ROUTINE ***
8ØØ FOR J=1 TO N: LPRINT A$;: NEXT J: RETURN
899 ' <<< LINE GRAPHICS AND SHADING >>>
9ØØ DATA Ø,Ø,Ø,Ø,15,Ø,8,Ø,8,Ø,8:  'a
91Ø DATA 8,Ø,8,Ø,15,Ø,Ø,Ø,Ø,Ø,Ø:  'b
92Ø DATA 8,Ø,8,Ø,12Ø,Ø,Ø,Ø,Ø,Ø,Ø: 'c
93Ø DATA Ø,Ø,Ø,Ø,12Ø,Ø,8,Ø,8,Ø,8: 'd
94Ø DATA 8,Ø,8,Ø,12Ø,Ø,8,Ø,8,Ø,8: 'e
95Ø DATA 8,Ø,8,Ø,15,Ø,8,Ø,8,Ø,8:  'f
96Ø DATA Ø,Ø,Ø,Ø,127,Ø,8,Ø,8,Ø,8: 'g
97Ø DATA 8,Ø,8,Ø,127,Ø,Ø,Ø,Ø,Ø,Ø 'h
98Ø DATA 8,Ø,8,Ø,127,Ø,8,Ø,8,Ø,8: 'i
99Ø DATA 8,Ø,8,Ø,8,Ø,8,Ø,8,Ø,8:   'j
1ØØØ DATA Ø,Ø,Ø,Ø,127,Ø,Ø,Ø,Ø,Ø,Ø:'k
1Ø1Ø DATA 84,Ø,17Ø,Ø,84,Ø,17Ø,Ø,84,Ø,17Ø:'l
1Ø99 ' <<< STRATA SOFTWARE >>>
11ØØ DATA Ø,121,Ø,73,Ø,73,Ø,73,Ø,79,Ø: 'm - S
111Ø DATA Ø,127,Ø,65,Ø,65,Ø,65,Ø,127,Ø:'n - O
112Ø DATA Ø,63,64,8,64,8,64,28,64,32,Ø:'o - F
113Ø DATA Ø,32,64,Ø,64,63,64,Ø,64,32,Ø:'p - T
114Ø DATA Ø,126,1,2,4,8,4,2,1,126,Ø:  'q - W
115Ø DATA Ø,7,8,16,36,64,36,16,8,7,Ø: 'r - A
116Ø DATA Ø,127,Ø,72,Ø,72,Ø,76,2,121,Ø:'s - R
117Ø DATA Ø,62,65,8,65,8,65,28,65,34,Ø:'t - E
1199 '<<< LOGO DATA >>>
12ØØ DATA Ø,1,2,4,9,18,36,72,-16,16,64,8,64,8,32,16,Ø -
     7,Ø,Ø,128
```

```
1210 DATA 0,126,1,0,126,1,-5,0,0,1,2,4,9,18,36,-
    16,8,32,4,32,4,16,8,0,128
1220 DATA 0,0,0,64,32,16,72,36,-3,16,4,34,65,0,0,65,34,-
    8,16,4,18,9,4,2,1,0,-9,0,0,128
1230 DATA -8,0,0,64,32,16,72,36,16,-
    7,4,16,36,65,0,0,1,66,36,16,-3,4,16,4,18,9,4,2,1,-
    2,0,0,128
1240 DATA 0,32,16,64,8,64,-15,8,32,72,16,32,,64,-
    ,0,0,127,0,0,127,0,0,0,128
1250 DATA -7,0,0,0,8,4,16,2,16,-15,2,8,18,36,72,16,32,
    64,-2,0,0,128
1299 ' <<< BOX DATA >>>
1300 DATA a,j,f,b,k," ",k,k,g,j,i,h,k," ",k,k, d,j,e,c
1310 DATA 11,8," ACCOUNT NO.","DATE"
1320 DATA j,j,f,j," "," ",k," ",j,j,i,j," "," ", k,"
    ",j,j,e,j
1330 DATA 11,11,20,8,8,15,"DATE","INVOICE NO."
1340 DATA "DESCRIPTION", "CHARGES", "PAYMENTS","BALANCE"
1350 DATA j,j,f,j," "," ",k," ",j,j,i,j, " "," ",k,"
    ",j,j,e,j
1360 DATA 22,11,11,12,18
1370 DATA "CURRENT","30 DAYS","60 DAYS", "90 DAYS","AMOUNT
    DUE"
```

Entering the program

You could enter the entire program straight from the listing, but it's much easier to start with the BARCHART program from the first half of this chapter. Many of the lines in the BARCHART program that have accumulated from previous chapters need no changes. These include:

110, 120, 150, and 390
699 to 740
799 to 1250

So let's get started. First,

```
DELETE 10-90
```

Some of the lines require slight changes. Make those now:

```
100 DIM A(18): LPRINT CHR$(27)"1";CHR$(27)"U1";
105 ' May need to change line spacing to
    CHR$(27)"3"CHR$(20);
130 LPRINT CHR$(27)"&"CHR$(0)"at";
140 FOR Y=1 TO 20: LPRINT CHR$(139);
160 NEXT Y: LPRINT CHR$(27)"C"CHR$(66);: GOSUB 700
```

Lines 130 and 140 prepare to define 20 characters. Line 160 sets the form length to 66 lines and fixes the Top Of Form at the current location of the print head. It then goes to the logo routine, which starts at line 700. The logo is called as a subroutine, one line at a time, with these changes:

```
DELETE 69Ø
77Ø RETURN
```

The other lines to be entered are in two long sections:

170 to 660
1299 to 1370

Enter them one at a time, replacing the corresponding lines from the BARCHART program.

When you are finished, SAVE the program as STATEMENT, and RUN it. If your output looks like Figure 19-2, let out a loud hooray; you have just graduated with flying colors!

Program overview

The STATEMENT program is intended for your enjoyment. Feel free to modify it and use it as you see fit. For those who want to understand its inner workings in detail, we provide a brief overview that is followed by a line-by-line description of the main portion of the program.

The program breaks down into several large blocks of routines and data:

Lines	*Routine*
100-150	Defines characters and does housekeeping
160-390	Prints the statement form
500-660	The box subroutine
700-770	The logo subroutine
800	The STRING$ subroutine
900-1010	Data for line graphics
1100-1170	Data for the "strata software" letters
1200-1250	Data for the logo
1300-1370	Data for the box routine

Line-by-line comments

Line 170 prints STRATA SOFTWARE in Master Select Mode 56.
Lines 180 and 190 set vertical and horizontal tab stops.
Line 200 stores the tab command in H$ and prints STATEMENT in expanded print.
Line 210 prints the second and third lines of the logo.
Lines 220-250 print more of the logo and the address in various mode combinations.
Line 260 sets new horizontal tab stops.
Line 270 sets some variables for the box subroutine at 500. That routine prints the box in the upper right corner of the sheet.
Line 280 does a vertical tab, then prints a string of 21 underline characters (ASCII 95) via subroutine 800.
Line 290 prints the subscripted AMOUNT REMITTED.
Line 300 sets new horizontal tab stops.
Line 310 does two vertical tabs then prints a superscripted message.
Line 320 cancels scripts and prints 80 hyphens with subroutine 800.
Line 330 calls the box routine.
Line 340 sets new horizontal tabs.
Line 350 calls the box routine.
Line 360 sets new horizontal tabs.
Line 370 prints STRATA SOFTWARE in a different print mode.
Line 380 thanks us in italic characters.
Line 390 resets all modes and stops the program.

Deciphering the box routine in lines 500 through 660 is left as an exercise for you. The following hints will get you started.

The subroutine at 500 is used to create three boxes of different sizes and characteristics, using the line graphics characters. The data stored in lines 1300 to 1370 determine which line graphics characters are used to print the boxes, the width of each cell, and the headings.

The variables sent to the subroutine are:

C -	the number of cells
H -	the height of the cells
F -	a flag for shading the headings
FT -	a flag that allows a horizontal tab to adjust the left margin of the box.

999 REM: THE END

In this chapter, we've presented two programs that utilize many of the features of the FX-80 printer. The programs demonstrate the tremendous potential of the powerful tool you have at your beck and call. We hope they inspire you to use the FX-80 in many creative ways to enhance your own programming applications.

Appendix A
ASCII Charts

I. Codes for U.S. Characters

Dec	Hex	Character or Function
0	00	none
1	01	none
2	02	none
3	03	none
4	04	none
5	05	none
6	06	none
7	07	BEL
8	08	BS
9	09	HT
10	0A	LF
11	0B	VT
12	0C	FF
13	0D	CR
14	0E	SO
15	0F	SI
16	10	none
17	11	DC1
18	12	DC2
19	13	DC3
20	14	DC4
21	15	none
22	16	none
23	17	none
24	18	CAN
25	19	none
26	1A	none
27	1B	ESC
28	1C	none
29	1D	none
30	1E	none
31	1F	none
32	20	ƀ (space)
33	21	! (Roman)
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	' (apostrophe)
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	, (comma)
45	2D	- (en dash)
46	2E	. (period)
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<

Dec	Hex	Character or Function	Dec	Hex	Character or Function
61	3D	=	101	65	e
62	3E	>	102	66	f
63	3F	?	103	67	g
64	40	@	104	68	h
65	41	A	105	69	i
66	42	B	106	6A	j
67	43	C	107	6B	k
68	44	D	108	6C	l
69	45	E	109	6D	m
70	46	F	110	6E	n
71	47	G	111	6F	o
72	48	H	112	70	p
73	49	I	113	71	q
74	4A	J	114	72	r
75	4B	K	115	73	s
76	4C	L	116	74	t
77	4D	M	117	75	u
78	4E	N	118	76	v
79	4F	O	119	77	w
80	50	P	120	78	x
81	51	Q	121	79	y
82	52	R	122	7A	z
83	53	S	123	7B	{
84	54	T	124	7C	¦ (vertical bar)
85	55	U	125	7D	}
86	56	V	126	7E	~ (tilde)
87	57	W	127	7F	DEL
88	58	X	128	80	none
89	59	Y	129	81	none
90	5A	Z	130	82	none
91	5B	[	131	83	none
92	5C	\	132	84	none
93	5D	]	133	85	none
94	5E	^	134	86	none
95	5F	— (em dash)	135	87	BEL
96	60	` (tick)	136	88	BS
97	61	a	137	89	HT
98	62	b	138	8A	LF
99	63	c	139	8B	VT
100	64	d	140	8C	FF

Dec	Hex	Character or Function
141	8D	CR
142	8E	SO
143	8F	SI
144	90	none
145	91	DC1
146	92	DC2
147	93	DC3
148	94	DC4
149	95	none
150	96	none
151	97	none
152	98	CAN
153	99	none
154	9A	none
155	9B	ESC
156	9C	none
157	9D	none
158	9E	none
159	9F	none
160	A0	b̸ (space)
161	A1	! (italic)
162	A2	"
163	A3	#
164	A4	$
165	A5	%
166	A6	&
167	A7	' (apostrophe)
168	A8	(
169	A9	)
170	AA	*
171	AB	+
172	AC	, (comma)
173	AD	– (en dash)
174	AE	. (period)
175	AF	/
176	B0	0
177	B1	1
178	B2	2
179	B3	3
180	B4	4
181	B5	5
182	B6	6
183	B7	7
184	B8	8
185	B9	9
186	BA	:
187	BB	;
188	BC	<
189	BD	=
190	BE	>
191	BF	?
192	C0	@
193	C1	A
194	C2	B
195	C3	C
196	C4	D
197	C5	E
198	C6	F
199	C7	G
200	C8	H
201	C9	I
202	CA	J
203	CB	K
204	CC	L
205	CD	M
206	CE	N
207	CF	O
208	D0	P
209	D1	Q
210	D2	R
211	D3	S
212	D4	T
213	D5	U
214	D6	V
215	D7	W
216	D8	X
217	D9	Y
218	DA	Z
219	DB	[
220	DC	\

Dec	Hex	Character or Function
221	DD	]
222	DE	^
223	DF	— (em dash)
224	E0	` (tick)
225	E1	a
226	E2	b
227	E3	c
228	E4	d
229	E5	e
230	E6	f
231	E7	g
232	E8	h
233	E9	i
234	EA	j
235	EB	k
236	EC	l
237	ED	m
238	EE	n
239	EF	o
240	F0	p
241	F1	q
242	F2	r
243	F3	s
244	F4	t
245	F5	u
246	F6	v
247	F7	w
248	F8	x
249	F9	y
250	FA	z
251	FB	{
252	FC	/ (vertical bar)
253	FD	}
254	FE	~ (tilde)
255	FF	DEL

II. Codes for Other Character Sets

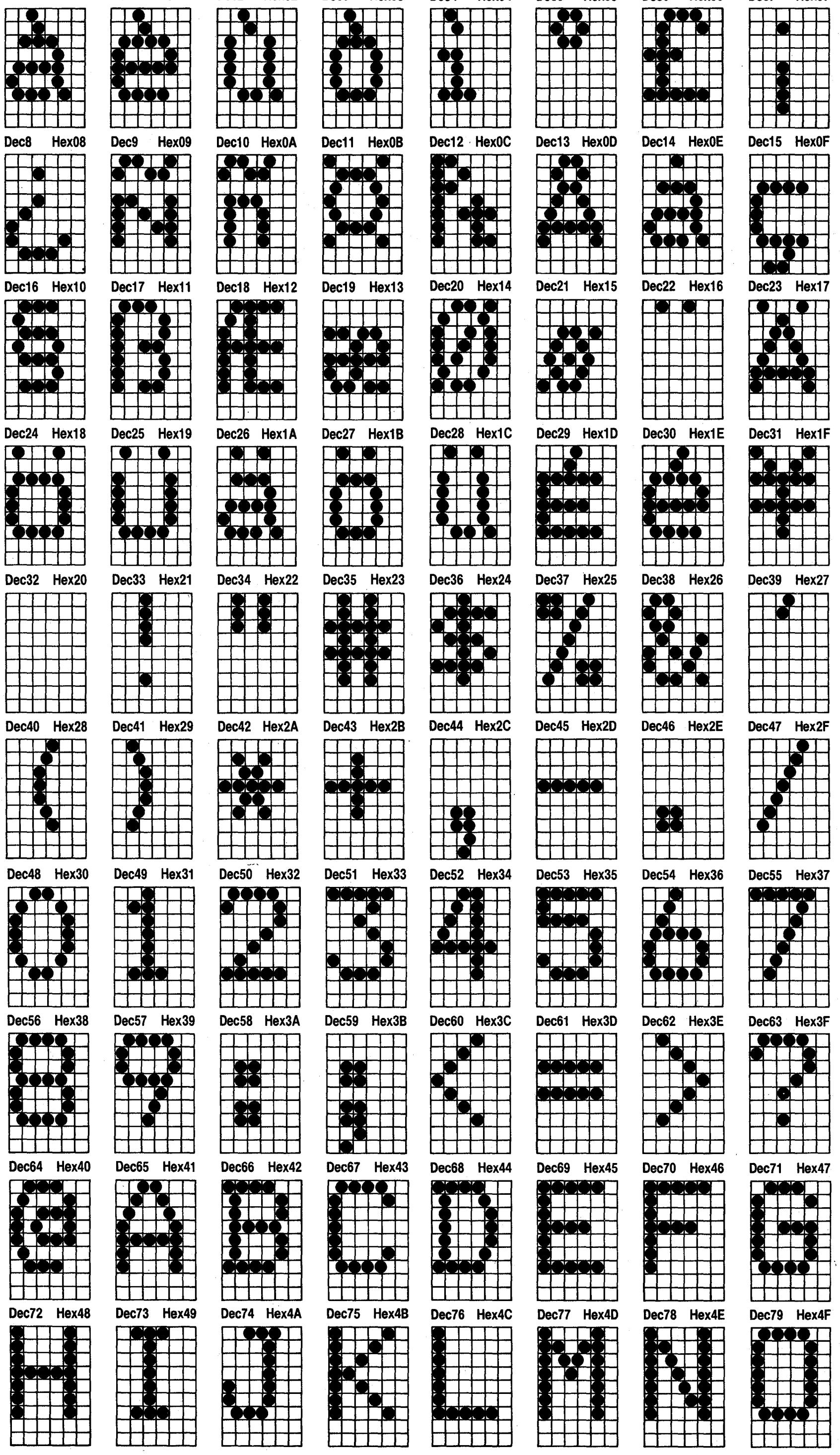
Memory locations 0 to 31 and 128 to 159 store the international character sets other than the one for the U.S. These characters are printable only with the ⟨ESC⟩"6", ⟨ESC⟩"I", or ⟨ESC⟩"R" sequences.

Dec	Hex	Character	Dec	Hex	Character
0	00	à	128	80	à
1	01	è	129	81	è
2	02	ù	130	82	ù
3	03	ò	131	83	ò
4	04	ì	132	84	ì
5	05	º	133	85	º
6	06	£	134	86	£
7	07	¡	135	87	¡
8	08	¿	136	88	¿
9	09	Ñ	137	89	Ñ
10	0A	ñ	138	8A	ñ
11	0B	¤	139	8B	¤
12	0C	₧	140	8C	₧
13	0D	Å	141	8D	Å
14	0E	å	142	8E	å
15	0F	ç	143	8F	ç
16	10	§	144	90	§
17	11	ß	145	91	ß
18	12	Æ	146	92	Æ
19	13	æ	147	93	æ
20	14	Ø	148	94	Ø
21	15	ø	149	95	ø
22	16	¨	150	96	¨
23	17	Ä	151	97	Ä
24	18	Ö	152	98	Ö
25	19	Ü	153	99	Ü
26	1A	ä	154	9A	ä
27	1B	ö	155	9B	ö
28	1C	ü	156	9C	ü
29	1D	§	157	9D	É
30	1E	é	158	9E	é
31	1F	¥	159	9F	¥

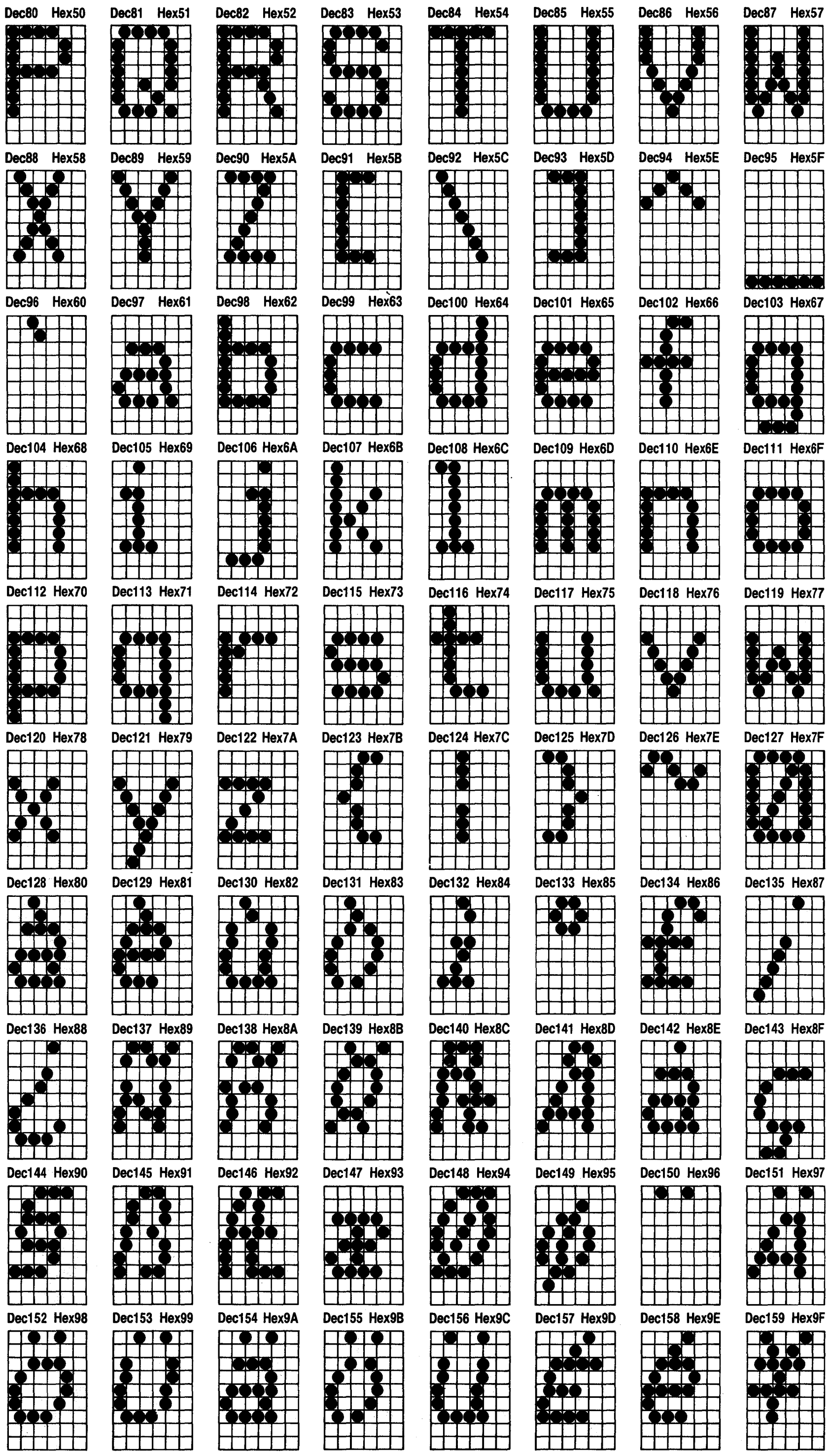
Appendix B
Character Fonts

This appendix shows the dot patterns for all alphanumeric characters from decimal 0 through 255; values are shown in decimal and also in hexadecimal (00 to FF). The dot-matrix grid is 9 rows by 6 columns.

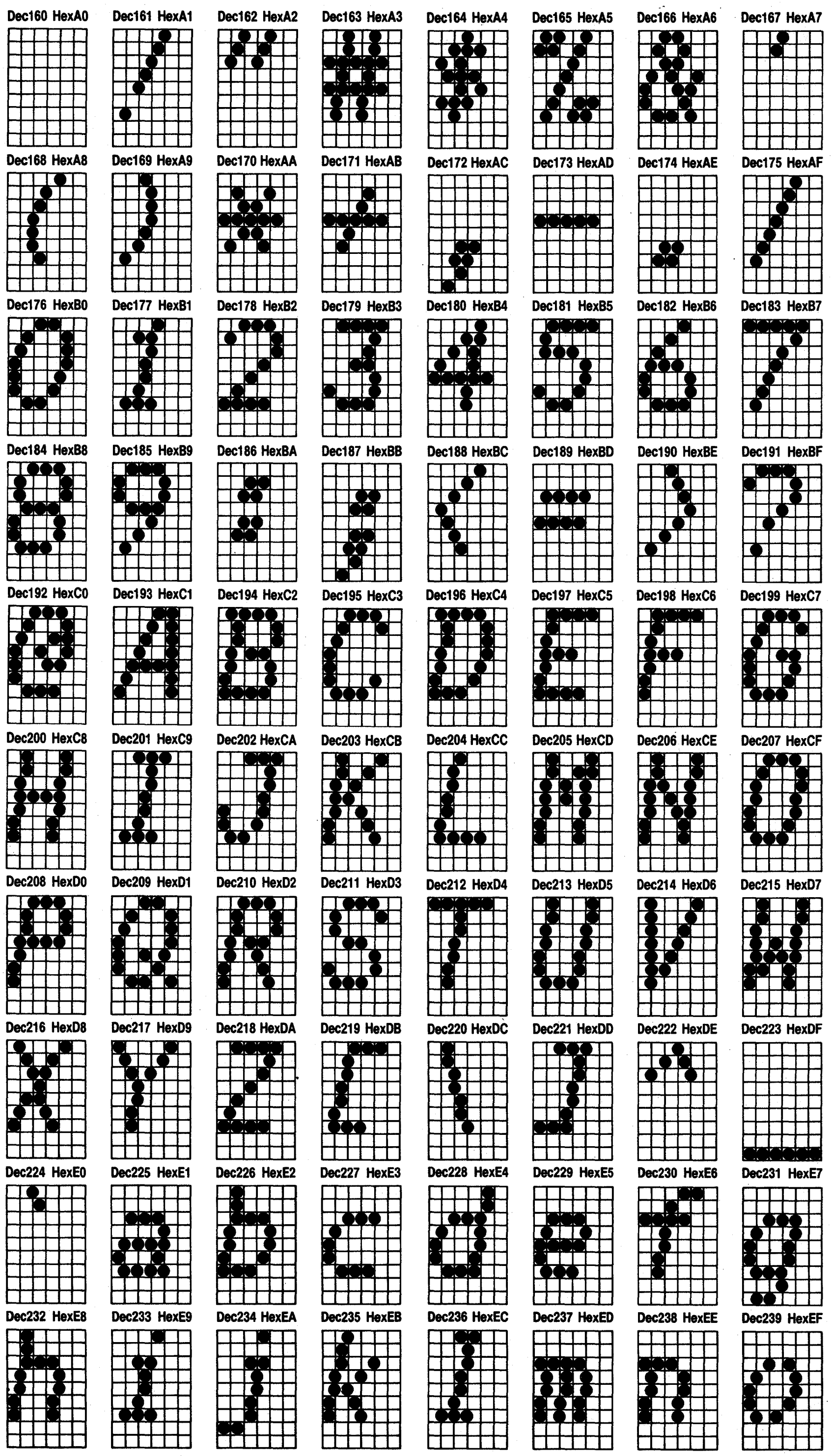
00 01 02 03 04 05 06 07

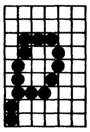


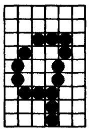
Dec0 Hex00 Dec1 Hex01 Dec2 Hex02 Dec3 Hex03 Dec4 Hex04 Dec5 Hex05 Dec6 Hex06 Dec7 Hex07
Dec8 Hex08 Dec9 Hex09 Dec10 Hex0A Dec11 Hex0B Dec12 Hex0C Dec13 Hex0D Dec14 Hex0E Dec15 Hex0F
Dec16 Hex10 Dec17 Hex11 Dec18 Hex12 Dec19 Hex13 Dec20 Hex14 Dec21 Hex15 Dec22 Hex16 Dec23 Hex17
Dec24 Hex18 Dec25 Hex19 Dec26 Hex1A Dec27 Hex1B Dec28 Hex1C Dec29 Hex1D Dec30 Hex1E Dec31 Hex1F
Dec32 Hex20 Dec33 Hex21 Dec34 Hex22 Dec35 Hex23 Dec36 Hex24 Dec37 Hex25 Dec38 Hex26 Dec39 Hex27
Dec40 Hex28 Dec41 Hex29 Dec42 Hex2A Dec43 Hex2B Dec44 Hex2C Dec45 Hex2D Dec46 Hex2E Dec47 Hex2F
Dec48 Hex30 Dec49 Hex31 Dec50 Hex32 Dec51 Hex33 Dec52 Hex34 Dec53 Hex35 Dec54 Hex36 Dec55 Hex37
Dec56 Hex38 Dec57 Hex39 Dec58 Hex3A Dec59 Hex3B Dec60 Hex3C Dec61 Hex3D Dec62 Hex3E Dec63 Hex3F
Dec64 Hex40 Dec65 Hex41 Dec66 Hex42 Dec67 Hex43 Dec68 Hex44 Dec69 Hex45 Dec70 Hex46 Dec71 Hex47
Dec72 Hex48 Dec73 Hex49 Dec74 Hex4A Dec75 Hex4B Dec76 Hex4C Dec77 Hex4D Dec78 Hex4E Dec79 Hex4F

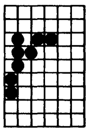

Dec80 Hex50
Dec81 Hex51
Dec82 Hex52
Dec83 Hex53
Dec84 Hex54
Dec85 Hex55
Dec86 Hex56
Dec87 Hex57
Dec88 Hex58
Dec89 Hex59
Dec90 Hex5A
Dec91 Hex5B
Dec92 Hex5C
Dec93 Hex5D
Dec94 Hex5E
Dec95 Hex5F
Dec96 Hex60
Dec97 Hex61
Dec98 Hex62
Dec99 Hex63
Dec100 Hex64
Dec101 Hex65
Dec102 Hex66
Dec103 Hex67
Dec104 Hex68
Dec105 Hex69
Dec106 Hex6A
Dec107 Hex6B
Dec108 Hex6C
Dec109 Hex6D
Dec110 Hex6E
Dec111 Hex6F
Dec112 Hex70
Dec113 Hex71
Dec114 Hex72
Dec115 Hex73
Dec116 Hex74
Dec117 Hex75
Dec118 Hex76
Dec119 Hex77
Dec120 Hex78
Dec121 Hex79
Dec122 Hex7A
Dec123 Hex7B
Dec124 Hex7C
Dec125 Hex7D
Dec126 Hex7E
Dec127 Hex7F
Dec128 Hex80
Dec129 Hex81
Dec130 Hex82
Dec131 Hex83
Dec132 Hex84
Dec133 Hex85
Dec134 Hex86
Dec135 Hex87
Dec136 Hex88
Dec137 Hex89
Dec138 Hex8A
Dec139 Hex8B
Dec140 Hex8C
Dec141 Hex8D
Dec142 Hex8E
Dec143 Hex8F
Dec144 Hex90
Dec145 Hex91
Dec146 Hex92
Dec147 Hex93
Dec148 Hex94
Dec149 Hex95
Dec150 Hex96
Dec151 Hex97
Dec152 Hex98
Dec153 Hex99
Dec154 Hex9A
Dec155 Hex9B
Dec156 Hex9C
Dec157 Hex9D
Dec158 Hex9E
Dec159 Hex9F

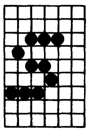
I. Alphanumeric Characters

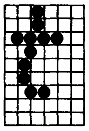


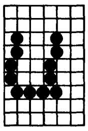

Dec240 HexF0

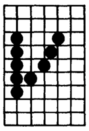

Dec241 HexF1

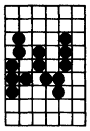

Dec242 HexF2

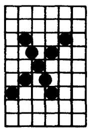

Dec243 HexF3

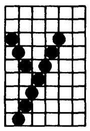

Dec244 HexF4

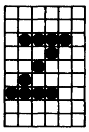

Dec245 HexF5

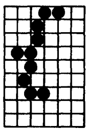

Dec246 HexF6

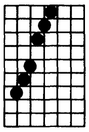

Dec247 HexF7

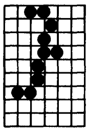

Dec248 HexF8

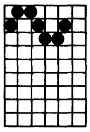

Dec249 HexF9

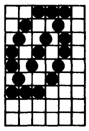

Dec250 HexFA


Dec251 HexFB


Dec252 HexFC


Dec253 HexFD


Dec254 HexFE


Dec255 HexFF

Appendix C
Proportional Character Widths

This chart shows the number of units used to print each character in Proportional Mode. A unit is the width of one of the 12 columns in a character matrix (about half a dot).

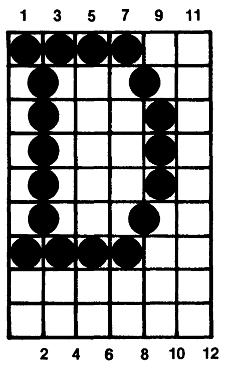


Figure C-1. Sample grid.

Proportional chart

Code	Symbol	Width
00	à	12
01	è	12
02	ù	11
03	ò	10
04	ì	6
05	°	8
06	£	12
07	¡	5
08	¿	12
09	Ñ	12
10	ñ	11
11	¤	12
12	₧	12
13	Å	12
14	å	12
15	ç	11
16	§	10
17	ß	11
18	Æ	12
19	æ	12
20	Ø	12
21	ø	12
22	¨	8
23	Ä	12
24	Ö	12
25	Ü	12
26	ä	12
27	ö	10
28	ü	11
29	§	12
30	é	12
31	¥	12
32	ƀ	12
33	!	5
34	"	8
35	#	12
36	$	12
37	%	12
38	&	12
39	'	5
40	(	6
41	)	6
42	*	12
43	+	12
44	,	7
45	-	12
46	.	6
47	/	10
48	0	12
49	1	8
50	2	12
51	3	12
52	4	12
53	5	12
54	6	12
55	7	12
56	8	12
57	9	12
58	:	6
59	;	6
60	<	10
61	=	12
62	>	10
63	?	12
64	@	12
65	A	12
66	B	12
67	C	12
68	D	12
69	E	12
70	F	12
71	G	12
72	H	12
73	I	8
74	J	11
75	K	12
76	L	12
77	M	12
78	N	12
79	O	12
80	P	12

Code	Symbol	Width	Code	Symbol	Width
81	Q	12	121	y	12
82	R	12	122	z	10
83	S	12	123	{	9
84	T	12	124	¦	5
85	U	12	125	}	9
86	V	12	126	~	12
87	W	12	127	Ø	12
88	X	10	128	à	11
89	Y	12	129	è	11
90	Z	10	130	ù	11
91	[	8	131	ò	11
92	\	10	132	ì	8
93	]	8	133	°	8
94	^	12	134	£	12
95	—	12	135	/	10
96	`	5	136	¿	11
97	a	12	137	Ñ	12
98	b	11	138	ñ	12
99	c	11	139	¤	12
100	d	11	140	Å	12
101	e	12	141	Å	12
102	f	10	142	å	11
103	g	11	143	ç	11
104	h	11	144	§	12
105	i	8	145	ß	11
106	j	9	146	Æ	12
107	k	10	147	æ	12
108	l	8	148	Ø	12
109	m	12	149	ø	11
110	n	11	150	¨	9
111	o	12	151	Ä	12
112	p	11	152	Ö	12
113	q	11	153	Ü	12
114	r	11	154	ä	11
115	s	12	155	ö	11
116	t	11	156	ü	12
117	u	12	157	É	12
118	v	12	158	é	11
119	w	12	159	¥	12
120	x	10	160	ƀ	12

Code	Symbol	Width
161	!	10
162	"	10
163	#	12
164	$	11
165	%	12
166	&	12
167	'	5
168	(	8
169	)	8
170	*	12
171	+	12
172	,	8
173	-	12
174	.	7
175	/	10
176	0	12
177	1	9
178	2	12
179	3	12
180	4	12
181	5	12
182	6	11
183	7	12
184	8	12
185	9	11
186	:	8
187	;	9
188	<	10
189	=	11
190	>	9
191	?	11
192	@	12
193	A	12
194	B	12
195	C	12
196	D	12
197	E	12
198	F	12
199	G	12
200	H	12
201	I	10
202	J	12
203	K	12
204	L	10
205	M	12
206	N	12
207	O	12
208	P	12
209	Q	12
210	R	12
211	S	12
212	T	12
213	U	12
214	V	11
215	W	12
216	X	12
217	Y	12
218	Z	12
219	[	11
220	\	7
221	]	11
222	^	10
223	—	12
224	`	5
225	a	11
226	b	11
227	c	11
228	d	12
229	e	11
230	f	12
231	g	11
232	h	11
233	i	9
234	j	10
235	k	11
236	l	9
237	m	11
238	n	10
239	o	11
240	p	11

Code	Symbol	Width
241	*q*	11
242	*r*	10
243	*s*	11
244	*t*	10
245	*u*	11
246	*v*	10
247	*w*	12
248	*x*	12
249	*y*	11
250	*z*	12
251	*{*	10
252	*/*	9
253	*}*	10
254	*~*	12
255	*Ø*	12

Appendix D
Control Codes in Numeric Order

Control codes require LPRINT CHR$(#).

Abbreviations: cpi = characters per inch
cps = characters per second

Notation: * a code in which "0" and "1" can be substituted for CHR$(0) and CHR$(1). For example, either
⟨ESC⟩"p"CHR$(1) or ⟨ESC⟩"p1"
turns Proportional Mode ON.

Dec	Hex	Symbol	Function
0	00	NUL	Used as a terminator for horizontal and vertical TABs.
7	07	BEL	Buzzes for .1 second.
8	08	BS	Backspace. Empties the printer buffer, then backspaces print head 1 space in the current pitch.
9	09	HT	Horizontal Tab. Empties the printer buffer, then the print head moves to the next tab stop.
10	0A	LF	Line Feed. Printer empties its buffer, performs a line feed at the current line spacing, and resets its buffer character count to 0.
11	0B	VT	Vertical Tab. Empties the printer buffer, then feeds the paper to the next vertical tab stop.
12	0C	FF	Form Feed. Empties the printer buffer, then advances the paper to the next logical Top Of Form (TOF).

	Dec	Hex	Symbol	Function
	13	0D	CR	Carriage Return. Prints the contents of the buffer and resets the buffer character count to 0. Restores the print head to the left margin. Automatic line feed can be added with switch 2-4.
	14	0E	SO	Shift Out. Turns Expanded Mode ON for the length of the line (unless cancelled by DC4 (20) or ‹ESC›"W0"). Works with all 3 pitches.
	15	0F	SI	Shift In. Empties the buffer and turns Compressed Mode (17.16 cpi) ON. Cannot work with Emphasized, Elite, Pica, or Proportional Modes. Stays on until cancelled by DC2 (18).
	17	11	DC1	Device Control 1. Places the printer in the active state: printer receives all data sent to it. This is the default condition.
	18	12	DC2	Device Control 2. Turns Compressed Mode OFF.
	19	13	DC3	Device Control 3. Places the printer in the inactive state until a DC1 code is received.
	20	14	DC4	Device Control 4. Turns the Expanded Mode set by CHR$(14) (Shift Out) OFF.
	24	18	CAN	Cancel. Cancels all text in the print buffer.
	27	1B	ESC	Escape. Prepares the printer to receive control codes.
	127	7F	DEL	Delete. Deletes the last text character in the printer buffer.
ESC	14	0E	SO	Same as Shift Out (CHR$(14)). Turns Expanded Mode ON for the length of 1 (only) print line.
ESC	15	0F	SI	Same as Shift In (CHR$(15)). Turns Compressed Mode ON.

	Dec	Hex	Symbol	Function
ESC	33	21	!	Master Print Mode Select (Master Select). Selects 16 unique print mode combinations. Format: 〈ESC〉"!"CHR$(n) where 0 〈 = n 〈 = 255. See the charts in Appendix F.
ESC	35	23	#	Accepts the 8th bit "as is" from the computer.
ESC	37	25	%	Activates a character set. Switch 1-4 must be off. Format: 〈ESC〉"%"CHR$(0)CHR$(0) selects the ROM set; and 〈ESC〉"%"CHR$(1)CHR$(0) selects the RAM set.
ESC	38	26	&	Defines characters in user RAM. Format: 〈ESC〉"&"CHR$(0)CHR$(c1)CHR$(c2); CHR$(a)CHR$(d1) . . . CHR$(d11); where CHR$(0) is for future use, c1 is the starting character, and c2 is the ending character. Each character in the range c1 - c2 requires an attribute byte (a) and 11 data bytes (d1 - d11).
ESC	42	2A	*	Turns Graphics Mode ON. Format: 〈ESC〉"*"CHR$(m)CHR$(n1)CHR$(n2); followed by n data numbers, where n = n1 + 256*n2, 0 〈 = n1 〈 = 255, 0 〈 = n2 〈 = 255, and m selects mode 0 - 6. See Table 13-1 for modes.
*ESC	45	2D	–	Turns Underline Mode ON. Format: 〈ESC〉" – "CHR$(n) where n = 0 turns underline OFF, n = 1 turns underline ON.
ESC	47	2F	/	Selects a vertical tab channel. Format: 〈ESC〉"/"CHR$(n) where 0 〈 = n 〈 = 7.
ESC	48	30	0	Sets line spacing to 1/8 inch (9 dots).
ESC	49	31	1	Sets line spacing to 7/72 inch (7 dots).

	Dec	Hex	Symbol	Function
ESC	50	32	2	Returns line spacing to the default of 1/6 inch (12 dots).
ESC	51	33	3	Sets line spacing to n/216 inch (1/216th inch is about 1/3rd of a dot). Stays on until changed. Format: 〈ESC〉"3"CHR$(n) where 0 〈 = n 〈 = 255.
ESC	52	34	4	Turns the italic character set ON.
ESC	53	35	5	Turns the italic character set OFF.
ESC	54	36	6	Enables the printing of the symbols which are stored in locations 128 - 159.
ESC	55	37	7	Causes codes 128 - 159 to print as control codes.
ESC	56	38	8	Disables the "paper out" sensor.
ESC	57	39	9	Enables the "paper out" sensor.
ESC	58	3A	:	Copies character set from ROM to RAM. Format: 〈ESC〉":"CHR$(n1)CHR$(n2)CHR$(n3); where n1, n2, and n3 are all 0. (They are included for future expansion.) The RAM character set must be activated with 〈ESC〉 " % ", and switch 1-4 must be off.
ESC	60	3C	〈	Turns Unidirectional Mode ON for 1 (only) line; prints from left to right.
ESC	61	3D	=	Sets the 8th bit to 0 (limits the range to 0 - 127).
ESC	62	3E	〉	Sets the 8th bit to 1 (limits the range to 128 - 255).
ESC	64	40	@	Master Reset Code, which resets all special modes to their power-up states, including Top Of Form. Clears all text and control codes from the print buffer.
ESC	65	41	A	Sets line spacing to n/72 inch (n dots). Format: 〈ESC〉"A"CHR$(n) where 0 〈 = n 〈 = 85.

	Dec	Hex	Symbol	Function
ESC	66	42	B	Sets up to 16 vertical tabs in the current line spacing. Tab settings are not affected by subsequent changes in line spacing. Format: 〈ESC〉"B"CHR$(n1)CHR$(n2) . . . CHR$(nk)CHR$(0) where 1 〈 = nk 〈 = 254. Terminate this tab sequence with CHR$(0) or a number less than that of the last tab (nk).
ESC	67	43	C	Sets the form length to n *lines* in the current line spacing. The default is 66 lines. Also resets Top Of Form. Format: 〈ESC〉"C"CHR$(n) where 1 〈 = n 〈 = 127.
ESC	67	43	C	Sets the form length to n *inches,* regardless of the current line spacing. The default is 11 inches. Also resets Top Of Form. Format: 〈ESC〉"C"CHR$(0)CHR$(n) where 1 〈 = n 〈 = 22.
ESC	68	44	D	Resets the current tabs and sets up to 32 horizontal tabs in the current pitch. Tabs may range up to the maximum width for the character and printer size. For example, the maximum tab for Pica characters on an 8-inch line is 79. Tab settings are not affected by subsequent changes in pitch. Format: 〈ESC〉"D"CHR$(n1)CHR$(n2) . . . CHR$(nk)CHR$(0) Terminate a tab sequence with CHR$(0) or a number less than that of the last tab (nk).
ESC	69	45	E	Turns Emphasized Mode ON. Cannot mix with Elite, Proportional, or Compressed Modes.
ESC	70	46	F	Turns Emphasized Mode OFF.
ESC	71	47	G	Turns Double-Strike Mode ON. Cannot mix with Proportional Mode.
ESC	72	48	H	Turns Double-Strike Mode OFF.

	Dec	Hex	Symbol	Function
*ESC	73	49	I	Enables printing of the characters that are stored in memory locations 0 - 31 and that are not used for control codes. Symbols stored in control-code locations must be printed with ⟨ESC⟩"R". Format: ⟨ESC⟩"I"CHR$(n) where n = 1 =⟩ prints characters, n = 0 =⟩ prints control codes.
ESC	74	4A	J	Forces an immediate line feed of n/216 inches without changing the current line spacing. Prints the contents of the buffer without a carriage return. Format: ⟨ESC⟩"J"CHR$(n) where 0 ⟨= n ⟨= 255.
ESC	75	4B	K	Normal Graphics Mode. Prints dot graphics, 480 dots per 8-inch line. Format: ⟨ESC⟩"K"CHR$(n1)CHR$(n2); followed by n data numbers, where n = n1 + 256*n2 0 ⟨= n1 ⟨= 255 0 ⟨= n2 ⟨= 255 For 480 dots, n1 = 224, n2 = 1.
ESC	76	4C	L	Turns Low-Speed Double-Density Graphics ON. Prints dot graphics, 960 dots per 8-inch line. Format: ⟨ESC⟩"L"CHR$(n1)CHR$(n2); followed by n data numbers, where n = n1 + 256*n2 0 ⟨= n1 ⟨= 255 0 ⟨= n2 ⟨= 255. For 960 dots, n1 = 192, n2 = 3.
ESC	77	4D	M	Turns Elite Mode (12 cpi) ON. Cannot mix with Pica, Proportional, Emphasized, or Compressed Modes.
ESC	78	4E	N	Sets skip-over-perforation to n lines. Format: ⟨ESC⟩"N"CHR$(n) where 1 ⟨= n ⟨= 127.
ESC	79	4F	O	Turns skip-over-perforation OFF.

	Dec	Hex	Symbol	Function
ESC	80	5A	P	Turns Elite Mode OFF. Returns to Pica unless Compressed Mode is active.
ESC	81	51	Q	Sets the column width. Also cancels all text that is in the print buffer. Format: 〈ESC〉"Q"CHR$(n) where 1 〈 = n 〈 = Maximum number of characters per line in the current pitch.
ESC	82	52	R	Selects an international character set by its country's number. See Tables 6-2 and 6-3. Format: 〈ESC〉"R"CHR$(n) where 0 〈 = n 〈 = 8.
*ESC	83	53	S	Turns Script Mode ON. Either type of Script is printed in Double-Strike; it cannot mix with Proportional Mode. Format: 〈ESC〉"S"CHR$(n) where n = 0 =〉 Superscript, n = 1 =〉 Subscript.
ESC	84	54	T	Turns Script Mode OFF.
*ESC	85	55	U	Turns Unidirectional Mode ON. Prints each line from left to right. Format: 〈ESC〉"U"CHR$(n) where n = 0 =〉 OFF, n=1 =〉 ON.
ESC	87	57	W	Turns Expanded Mode ON; stays ON until turned OFF. Cannot be turned off with DC4 (CHR$(20)). Format: 〈ESC〉"W"CHR$(n) where n = 0 =〉 OFF, n = 1 =〉 ON.
ESC	89	59	Y	Turns High-Speed Double-Density Graphics ON; gives the same density as 〈ESC〉"L", but cannot print 2 adjacent dots in the same row.

	Dec	Hex	Symbol	Function
ESC	90	5A	Z	Turns Quadruple-Density Graphics Mode ON. Prints dot graphics, 1920 dots per 8-inch line. Format: ⟨ESC⟩"Z"CHR$(n1)CHR$(n2); followed by n data numbers, where n = n1 + 256*n2, 0 ⟨= n1 ⟨= 255, 0 ⟨= n2 ⟨= 255. For 1920 dots, n1 = 128, n2 = 7.
ESC	94	5E	^	Turns Nine-Pin Graphics ON. Format: ⟨ESC⟩"^"CHR$(d)CHR$(n1)CHR$(n2); followed by 2 times n data numbers where n = n1 + 256*n2, 0 ⟨= n1 ⟨= 255, 0 ⟨= n2 ⟨= 255. The printer expects 2 data numbers for each column of print. The d selects the density, where d = 0 =⟩ single density, d = 1 =⟩ double density.
ESC	98	62	b	Sets the vertical tab for channel n. Format: ⟨ESC⟩"b"CHR$(n); where 0 ⟨= n ⟨= 7, and n = 0 is the same as ⟨ESC⟩"B".
*ESC	105	69	i	Turns Immediate Mode ON. Prints each character immediately as it is received by the printer. Format: ⟨ESC⟩"i"CHR$(n) where n = 0 =⟩ OFF, n = 1 =⟩ ON.
ESC	106	6A	j	Causes immediate reverse line feed in increments of 1/216" without a carriage return. Similar to ⟨ESC⟩"J". Format: ⟨ESC⟩"j"CHR$(n) where 0 ⟨= n ⟨= 255.

ESC	108	6C	l	Sets the left margin. Also cancels all text that is in the print buffer. Format: 〈ESC〉"l"CHR$(n) where n ranges from 0 - 78 in Pica, 0 - 93 in Elite, and 0 - 133 in Compressed
*ESC	112	70	p	Turns Proportional Mode ON. Cannot mix with Elite, Emphasized, Compressed, Script, or Double-Strike Modes. Format: 〈ESC〉"p"CHR$(n) where n = 0 =〉 OFF, n = 1 =〉 ON.
ESC	115	73	s	Turns Half-Speed (80 cps) ON. Used to reduce noise. Format: 〈ESC〉"s"CHR$(n) where n = 1 =〉 80 cps, n = 0 =〉 160 cps.

Codes from 128 - 255 simply mirror their counterparts in the range 0 - 127. Simple control codes are listed below for ready reference.

135	87	BEL	Bell.
136	88	BS	Backspace.
137	89	HT	Horizontal Tab.
138	8A	LF	Line Feed.
139	8B	VT	Vertical Tab.
140	8C	FF	Form Feed.
141	8D	CR	Carriage Return.
142	8E	SO	Shift Out (Expanded Mode).
143	8F	SI	Shift In (Compressed Mode).
145	91	DC1	Device Control 1. Printer selected.
146	92	DC2	Device Control 2. Compressed Mode OFF.
147	93	DC3	Device Control 3. Printer deselected.
148	94	DC4	Device Control 4. Expanded Mode (CHR$(14) only) OFF.
152	98	CAN	Cancel.
155	9B	ESC	Escape.
255	FF	DEL	Delete.

ESC = CHR(27)
0 = 48
1 = [illegible]
2 = [illegible]

Appendix E
Control Codes by Function

Print Width Commands

〈ESC〉"M"	Turns Elite Mode ON.
〈ESC〉"P"	Turns Elite Mode OFF.
CHR$(15)	Turns Compressed Mode ON.
〈ESC〉 CHR$(15)	Same as CHR$(15).
CHR$(18)	Turns Compressed Mode OFF.
CHR$(14)	Turns One-Line Expanded Mode ON.
〈ESC〉 CHR$(14)	Same as CHR$(14).
CHR$(20)	Turns One-Line Expanded Mode OFF.
〈ESC〉"W1"	Turns Continuous Expanded Mode ON.
〈ESC〉"W0"	Turns Expanded Mode OFF.

Print Quality Commands

〈ESC〉"E"	Turns Emphasized Mode ON.
〈ESC〉"F"	Turns Emphasized Mode OFF.
〈ESC〉"G"	Turns Double-Strike Mode ON.
〈ESC〉"H"	Turns Double-Strike Mode OFF.
〈ESC〉"S1"	Turns Superscript Mode ON.
〈ESC〉"S0"	Turns Subscript Mode ON.
〈ESC〉"T"	Turns either Script Mode OFF.
〈ESC〉"p1"	Turns Proportional Mode ON.*
〈ESC〉"p0"	Turns Proportional Mode OFF.*
〈ESC〉"-1"	Turns Underline Mode ON.
〈ESC〉"-0"	Turns Underline Mode OFF.
〈ESC〉"4"	Turns italic character set ON.
〈ESC〉"5"	Turns italic character set OFF.

*If your computer cannot generate lower-case letters, use the equivalent decimal values.

Selecting Print Modes

〈ESC〉"!" CHR$(n)	Master Print Mode Select (Master Select): n = 0 to 255. See Appendix F for mode combinations.
〈ESC〉"@"	Master Reset Code.

Special Printer Features

CHR$(7)	Sounds the bell for .1 second.
CHR$(8)	Backspaces.
CHR$(17)	Enables the printer to receive data (default).
CHR$(19)	Disables the printer from receiving data.
CHR$(24)	Cancels the text in the print buffer.
CHR$(127)	Deletes the most recent text character in the print buffer.
〈ESC〉"〉"	Sets the high-order bit ON.
〈ESC〉"="	Sets the high-order bit OFF.
〈ESC〉"#"	Accepts the 8th bit "as is" from the computer.
〈ESC〉"R" CHR$(n)	Selects the international character set n. See Tables 6-2 and 6-3.
〈ESC〉"i1"	Turns Immediate Mode ON.*
〈ESC〉"i0"	Turns Immediate Mode OFF.*
〈ESC〉"s1"	Turns Half-Speed Mode ON.*
〈ESC〉"s0"	Returns to normal speed.*

Paper Feed Commands

CHR$(10)	Produces a line feed.
〈ESC〉"0"	Sets line spacing to 1/8 inch.
〈ESC〉"1"	Sets line spacing to 7/72 inch.
〈ESC〉"2"	Sets line spacing to 1/6 inch (default).
〈ESC〉"A"CHR$(n)	Sets line spacing to n/72 inch.
〈ESC〉"3"CHR$(n)	Sets line spacing to n/216 inch.
〈ESC〉"J"CHR$(n)	Produces an immediate one-time line feed of n/216 inch without a carriage return.
〈ESC〉"j"CHR$(n)	Produces an immediate one-time reverse feed of n/216 inch without a carriage return.*

Forms Control Commands

CHR$(12)	Produces a form feed.
CHR$(13)	Produces a carriage return.

*If your computer cannot generate lower-case letters, use the equivalent decimal values.

⟨ESC⟩"8"	Turns the paper-out sensor OFF.
⟨ESC⟩"9"	Turns the paper-out sensor ON.
⟨ESC⟩"C"CHR$(n)	Sets the form length in lines: n = 0 - 127.
⟨ESC⟩"C"CHR$(0)CHR$(n)	Sets the form length in inches: n = 0 - 22.
⟨ESC⟩"N"CHR$(n)	Produces a variable skip-over-perforation. n = 0 - form length or 127.
⟨ESC⟩"O"	Turns skip-over-perforation OFF.

Formatting Commands

TAB(n)	BASIC tab to column n. Varies with the system.
⟨ESC⟩"D"CHR$(n1) . . . CHR$(nk)CHR$(0)	Sets the horizontal tab stops at n1, n2, . . ., nk, where k ⟨ = 32, 0 ⟨ nk ⟨ margin length.
CHR$(9)	Activates a horizontal tab.
⟨ESC⟩"B"CHR$(n1). . .CHR$(nk)CHR$(0)	Sets the vertical tab stops at n1 - nk, where k ⟨ = 16, 0 ⟨ nk ⟨ form length.
⟨ESC⟩"b"CHR$(n)CHR$(n1)CHR$(n2) . . . CHR$(nk)CHR$(0)	Stores vertical tab stops in channel n, where 0 ⟨ = n ⟨ = 7. Channel 0 is the same as ⟨ESC⟩ "B".*
⟨ESC⟩"/"CHR$(n)	Selects channel n.
CHR$(11)	Activates a vertical tab.
⟨ESC⟩"Q"CHR$(n)	Sets the right margin at n, where n ranges from 2 - 80 in Pica, 3 - 96 in Elite, and 4 - 137 in Compressed.
⟨ESC⟩"l"CHR$(n)	Sets the left margin at n, where n ranges from 0 - 78 in Pica, 0 - 93 in Elite, and 0 - 133 in Compressed.*
⟨ESC⟩"U1"	Turns Continuous Unidirectional Mode ON.
⟨ESC⟩"U0"	Turns Continuous Unidirectional Mode OFF.
⟨ESC⟩"⟨"	Turns One-Line Unidirectional Mode ON.

*If your computer cannot generate lower-case letters, use the equivalent decimal values.

Graphics Modes

CHR$(2^X)	Fires pin X when sent as graphics data. 0 <= X <= 7.
<ESC>"K"CHR$(n1)CHR$(n2);	Turns Single-Density Graphics Mode ON; width = n1 + 256*n2
<ESC>"L"CHR$(n1)CHR$(n2);	Turns Double-Density Graphics Mode ON.
<ESC>"Y"CHR$(n1)CHR$(n2);	Turns High-Speed Double-Density Graphics ON.
<ESC>"Z"CHR$(n1)CHR$(n2);	Turns Quadruple-Density Graphics Mode ON.
<ESC>"*"CHR$(n)CHR$(n1)CHR$(n2);	Selects Graphics Density n, where:

0 = 480 dots per line	4 = 640 dpl
1 = 960 dpl (same as <ESC>"L")	5 = 576 dpl (1-to-1 ratio)
2 = 960 dpl (same as <ESC>"Y")	6 = 720 dpl
3 = 1920 dpl (same as <ESC>"Z")	

<ESC>"^"CHR$(0)CHR$(n1)CHR$(n2);	Turns Single-Density Nine-Pin Graphics Mode ON.
<ESC>"^"CHR$(1)CHR$(n1)CHR$(n2);	Turns Double-Density Nine-Pin Graphics Mode ON.
<ESC>"%"CHR$(n1)CHR$(n2)	Selects a character set: n1 selects ROM (0) or RAM (1); n2 is 0.
<ESC>"&"CHR$(n)CHR$(c1)CHR$(c2)CHR$(A)CHR$(d1) . . . CHR$(d11). . .	Defines characters c1 to c2 in RAM area; n is 0. Each character requires an attribute byte (A), followed by 11 data numbers (d1 to d11).
<ESC>":"CHR$(n1)CHR$(n2)CHR$(n3)	Copies ROM characters to the user RAM area. All numbers must be 0.
<ESC>"I1"	Enables printing of the symbols that are stored in locations 0 - 31 and that are not used as control codes.
<ESC>"I0"	Disables printing of codes 0 - 31 as characters.
<ESC>"6"	Enables printing of characters stored in locations 128 - 159.
<ESC>"7"	Causes codes 128 - 159 to print as control codes.

Appendix F
Mixing Print Modes

One of the most pleasing aspects of the FX-80 is its wide variety of print modes. By mixing modes as shown in Table F-1, you can print characters in 128 different type styles. The details are in Chapter 5; a short summary is provided here.

Table F-1. Arriving at 128 Type Styles.

4 type modes	2 strike modes	3 combining modes	2 script modes (Double-Strike only)
Elite or Emphasized Pica or Compressed or Pica	Single-Strike or Double-Strike	Expanded (E) and/or Italics (I) and/or Underline (U)	Superscript or Subscript
4 pitches =4	x 2 strikes =8	x 8 combinations E,I,U,EI,EU,IU, EIU (all) or none =64	+ 2 Script Modes x 32 Double-Strike combinations =64 64 + 64 = 128

Methods for Mixing

The Master Print Mode Select code can be used to select 16 unique print modes. The format is

```
<ESC>"!"CHR$(n)
```

where n ranges from 0 to 255.

Table F-2 shows the mode combinations selected by different values of n.

Table F-2. Master Select and the 256 ASCII Codes.

	WEIGHT			
PITCH	Single-Strike	Emphasized	Double-Strike	Double-Strike Emphasized
Pica (10 cpi)	0 2 64 66 128 130 192 194	8 10 12 14 72 74 76 78 136 138 140 142 200 204 206 212	16 18 80 82 144 146 208 210	24 26 28 30 88 90 92 94 152 154 156 158 216 218 220 222
Elite (12 cpi)	1 3 5 8 9 11 13 15 65 67 69 71 73 75 77 79 129 131 133 135 137 139 141 143 193 195 197 199 201 203 205 207	Elite takes precedence over Emphasized.	17 19 21 23 25 27 29 31 81 83 85 87 89 91 93 95 145 147 149 151 153 155 157 159 209 211 213 215 217 219 221 223	Elite takes precedence over Emphasized.
Com-pres-sed (17.16 cpi)	4 6 68 70 132 134 196 198	Emphasized takes precedence over Compressed.	20 22 84 86 148 150 212 214	Emphasized takes precedence over Compressed.
Ex-panded (5 cpi)	32 34 96 98 160 162 224 226	40 42 44 46 104 108 110 166 168 172 174 232 234 236 238	48 50 112 114 176 178 240 242	56 58 60 62 120 122 124 126 136 137 188 190 248 250 252 254
Ex-panded Elite (6 cpi)	33 35 37 39 41 43 45 47 97 99 101 103 105 107 109 111 161 163 165 167 169 171 173 175 225 227 229 231 233 235 237 239	Elite takes precedence over Emphasized.	49 51 53 55 57 59 61 63 113 115 117 119 121 123 125 127 177 179 181 183 185 187 189 191 241 243 245 247 249 251 253 255	Elite takes precedence over Emphasized.
Ex-panded Com-pressed (8.58 cpi)	56 38 100 102 164 166 228 234	Emphasized takes precedence over Compressed.	52 54 116 118 180 182 244 246	Emphasized takes precedence over Compressed.

By using the character form of some of the numbers, you can shorten the command string. Table F-3 shows some convenient characters to be used for the various combinations.

Table F-3. Master Select Quick Reference Chart

PITCH	WEIGHT							
	Single-Strike		Emphasized		Double-Strike		Double-Strike Emphasized	
	Num.	ASCII	Num.	ASCII	Num.	ASCII	Num.	ASCII
Pica	0	@	8	H	16	P	24	X
Elite	1	A	←		17	Q	←	
Compressed	4	D	↑		20	T	↑	
Expanded Pica	32	b̸	40	*	48	0	56	8
Expanded Elite	33	!	←		49	1	←	
Expanded Compressed	36	$	↑		52	4	↑	

Note: Arrows point to the modes which take precedence.

The rest of the 128 unique mode combinations can be achieved by using one of the selections from Table F-3 with the code for one or more of the following modes:

Superscript	Italic
Subscript	Underline

Only two constraints must be observed when combining these four modes with the others: 1) the two script modes cannot be used at the same time; 2) the script modes are always printed in Double-Strike. Other than that, anything goes. The upshot is that 128 unique modes are available on your FX-80 printer! See the print sample at the beginning of the manual, which shows the full array.

Mode Conflicts and Priorities

To better understand the way the FX-80 print modes work, consider that each mode except Pica (Pica is the default) has a separate switch that can be turned on and off via software. Once the switch is on, it stays on until turned off. When two modes that conflict are turned on at the same time, the printer must choose which one to use.

For example, suppose you turn on both Elite and Compressed Pitches. Since the printer can only print one pitch at a time, it must make a choice; in this case, the printer chooses Elite. The Compressed switch, however, is still on even though it doesn't show on your printout. When the Elite switch is turned off, the Compressed switch will take charge—unless some other mode that has priority over Compressed Mode is active.

Table F-4 shows the internal priority list maintained by the printer.

Table F-4

$$\begin{tikzcd}
\text{Elite} \arrow[d] \\
\text{Proportional (Emphasized)} \arrow[d] \\
\text{Emphasized} \arrow[d] \\
\text{Compressed}
\end{tikzcd}$$

$$\begin{tikzcd}
 & \text{Proportional} \arrow[dl] \arrow[dr] & \\
\text{Script} & & \text{Double-Strike}
\end{tikzcd}$$

Note: Elite Mode takes precedence over the modes beneath it. For example, if both Elite and Compressed are "on," the printing will be in Elite. If Elite is turned off at this point, the printing will change to Compressed.

The first chain shows that:

Elite has priority over Proportional, Emphasized, and Compressed.

Proportional has priority over Emphasized and Compressed.

Emphasized has priority over Compressed. (Yes, everyone seems to win out over Compressed.)

The second chain shows that Proportional also has priority over Script and Double-Strike Modes.

Summary Notes:

(1) Pica is the default pitch when Elite and Compressed are turned off.

(2) When two modes conflict, the one of lesser priority is masked (not cancelled). For example, Compressed and Emphasized cannot be printed at the same time. The chart shows that when both are active, printing is in Emphasized. When Emphasized is cancelled, printing is in Compressed.

(3) Italic, Underline, and Expanded Modes combine with all the above print modes.

(4) Proportional characters are always printed in Emphasized.

(5) Script characters are always printed in Double-Strike.

Appendix G Summary of Differences between the FX-80 and the MX Graftrax Plus

Features New to the FX-80

Elite Pitch

The FX-80 sports a 12 character-per-inch (cpi) Elite pitch which matches the Elite font used on many typewriters.

Reverse Line Feed

⟨ESC⟩"j" activates a reverse line feed of n/216 inch at the location of the print head when the code is received. Neither this command nor the ⟨ESC⟩"J" (forward line feed) returns the carriage to the left; each moves only in the current column.

Proportional Spacing

Proportional spacing on the FX-80 strips off the excess space between Pica characters for letter-quality printing. Proportional characters are printed in Emphasized Mode.

Selecting the Printer

With Switch 2-1 off, you can "select" or turn on printing with CHR$(17) (DC1) and "deselect" or turn off printing with CHR$(19) (DC3). When the printer is in deselected mode, all output is ignored.

User-Definable Characters

The FX-80 allows you to design your own characters and store them in a special area of RAM. These characters can be used in lieu of or in combination with the standard FX-80 character set. (There is a provision for copying the standard ROM characters into RAM so that you can redefine only a few characters if you choose. This feature is ideal for creating special character fonts or unique symbols.)

The designs are limited to the standard character matrix of 9 by 11 dots. Characters can only be 8 dots tall, but they can be printed with either the top or the bottom 8 pins of the print head. The width of each character consists of 6 regular dot positions plus 5 intermediate positions (for a total of 11).

To allow space between the characters, each character design should leave the last two columns blank.

User-defined characters can be joined together with no gaps between them if they are printed in Proportional Mode.

All of the ASCII codes (0 - 255) can be redefined, but the control codes (0 - 31 and 128 - 159) require special commands to be printed as normal characters.

Defining your own characters is covered in Chapter 17.

Vertical Tabs

You can set up to 16 vertical tab stops on the FX-80. In addition, up to 8 sets (or channels) of vertical tabs can be stored in memory at one time. The active channel can be selected by number with a single command.

Hex Dump

The FX-80 has a fantastic program debugging tool known as the hex dump mode. Simply hold down the FF button and turn the printer on. The printer is now in hex dump mode. In this mode, all codes sent to the printer are echoed onto the paper in their hexadecimal format (e.g., 5B, 2F, AA, AF). This lets you see exactly what the printer is receiving from the computer. Turn the printer off to return to normal printing.

Immediate Printing

The FX-80 can be used as a typewriter with real-time printing, character by character.

New Dot Graphics Modes

The FX-80 has more graphics densities than the MX:

72 dots per inch
80 dpi
90 dpi
120 dpi at high speed
240 dpi

The 72 dpi is a one-to-one ratio of horizontal dots to vertical dots. This can be used to correct distortions caused by the aspect ratio of CRT screens. The 80 and 90 dpi modes are designed for the CRT ratios of several popular computers, including the Epson QX-10, the DEC, the Corvus Concept, and the NEC.

There is also a Nine-Pin Graphics Mode. This is especially useful for speeding up screen dumps; it is covered in Chapter 13.

Left Margin

You can set the left margin: from 0 - 78 in Pica, 0 - 93 in Elite, and 0 - 133 in Compressed.

Master Print Mode Select

A Master Select function contains several combined type styles for easy access. There are 16 unique Master Print Modes available with a simple 〈ESC〉"!" CHR$(n) sequence. These can also be combined with such other modes as Underline and Italic. See Chapter 5 for details.

FX-80 Enhanced Versions of Graftrax Plus Features

Software

The FX-80 prints at 160 characters per second, compared to 80 cps on the MX printers. For quiet operation, this can be reduced to half speed with the 〈ESC〉"s" command.

FX characters are printed in a 9 x 11 matrix (instead of 9 x 7) to produce a better quality print.

An underlined space can be placed at the beginning or end of a line. Formerly, if no characters were printed on either side of the underlined space, the print head would pass over it.

〈ESC〉"3"CHR$(0) and 〈ESC〉"A"CHR$(0) set the line feed to 0 until reset. It is no longer necessary to change an internal switch to supress a line feed, or to ground wire 14 of the cable in the parallel port.

Codes with ON/OFF options (such as Underline, Unidirectional, Super/Subscript, Expanded, Proportional, and Half-Speed Modes) acknowledge the printed 1 and 0 (ASCII 48 and 49) as well as ASCII 0 and 1. (Note that Proportional and Half-Speed Modes are new to the FX-80.) Thus ⟨ESC⟩"s0" is the same as ⟨ESC⟩"s"CHR$(0).

Horizontal and vertical tab setting sequences can be terminated with any value less than or equal to the last designated tab stop.

Both horizontal and vertical tab stops are set in the current pitch and line spacing. Once set, their positions on the page do not change regardless of subsequent changes in the line spacing or pitch.

Note that the default tab settings do change from single width to Expanded print. See Chapter 9.

Backspace works in all 6 pitches. (In Graftrax Plus, it only went back one Pica position when the printer was in Expanded Mode.)

There are 128 unique typestyles available in the FX-80. Different combinations of Elite, Expanded, Pica, Double- Strike, Emphasized, Script, etc. . . . are covered in Chapter 5. (Note that the Elite pitch is new to the FX-80.)

Superscript and Subscript, when canceled with ⟨ESC⟩"T", return to the previous mode—either Single- or Double-Strike. (In Graftrax Plus they returned only to Double-Strike.) Also, ⟨ESC⟩"H" no longer cancels Superscript and Subscript.

Script characters can also be printed in Expanded and Emphasized modes on the FX-80, which was not possible on the MX printers.

Switches 2-3 and 2-4 have exchanged functions. Switch 2-3 now controls skip-over-perforation; Switch 2-4 controls automatic line feed with carriage return.

The Double-Strike creep noted in the Graftrax Manual (pages 6-9 and 6-10) has been eliminated.

Seven-bit computers can enter Graphics Mode on the FX-80 with the high-order bit set, although to little purpose.

N_2 in the ⟨ESC⟩"K"CHR$($N_1$)CHR$(N_2) sequence no longer works modulo 8. For example, 128 no longer works as a 0, so sending a 128 for N_2 sets the number of columns to $N_1 + 256*128 = N_1 + 32768$. Yipes! If you have the nerve to send it that much data, the printer prints one complete line of graphics (480 columns in ⟨ESC⟩"K") and ignores everything else until the graphics requirement is filled. Then it continues printing on the next line as if nothing strange had happened.

Table G-1 compares the control codes available on the FX and MX printers.

Hardware

The FX-80 has separate alarms for different error conditions. The sounds and their errors are covered in Appendix H.

The FX-80 has built-in pin feed and friction feed. A tractor unit is available for printing mailing labels or any pin-feed paper smaller than 8 ½ inches wide.

Table G-1. Control Code Comparison

The control codes of MX Graftrax Plus and the FX-80 are listed below.

Code	Function	MX GT +	FX
BEL	Sounds bell	*	*
BS	Backspace	*	*
HT	Horizontal Tabulation	*	*
LF	Line Feed	*	*
VT	Vertical Tabulation	*	*
FF	Form Feed	*	*
CR	Carriage Return	*	*
SO	Shift Out (Expanded)	*	*
SI	Shift In (Compressed)	*	*
DC1	Places the printer in an active state		*
DC2	Cancels Compressed Mode	*	*
DC3	Deactivates the printer		*
DC4	Cancels Expanded Mode (SO)	*	*
CAN	CANcels text in print buffer		*
DEL	DELetes last character	*	*
ESC	Changes print modes	*	*
ESC SO	Same as SO		*
ESC SI	Same as SI		*
ESC !	Selects mode combinations		*
ESC #	MSB mode cancel	*	*
ESC %	Selects active character set (internal ROM or user RAM)		*
ESC :	Copies from ROM to user RAM		*
ESC &	Defines user characters		*
ESC *	Selects dot Graphics Modes		*
ESC /	Selects active vertical tab channel		*
ESC -	Underline Mode	*	*
ESC ∧	9-Pin Graphics Mode		*

Code	Function	MX GT +	FX
ESC 0	1/8″ line spacing	*	*
ESC 1	7/72″ line spacing	*	*
ESC 2	1/6″ line spacing	*	*
ESC 3	n/216″ line spacing	*	*
ESC 4	Italic character font on	*	*
ESC 5	Italic character font off	*	*
ESC 6	Deactivates high-order control codes		*
ESC 7	Restores high-order control codes		*
ESC 8	Paper end detector off	*	*
ESC 9	Paper end detector on	*	*
ESC ⟨	One-line Unidirectional Mode	*	*
ESC ⟩	MSB set	*	*
ESC =	MSB reset	*	*
ESC @	Master Reset	*	*
ESC A	n/72″ line spacing	*	*
ESC B	Vertical tab setting		*
ESC C	Page length	*	*
ESC D	Horizontal tab setting	*	*
ESC E	Emphasized Mode on	*	*
ESC F	Emphasized Mode off	*	*
ESC G	Double-Strike Mode on	*	*
ESC H	Double-Strike Mode off	*	*
ESC I	Control code select		*
ESC J	Immediate temporary line feed	*	*
ESC K	Single-Density Graphics Mode	*	*
ESC L	Double-Density Graphics Mode	*	*
ESC M	Elite Mode on		*
ESC N	Skip-over-perforation on	*	*
ESC O	Skip-over-perforation off	*	*
ESC P	Elite Mode off		*
ESC Q	Column length set	*	*
ESC R	International characters		*
ESC S	Super/Subscript on	*	*
ESC T	Super/Subscript reset	*	*
ESC U	Unidirectional printing	*	*
ESC W	Expanded Mode (continuous)	*	*
ESC Y	Double-Speed Double-Density Graphics Mode		*
ESC Z	Quadruple-Density Graphics Mode		*
ESC b	Set vertical tab channel		*
ESC i	Immediate print (typewriter mode)		*
ESC j	Immediate temporary reverse paper feed		*
ESC l	Set left margin		*
ESC p	Proportional Mode on/off		*
ESC s	Half-Speed on/off		*

Appendix H
Hex Dumping and Problem Codes

The Seven-Bit Limitation

BASIC programs on the Apple II Plus (and perhaps some other computers) can send to the printer only 7 bits at a time instead of the normal 8 (machine-language programs can send 8). Because of this limitation, the CHR$ function on these computers is unable to deliver the entire range of ASCII codes (0 - 255) to the printer; it can send only the codes 0 - 127. The FX-80 can interpret these codes as either 0 - 127 or as 128 - 255—but not as both ranges at once. The low range is automatically active unless the high range is set by the user.

The computer converts all codes outside of the active range to their equivalent active range code by adding or subtracting 128.

The following simple test can be used to determine the status of your computer:

```
    1Ø FOR X=16Ø TO 254
    2Ø LPRINT CHR$(X);
    3Ø NEXT X
```

If you get italic characters when you RUN this, your system is capable of generating all 256 ASCII codes (it's an 8-bit system). If you get standard roman characters, then your system is forcing all codes into the lower range; you should pay close attention to the following.

As you have discovered in this manual, the FX-80 sometimes requires full 8-bit capability. Users of 7-bit computers will have to find ways to cope with these situations. In most cases, the 7-bit limitation is not totally debilitating, but rather more of an annoyance. Often you can achieve the same results as 8-bit users can, but it takes a little more coding or manipulating of the high-order bit.

The FX-80 has three commands that all 7-bit machines can use to temporarily set the eighth bit. These commands are covered in detail in Chapter 6 under "Solving the Seven-Bit Dilemma." In brief, the high-order bit is set with the following command:

```
    LPRINT CHR$(27)">";
```

This command converts all codes to the upper range. That works great as long as you don't need both a high and a low code in the same command. For example, the command sequence for entering Graphics Mode requires one number from 0 - 255 and a second number from 0 - 7. If your application requires the first number to be greater than 127, you'll have to set the high-order bit before entering Graphics Mode. But this also makes the second number (and the subsequent graphics data) greater than 127. This sets the graphics width to some astronomical figure (over 32,000). Unfortunately, there is no way to switch the high-order bit back and forth in the middle of an ESCape sequence. So 7-bit users are well advised not to use the high-order bit in Graphics Mode.

Other problems concerning the 7-bit limitation are as follows:

Tabs can't be set in both code ranges.
Width settings are limited when entering Graphics Mode.
The top pin of the print head is unusable in Graphics Mode (details in Chapter 11).
User-defined characters can't be printed with the top 8 pins (the standard position for most characters).
Eight pins can't be used in defining characters.

Hex Dump

The hex dump mode echos each code received by the FX-80 onto the paper as a string of hexadecimal values. You turn hex mode on by holding the FF button down while you turn the printer on.

The hex dump is useful for determining which codes are creating problems for a particular system. Some computers change certain codes when sending them to the printer from BASIC programs. A hex printout of a program shows exactly what the printer is receiving, regardless of what the computer is sending (or not sending). The following program lets you see what your computer's problem codes are.

Put the printer in hex dump mode and then RUN this program.

```
10 FOR X=0 TO 255
20 LPRINT CHR$(X);
30 NEXT X
```

If your system passes the codes faithfully on to the printer, your output should look like Table H-1 (take your printer off-line to print the last line).

Table H-1. Best-case Hex Dump.

```
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  40  41  42  43  44  45  46  47  48  49  4A  4B  4C  4D  4E  4F
50  51  52  53  54  55  56  57  58  59  5A  5B  5C  5D  5E  5F  60  61  62  63
64  65  66  67  68  69  6A  6B  6C  6D  6E  6F  70  71  72  73  74  75  76  77
78  79  7A  7B  7C  7D  7E  7F  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  0A
```

Most BASICs, however, are not quite that straightforward. For example, the TRS-80 Model III prints Table H-2:

Table H-2. Model III Hex Dump.

```
00  01  02  03  04  05  06  07  08  09  0D  0B  0A  0A  0A  0A  0A  0A  0A  0A
0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A
0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A
0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  0A  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  40  41  42  43  44  45  46  47  48  49  4A  4B  4C  4D
4E  4F  50  51  52  53  54  55  56  57  58  59  5A  5B  5C  5D  5E  5F  40  61
62  63  64  65  66  67  68  69  6A  6B  6C  6D  6E  6F  70  71  72  73  74  75
76  77  78  79  7A  7B  7C  7D  7E  7F  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
```

Notice that the decimal code 10 is being received as hex code 0D (which is really decimal code 13). In addition, ASCII code 12 is coming across as a series of line feeds (decimal 10, 0A in hex).

The hex mode prints 20 characters per line. If less than 20 characters are sent, the printer sits in a holding pattern, awaiting more data. Taking the printer off line will dump these characters to the paper.

The hex mode can be very useful in debugging a program. Just place the printer in hex mode, RUN the program, and the printer will show you exactly what it is receiving.

Problem Codes

Once you've determined the problem codes peculiar to your system, either by using the hex test above or by trial and error, you can start overcoming them.

Because each computer has a different set of problem codes, it is not possible, in one short appendix, to provide solutions for all the computers out there. But we can point out possible ways for you to solve your problems.

There are generally two approaches. First, most computers can avoid problem codes by sending each code directly to the printer. This bypasses the BASIC interpreter and avoids the interface (where most of the problems lie).

Unfortunately, this process is also different for each computer. We'll take a stab at a couple here; if yours is not one of them, don't worry. Remember that the concept is the same; consult your computer's manual to determine if you can do the same on your system.

A second approach is to change the printer driver program in your system. This requires a knowledge of machine language and of the way your computer works. We show a sample driver below. The idea behind a new printer driver is to pass the codes (as issued by a BASIC program) directly to the printer rather than intercept and possibly change them on the way through your computer.

We show examples of both methods below. Perhaps you can adapt the ideas to your system.

POKEing Codes

The TRS-80 Model I version of the CHR$ function does not correctly pass on the values of 0, 10, 11, and 12. Zero is a particular problem as it is very important to the ESC codes of the FX-80 yet is totally ignored by the TRS-80.

These codes can be sent directly to the printer by POKEing them to a special memory location where they are immediately forwarded to the printer. The format is:

```
POKE 14312,N
```

where N is the decimal value of the code you wish to send to the printer. This works fine as long as the printer is ready to receive the data when you are ready to send it. On the TRS-80, the printer's readiness is assured if location 14312 contains a decimal 63.

It is best to first test to see if the printer is ready with:

```
100 IF PEEK(14312)<>63 THEN 100
```

This puts the program into a continuous loop until the printer is ready to receive data. If data is sent while the printer is "out to lunch," it will be lost.

To show how similar these commands can be from system to system, here is the same concept implemented on the Apple II Plus:

```
100 IF PEEK(49601)>127 THEN 100
200 POKE 49296,N
```

The printer's status is stored in location 49601 and the outgoing values are sent to 49296.

Using a Special Printer Driver

An even better (but more difficult) way to overcome these problems is to modify the printer driver so that the codes are passed correctly to the printer without any PEEKs or POKEs. Unfortunately, modifying a printer driver requires a knowledge of machine-language programming and a familiarity with your computer's operating system. If you are not well versed in these areas, join the club. Perhaps your computer dealer knows of a solution. Those who have mastered the machines, let it be known in the trade journals.

The following PRINTER DRIVER is written for the Model I TRS-80 by Bob Boothe and reprinted with the kind permission of 80 Microcomputing. The program pokes a machine-language printer driver program (stored in line 10) into memory, then tells the system where its new driver is located. Once the program is run, all codes sent by any BASIC program are sent directly to the printer—including 0s, 10s, and 12s.

```
10 DATA 21E837CB7E20FC211100397E32E837C9
20 READ B$: A=16571
30 FOR P=1 TO LEN(B$) STEP 2
40 B=ASC(MID$(B$,P,1)) - 48
50 IF B>9 THEN B=B - 7
60 T=ASC(MID$(B$,P + 1,1)) - 48
70 IF T>9 THEN T=T-7
80 POKE A,B*16 + T
90 A=A+1
100 NEXT P
110 POKE 16422,187
120 POKE 16423,64
```

This driver will also work on the Model III—with one change in line 10: change 32E837 to D3FB. That's all there is to it! Once the program is run, you can kiss problems with radical codes goodbye.

One More Solution for the Apple

The Apple uses CHR$(9) to "initialize" the printer. This code and the following character or characters are intercepted by the printer interface card and used to change modes (somewhat like the way the printer uses ESCape codes). Apple users can divert all output to the printer instead of the screen by sending the following line to the printer interface card:

```
PR#1
PRINT CHR$(9)"80N"
```

. . . and type anything, followed by ENTER.

Pretty slick, huh? The CHR$(9)"80N" code directs all subsequent output to the printer, up to 80 characters per row. You can cancel this by typing:

```
PRINT CHR$(9)"I"
```

or

```
PR#0
```

But the printer also uses the CHR$(9) code to activate horizontal tabulation. The problem is finding a way to sneak the CHR$(9) past the interface board so that it can be used by the printer.

You can avoid the problem by changing the printer initialization code to something else; this frees the CHR$(9) for horizontal tabulation. Try this:

```
PR#1
PRINT CHR$(9); CHR$(1)
```

Now the printer will initialize with CHR$(1).

Error Detection

The FX-80 will warn you if certain error conditions exist by sounding the internal bell. Each of the four error conditions has a unique sound. Listen closely . . .

1. A short circuit between the collator and the emitter of the head transistor produces:

   PEE, PEE, PEE, PEE

2. Detection of high voltage produces:

   PI, PI, PI, PEE

3. A slave CPU error produces:

   PI, PI, PI, . . . PI, PI, PI

4. The paper running out produces:

   PI, PI, PI, PI (repeated five times)

Note: Except for paper-out sensing, the bell will sound regardless of switch settings.

Appendix I
Troubleshooting the Sample Programs

Problems in the Programs

This appendix can help you identify and resolve potential problem areas in the sample programs. In programs where well known systems are likely to encounter problems, we've referred you to this appendix. We've elected to handle solutions to recurring problems here rather than disrupt the flow of the tutorial for all users.

The entries in this appendix are keyed to specific programs in the manual. In some cases, an alternate approach is suggested for a particular computer; in others, you are referred to Appendix H, which includes a discussion of the major problem codes and examples of ways to deal with them.

Page 69

Model I users—change line 20 to:

```
20 LPRINT CHR$(27)"W"CHR$(48+J)
```

Figure 6-1

Apple systems use the CHR$(9) as a sort of escape code (called the printer initialization code), but the FX-80 uses CHR$(9) to activate horizontal tabs. To avoid a conflict of interest, Apple users must change the printer initialization code. Try:

```
PR#1
PRINT CHR$(9);CHR$(1)
```

This makes CHR$(1) the new printer initialization code, which frees CHR$(9) for horizontal tab duties.

TRS-80 Model I users have problems with CHR$(0) in line 70. The best solution to this problem is to use the Printer Driver listed in Appendix H. An alternate solution is to POKE the 0 directly to the printer:

```
70 FOR Y=0 TO 8: LPRINT CHR$(27)"R";
74 IF PEEK(14312)<>63 THEN 74
76 POKE 14312,Y;
```

Page 83 (top)

Model I users can use the Printer Driver from Appendix H or change line 30 to:

```
30 LPRINT TAB(6*X);"STAIR";CHR$(27)"A";
32 IF PEEK(14312)<>63 THEN GOTO 32
34 POKE 14312,0
36 LPRINT "STEPS"
```

Page 93

Model I users can use the Printer Driver or POKE the CHR$(0) in line 10.

Model I and Model III users can avoid the problem-ridden CHR$(12) as a form feed by using its high-order counterpart. Change line 30 to:

```
30 LPRINT CHR$(140)"TWO-INCH FORM"
```

Page 108

Apple users—change your printer initialization code to CHR$(2). See Appendix H.

Page 120

Model I users can employ the Printer Driver to avoid CHR$(0). Apple users can change the printer initialization code (9) to some unused number.

Page 134 **Page 138 (top)** **Page 141**

TRS-80 Model I users can use the Printer Driver for programs in these chapters.

Page 157

Model I users can use the Printer Driver to avoid CHR$(0). Apple users—change the printer initialization code (9) to some unused number.

Page 173

TRS users—use the Printer Driver for all programs in Chapter 15.

Page 188

One way to handle trouble codes for this program is to change them to similar patterns with less troublesome numbers. For example:

```
180 NEXT DOT: IF P=12 THEN P=4
200 N1=C: N2=0: IF P0=12 THEN P0=4
```

Page 197

Model I users—run the Printer Driver for this chapter.

Page 206 (top)

7-bit computer users—add these lines to the program:

```
3 LPRINT CHR$(27)">"
5 LPRINT CHR$(27)"="
```

Page 206 (bottom)

Sorry 7-bit'ers, this is one of the programs you cannot RUN. Setting the MSB to print 128 and 131 will not work because the CHR$(0) will then be read as CHR$(128) . . . the printer will not understand.

Page 211 **Page 226**

Model I users—again see Appendix H for the Printer Driver.

Appendix J
General Troubleshooting

In this appendix we list several typical problems encountered by users in various printing situations—along with possible solutions.

Problem	Solution
Setting Print Styles	
Can't get Compressed print.	Cancel Emphasized, Elite, Proportional, and/or Script Modes. They have priority over Compressed. See Table F-4 for priorities.
Doesn't go back to Pica print when Proportional Mode is cancelled.	Proportional print "masks" other modes. When it is cancelled, the printer assumes the mode that it was in prior to Proportional. To get back to Pica, cancel all other modes.
Changing Form Measurements	
The 〈ESC〉"C" command is not working properly.	Don't set form lengths of 0 or 128.
The 〈ESC〉"N" skip-over-perforation doesn't work.	Don't set the SOP larger than the form length.
Tabbing	
Vertical tabs don't work correctly.	Can't set vertical tabs greater than the form length. Each tab sequence must be terminated by a CHR$(0) or a number less than the last tab value.

Problem	Solution
Horizontal tabs don't work correctly.	Each tab sequence must be terminated by a CHR$(0) or a number less than that of the last tab value.
Horizontal tabs are incorrect when changing pitch.	Tabs are set according to current print pitch. Changes in pitch do not affect the position of the tabs on the page.
Graphics	
Strange things print.	Computers have problems sending codes from 8 - 13 and 0. They are best avoided if possible. Otherwise, you can POKE them directly to the printer.
Printer "freezes" in Graphics Mode.	The printer expects a certain number of pin patterns, determined by n1 and n2. It will wait patiently until the quota is full. Note that 9-Pin Graphics Mode requires 2 bytes for each column of graphics.
Having trouble getting a full page in width.	Seven-bit computers are limited to widths of 0 - 127, 256 - 383, 512 - 639, etc. See Chapter 11.
Getting funny stuff in the pin patterns.	Seven-bit computers cannot use the 8th pin (128). If you have a 7-bit computer and your pin sequence is larger than 127, change it!
Having trouble getting into Graphics Mode.	Make sure the high-order bit is OFF. For 7-bit computers, that means sending the code: 〈ESC〉"=".
User-Defined Characters	
The last character is swallowed by the printer . . . nothing gets printed.	Make sure the attribute byte is sent before the 11-pin patterns for EACH character.

Problem	Solution
Characters are one dot too high or low.	Use the correct setting for the attribute byte. An attribute byte less than 128 makes the bottom 8 pins active (good for descenders). An attribute byte of 128 or greater activates the top 8 (matching ROM characters without descenders).
Characters are running too close together.	Must reserve space between characters within the character design. Typically, the last 2 columns are defined as 0. See ROM characters in Appendix B.
Some of the dots are being ignored by the printer.	Two dots in the same row cannot be printed in adjacent columns.

1 2 3 4 5 6 7 8

↑ ↑ V V Λ Λ V Λ

1 2 3 4

V V V V

Appendix K
Switch Settings

The FX-80 has two sets of internal switches which are used by the printer to determine the default mode on power-up. The switches are under the upper right vent. As outlined in Chapter 1, the vent screw must be removed with a phillips-head screwdriver in order to take the cover off.

Since switch settings are only checked by the printer on power-up, all switch setting should be done with the power off. The printer will not recognize changes made in switch settings when the power is on until the printer is cycled off and then on again.

The factory sets and numbers the switches in the following way:

Table K-1. The DIP Switches.

Switch 1

1-8	ON ▨	International character	OFF
1-7	ON ▨	International character	OFF
1-6	ON ▨	International character	OFF
1-5	Emphasized	Print mode	Normal ▨
1-4	2K buffer	RAM memory	User-defined Characters ▨
1-3	Inactive	Paper-out sensor	Active ▨
1-2	Ø	Zero font	0 ▨
1-1	ON	Compressed Mode	OFF ▨

Switch 2

2-4	CR + LF	Automatic line feed	CR only ▨
2-3	ON	Skip-over-perforation	OFF ▨
2-2	ON ▨	Bell	OFF
2-1	ON ▨	Printer select	OFF

Note: The shaded boxes show the factory settings.

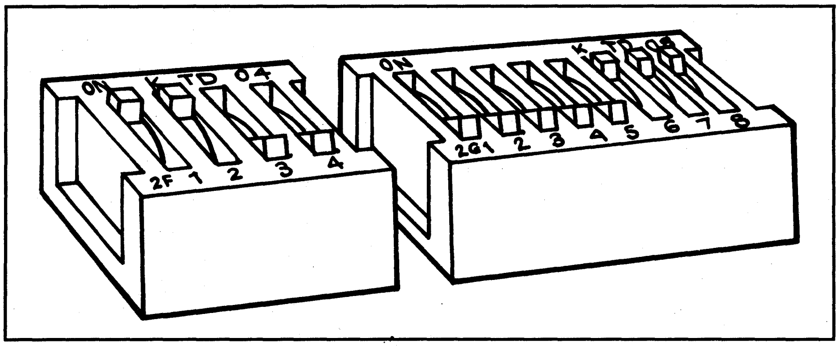


Figure K-1. Factory setting of the DIP switches.

Examining the Switches

Switches 1-6, 1-7, and 1-8 determine the active international character set as shown below:

SW 1-6	SW 1-7	SW 1-8	COUNTRY
ON	ON	ON	USA
ON	ON	OFF	FRANCE
ON	OFF	ON	GERMANY
ON	OFF	OFF	ENGLAND
OFF	ON	ON	DENMARK
OFF	ON	OFF	SWEDEN
OFF	OFF	ON	ITALY
OFF	OFF	OFF	SPAIN

See Chapter 6.

Switch 1-5: When ON, Emphasized Mode becomes the default. When OFF, normal (Pica) is the default mode. All switch modes are overridden by software commands.

Switch 1-4: When ON, makes a 2K communications buffer available. When OFF, memory can be used for user-defined characters.

Switch 1-3: When ON, paper-out detection is deactivated, and printing will continue when paper is out (printer stays on-line). When OFF, printing stops when paper is out, and bell sounds if switch 2-2 is on.

Switch 1-2: When ON, a slashed Ø is printed. When OFF, a normal 0 is printed.

Switch 1-1: When ON, Compressed Mode becomes the default. When OFF, normal (Pica) characters are the default. If both switch 1-5 and switch 1-1 are ON, Emphasized Mode takes priority over Compressed.

Switch 2-4: When ON, the printer does an automatic line feed with every carriage return. When OFF, no line feed is added (line feed must be provided by computer).

Switch 2-3: When ON, a 1″ paper feed is performed at the bottom of every form. Note that the top of form is set when the printer is turned ON. This switch is used primarily to automatically skip over the paper perforations of fanfold paper.

Switch 2-2: When ON, the printer's buzzer sounds when the paper-out condition is detected or CHR$(7) is sent. When OFF, CHR$(7) or paper-out doesn't sound buzzer, but printer goes off-line and paper stops feeding for the paper-out condition.

Switch 2-1: When ON, the printer actively processes commands sent from the computer; it cannot be deactivated with software codes. When OFF, the printer can be activated and deactivated by external software codes. CHR$(17) (DC1) activates or turns on printing, and CHR$(19) (DC3) deactivates or turns off printing. While the printer is inactive, all input data is ignored (until the printer is reactivated by CHR$(17)).

Appendix L
Customizing the FX-80

The FX-80 can be adapted to fit a variety of needs. The print capability can be altered both by hardware switches that change default conditions and by software codes that activate or deactivate different print modes. In this Appendix we look at a few typical applications and adjustments of the printer which may be appropriate.

Spread-Sheet Programs

Three changes in the standard configuration of the printer may be of some use to those who make heavy use of spread-sheet programs. Changing switch 1-2 to ON will print all 0s with slashes, making it easy to distinguish between 0s and O's. Switch 1-4 ON makes the 2K buffer available so that you can work with the computer while the sheet finishes printing. Also, Switch 1-1 ON changes the power-up pitch to Compressed so that you can squeeze more characters into a line.

Word Processing

If your printer is used primarily for word processing, you may want to make some permanent adjustments. Since most word processors have their own page formatting controls, you should probably leave switch 2-3 (automatic skip-over-perforation) OFF.

When switch 1-5 is on, it activates Emphasized Mode on power-up to give your printouts a better look. Keep in mind that this puts more wear on your ribbon than usual.

An even better look can be achieved with Proportional print. Since there is no hardware switch for this mode, you'll have to send the required codes to activate Proportional Mode from BASIC (or perhaps from your word processor). But don't try to right-justify your text unless the word processing program is specifically designed to justify characters with Epson's Proportional Mode.

Another option—for those who use foreign characters in their correspondence—is to select the appropriate country with switches 1-6, 1-7, and 1-8. Eight of the 9 countries can be selected (all except Japan) according to the chart in Appendix K. This will work as long as your word processor can send out the required codes. See Chapter 6.

Single-Sheet Forms

For extensive use of forms or letters that require a single sheet of letterhead paper, you can deactivate the paper-out sensor by turning switch 1-3 on. This will keep the printer from shutting down before it reaches the end of the page. By causing the printing to pause at the end of a page, you will have time to insert the next sheet. (Some word processors pause automatically after each page.)

BASIC Program Listings

Set switch 1-2 ON to print all 0s with slashes. Set switch 2-3 ON or use ⟨ESC⟩"N"CHR$(n) to activate the skip-over-perforation feature.

Quiet Printing

The protective lid dampens the noise output. The Half-Speed Mode can be activated with ⟨ESC⟩"s1"—this also cuts down on the noise level. And for those late night sessions, you may want to turn the bell off by turning switch 2-2 OFF.

Graphics and User-Defined Characters

Set switch 1-4 OFF to enable you to define characters.

Appendix M
Printer Maintenance

Always

Always keep your printer in a safe and clean location. Keep it away from:

Dust and grease.
Heaters and furnaces. Safe temperature range is 5°C (41°F) to 35°C (95°F).
Large electrical machines.
Crowded areas (the printer gets uptight if its paper is cramped).
Leaky ceilings.
Rambunctious children or pets.

Now and Then

Clean particles and dust from the printer every so often with a soft cloth or brush. Use a mild cleanser for the outside framework and, after removing the ribbon cartridge, denatured alcohol for the inside.

The ribbon cartridge should live to print about 3 million characters (a ripe old age). See your Epson dealer for replacements.

Rarely

Once in a great while your printer should be lubricated. Epson recommends two lubricants: O-2 and G-2. Every 6 months or one million lines (count 'em!), use O-2 lubricant on the shafts and platen bearings. Use G-2 elsewhere every five million lines (or when you take the printer in for an overhaul).

Changing The Print Head

The print head lives to be about 100,000,000 characters old (assuming an average of 14 dots per character). See your Epson dealer for FX-80 print head replacements.

To remove the old print head:

If you have been printing, turn the printer OFF for about 15 minutes to allow the print head to cool. Next, remove the protection lid and ribbon cartridge. Flip the head lock lever shown in Figure M-1 toward the back of the printer.

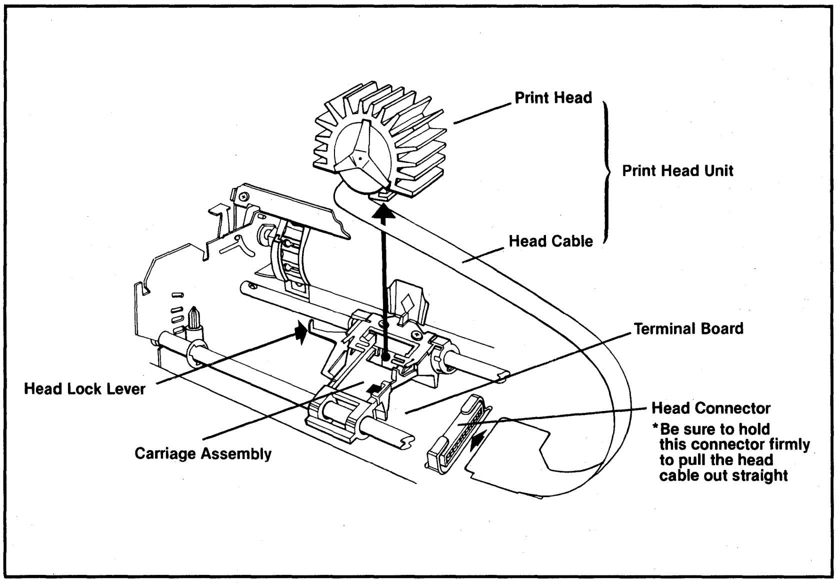


Figure M-1.

Now pull the cable from the connector block. Hold the block firmly because it has to stay put! Pull the print head straight up and off.

To install the new print head:

Place the new print head onto the head mount and flip the locking lever back toward the front of the printer. Connect the cable to the block. That's all there is to it!

Appendix N
Technical Specifications

Printing

Printing method	Impact dot matrix
Printing speed	160 characters per second
Paper feed speed	Approximately 150 ms/line (at 1/6 inch/line)
Printing direction	Bidirectional, logic seeking Unidirectional (left to right) in bit-image mode
Character set	ASCII 96 32 special international characters (8 international character sets) 96 alternate italic characters
Character size	**11 (wide) x 9 (high) dot matrix** 2.1 mm x 3.1 mm (Pica, the default) 4.2 mm x 3.1 mm (Expanded Pica) 1.05 mm x 3.1 mm (Compressed) 2.1 mm x 3.1 mm (Expanded Compressed) 1.4 mm x 3.1 mm (Elite) 2.8 mm x 3.1 mm (Expanded Elite) The pitch width x 1.6 mm (Super/Subscript)
Line spacing	1/6 inch (the default) Programmable in increments of 1/72 and 1/216 inch
Column width	80 characters (Pica, the default) 40 characters (Expanded) 132 characters (Compressed) (137 characters using 〈ESC〉"Q") 66 characters (Compressed/Expanded) 96 characters (Elite) 48 characters (Elite/Expanded)

Paper

Paper feed	Adjustable sprocket feed Friction feed Optional tractor feed
Paper thickness	0.3 mm (0.012″) maximum
Number of copies	One plus 2 carbon copies
Paper width	9.5″ to 10.0″ (without the optional tractor unit) 4.0″ to 9.0″ (with tractor unit)

Printer

Ribbon	Cartridge ribbon, black
Ribbon life expectancy	3,000,000 characters
MTBF	5,000,000 lines (excluding print-head life)
Print head life	100,000,000 characters
Dimensions	100 mm(H)x420 mm(W)x347 mm(D)
Weight	7.5 kg (main unit only) 0.5 kg (for tractor unit)
Power	AC 120V plus or minus 10%
Frequency	49.5 to 60.5 Hz
Temperature	Operating 5°C to 35°C (41°F to 95°F) Storage -30°C to 70°C (-22°F to 158°F)
Humidity	Operating 10% to 80% (no condensation) Storage 5% to 85% (no condensation)
Shock	Operating 1 G (less than 1 msec.) Storage 2 G (less than 1 msec.)
Vibration	Operating 0.25 G, 55Hz (Max.) Storage 0.50 G, 55Hz (Max.)
Insulation resistance	10 megaohms between AC power line and chassis
Dielectric strength	No trouble when AC 1 KV (R.M.S.) 50 or 60 Hz is applied for more than 1 minute between AC power line and chassis

Parallel Interface

Interface	Centronics-compatible, 8-bit parallel. (Compatible with EPSON MX-series)
Synchronization	By externally supplied strobe pulses
Handshaking	By ACKN or BUSY signals
Logic level	Input data and all interface control signals are compatible with the TTL level
Parallel connector	57-30360 (Amphenol)

Appendix O
The Parallel Interface

The FX-80 uses a parallel interface to communicate with the computer; this appendix describes it.

Connector pin assignments and a description of respective interface signals are shown in Table O-1.

Table O-1. Pins and Signals

Signal Pin#	Return Pin#	Signal		Direction
1	19	$\overline{\text{STROBE}}$	IN	$\overline{\text{STROBE}}$ Pulse to read data in. Pulse width must be more than 0.5 microseconds at the receiving terminal.
2	20	DATA 1	IN	(Signal Pin Nos. 2 - 9) These signals represent information of the 1st to 8th bits of parallel data, respectively. Each signal is at HIGH level when data is logical 1 and LOW when it is logical 0.
3	21	DATA 2	IN	
4	22	DATA 3	IN	
5	23	DATA 4	IN	
6	24	DATA 5	IN	
7	25	DATA 6	IN	
8	26	DATA 7	IN	
9	27	DATA 8	IN	
10	28	$\overline{\text{ACKNLG}}$	OUT	Approximately 5-microsecond pulse. LOW indicates that data has been received and that the printer is ready to accept more data.
11	29	BUSY	OUT	A HIGH signal indicates that the printer cannot receive data. The signal goes HIGH in the following cases: 1) During data entry. 2) During printing. 3) Off-Line. 4) During printer-error state.

Table O-1, continued.

Signal Pin#	Return Pin#	Signal		Direction
12	30	PE	OUT	A HIGH signal indicates that the printer is out of paper.
13	—	SLCT	OUT	Logic HIGH level.
14	—	$\overline{\text{AUTO FEED XT}}$	IN	When this signal is LOW, the paper is automatically fed 1 line after printing. (The signal level can be fixed to this by setting DIP switch 2-3 to ON.)
15 —	NC	—		Unused.
16	—	OV	—	Logic GND level.
17	—	CHASSIS GND	—	Printer's chassis ground, which is isolated from the logic ground.
18	—	NC	—	Unused.
19 - 30	—	GND	—	Twisted-pair return signal ground level.
31	—	$\overline{\text{INIT}}$	IN	When this level becomes LOW, the printer controller is reset to its initial state and the print buffer is cleared. This level is usually HIGH; its pulse width must be more than 50 microseconds at the receiving terminal.
32	—	$\overline{\text{ERROR}}$	OUT	This level becomes LOW when the printer is in: 1) Paper-end state. 2) Off-line. 3) Error state.
33	—	GND	—	Same as for Pins 19 - 30.
34	—	NC	—	Unused.
35	—	+5V	—	Pulled up to +5V through 3.3K ohm resistance.
36	—	$\overline{\text{SLCT IN}}$	IN	Data entry to the printer is possible only when this level is LOW; DIP switch 1-8 is set for this at the factory.

Notes:

1. "Direction" refers to the direction of signal flow as viewed from the printer.
2. "Return" denotes "twisted-pair return" and is to be connected at signal ground level. For the interface wiring, be sure to use a twisted-pair cable for each signal and to complete the connection on the return side. To prevent noise, these cables should be shielded and connected to the chassis of the host computer and the printer, respectively.
3. All interface conditions are based on TTL level. Both the rise and the fall times of each signal must be less than 0.2 microseconds.
4. Data transfer must be carried out by consulting with the $\overline{\text{ACKNLG}}$ or BUSY signal. (Data transfer to this printer can be carried out only after configuration of the $\overline{\text{ACKNLG}}$ signal or when the level of the BUSY signal is LOW.)

Data Transfer Sequence

Interface timing

Figure O-1 shows the timing for the parallel interface.

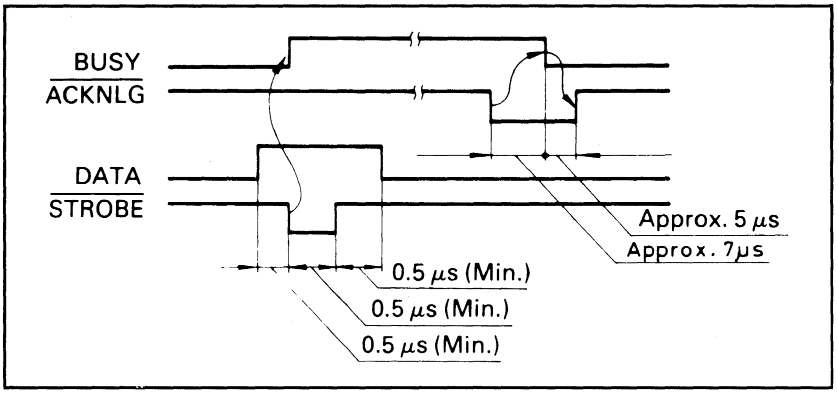


Figure O-1. Parallel interface timing.

Signal interrelations

Table O-2 shows the way data entry is handled in the On-Line and Off-Line states. The relationships between seven signal sets are also shown.

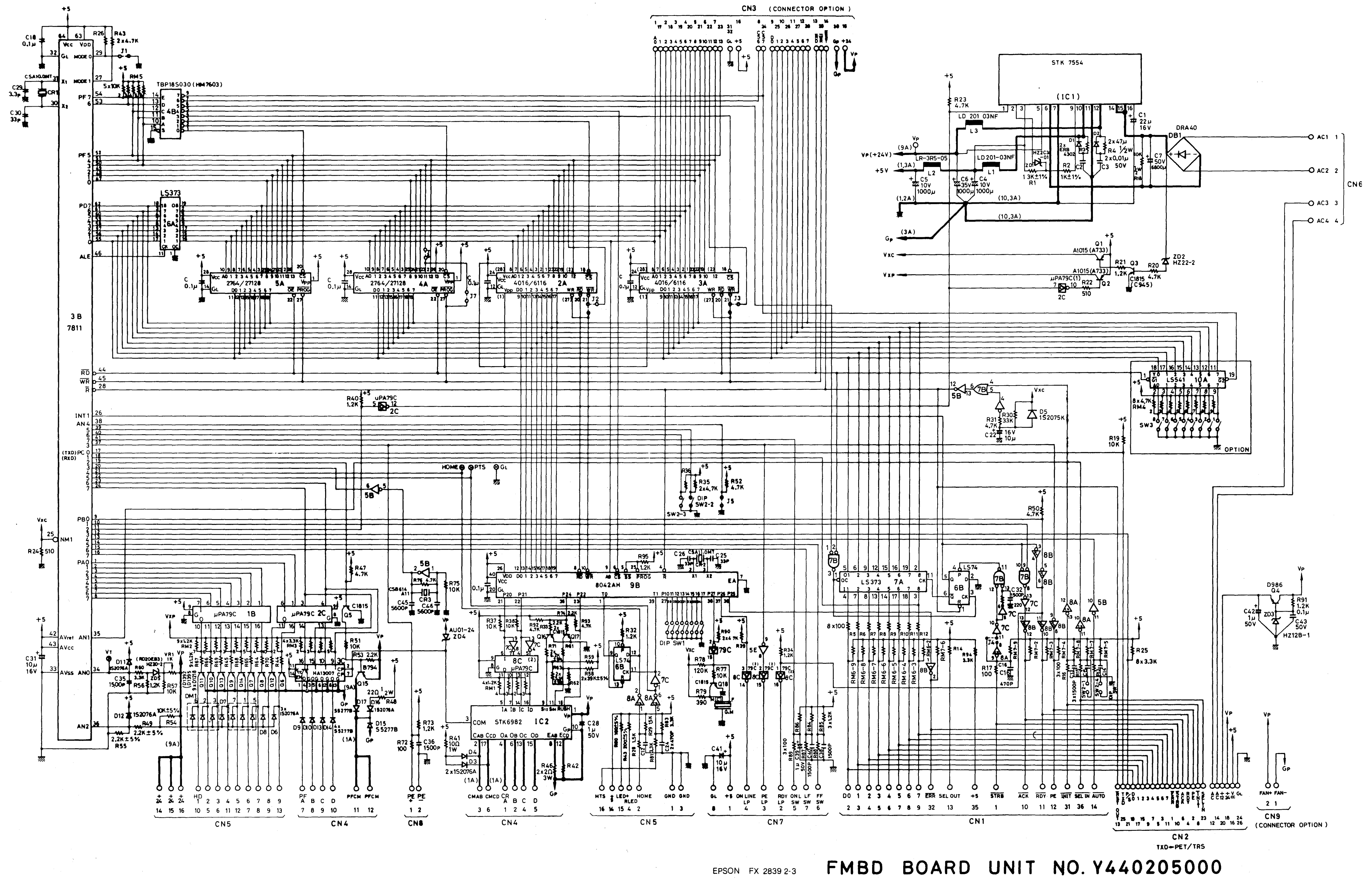
Table O-2. Signal Interrelations.

On-Line	$\overline{\text{SLCT IN}}$	DC1/DC3	$\overline{\text{ERROR}}$	BUSY	$\overline{\text{ACKNLG}}$	DATA ENTRY
OFF	HIGH/LOW	DC1/DC3	LOW	HIGH	Not generated	Disabled
ON	HIGH	DC1	HIGH	LOW/HIGH	Generated after data entry	Enabled (normal entry)
ON		DC3	HIGH	same	same	Enabled*
ON	LOW	DC1/DC3	HIGH	same	same	Enabled (normal entry)

*Data entry enabled, but the input data will be disregarded until DC1 is input.

Note: $\overline{\text{ERROR}}$ status is assumed to result only in Off-Line state, and the $\overline{\text{ERROR}}$ status does not always mean $\overline{\text{SLCT IN}}$.

Appendix P
Schematics



EPSON FX 2839 2-3

FMBD BOARD UNIT NO. Y440205000

Index

ESCape "&" CHR$(n) CHR$(c1) CHR$(c2) CHR$(A) CHR$(D1) . . . CHR$(D11) Defines characters c1 to c2 in RAM area: *See* User-defined characters.
ESCape "*" CHR$(m) CHR$(n1) CHR$(n2). Selects Graphics Density n. *See* Graphics Mode.
ESCape "-0". Turns Underline Mode off. *See* Underline Mode.
ESCape "-1". Turns Underline Mode on. *See* Underline Mode.
ESCape "/" CHR$(n). Selects channel n. *See* Tabs, vertical.
ESCape "0". Sets line spacing to 1/8". *See* line spacing.
ESCape "1". Sets line spacing to 7/72". *See* line spacing.
ESCape "2". Sets line spacing to 1/16". *See* line spacing.
ESCape "3" CHR$(n). Sets line spacing to n/216". *See* line spacing.
ESCape "4". Turns on Italic Mode. *See* Italic Mode.
ESCape "5". Turns Italic Mode off. *See* Italic Mode.
ESCape "6". Enables printing of control codes 128-159. *See* ASCII codes; User-defined characters.
ESCape "7". Returns codes 128-159 to control codes. *See* ASCII codes; User-defined characters.
ESCape "8". Turns paper-out sensor off. *See* Paper-out sensor.
ESCape "9". Turns the paper-out sensor on. *See* paper-out sensor.
ESCape ":" CHR$(n1) CHR$(n2) CHR$(n3). Copies ROM characters to the user RAM area. *See* User-defined characters.
ESCape "<". Turns on Unidirectional Mode. *See* Unidirectional mode.
ESCape "=". Sets the high-order bit off. *See* Bit, high-order.
ESCape ">". Sets the high-order bit on. *See* Bit, high-order.
ESCape "@". Sets Master Reset Code. *See* Master Reset Code.
ESCape "A" CHR$(n). Sets line spacing to n/72". *See* line spacing.
ESCape "B" CHR$(n1) CHR$(nk) CHR$(0). Sets the vertical tab stops (Vtabs). *See* Tabs, vertical.
ESCape "C" CHR$(0) CHR$(n). Sets the form length in inches. *See* form length.
ESCape "C" CHR$(n). Sets the form length in lines. *See* form length.
ESCape "D" CHR$(n1). Sets the horizontal tabs. *See* Tabs, horizontal.
ESCape "E". Turns Emphasized Mode on. *See* Emphasized Mode.
ESCape "F". Turns Emphasized Mode off. *See* Emphasized Mode.
ESCape "G". Turns Double-Strike Mode on. *See* Double-Strike Mode.
ESCape "H". Turns Double-Strike Mode off. *See* Double-Strike Mode.
ESCape "I0". Returns codes 0-31 to control codes. *See* ASCII codes; User-defined characters.
ESCape "I1". Enables printing of control codes 0-31. *See* ASCII codes; User-defined characters.
ESCape "J" CHR$(n). Produces an immediate one-time line feed of n/216" without a carriage return. *See* Line Feed.
ESCape "K" CHR$(n1) CHR$(n2). Turns Graphics Mode, Single-Density on. *See* Graphics mode.
ESCape "L" CHR$(n1) CHR$(n2). Turns Graphics mode, Double-Density on. *See* Graphics Mode.
ESCape "M". Turns Elite Mode on. *See* Elite Mode.
ESCape "N" CHR$(n). Produces a variable Skip-Over-Perforation (SOP). *See* skip-over-Perforation.
ESCape "O". Turns Skip-Over-Perforation off. *See* Skip-Over-Perforation.
ESCape "P". Turns Elite Mode off. *See* Elite Mode.
ESCape "Q" CHR$(n). Sets the right margin. *See* margins.
ESCape "R" CHR$(n). Selects the international charater set n. *See* international character set.
ESCape "S0". Turns Subscript Mode on. *See* Script mode.
ESCape "S1". Turns Superscript Mode on. *See* Script mode.
ESCape "T". Turns either script mode off. *See* Script Mode.
ESCape "U0". Turns Continuous Unidirectional Mode off. *See* Unidirectional mode.

EPSON

FX-80 Printer

User's Manual

Chapters include:

EPSON AMERICA, INC.
3415 Kashiwa Street
Torrance, CA 90505

P 8294014

Printed in Japan 83.07-70