SR-10/15
USER'S MANUAL

NOT INTENDED FOR SALE

Federal Communications Commission
Radio Frequency Interference
Statement

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

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

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

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

A note about the programs in this manual:

This manual contains several programs that help to demonstrate the versatility of the SR-10/15 printers. Star Mfg. Co., Ltd. has made every effort to insure that the programs are functional and accurate. However, Star Mfg. Co., Ltd. cannot guarantee their accuracy or suitability to any particular application.

Trademark Acknowledgement

SR-10, SR-15: Star Mfg. Co., Ltd.
grafstar: Star Micronics
Apple, Apple II, Apple II+, Apple IIe, Applesoft: Apple computer Inc.
Commodore C-64: Commodore Business Machines, Inc.
Compaq: Compaq Computer corporation
CP/M: Digital Research
IBM Personal Computer, IBM PC: International Business Machines Corp.
Kaypro: Kaypro Computer Corporation
Microsoft BASIC: Microsoft Corporation
Osborne 1: Osborne Computer Corporation
TRS-80: Radio Shack, a division of Tandy Corporation

A Special Message to the New Owner

You're to be congratulated on selecting the printer of choice for both the sophisticated as well as the first-time user/owner — the new *SR-10/15*!

Right now, before you even start readying your SR-10/15 for action, we'd like to impress you with these two thoughts:

1. In as few words as possible, we'll highlight the several special features that SR-10/15 offers you, and
2. We'll show you how this manual can help you get the most from your SR-10/15, while saving you time, effort, and money.

Taking up the special features first, so they'll be fresh in your mind as you ramble through this manual . . . specifically . . .

Speed — At 200 characters per second top printing speed, It's one of the fastest in its class. And SR-10/15 is smart too: when printing blank spaces, SR-10/15 speeds up to a blistering 240 CPS!

634 Characters — Allows printing in no less than nine different fonts of type faces, including a brand new face which we call . . .

Near Letter Quality — A solid black dot-free, high-resolution type face that looks more like typewriter than computer-generated printing. Perfect for correspondence.

Faster Paper Handling — More economical, too. Automatic feeding for both single sheets and sprocket paper. And the unique built-in tractor design — behind the platen — avoids wasting a sheet each time you start printing, as in conventional loading. It also permits "reverse paper feed", for multiple colunm printing or other special applications, with a neater appearance, too.

Graphics — If you're designing your own, you'll be delighted at finding *six* different dot graphic densities with varying degrees of resolution or sharpness. There's even a *quadruple* density, with 240 dots per inch horizontal by 72 dots per inch vertical! And, you can print double density graphics at *double speed* !

Macro Instruction — A real timesaver on the keyboard. This feature allows you to define a sequence of codes and call (transmit) that entire sequence with a *single* code.

Easy Everything! — All the DIP switches are quickly accessible for ease in connecting your computer and changing print parameters; the ink ribbon comes in its own enclosed cartidge, ready to snap into place; paper is machine-fed, not cranked into place manually. Easy is the word for SR-10/15!

We think you'll also find this manual easy and pleasant to use. We've gone to great lengths to make it so. As a first example, look over the table of contents and you'll see what we mean. Whether greenhorn or wizard, *everybody* will find what they need to know to fulfill their expectations. We suggest that each new user/owner, *before you even unpack the box,* read or at least scan Chapters 2 and 3 – "Getting to Know Your SR-10/15" and "Getting Started with SR-10/15" – as well as Chapter 1, "Setting Up SR-10/15." *Now* you can unpack the box and start putting things together.

When you're ready to connect your computer to your SR-10/15, look at Appendix J for directions applying to your make of computer.

For you who wish to design your own characters, do your own plotting, your own infinite variety of dot graphic patterns and densities, you'll have a ball! For you, Chapters 5 through 10 are a must, and of course everybody should look at Chapter 11, which tells how to maintain your SR-10/15 for a long and carefree life.

In this manual there are plenty of example programs to demonstrate and show off all of SR-10/15's features. Since many SR-10/15 users have IBM Personal Computers (or the equivalent) all the example programs are written in Microsoft BASIC for the IBM. But throughout the manual, users of other computers will find hints on how to make SR-10/15 work with their computer.

So, gentle reader, with this manual we hand you the key to the wonderful world of SR-10/15. May you enjoy years of handsome, fast, and carefree printing!

Table of Contents

Table of Tables

CHAPTER 1

SETTING UP SR-10/15

In this chapter, we'll show you how to unpack your new SR-10/15 printer, set it up in the right location, and get it ready for you to load it with paper and start printing. But first . . .

WHERE SHALL WE PUT IT?

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

- Placing the printer on a flat surface
- Keeping it out of direct sunlight and away from heat-producing appliances
- Using it only in temperatures where you are comfortable
- Avoiding areas with a lot of dust, grease, or humidity
- Giving it "clean" electricity. Don't connect it to the same circuit as large, noise-producing motors
- Power supply voltage should be the same voltage that's specified on the identification plate – not over 10% more or less than the recommended AC voltage.
  Warning: Extremely high or low voltage can damage your printer.

WHAT HAVE WE HERE?

Now let's take a look at what's in the carton. Take it slow and easy, and check each item in the box against Figure 1-1. There should be exactly 8 items.

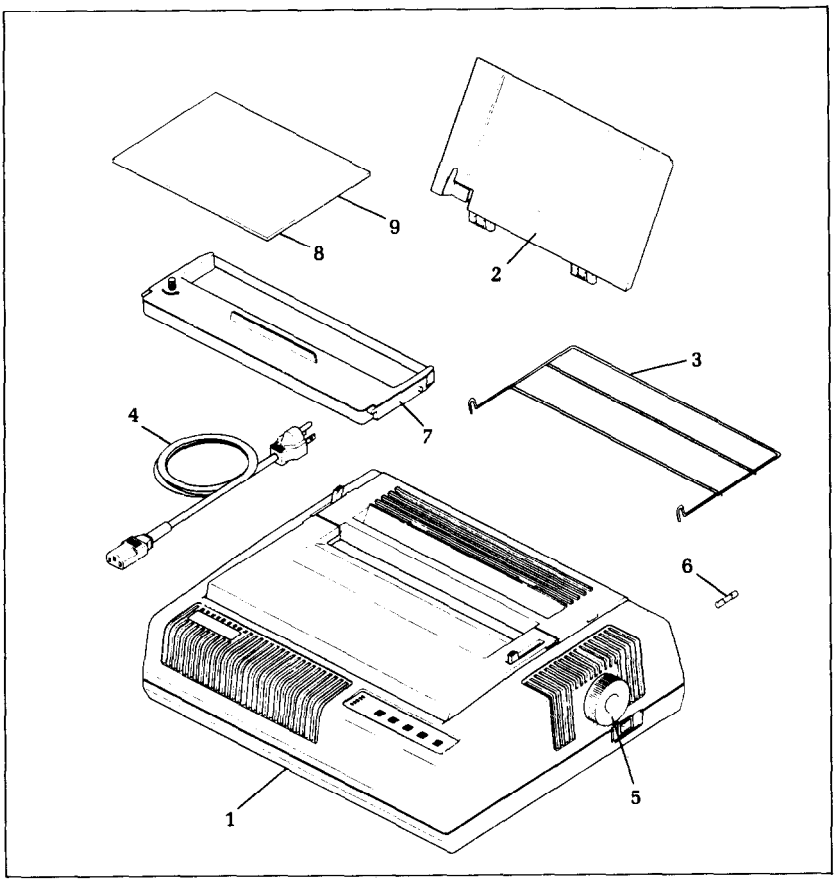


Figure 1-1. Inside the carton you should have received: 1) SR-10/15 printer, 2) cut sheet guide, 3) continuous paper guide, 4) power cord, 5) platen knob, 6) spare fuse, 7) ribbon cartridge, and 8) this user's manual.

Let's move on the next step . . .

■ Removing the printer covers

What are covers for, really? Primarily, for two reasons: one, to keep dust and dirt away from the delicate "innards," and two, to keep the noise level down. The front cover *must* be on or SR-10/15 will not print. So, you should keep the covers on all the time, except when setting the ink ribbon cartridge in place, loading paper, or making other adjustments when the cover might be in the way.

SR-10/15 has *two* covers, front and back. Both operate in the same way. To remove them, lift up the free end (nearest the center of the printer) so that the cover makes approximately a 45° angle

with the printer frame, then with a slight rocking motion, lift it straight up and off the machine. To replace, just reverse the procedure. Figure 1-2 illustrates the proper position and movement for both removal and replacement of the covers.

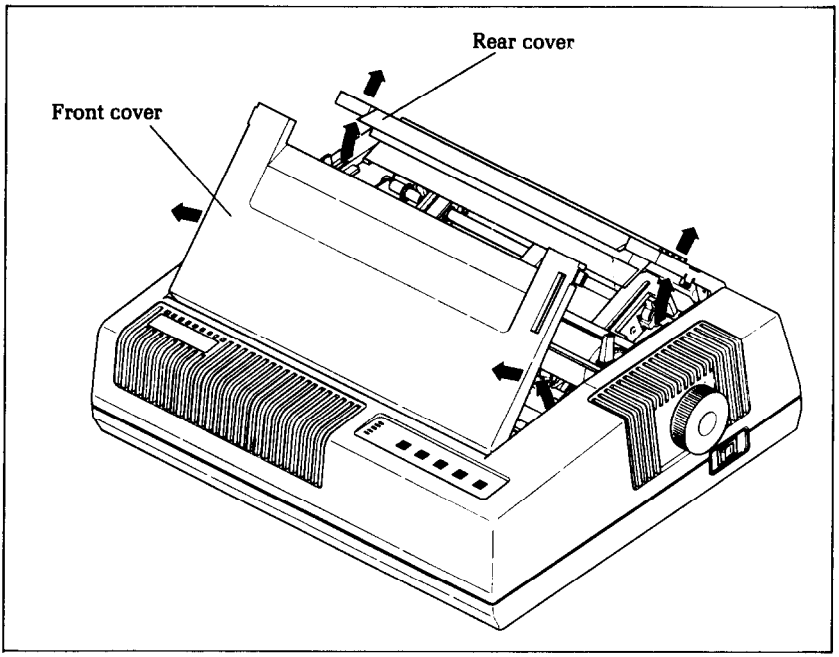


Figure 1-2. Remove the printer covers by tilting them up to about 45°, then lifting straight up.

■ Removing packing and shipping screws

There are three (on an SR-10) or four (on an SR-15) shipping screws on the bottom of the printer, used to hold the internal chassis securely to the external frame during shipping. To get at these, carefully place the printer upside down on a soft surface like a foam cushion. Remove the screws with a Phillips screwdriver as shown in Figure 1-3.

Next, remove the front cover, and remove the spiral tube on the carriage stay which protects the print head, per Figure 1-4.

You'll be smart to save these screws, along with the rest of the packing material and the shipping carton, in case you ever have to ship the printer. Tape the screws somewhere on the carton or packing.

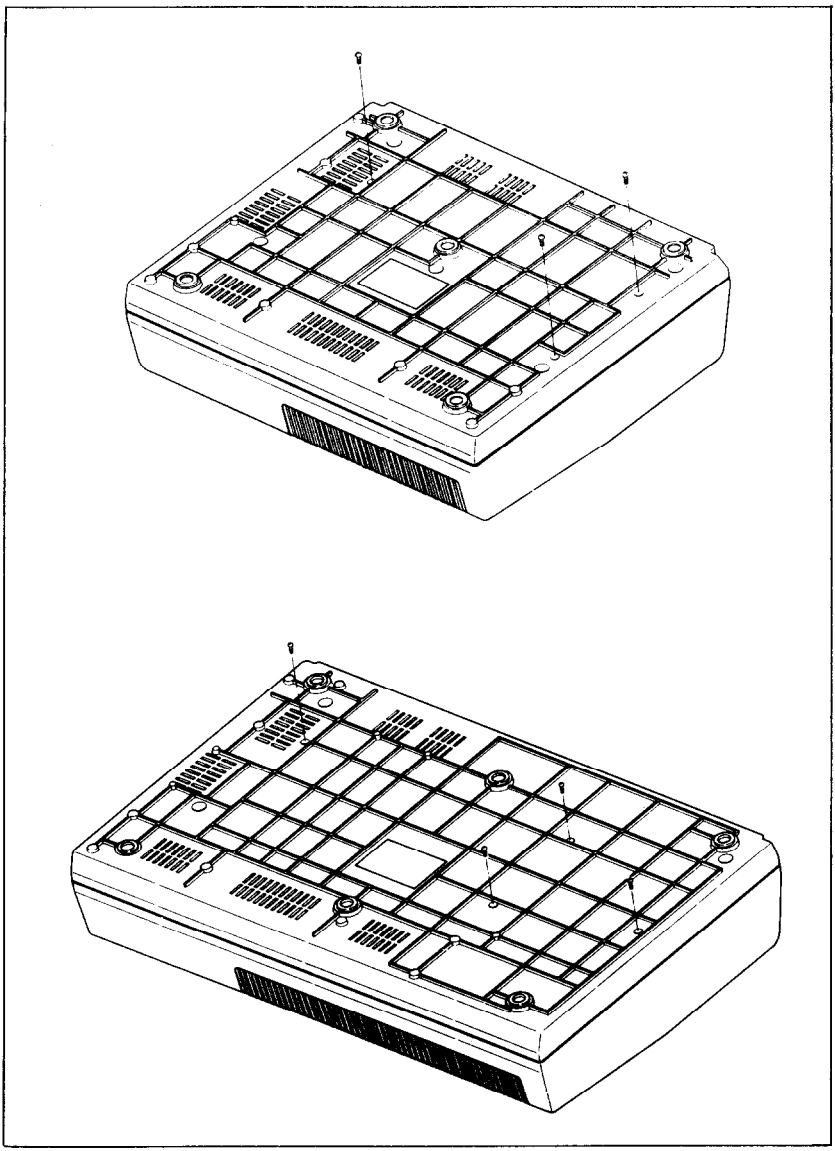


Figure 1-3. SR-10 has three screws which secure the chassis during shipping; SR-15 has four. They should be removed before use.

■ Installing the platen knob

This is the knob that turns the rubber platen cylinder. It fits into the hole on the right side of the printer case. Just match the odd-shaped hole in the knob with the same shape on the shaft you'll see inside the hole in the case, and press it on firmly. Give

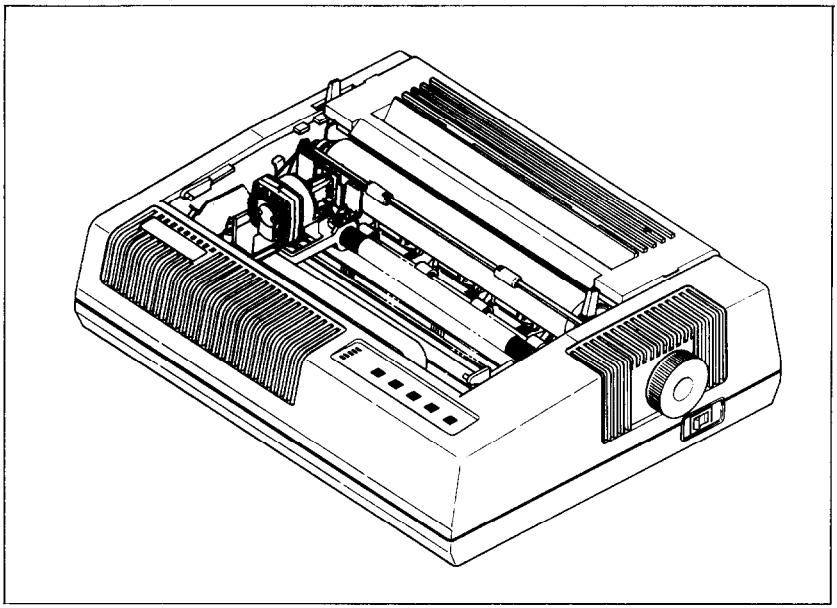


Figure 1-4. Remove the spiral tube on the carriage stay that protects SR-10/15's print head.

the knob a few turns to see that it's turning the platen easily and smoothly.

■ Installing the ribbon cartridge

The ribbon cartridge greatly simplifies installing the ink ribbon. For easy installation, though, it's wise to follow the sequence and diagrams shown here.

1. Turn the power switch *off* , and remove the front cover (as explained earlier.)
2. Slide the print head gently with your fingers to the approximate center of its pathway.
3. Note the position of the guide pins on the cartridge as shown in Figure 1-5. Then hold the cartridge at each end, with the ribbon facing away from you, and insert the guide pins into the cut-out hooks of the printer frame. You'll find this easier if you tilt the cartridge forward as you do this, as Figure 1-6 shows.
4. Using the guide pins as a fulcrum, lightly press the cartridge down until the two holder springs snap shut to hold the cartridge firmly in place.

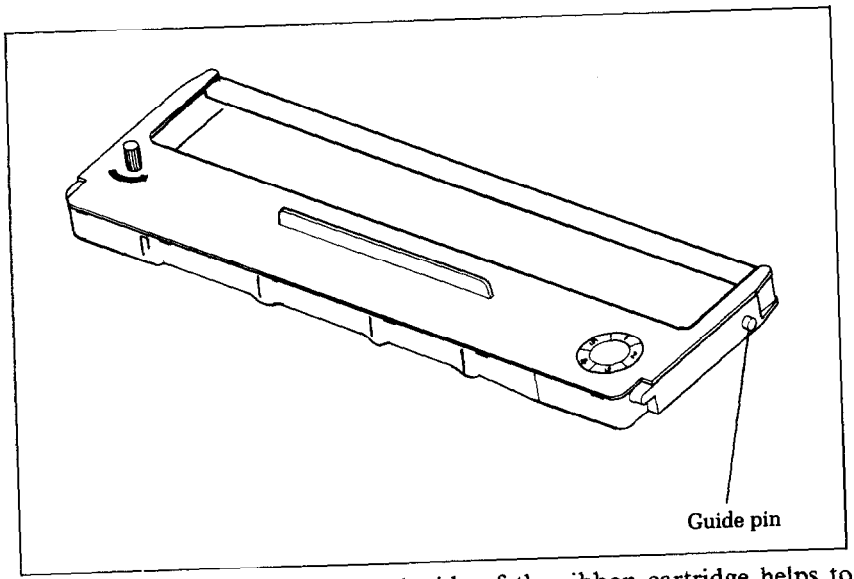


Figure 1-5. A guide pin on each side of the ribbon cartridge helps to align the cartridge during installation.

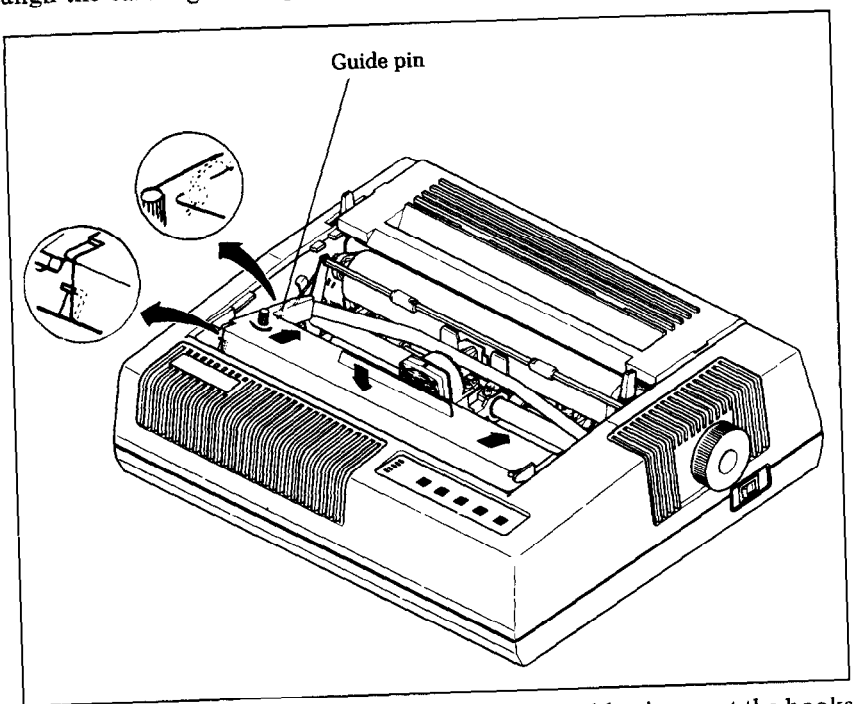


Figure 1-6. Tilt the ribbon cartridge in until the guide pins meet the hooks in the printer frame, then lower the front edge until the holder springs hold it in place.

5. Now thread the ribbon carefully between the print head and the ribbon guide next to the platen. (Take a good look at Figure 1-7.) You might want to use a ball point pen to lightly press the ribbon guide against the platen (runbber roller) while you insert the ribbon into the thin space between the print head and ribbon guide. ***Important:*** Center the ribbon vertically in the middle of the print head to avoid misprints or the ribbon coming off during printing.
6. Turn the spool gear knob in the direction of the arrow printed on the top left side of the cartridge to take up the slack in the ribbon; continue turning the spool gear four or five times to verify that everything is properly set and ready to roll.
7. As a final step, replace the front cover. As you'll learn in Chapter 2, SR-10/15 refuses to print unless the front cover is securely in place! A glowing, "pause" lamp warns of a loose cover. When this occurs, do the obvious thing: fasten the cover securely, press the pause button to douse the green light, and you're back in business!

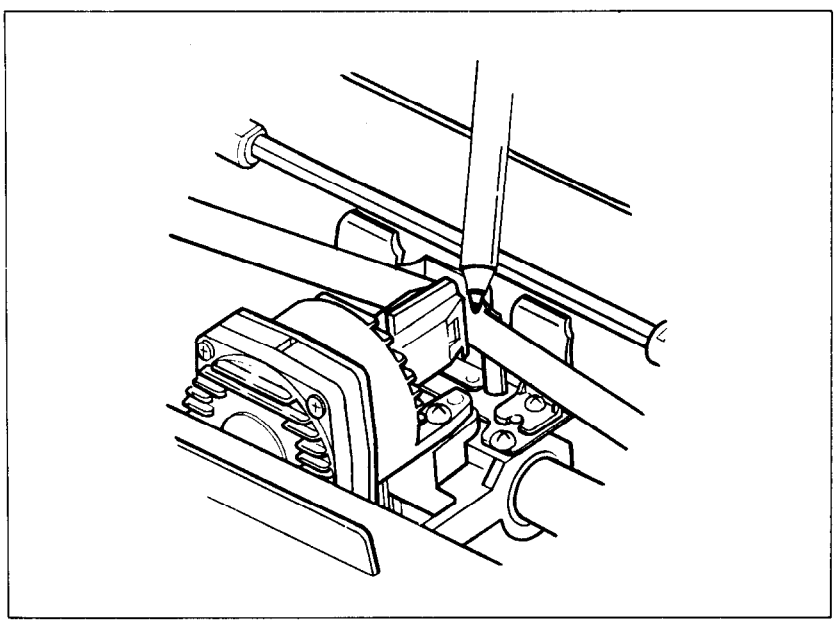


Figure 1-7. Use a ball point pen to place the ribbon between the print head and the ribbon guide. It's important that the ribbon is centered vertically between the print head and the ribbon guide.

CONNECTING SR-10/15 TO YOUR COMPUTER

To complete the installation, you'll need to connect SR-10/15 to your computer. Figure 1-8 shows where the cable connects, but there's more that you need to know. Find the appendix and follow the guidelines for making connections ("interfacing") and for setting the DIP switches. If you cannot connect to your computer, then your Star dealer will give you advice on connecting SR-10/15 to your computer.

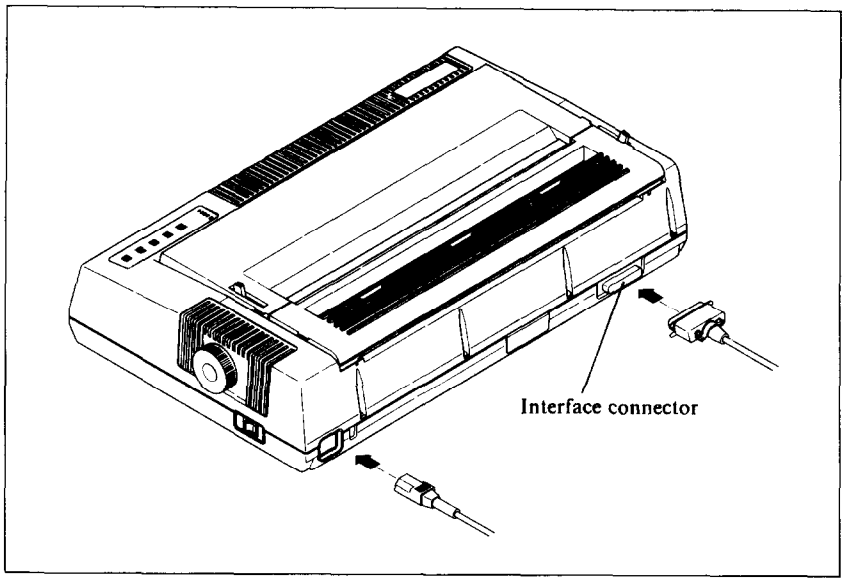


Figure 1-8. SR-10/15 has parallel interface as standard.

Then, in Chapter 2, you'll learn how to load paper (here's where you'll use the paper guides) and operate SR-10/15.

CHAPTER 2

GETTING TO KNOW YOUR SR-10/15

The more you learn about SR-10/15 and its sophisticated features, old and new, the better SR-10/15 is going to perform for you. Remember, it's not just what you know – it's what you know *how to use!* So, let's start getting acquainted!

Subjects we'll cover in this chapter include:

- **Components and controls**
- **Paper-out and front-cover-open detectors**
- **Paper selection and loading**
- **Adjusting the gap – for different paper thickness**
- **Self-test – printout of available characters**
- **Some tips for smoother operation**

COMPONENTS AND CONTROLS

First, the components. You saw most of these when you unpacked you printer. Now we'll give you a condensed rundown on what they do. (For details on your initial set-up of SR-10/15, with all components in place, see Chapter 1.)

PRINTER COVERS – There are two, front and rear. Their function is to protect the ribbon and print head from dust and dirt, and also to reduce the sound level.

SINGLE SHEET GUIDE – As you've guessed, this plastic rack is used to support and guide the single sheets during printing.

SPROCKET PAPER GUIDE – This wire rack serves the same function, but for sprocket paper.

INK RIBBON CARTRIDGE – A neat and tidy timesaver, which snaps into place within a few seconds.

POWER CORD – Connects the printer to its power source, usually a wall outlet. It's located at the right rear.

PRINT HEAD – This is the unit which does the actual printing.

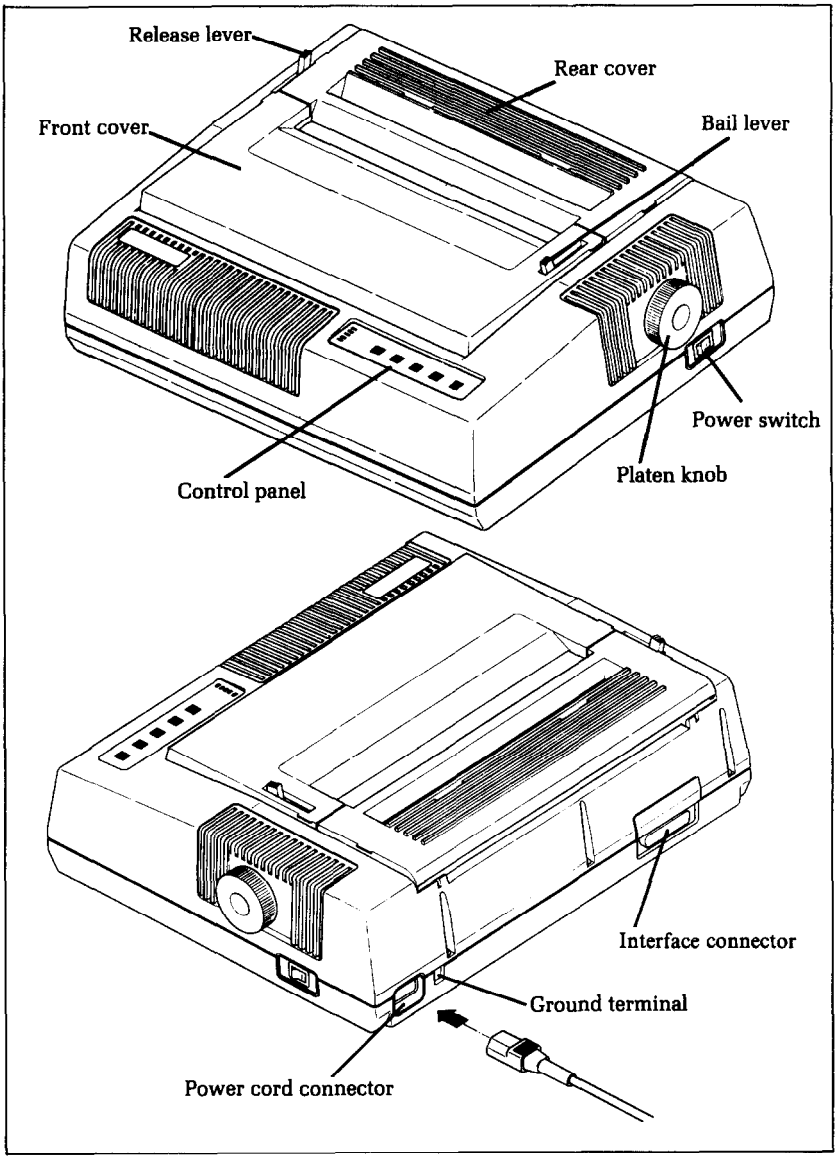


Figure 2-1. Front and rear views of SR-10.

Like a typewriter, the print head prints through an ink ribbon.
TRACTOR — This built-in unit sits in the rear of your printer, under the rear cover. Its sprocket wheels carry the sprocket-feed paper on its pathway through the printer.
PLATEN — This is the rubber cylinder that carries paper to the print head.

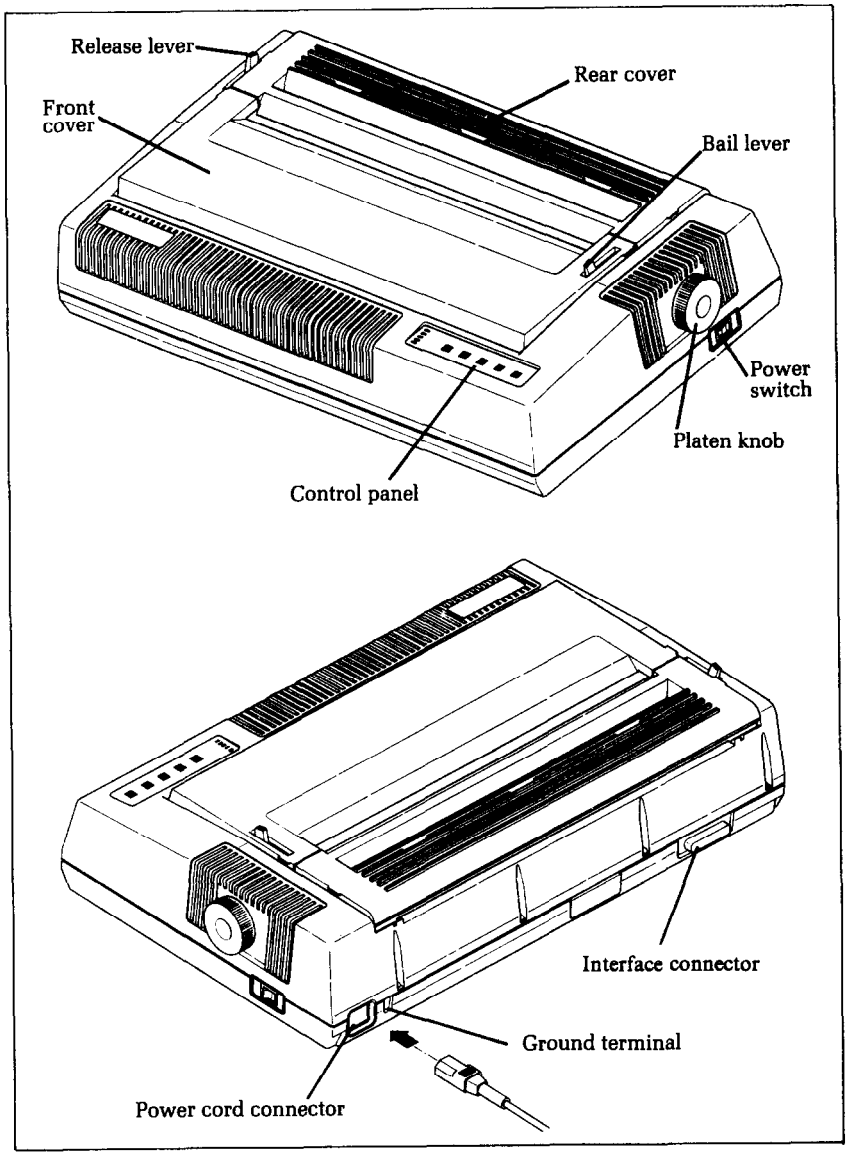


Figure 2-2. Front and rear views of SR-15.

INTERFACE CONNECTOR — Around on the back, this is the place where you connect your computer to SR-10/15, so that they are able to communicate with each other.

Now let's take a tour around the controls, starting with the control panel board, located at the right front. There are 5 lamps and 5 buttons on the panel:

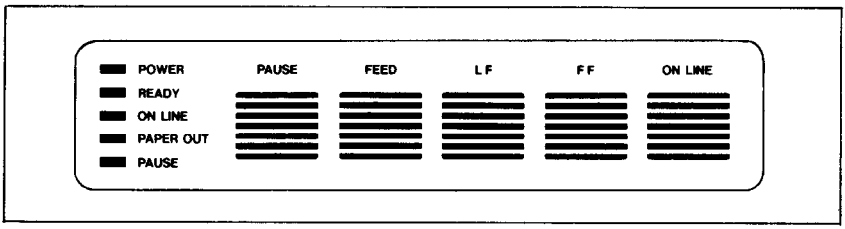


Figure 2-3. SR-10/15's controls.

POWER LAMP — Glows green when the power is *on.*
READY LAMP — Glows green when the printer is ready to accept data. This light flickers during transmission. Don't worry about the flicker; it's normal!
ON LINE LAMP — Glows green when the communication lines to your computer are open.
PAPER-OUT LAMP — Glows red when the printer is out of paper and stops printing. It works only when you're using sprocket paper.
PAUSE LAMP — A very important control! It glows green when the pause button has been pressed or when the front cover has been opened. When the pause lamp is on, you can feed paper with the LF, FF, or Feed buttons – but there's no printing possible. When the pause lamp is off, the printer will print – but you can't feed paper.
PAUSE BUTTON — Basically, this button allows you to change the printer status from "printing" to "not printing" or vice versa, with the results stated above under the Pause Lamp heading. This allows you to stop printing to advance the paper – a few lines or to the top of the next page.
FEED BUTTON — This is used for automatic feeding of single sheets, which is described in detail later in this chapter.
LF BUTTON — Stands for "Line Feed," and allows you to advance the paper one line at a time when the pause lamp is on. If you hold the button down, you'll get consecutive line feeds, one after the other.
FF BUTTON — Stands for "Form Feed." When you tap this button while the pause lamp is on, you advance the paper to the top of a new page or "form."
ON LINE BUTTON — Lets you change the printer status between "off line," and "on line". When it's on line, the printer can receive

data from the computer. When it's off line, the printer sends a signal to the computer indicating that it cannot accept data. When you turn the power switch *on,* you are automatically on line.

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

POWER SWITCH — Towards the back on the right side. This turns on the electricity to your machine.

PLATEN KNOB — Middle, right side. Lets you manually turn the platen, just like a typewriter. CAUTION: Turn this knob only with power switch *off.* Turning it with the power on could damage the platen drive gears.

RELEASE LEVER — On top, near the left rear corner. You'll be using this particular control often. What it does is control the pressure of the paper against the platen. Its position is crucial to feeding the different paper types — sprocket and single sheets. It has three settings: "Friction," "Set," and "Tractor." The first two are used for single sheet printing, and the Tractor position for sprocket paper. This will be fully explained in the section describing paper loading procedures.

BAIL LEVER — The bail is the movable bar that presses the paper against the platen during printing, and when moved away from the platen, allows the paper to reach its proper position during the loading operation. The lever which controls it is on the right side of the platen.

PAPER-OUT DETECTOR — This sensor automatically stops printing and tells you when the printer runs out of sprocket paper. The paper-out lamp glows red and a beep tone alerts you when the printer runs out of paper. The pause lamp also glows, so you are ready to load more paper. The lamp also glows if the release lever is not set in the tractor position for sprocket paper loading.

FRONT-COVER-OPEN DETECTOR — When the front cover is not fully closed, this magnetic detector causes the pause lamp to glow, and printing is interrupted (or won't begin). If this happens, printing may be re-started by securely closing the cover and pressing the pause button.

DIP SWITCHES — Primarily, these switches are used in interfacing SR-10/15 to your particular brand of computer. But there are also switches to set the power-on default settings for print style, and page size. See the appendix for a complete explanation.

PAPER SELECTION AND LOADING

Now we'll look at paper. Your SR-10/15 can handle single sheets – standard-size stationery, multi-part carbonless business forms, or almost any other kind of cut sheet. You can also print on "computer paper" with the holes along the sides, which is also called sprocket, punched, or perforated fan-fold. The loading procedures are quite different for single sheet and sprocket paper. We'll try to keep it short and sweet, but without sacrificing clarity and preciseness in our explanations.

■ Loading single sheets

Start with the proper paper. Paper width must be between 5½ and 8½ inches (5½ and 14½ inches for the SR-15), and paper thickness between .07 mm and .10mm (16 pound to 24 pound bond falls in this range). Loading is done automatically and instantly by pushing the Feed button. Here's the correct sequence.

1. Attach the single sheet guide to the printer (Figure 2-4).
2. To set the margin, use the little metal guide (shown in Figure 2-5) in one of its 3 positions.
3. Put the release lever in the "set" position. This step is very important for proper sheet alignment.
4. Putting the left edge of the sheet against the metal guide, insert a sheet into the paper chute until the bottom edge of the paper touches the paper stopper. (The set position of the release lever permits you to get the paper in straight.)
5. Now, push the release lever away from you to the "friction" position. This grips the paper securely for proper feeding.
6. Make sure that the bail is resting against the platen (you should push the bail lever away from the front of the printer). SR-10/15 will automatically lift it out of the way at the proper time!
7. With the power on, press the Feed button, and the paper *automatically* moves around the platen to the correct position to start printing, just one inch from the top edge of the sheet!

Note: If you'd like to start the first line of printing lower down on the sheet, as for letter correspondence for example, just press the Pause, then the LF (line feed) button to move the paper to the desired starting point. Hold down the LF button for multiple line feeds.

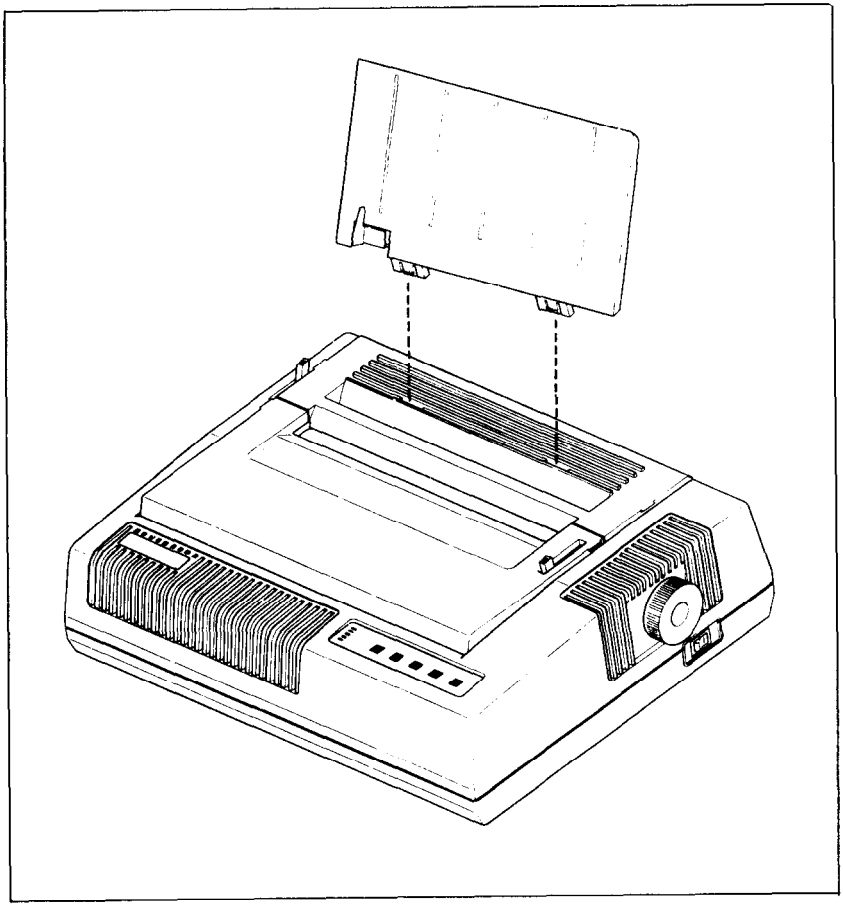


Figure 2-4. Use the single sheet guide for loading cut paper.

Table 2-1
Left margin on the single sheet guide

Position of Guide	Distance from Left-Hand Edge of Paper	
	For SR-10	**For SR-15**
Left	Approx. .6 inch	Approx. .8 inch
Middle	Approx. .3 inch	Approx. .5 inch
Right	Approx. .1 inch	Approx. .3 inch

■ Loading sprocket-feed paper

Continuous paper feeds into the printer from the rear. So, the paper should be stacked directly back of the printer, either on the same surface, if there's room, or on the floor.

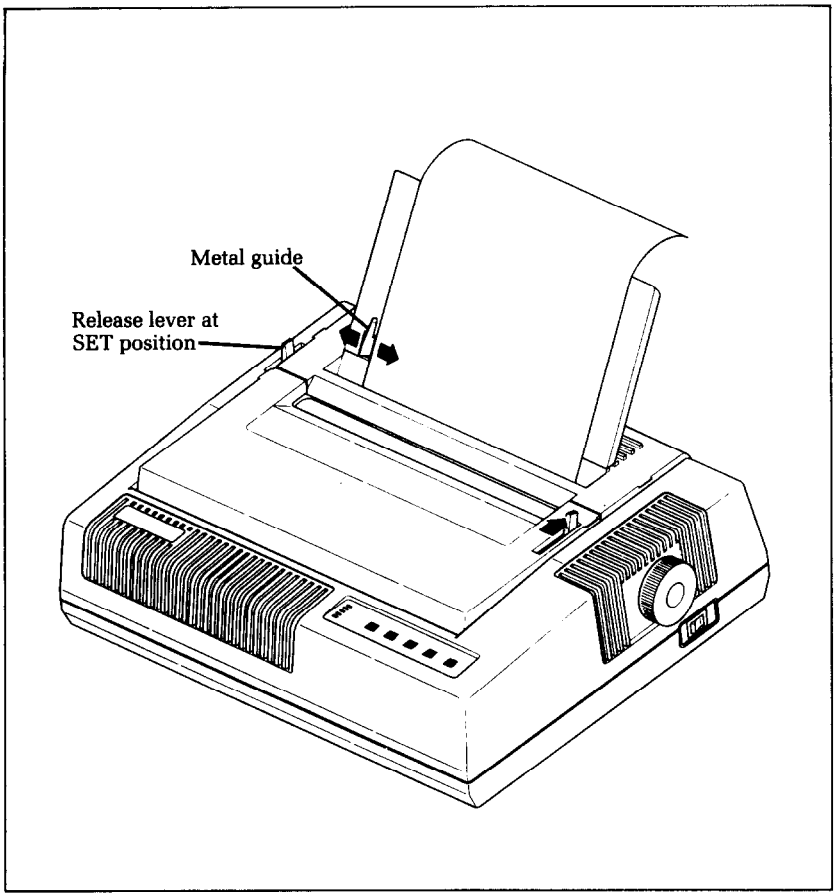


Figure 2-5. The metal guide is used to align the left margin.

Here's the proper sequence for loading:

1. Turn off the power and remove the rear cover. (After you've practiced a few times, you'll find it easy to load paper by just opening the cover.)
2. Attach the wire paper guide to the rear of the upper case, as shown in Figure 2-6.
3. Pull the release lever towards you to put it in the "tractor" position.
4. Pull the bail lever towards you to the open position.
5. Open the tractor covers, located on top of the left- and right-hand sprocket units (Figure 2-7).
6. Flip the sprocket clamp levers towards the rear. This unlocks the sprocket wheels to move left and right so you can align them with the holes in the paper.

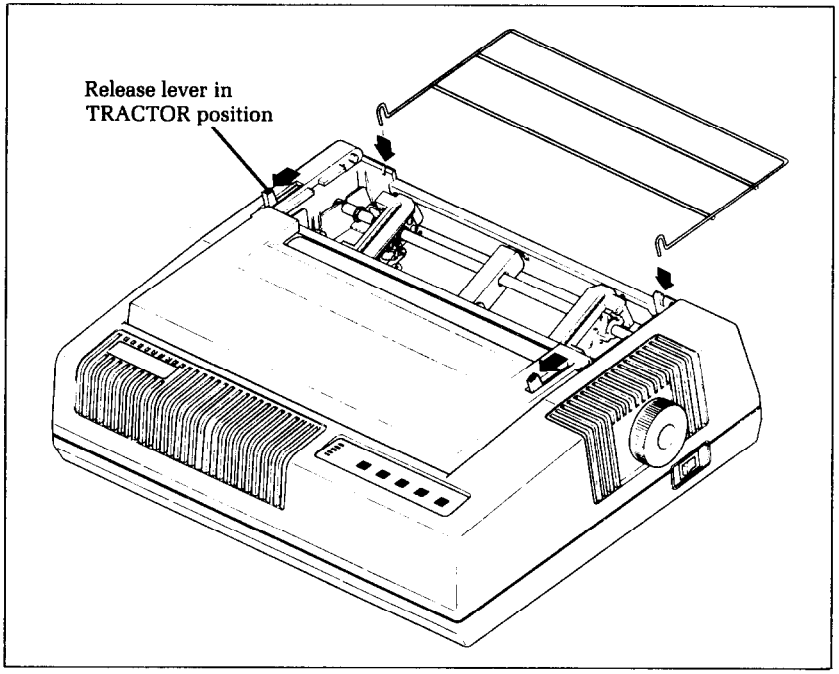


Figure 2-6. The wire paper guide keeps continuous paper away from the cables.

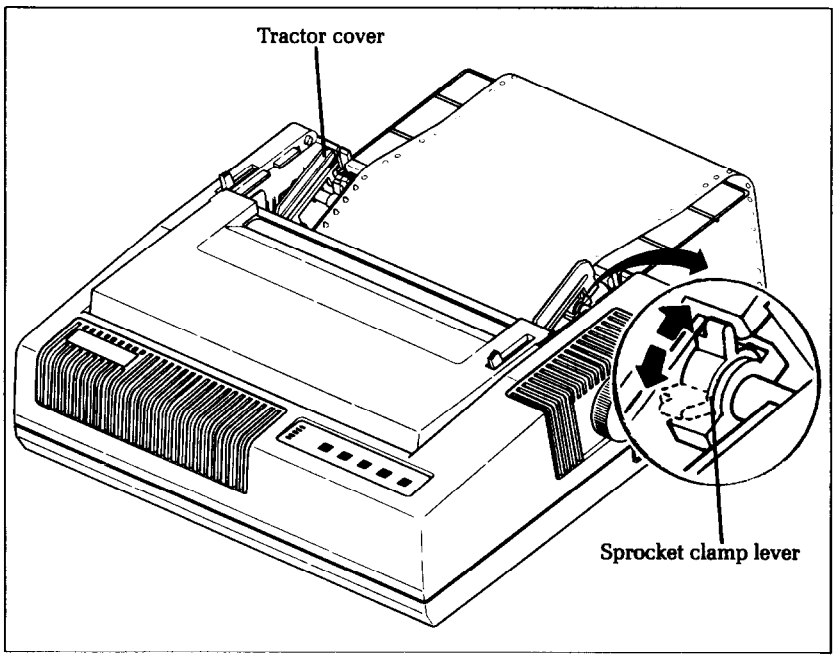


Figure 2-7. The tractors, which guide the paper, are underneath the rear cover.

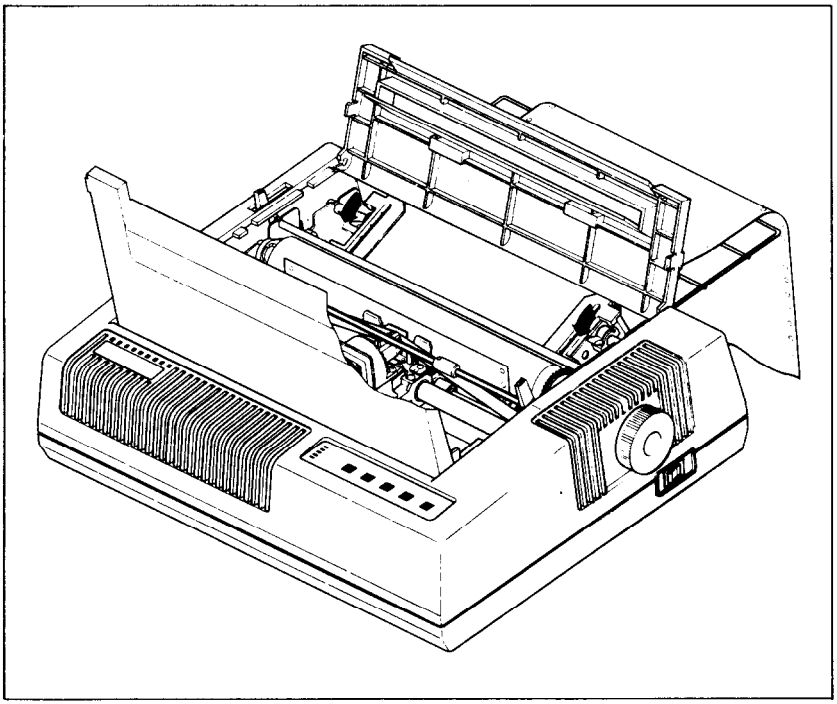


Figure 2-8. With the tractors in place, you're ready to close the covers and advance the paper.

7. Bring the paper up from the back, over the wire guide, and into the back of the printer. When the holes in the paper fit snugly over the nubby teeth in both sprockets, close the tractor covers and snap the clamp levers back into their locked positions (Figure 2-8).
8. Now we'll feed the paper around the platen *automatically*. To do this, close the rear cover, turn on the power, then push the Pause button and hold down the LF button until the paper moves smoothly into position.
9. Close the bail lever (push away from you). The top edge of the paper should line up with the cutter edge of the front cover so that printing will start one inch from the top edge.

RIBBON INSTALLATION

This is described in two places: installation of the ribbon cartridge is explained in Chapter 1; replacing the ink ribbon *inside* the ribbon cartridge casing is described in Chapter 11 ("Maintenance").

ADJUSTING THE GAP

The gap is the space between the print head and the platen. Adjusting the gap is simply adjusting the printer to accommodate different thicknesses of paper.

To make this adjustment, move the adjustment lever which is under the front cover, immediately in front of the release lever shown in Figure 2-9. Pulling the adjustment lever towards you will widen the gap; pushing it away from you will narrow the gap.

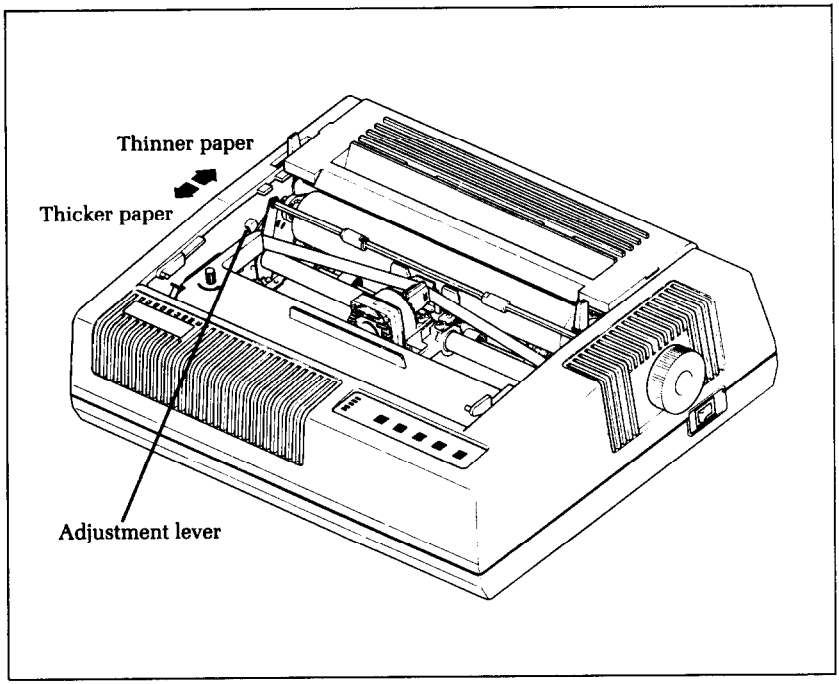


Figure 2-9. The adjustment lever allows for different thicknesses of paper.

Five positions are available; you can feel the lever clicking into the various notches. The second step (illustrated) is the one most commonly used for single sheets of paper. The lever is nearly straight up in this position.

You shouldn't encounter any difficulty in finding the right gap setting to fit your paper. If necessary, experiment; you'll soon find the best position for the paper you're using.

SELF-TEST

The "self-test" is a trial run of your beautiful new machine. SR-10/15 carries a built-in program that prints out sample lines of letters, numbers, and other characters — to show you that everything's in good working order. It also serves as a display of the characters available in the SR-10/15. And finally, it's a "warm-up" that permits you to check your installation of ribbon and paper, and the adjustment of the print head gap.

Best of all, you don't have to wait another minute — you can print the self-test without hooking up the SR-10/15 to your computer! It's as simple as 1, 2, 3...

1. Plug the printer's power cord into an electrical outlet.
2. Insert a sheet of paper (or sprocket paper, either one).
3. While holding down the LF button, turn the power switch on.

Were you surprised? It's speedy, isn't it? 200 characters a second, to be exact (when printing normal pica type).

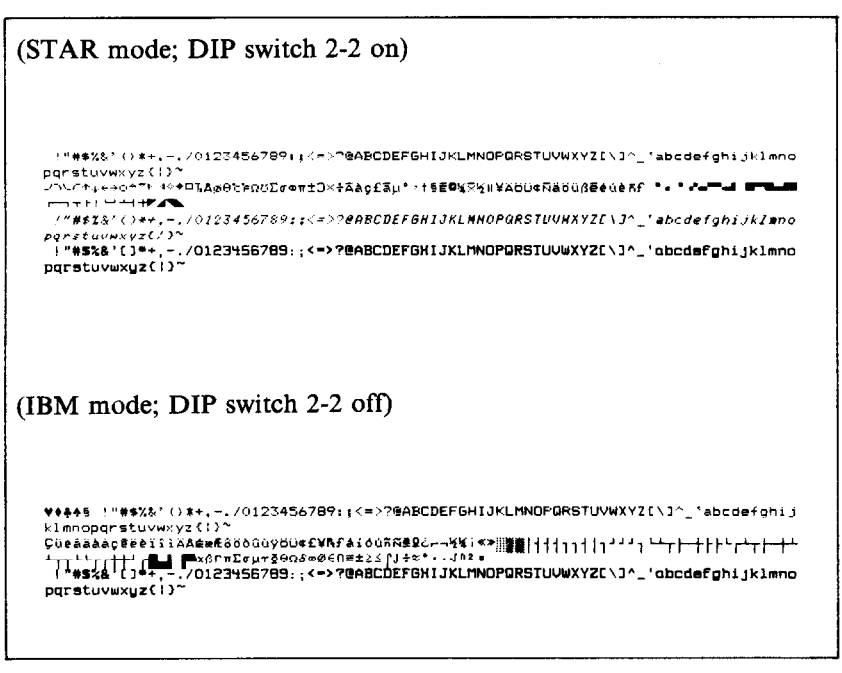


Figure 2-10. SR-10/15's self-test gives a preview of its capabilities.

SOME TIPS FOR SMOOTHER OPERATION

Here are some ideas that might save time and trouble with SR-10/15.

- When setting the left-hand margin on *sprocket* paper, you'll find the bail bar is marked with pica size unit measurements, so it's a handy reference. (There are 10 pica characters to the inch, so the markings 10, 20, 30 and so on also correspond exactly to inches, 1, 2, 3, etc.)
- The sprocket paper is perforated in page size units, to facilitate easy folding (that's the way it comes, in a stack). It is this edge that you should align with the front cover cutter edge so that printing will start just one inch below that point.
- When loading sprocket paper, never place the release lever in either the "set" or "friction" position. You'll know when this happens by the beep tone and the paper-out lamp glowing red. Use the "tractor" setting at all times when loading or running sprocket paper.
- When you use multi-layer paper, such as a 3-part carbonless form, you should adjust the print head gap to fit the greater paper thickness, as explained earlier in this chapter.
- If paper should jam when loading sprocket paper, it's usually because you forgot to put the bail lever in the open position (by pulling it towards you). Best thing to do then is to turn the power *off*, open the front cover, and roll the paper *backwards* by turning the platen knob.
- If the printing is faint, first check the thickness adjustment lever, then try a new ribbon. If it's *still* too faint, perhaps it's *finally* time for a new print head.

CHAPTER 3

GETTING STARTED WITH SR-10/15

In this chapter you'll learn about:
- **Using SR-10/15 with commercial software**
- **ASCII codes**

You have assembled and tested your printer, and seen a quick sample of SR-10/15 's capabilities in the self-test. Now it's time to do what you bought SR-10/15 to do: print information from your computer.

USING COMMERCIAL SOFTWARE

Many of you purchased SR-10/15 to use with commercial software. You made a good choice because SR-10/15 is compatible with most commercial programs, from word processing programs to spreadsheet programs to accounting programs.

Many of these programs have a routine for describing your printer. These routines are often in "installation programs". They typically give you a choice of printers or printer types to pick from. Some typical descriptions that you might pick for SR-10/15 are: "TTY type printer with backspace", "IBM-dot matrix printer", "Centronics-type printer", "Dot matrix ASCII printer". SR-10/15 should work fine with any of these descriptions.

Some printer lists are not very clear, and may not include anything that you think describes SR-10/15, If you can't decide which description best fits SR-10/15 we recommend that you narrow the list to two or three choices (you can quickly eliminate all the daisywheel printer types) and then experiment. You won't hurt anything if you guess wrong; it just won't work right. This should quickly tell you if your guess is right. If all else fails, though, your Star dealer will be happy to give you some advice.

Some programs don't ask you what kind of printer you have, but instead they ask some questions about what your printer can do. Here are the answers to the "most asked" questions. SR-10/15 *can* do a "backspace". SR-10/15 *can* do a "hardware form feed".

With these questions answered, you are ready to start printing. Read the manual that came with your commercial software to see how to make it send information for SR-10/15 to print. This is all you need to know to use SR-10/15 as a regular printer. But SR-10/15 isn't just a regular printer. SR-10/15 has many capabilities that your commercial software isn't aware of. A little later we will see what it takes to use some of SR-10/15's advanced features with commercial software.

■ First, some terminology

SR-10/15 knows what to print because it knows how to interpret the codes that the computer sends to it. These codes are numbers that the computer sends to SR-10/15. Both the computer and SR-10/15 know the meaning of these codes because they are a set of standard codes used by almost all microcomputers. This set of codes is the *American Standard Code for Information Interchange,* which is usually referred to as ASCII (pronounced *ask-key*). There are ASCII codes for all the letters of the alphabet, both lower case and capital, the numbers from 0 to 9, most punctuation marks, and some (but not all) of SR-10/15's functions.

ASCII codes are referred to in several different ways, depending on the way they are used. Some times these codes are treated as regular numbers. For example, the letter "A" is represented by the number 65 in ASCII. Appendix F shows all of the ASCII codes.

In BASIC, ASCII codes are used in the CHR$ function. This function is used to print the character that is represented by the number in the CHR$ function. The BASIC statement PRINT CHR$ (65) will print an "A" on the terminal.

In some other programming languages, ASCII codes are referred to by their *hex* value. "Hex" is short for *hexadecimal* which is a base-16 number system (our usual numbers are base-10). Since hex needs 16 digits, it uses the numbers 0 through 9 and then it uses the letters A through F for digits. The ASCII code for the letter "A" is 41 in hex.

Of course, most of the time we don't even need to think about this code system. Our computers are smart enough to know that

when we press the "A" key on our keyboard we want to print the letter "A". The computer takes care of all the rest.

But there are a number of ASCII codes that don't have keys on the keyboard. The most important of these codes are the codes that have ASCII values below 32. These codes control many of SR-10/15's functions. Even though there aren't keys for these codes, most keyboards can send these codes. It's done by holding down the "control" key (many times marked CTRL) and simultaneously pressing a letter key. The particular letter key that is pressed determines what code is sent. Control and A sends ASCII code 1, control and B sends ASCII code 2, and so on. Because of the way they are created, these codes are often referred to as "control-A" etc.

So there are four common ways of referring to the same set of codes: the character or name of the code, the decimal ASCII value, the hexadecimal ASCII value, and the "control-" value.

For example, the code that causes SR-10/15 to advance the paper one line is ASCII 10 (decimal). This code is commonly referred to by all the following names:

line feed	—	its name
< LF >	—	the abbreviation of its name
ASCII 10	—	its decimal value
ASCII 0AH	—	its hexadecimal value (the H signifies hex)
CHR$(10)	—	the way it's used in BASIC
control-J	—	the way you send it from a keyboard.

There's a chart in Appendix F that shows these side-by-side so that you can convert back and forth.

The reason that we are telling you all this about ASCII codes is that people are not very consistent about how they describe ASCII codes. We are going to help you use SR-10/15 with commercial software, but we don't know what its documentation is going to call the various codes. So if you know all the different things that the codes might be called, it will be easier to figure out what it is trying to tell you.

Now, armed with the knowledge of what to look for, you can delve into the manuals of your commercial software and dig out the secrets of how to send "control codes" to your printer. When you find the method that your program uses, then you can shop through this manual to find the function that you want to use.

By translating the codes from the system that we use, to the system that your commercial software uses, you should be able to use many of SR-10/15's advanced features. It may help, however if we look at a couple of examples.

■ The escape code

There's one particular ASCII code that we are going to be using more than all the rest. This is ASCII 27, which is called *escape*. With all of SR-10/15's advanced features, there weren't enough single ASCII codes to go around. So escape is used to start sequences of control codes that open a wider range of functions to us.

While you must call this code CHR$(27) in BASIC, we are going to refer to it as <ESC> in this book. This will make it much easier to recognize when we use it.

A typical *escape code sequence* starts with <ESC> which is followed by one or more codes. As an example, the escape code sequence to turn on emphasized print is:

```
<ESC> "E"
```

We'll learn more about these escape code sequences and how to use them in the chapters that follow.

■ Using this book without learning BASIC

Throughout the latter part of this book we will be teaching you how to use all of SR-10/15's features using the BASIC programming language in our examples. This is because it is easy to communicate with SR-10/15 from BASIC and because, despite its shortcomings, BASIC is the nearest thing to a universal language among users of personal computers. But it's not the only way to communicate with SR-10/15. Even if you don't know BASIC, you can learn how to use SR-10/15's features by reading on. When you find a function that you want to use, just apply what you already know about translating from one name for codes to another. The examples will still show you how the commands are used, even if you are not using BASIC.

CHAPTER 4

CONTROLLING SR-10/15 WITH BASIC

Throughout the rest of this book we will be teaching you how to use SR-10/15 's features using the BASIC programming language in our examples. It is easy to communicate with SR-10/15 from BASIC and, though it has its detractors, BASIC is the nearest thing to a universal language among users of personal computers. But remember that it's not the only way to communicate with SR-10/15, as we have already seen.

Subjects covered in this chapter include:

- **Listing BASIC programs on the printer**
- **Printing from BASIC**
- **CHR$ function**
- **Problem codes**
- **Command syntax used in this manual**
- **Selecting the right software mode**

All of the examples in this manual are written in Microsoft BASIC (specifically, Microsoft BASIC for the IBM Personal Computer). With minor modifications, the examples can be adapted to run in any version of BASIC. In this chapter, we'll tell you what modifications need to be made and how to do it. In this chapter we assume that you have some familiarity with BASIC.

SOME BASICS ABOUT BASIC

Probably the simplest thing to do with your printer in BASIC is to list a program on the printer. But in this world of proliferating microcomputers even this presents a problem. It seems that every computer uses a different system of communicating with the printer. We are going to tell you about some of the more common

ways, and hope that between this and your computer's BASIC manual you will be able to stay with us.

First on our list is Microsoft BASIC's way of communicating with the printer. They just add an "L" to the beginning of the LIST and PRINT commands, making them LLIST and LPRINT. This method is used by more computers than any other and so we will use it throughout this book, after telling the rest of you how to follow along.

Microsoft BASIC is used by TRS-80 computers, IBM-PC computers, many CP/M computers, and many other computers. (Look in your BASIC manual; it will probably say if it's Microsoft BASIC.)

Next we need to talk about Apple II computers. They have a real simple system. To list a program that you have loaded into memory, just type:

```
PR#1
LIST
PR#Ø
```

The PR#1 says "send everything to the printer," the LIST sends it, and the PR#0 says "OK, back to the screen now."

Some other computers require you to open the printer as a numbered device, and then direct the output to that device. For example, to list a program on the printer with a Commodore C-64 computer you type the following:

```
OPEN4,4
CMD4
LIST
CLOSE4
```

This says that the printer is device 4, directs the output to it, lists the program, and finally closes device 4.

The appendix gives more information about listing programs on various computers. Find the appendix that tells how your computer works, and try it.

Now that we all know how our computers address the printer, let's try listing a BASIC program. Load a BASIC program and

LLIST it (or however your computer does it). We've crossed the first major hurdle—learning how to list programs on SR-10/15. Now we are ready to jump into the world of programming with SR-10/15. But first, there are a few fundamentals that we need to cover.

■ Establishing communications

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

Let's try what we learned. Type the following:

```
NEW
1Ø LPRINT "TESTING"
RUN
```

Remember—we use LPRINT; you may have to use something else!

At any rate, you should have the word "TESTING" on your printer. Quite an achievement, isn't it? Let's get done with this simple stuff so that we can go on to something interesting.

■ The CHR$ function

We mentioned CHR$ in Chapter 3 as one way to express ASCII codes. We are going to use it a lot in communicating with SR-10/15. SR-10/15 uses many of the ASCII code that don't represent letters and numbers. The CHR$ function gives us an easy way to send these codes to the printer. Try this to see how the CHR$ function works:

```
NEW
1Ø LPRINT CHR$(83)
RUN
```

That should print an "S" for Star. If you check the chart in Appendix B you will see that 83 is the ASCII code for "S".

■ Control codes

SR-10/15 uses many of the non-printing ASCII codes for control codes. These codes perform a function rather than printing a character. Let's try an easy one right now:

```
NEW
1Ø LPRINT CHR$(7)
RUN
```

Where did that noise come from? That's SR-10/15's bell. We will learn more about it in Chapter 8. We just wanted to illustrate a code that causes SR-10/15 to perform a function.

■ The escape code

There's one ASCII code that we are going to be using more than all the rest. This is ASCII 27, which is called *escape*. In BASIC it is CHR$(27). With all of SR-10/15's advanced features, there weren't enough single ASCII codes to access all of them. So escape is used to start sequences of control codes that open a wider range of functions to us.

While you must call this code CHR$(27) in BASIC, we are going to refer to it as <ESC> in this book. This will make it much easier to recognize when we use it.

A typical escape code sequence starts with <ESC> which is followed by one or more CHR$ codes. As an example, the escape code sequence to turn on emphasized print is:

```
<ESC> CHR$(69)
```

In a program, this would look like this:

```
NEW
1Ø LPRINT CHR$(27) CHR$(69);
2Ø LPRINT "TESTING"
RUN
```

Try this program. It will print the word **TESTING** in emphasized print.

Some of you fast students may have noticed that CHR$(69) is the same as "E". That's right, the program will work just as well if line 10 is changed like this:

```
1Ø LPRINT CHR$(27) "E";
```

That's just another form of the same ASCII code, and it's all the same to SR-10/15.

Here's another shortcut for BASIC programmers: since <ESC> is used so often, assign it to a variable. In a long program, typing ESC$ is much easier than typing CHR$(27) each time! Now our program looks like this:

```
5 ESC$=CHR$(27)
1Ø LPRINT ESC$ "E";
```

Turn your printer off and back on now, or you will be printing in emphasized for quite a while!

■ Some problem codes

Before we go too far we need to mention some codes that may cause you problems. Like most of the subjects in this chapter, we have to be a little vague because of the differences in computers. Nearly all BASICs change some of the ASCII codes between your BASIC program and your printer. Some turn CHR$(10) (a line feed) into a CHR$(13) (a carriage return) before sending it on. Some other problem codes are 0, 7, and 9 through 13.

COMMAND SYNTAX USED IN THIS MANUAL

Because SR-10/15 users will be running such a wide variety of applications we just couldn't show the precise method of sending printer control codes to SR-10/15 for every one of them! Instead, as we introduce you to each command, we will show the commands like this example:

```
<ESC> "W" 1
```

This is the command to turn on expanded print <ESC>, as we mentioned earlier, is the way we will indicate the escape code, which is ASCII code 27.

A letter or number enclosed in quote marks (such as the "W" above) means that character should be sent to the printer (without the quote marks). In our example, you should send a capital W following the escape code. In BASIC, you could do this in a couple of ways: by sending the character itself (e.g. LPRINT "W";), or by using the CHR$ function to send the ASCII code for the character (e.g. LPRINT CHR$(87);).

Many of SR-10/15's commands end with a 1 or 0. When shown as in the above example (i.e. no quotes and no "CHR$"), you can use either ASCII code 1 (i.e.CHR$(1)) or the character "1" (which is ASCII code 49). The same idea applies to commands ending with 0.

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

```
LPRINT CHR$(27) "W" CHR$(1)
LPRINT CHR$(27) "W" CHR$(49)
LPRINT CHR$(27) "W1"
```

There are three commands that require the use of ASCII code 0; the character "0" (ASCII code 48) cannot be substituted. In these cases, instead of an unadorned 0 we will show CHR$(0) each time these commands are referenced. The commands are <ESC> "C" CHR$(0) n (set page length to n inches), <ESC> "D"... CHR$(0) (set horizontal tabs), and <ESC> "P"... CHR$(0) or <ESC> "B"... CHR$(0) (set vertical tabs).

There are other non-printing codes that are used (such as ASCII code 15, which is used to turn on condensed pitch). These commands will be introduced using the BASIC CHR$ function (e.g.CHR$(15)).

SELECTING THE RIGHT SOFTWARE MODE

For SR-10/15 to correctly respond to control codes, you must make sure that you've set its DIP switch properly. If you're using the IBM Personal Computer, you should put SR-10/15 in IBM mode. If you're using other computer, you should put SR-10/15

in STAR mode. There are some cases where you would use IBM mode with your computer, if you wish.

Chapters 5 through 10 discuss the control codes you send to SR-10/15 to control printing style, horizontal and vertical formatting, graphics, and other features. You'll find that many features have different codes for IBM mode and STAR mode.

That's it for the basics. You are ready to learn how to use the many features of SR-10/15.

CHAPTER 5

PRINTING TEXT WITH SR-10/15

Beginning with this chapter we will be exploring all the features of SR-10/15.

In this chapter we'll cover:

- **Near letter quality characters**
- **Italics**
- **Underlining**
- **Superscript and subscripts**
- **Print pitch**
- **Print emphasis**

All our examples will be given in Microsoft BASIC as used by the IBM Personal Computer, but remember that you don't need to know BASIC to use SR-10/15's features. Just use the same ASCII codes as we do in our examples.

You have already printed a few lines on your SR-10/15 printer. Now it's time to start looking at the many variations of printing style that you have available to you.

SOME SPECIAL KINDS OF TEXT

■ Near Letter Quality characters

SR-10/15's Near Letter Quality (sometimes abbreviated as NLQ) character set is ideal for correspondence and other important printing, for it takes a keen eye to detect that it is from a dot matrix printer. Normally (unless you have turned DIP switch 1-4 off), SR-10/15 prints draft quality characters.This is adequate for most work and it prints fastest. But for the final printout, try NLQ. The program below shows how.

```
1Ø 'Demo near letter quality character set.
2Ø LPRINT CHR$(27) "B" CHR$(4)  ; 'Select NLQ.
3Ø LPRINT "This line shows NEAR LETTER QUALITY!"
4Ø LPRINT CHR$(27) "B" CHR$(5)  ; 'Select draft.
5Ø LPRINT "This line shows standard print."
```

In this program, line 20 selects NLQ characters with the < ESC > "B" CHR$(4) command. Line 30 prints a sample before line 40 switchs SR-10/15 back to draft printing with an < ESC > "B" CHR$(5). When you run the program you should get this:

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

If you are using with the IBM mode, change the following lines to the program given above.

```
2Ø LPRINT CHR$(27) "4"  ; 'Select NLQ.
4Ø LPRINT CHR$(27) "5"  ; 'Select draft.
```

Table 5-1
Near letter quality commands

Function	Mode	Control code
Near letter quality ON	STAR	< ESC > "B" CHR$(4)
	IBM	< ESC > "4"
Near letter quality OFF	STAR	< ESC > "B" CHR$(5)
	IBM	< ESC > "5"

■ Italic printing

Italic letters are letters that are slanted to the right. SR-10/15 can print all of its letters except NLQ characters in *italic* as well as the roman (standerd) letters you are accustomed to. Italics can be used to give extra emphasis to certain words. The command codes to turn italic on and off are shown in Table 5-2.

Table 5-2
Italic commands

Function	Mode	Control code
Italic ON	STAR	< ESC > "4"
	IBM	< ESC > "I" 1
Italic OFF	STAR	< ESC > "5"
	IBM	< ESC > "I" 0

Use this program with STAR mode to see italic characters:

```
1Ø 'Demo italic and roman.
2Ø LPRINT CHR$(27) "4"  ; 'Italic on.
3Ø LPRINT "This line is in ITALIC characters."
4Ø LPRINT CHR$(27) "5"  ; 'Italic off.
5Ø LPRINT "This line is in ROMAN (normal) characters."
```

Here is what you should get:

```
This line is in ITALIC characters.
This line is in ROMAN (normal) characters.
```

This program is easy; line 20 turns italic on with <ESC> "4", and line 40 turns it off with <ESC> "5".

■ Underlining

Not only can SR-10/15 print all styles ot printing in both roman and italic, but it can underline them too. The control codes are shown in Table 5-3.

Table 5-3
Underline commands

Function	Mode	Control code
Underline ON	STAR	<ESC> "–" 1
	IBM	<ESC> "–" 1
Underline OFF	STAR	<ESC> "–" 0
	IBM	<ESC> "–" 0

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

```
1Ø 'Demo underlining.
2Ø LPRINT CHR$(27) "-" CHR$(1) ; 'Underline on.
3Ø LPRINT "This phrase is UNDERLINED;" ;
4Ø LPRINT CHR$(27) "-" CHR$(Ø) ; 'Underline off.
5Ø LPRINT " this is not."
```

It should come out like this:

```
This phrase is UNDERLINED; this is not.
```

In this program underline is turned on in line 20 with <ESC> "–" CHR$(1), and then off in line 40 with <ESC> "–" CHR$(0). There's a new little wrinkle in this program, though. It all printed on one line. The semicolons at the end of the first three lines told BASIC that those lines were to be contin-

ued.Therefore, BASIC didn't send a carriage return and line feed at the end of those lines. We just did this to illustrate that all these control codes can be used in the middle of a line. It's easy to underline or *italicize* only part of a line.

■ Superscripts and subscripts

SR-10/15 can print in two different heights of characters. The smaller characters are called *superscripts* and *subscripts* and are half the height of normal characters. *Superscripts* print even with the tops of regular printing while *subscripts* print even with the bottom of regular printing. They are frequently used to reference footnotes, and in mathematical formulas.

Table 5-4 has the codes for using superscripts and subscripts.

Table 5-4
Superscript and subscript commands

Function	Mode	Control code
Superscript ON	STAR	<ESC> "S" 0
	IBM	<ESC> "S" 0
Subscript ON	STAR	<ESC> "S" 1
	IBM	<ESC> "S" 1
Super & subscript OFF	STAR	<ESC> "T"
	IBM	<ESC> "T"

Try this program to see them work:

```
10 'Demo subscripts and superscripts.
20 LPRINT "Look! " ;
30 LPRINT CHR$(27) "S" CHR$(0) ; 'Superscript on.
40 LPRINT "Superscripts " ;
50 LPRINT CHR$(27) "T" ; 'Cancel superscripts.
60 LPRINT "& " ;
70 LPRINT CHR$(27) "S" CHR$(1) ; 'Subscripts on.
80 LPRINT "subscripts " ;
90 LPRINT CHR$(27) "T" ; 'Cancel subscripts.
100 LPRINT "on one line."
```

```
Look! Superscripts & subscripts on one line.
```

Here line 30 turns on superscripts with <ESC> "S" CHR$(0). It's turned off in line 50 with <ESC> "T". Then, between printing text, subscripts are turned on in line 70 with <ESC>

"S" CHR$(1), and finally off in line 90. Again, everything prints on one line because of the semicolons.

CHANGING THE PRINT PITCH

In "printer talk," character width is called *pitch.* Normally, SR-10/15 prints 10 characters per inch. This is called *pica* pitch because it's the same spacing as a standard pica typewriter.

SR-10/15 can also print 12 characters per inch. This is called *elite* pitch because it is the same spacing as an elite typewriter.

Condensed print is approximately 17 characters per inch. Condensed pitch allows you to get 136 columns of printing on an 8½ inch page.

Proportional spacing provides an alternative to the block-style output of a defined pitch. It moves its print head only as far as each character needs. Thus, the print head moves further for "M"s and "W"s than for "l"s and "i"s.

The table below shows four options of this command.

Table 5-5
Print pitch commands

Pitch	Characters/inch	Mode	Control code
Pica	10	STAR	<ESC> "B" CHR$(1) or CHR$(18)
		IBM	<ESC> "P" or CHR$(18)
Elite	12	STAR	<ESC> "B" CHR$(2)
		IBM	<ESC> "M"
Condensed	17	STAR	<ESC> "B" CHR$(3) or CHR$(15)
		IBM	CHR$(15)
Proportional ON		STAR	<ESC> "p" 1
		IBM	<ESC> "p" 1
Proportional OFF		STAR	<ESC> "p" 0
		IBM	<ESC> "p" 0

Let's see how these four pitches look. Try this program with STAR mode:

```
1Ø 'Demo all pitches.
2Ø LPRINT CHR$(27) "B" CHR$(3) ; 'Select condensed
   pitch.
```

```
3Ø LPRINT "This line is CONDENSED pitch."
4Ø LPRINT CHR$(27) "B" CHR$(2) ; 'Select elite pitch.
5Ø LPRINT "This line is ELITE pitch."
6Ø LPRINT CHR$(27) "p" CHR$(1) ; 'Select proportional.
7Ø LPRINT "This line is PROPORTIONAL spacing."
8Ø LPRINT CHR$(27) "p" CHR$(Ø) ; 'Cancel proportional.
9Ø LPRINT CHR$(27) "B" CHR$(1) ; 'Select pica pitch.
1ØØ LPRINT "This line is PICA pitch (normal)."
```

When you run this program you should get this:

```
This line is CONDENSED pitch.
This line is ELITE pitch.
This line is PROPORTIONAL spacing.
This line is PICA pitch (normal).
```

Line 20 turns on condensed pitch with < ESC > "B" CHR$(3). Line 30 prints a line at 17 characters per inch. The < ESC > "B" CHR$(2) in line 40 changes SR-10/15 to elite pitch and line 50 prints a line in elite pitch. Line 60 turns on proportional spacing with < ESC > "p" CHR$(1), and line 70 prints a line with proportional spacing. Line 80 and line 90 reset SR-10/15 to pica pitch and line 100 prints a line in pica pitch.

Pica pitch and condensed pitch can be set with "shortcut" codes. Instead of using < ESC > "B" CHR$(*n*), you can set them with a single code. CHR$(18) sets pica pitch and CHR$(15) sets condensed pitch. You can not set elite pitch with a single code.

■ Expanded print

Each of SR-10/15's four print pitches can be enlarged to twice its normal width. This is called expanded print.Try this program to see how it works:

```
1Ø 'Demo expanded mode.
2Ø LPRINT "Demonstration of " ;
3Ø LPRINT CHR$(14) ; 'Expanded mode on.
4Ø LPRINT "EXPANDED" ;
5Ø LPRINT CHR$(2Ø) ; 'Expanded mode off.
6Ø LPRINT " printing."
7Ø LPRINT "Notice that " ;
8Ø LPRINT CHR$(14) ; 'Expanded mode on.
9Ø LPRINT "EXPANDED mode"
1ØØ LPRINT "automatically turns off at end of a line."
```

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

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

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

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

```
10 'Demo permanent expanded mode
20 LPRINT CHR$(27) "W" CHR$(1) ; 'Expanded mode on
   permanently.
30 LPRINT "Permanent expanded"
40 LPRINT "mode stays on until"
50 LPRINT "it is " ;
60 LPRINT CHR$(27) "W" CHR$(0) ; 'Expanded mode off.
70 LPRINT "turned off."
```

Now the results look like this:

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

When you turn on expanded print with < ESC > "W" CHR$(1) it stays on until you turn it off with < ESC > "W" CHR$(0).

Table 5-6
Expanded print commands

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

By combining expanded print with the four pitches, SR-10/15 has eight different character widths available.

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

```
1Ø 'Demo pitches in combination with expanded mode.
2Ø LPRINT CHR$(27) "W" CHR$(1) ; 'Permanent expanded
   mode on.
3Ø LPRINT CHR$(27) "B" CHR$(3) ; 'Select condensed
   pitch.
4Ø LPRINT "This line is EXPANDED CONDENSED pitch."
5Ø LPRINT CHR$(27) "B" CHR$(2) ; 'Select elite pitch.
6Ø LPRINT "This is EXPANDED ELITE."
7Ø LPRINT CHR$(27) "B" CHR$(1) ; 'Select pica pitch.
8Ø LPRINT "This is EXPANDED PICA."
9Ø LPRINT CHR$(27) "p" CHR$(1) ; 'Select proportional
1ØØ LPRINT "This is EXP. PROPORTIONAL."
11Ø LPRINT CHR$(27) "p" CHR$(Ø) ; 'Cancel proportional.
12Ø LPRINT CHR$(27) "W" CHR$(Ø) ; 'Permanent expanded
    mode off.
13Ø LPRINT "This is UNEXPANDED PICA pitch (default)."
```

If you are using with the IBM mode, change the following lines to the program given above.

```
3Ø LPRINT CHR$(27) CHR$(15) ; 'Select condensed pitch.
5Ø LPRINT CHR$(27) "M" ; 'Select elite pitch.
7Ø LPRINT CHR$(27) "P" ; 'Select pica pitch.
```

Here's what you should get from this program:

```
This line is EXPANDED CONDENSED pitch.
This is EXPANDED ELETE.
This is EXPANDED PICA.
This is EXP. PROPORTIONAL.
This is UNEXPANDED PICA pitch (default).
```

In addition, the NLQ characters can be printed with expanded print as shown below.

```
This is normal pica.
This is normal NLQ.
This is EXPANDED NLQ.
This is EXPANDED pica.
```

MAKING SR-10/15 PRINT DARKER

SR-10/15 has very good print density when it's just printing regularly. But sometimes you may want something to stand out from the rest of the page. SR-10/15 provides two ways to do this: double-strike and emphasized print. Both of these go over the characters twice, but they use slightly different methods to darken the characters. Let's try them and see what the difference is.

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

Table 5-7
Print emphasis commands

Function	Mode	Control code
Double-strike ON	STAR	<ESC> "G"
	IBM	<ESC> "G"
Double-strike OFF	STAR	<ESC> "H"
	IBM	<ESC> "H"
Emphasized ON	STAR	<ESC> "E"
	IBM	<ESC> "E"
Emphasized OFF	STAR	<ESC> "F"
	IBM	<ESC> "F"

Try them now with this little program:

```
1Ø 'Demo double-strike and emphasized.
2Ø LPRINT CHR$(27) "G" ; 'Double strike on.
3Ø LPRINT "This line is DOUBLE-STRIKE printing."
4Ø LPRINT CHR$(27) "E" ; 'Emphasized on.
5Ø LPRINT "This line is DOUBLE-STRIKE and EMPHASIZED."
6Ø LPRINT CHR$(27) "H" ; 'Double strike off.
7Ø LPRINT "This line is EMPHASIZED printing."
8Ø LPRINT CHR$(27) "F" ; 'Emphasized off.
9Ø LPRINT "This line is normal printing."
```

Run this program. The results will look like this:

```
This line is DOUBLE-STRIKE printing.
This line is DOUBLE-STRIKE and EMPHASIZED.
This line is EMPHASIZED printing.
This line is normal printing.
```

Line 20 turns on double-strike with <ESC> "G" and line 30 prints a line of text. In line 40 emphasized is turned on with <ESC> "E". Line 50 prints a line of text in double-strike *and* emphasized. Line 60 then turns double-strike off with <ESC> "H" so that line 70 can print in emphasized only. Finally, line 80 turns emphasized off, so that SR-10/15 is set for normal printing.

Look closely at the different lines of printing. In the line of double-strike printing each character has been printed twice, and they are moved down just slightly the second time they are printed. In emphasized printing, they are moved slightly to the right the second time SR-10/15 prints. The last line combined both of these so that each character was printed 4 times. Now that's pretty nice printing, isn't it?

MIXING MODES

We have learned how to use SR-10/15's many different printing modes individually.

Star's engineers have given a unique control command that lets you choose at will between any of different printing styles. This command is called Master Select. The Master Select command consists of <ESC> "?" followed by a single ASCII code. (At IBM mode, use "!" instead of "?".) The value of the ASCII code determines the printing style that is selected, as shown the table below.

Let's see how this master select looks. Try this program with STAR mode:

```
1Ø 'Demo master select print
2Ø LPRINT CHR$(27) "?" CHR$(16) ;
3Ø LPRINT "This line is DOUBLE-STRIKE printing."
4Ø LPRINT CHR$(27) "?" CHR$(24) ;
5Ø LPRINT "This line is DOUBLE-STRIKE and EMPHASIZED."
6Ø LPRINT CHR$(27) "?" CHR$(8)  ;
7Ø LPRINT "This line is EMPHASIZED printing."
8Ø LPRINT CHR$(27) "?" CHR$(2)  ;
9Ø LPRINT "This line is normal printing."
```

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

Table 5-8
Master Select and the 256 ASCII Codes

Pitch	Normal	Emphasized	Double-strike	Double-strike & Emphasized
Pica	0, 2, 64, 66, 128, 130, 192, 194	8, 10, 12, 14, 72, 74, 76, 78, 136, 138, 140, 142, 200, 202, 204, 206	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	1, 3, 5, 7, 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.
Condensed	4, 6, 68, 70, 132, 134, 196, 198	Emphasized takes precedence over condensed.	20, 22, 84, 86, 148, 150, 212, 214	Emphasized takes precedence over condensed.
Expanded Pica	32, 34, 96, 98, 160, 162, 224, 226	40, 42, 44, 46, 104, 106, 108, 110, 168, 170, 172, 174, 232, 234, 236, 238	48, 50, 112, 114, 176, 178, 240, 242	56, 58, 60, 62, 120, 122, 124, 126, 184, 186, 188, 190, 248, 250, 252, 254
Expanded Elite	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 Condensed	36, 38, 100, 102, 164, 166, 228, 230	Emphasized takes precedence over condensed.	52, 54, 116, 118, 180, 182, 244, 246	Emphasized takes precedence over condensed.

SUMMARY

Control code	Function
<ESC> "B" CHR$(4)	Near letter quality on (for STAR mode)
<ESC> "B" CHR$(5)	Near letter quality off (for STAR mode)
<ESC> "4"	Near letter quality on (for IBM mode)
<ESC> "5"	Near letter quality off (for IBM mode)
<ESC> "4"	Italic on (for STAR mode)
<ESC> "5"	Italic off (for STAR mode)
<ESC> "I" 1	Italic on (for IBM mode)
<ESC> "I" 0	Italic off (for IBM mode)
<ESC> "–" 1	Underline on
<ESC> "–" 0	Underline off
<ESC> "S" 0	Superscript on
<ESC> "S" 1	Subscript on
<ESC> "T"	Super & subscript off
<ESC> "B" CHR$(1)	Sets pica pitch (for STAR mode)
<ESC> "P"	Sets pica pitch (for IBM mode)
<ESC> "B" CHR$(2)	Sets elite pitch (for STAR mode)
<ESC> "M"	Sets elite pitch (for IBM mode)
<ESC> "B" CHR$(3)	Sets condensed pitch (for STAR mode)
<ESC> "p" 1	Proportional on
<ESC> "p" 0	Proportional off
CHR$(18)	Sets pica pitch
CHR$(15)	Sets condensed pitch
CHR$(14)	One line expanded
<ESC> CHR$(14)	One line expanded
CHR$(20)	One line expanded off
<ESC> "W" 1	Expanded on
<ESC> "W" 0	Expanded off
<ESC> "G"	Double-strike on
<ESC> "H"	Double-strike off
<ESC> "E"	Emphasized on
<ESC> "F"	Emphasized off
<ESC> "?" *n*	Master select (for STAR mode)
<ESC> "!" *n*	Master select (for IBM mode)

CHAPTER 6

LINE SPACING AND FORMS CONTROL

We have learned how to print in many different ways, but so far we haven't looked at how to position the printing on the page.

In this chapter we will learn how to:

- **Change the vertical spacing**
- **Change the length of the page**
- **Set top and bottom margins**

STARTING NEW LINES

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

There are two codes that are used to end one line and start another. They are *carriage return* (CHR$(13)) and *line feed* (CHR$(10)). Like the espace code, they have been given abbreviations which you'll find in many texts (including this one): <CR> and <LF>. The codes are simple, but their action is a little confusing (especially with BASIC). Carriage return is the easiest. Each time that the printer receives a CHR$(13) it returns the print head to the left margin. It does not advance the paper (if DIP switch 2-3 is on; see below).

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

Now to add a little confusion — most (but not all) versions of BASIC add a line feed (CHR$(10)) to every carriage return (CHR$(13)) that they send. If your version of BASIC doesn't do this, then you should turn DIP switch 2-3 off so that SR-10/15

will add the line feed for you. When you have DIP switch 2-3 off the printer will do the same thing when it receives a carriage return as it does when it receives a line feed.

If you find that your printer double spaces when it should single space, then you probably need to turn DIP switch 2-3 on.

■ Reverse line feeds

Your SR-10/15 printer has a unique capability: it can move the paper up or down! Its unique tractor design allows the paper to be fed in either direction without jamming. This allows you to move around the page at will. You can use this feature to print several columns of text side by side, or print a graph and then move back up and insert descriptive legends. As you experiment you're bound to come up with more uses!

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

Table 6-1
Line feed commands

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

CHANGING LINE SPACING

When you turn SR-10/15 on the line spacing is set to 6 lines per inch. This is fine for most printing applications, but sometimes you may want something different. SR-10/15 makes it easy to set the line spacing to whatever value you want.

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

```
NEW
1Ø FOR I = 1 TO 25
2Ø IF I = 13 THEN 5Ø
3Ø LPRINT CHR$(27) "A" CHR$(I);
4Ø LPRINT "This line spacing is set to" I
5Ø NEXT
6Ø LPRINT "Line spacing is set to 1/6 inch (normal)."
7Ø LPRINT CHR$(27) "2"
```

This is what you will get:

```
This line spacing is set to 1
This line spacing is set to 2
This line spacing is set to 3
This line spacing is set to 4
This line spacing is set to 5
This line spacing is set to 6
This line spacing is set to 7
This line spacing is set to 8
This line spacing is set to 9
This line spacing is set to 10
This line spacing is set to 11
This line spacing is set to 12
This line spacing is set to 14
This line spacing is set to 15
This line spacing is set to 16
This line spacing is set to 17
This line spacing is set to 18
This line spacing is set to 19
This line spacing is set to 20
This line spacing is set to 21
This line spacing is set to 22
This line spacing is set to 23
This line spacing is set to 24
This line spacing is set to 25
Line spacing is set to 1/6 inch (nomal).
```

Line 30 changes the line spacing. The command <ESC> "A" CHR$(*n*) changes the line spacing to *n*/72 of an inch. The loop that is started in line 10 increases the value of *n* (the variable I in the program) each time it is executed. So the line spacing increases as the program continues. Line 20 just shortcuts the loop when I = 13, since BASIC won't let us send CHR$(13) without adding an unwanted CHR$(10) to it. Finally, the <ESC> "2" in line 60 resets the line spacing to 6 lines per inch. This is a shortcut that is the same as <ESC> "A" CHR$(12).

When you run this program with IBM mode, you cannot get the printout as shown above.

The command <ESC> "A" CHR$(*n*) in IBM mode only defines the line spacing as *n*/72 of an inch; the <ESC> "2" command changes the line spacing to the amount defind by the previous <ESC> "A".

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

```
3Ø LPRINT CHR$(27) "A" CHR$(I); : LPRINT CHR$(27)
   "2";
7Ø LPRINT CHR$(27) "A" CHR$(12); : LPRINT CHR$(27)
   "2"
```

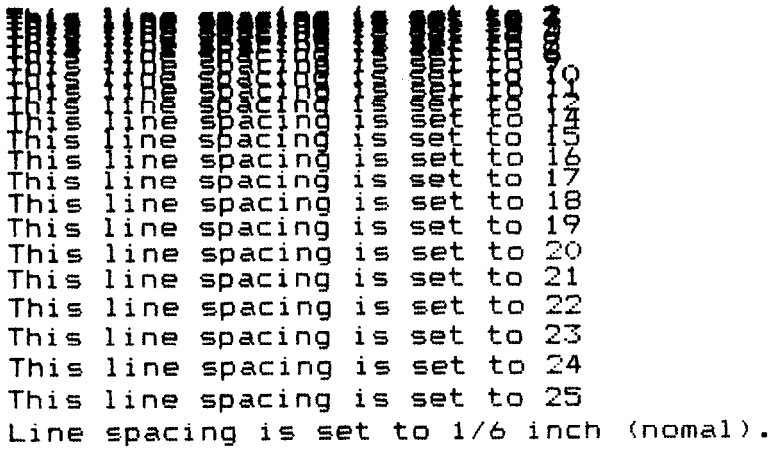
You may wonder why they picked 1/72 of an inch as the increment for the line spacing command. There's a good reason: the dots that the printer makes are 1/72 inch apart. So this means that you can vary the line spacing in increments as fine as one dot—unless you want finer spacing, like one half dot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

The <ESC> "3" CHR$(*n*) command sets the line spacing in increments of 1/144 inch (STAR mode) or 1/216 inch (IMB mode). Change line 30 in your program so it is like this:

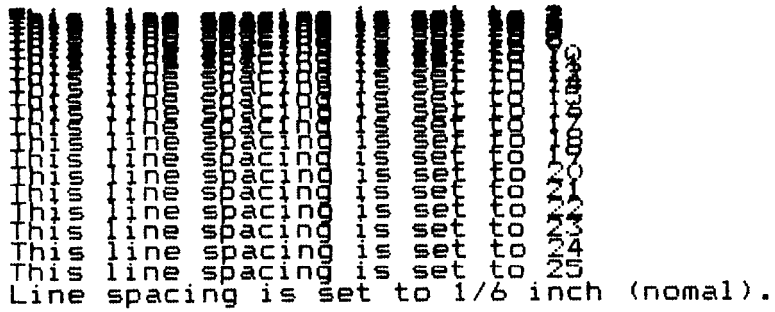
```
3Ø LPRINT CHR$(27) "3" CHR$(I);
```

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

(STAR mode)



(IBM mode)



The program works just the same as before, but the line spacing are just half (STAR mode) or one-third (IBM mode) what they were. This is because < ESC > "3" CHR$($n$) sets the line spacing to n/144 inch in the STAR mode, or n/216 inch in the IBM mode.

Table 6-2 shows all the line spacing commands, including several "shortcut" commands for commonly used line spacings.

Table 6-2
Line spacing commands

Function	Mode	Control code
Set line spacing to 1/8 inch	STAR	<ESC> "0"
	IBM	<ESC> "0"
Set line spacing to 7/72 inch	STAR	<ESC> "1"
	IBM	<ESC> "1"
Set line spacing to 1/6 inch	STAR	<ESC> "2"
	IBM	(not attached)
Set line spacing to *n*/72 inch	STAR	<ESC> "A" CHR$(*n*)
	IBM	(not attached)
Define line spacing to *n*/72 inch	STAR	(not attached)
	IBM	<ESC> "A" CHR$(*n*)
Set to <ESC> "A" definition	STAR	(not attached)
	IBM	<ESC> "2"
Set line spacing to *n*/144 inch	STAR	<ESC> "3" CHR$(*n*)
	IBM	(not attached)
Set line spacing to *n*/216 inch	STAR	(not attached)
	IBM	<ESC> "3" CHR$(*n*)
One-time line feed of *n*/144 inch	STAR	<ESC> "J" CHR$(*n*)
	IBM	(not attached)
One-time line feed of *n*/216 inch	STAR	(not attached)
	IBM	<ESC> "J" CHR$(*n*)
One-time reverse line feed of *n*/144 inch	STAR	<ESC> "j" CHR$(*n*)
	IBM	(not attached)
One-time reverse line feed of *n*/216 inch	STAR	(not attached)
	IBM	<ESC> "j" CHR$(*n*)
Advance paper *n* lines	STAR	<ESC> "a" CHR$(*n*)
	IBM	<ESC> "a" CHR$(*n*)

Note: If your computer does not support lowercase characters, use CHR$(106) and CHR$(97) for "j" and "a," respectively.

■ Moving down the page without a carriage return

So far, all the commands that move the paper also move the print head to the left margin. And normally this is what you want. Sometimes, though, you may wish to move down the page

without moving the printhead back to the left margin. The following commands do just that.

The <ESC> "J" CHR$(*n*) command causes the printer to make one line feed of *n*/144 inch (STAR mode), or *n*/216 inch (IBM mode), but does not *change* the setting of the line spacing. Try this program to see how it works:

```
1Ø 'Demo one-time line feeds.
2Ø LPRINT "Line number 1."
3Ø LPRINT "Line number 2." ;
4Ø 'One time line feed.
5Ø LPRINT CHR$(27) "J" CHR$(1ØØ) ;
6Ø LPRINT "Line number 3."
7Ø LPRINT "Line number 4."
```

Here is what SR-10/15 will produce:

```
Line number 1.
Line number 2.


                  Line number 3.
Line number 4.
```

The <ESC> "J" CHR$(100) in line 50 changes the spacing to 100/144 inches (100/216 inches for IBM mode) for one line only without moving the printhead. The rest of the lines printed with the normal line spacing. Notice that both line 30 and line 50 end with semicolons. This prevents the normal line feed from occuring.

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

```
4Ø 'One time reverse line feed.
5Ø LPRINT CHR$(27) "j" CHR$(1ØØ) ;
```

```
                    Line number 3.
Line number 4.

Line number 1.
Line number 2.
```

The <ESC> "a" CHR$(*n*) command advances the paper *n* lines (using whatever the current line spacing is) without moving the printhead. Change line 40 and 50 of your program so that they are like this.

```
4Ø 'Advance paper 3 lines.
5Ø LPRINT CHR$(27) "a" CHR$(3) ;
```

Now when you run the program the results will look like this.

```
Line number 1.
Line number 2.


                    Line number 3.
Line number 4.
```

The new line 50 moves the paper up 3 lines, but the printhead doesn't move. Therefore, line 60 prints its message starting in the column that the printhead was left in at the end of line 30.

FORMS CONTROLS

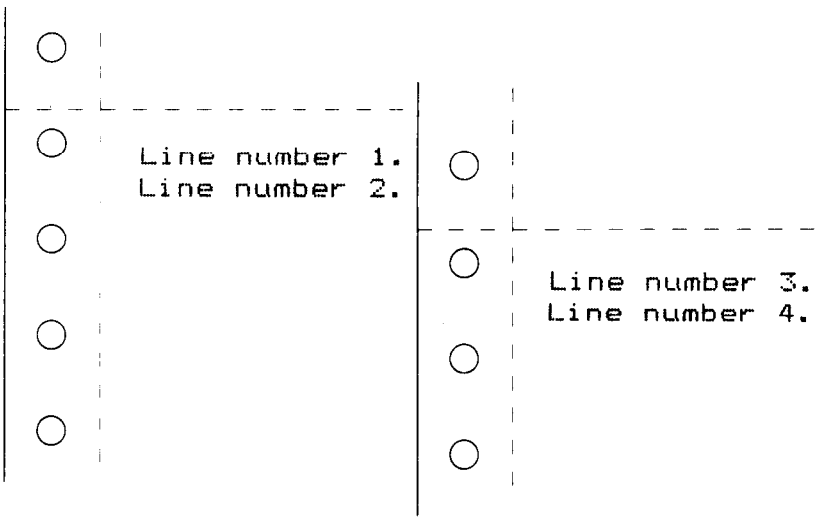
We have seen how to control the spacing between lines on a page. SR-10/15 also has commands that control the placement of printing on the page, and even adjust for different size pages.

■ Form feed

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

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

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



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

A note to TRS-80 users: CHR$(12) is a problem code for the TRS-80. To send a form feed command to SR-10/15 you must add 128 to it making it CHR$(140). Use CHR$(140) where we use CHR$(12) in these programs.

■ Reverse form feed

Just as SR-10/15 can perform a reverse line feed, it can do a reverse form feed. This code moves the paper so that the print head is positioned at the top of the current page. This can be used, for example, to print text in a multi-column magazine format; print the first column, then reverse form feed back to the top of the page to start the second column. The code for reverse form feed is easy to remember: <ESC> <FF>.

Table 6-3
Form feed commands

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

CHANGING THE PAGE LENGTH

You may have some computer forms that you wish to use with SR-10/15 that are not 11 inches high. That's no problem, because you can tell SR-10/15 how high the forms are that you are using. There are two commands for doing this, shown in this table:

Table 6-4
Form length commands

Function	Mode	Control code
Set the page length to *n* lines	STAR	<ESC> "C" CHR$(*n*)
	IBM	<ESC> "C" CHR$(*n*)
Set the page length to *n* inches	STAR	<ESC> "C" CHR$(0) CHR$(*n*)
	IBM	<ESC> "C" CHR$(0) CHR$(*n*)

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

```
1Ø 'Demo variable form lengths.
2Ø LPRINT CHR$(27) "C" CHR$(Ø) CHR$(7) ; 'Form length 7
   inches.
3Ø LPRINT "Pay to the order of:"
4Ø LPRINT CHR$(12) ; 'Form feed.
5Ø LPRINT "Pay to the order of:"
```

This program should print "Pay to the order of:" twice, and they should be 7 inches apart. Line 20 sets the form length to 7 inches. After line 30 prints, line 40 sends a form feed advance the paper to the top of the next form. Line 50 then prints its message.

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

TOP AND BOTTOM MARGINS

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

Of course SR-10/15 has a solution to this predicament. SR-10/15 can keep track of the position on the page, and advance the paper so that you won't print too near the perforation. There are two commands to do this. One controls the space at the top of the page and the other controls the space at the bottom of the page. The control codes are given in the following table.

Table 6-5
Top and bottom margin commands

Function	Mode	Control code
Set top margin	STAR	<ESC> "R" CHR$(*n*)
	IBM	<ESC> "r" CHR$(*n*)
Set bottom margin	STAR	<ESC> "N" CHR$(*n*)
	IBM	<ESC> "N" CHR$(*n*)
Clear top and bottom margins	STAR	<ESC> "O"
	IBM	<ESC> "O"

In both cases the value of *n* tells SR-10/15 how many lines to skip, although there is a slight difference in the usage. When you set the top margin with <ESC> "R" CHR$(*n*) in STAR mode, or with <ESC> "r" CHR$(*n*) in IBM mode, the value of *n* tells SR-10/15 what line to start printing on. When you set the bottom margin with <ESC> "N" CHR$(*n*), the value of *n* tells SR-10/15 how many blank lines should be left at the bottom of the page.

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

```
1Ø 'Demo top and bottom margins
2Ø LPRINT CHR$(12) ; 'Form feed.
3Ø FOR I = 1 TO 15Ø
4Ø LPRINT "This is line" I
5Ø NEXT I
6Ø LPRINT CHR$(12) ; 'Form feed.
```

When you run this program it will print 150 lines right down the page and across the perforations. When it's done line 60 sends a form feed to advance the paper to the top of the next page. Look at the lines that have printed near the perforations. Separate the sheets and see if any of the lines have been torn in half. These are the problems that the top and bottom margins will solve.

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

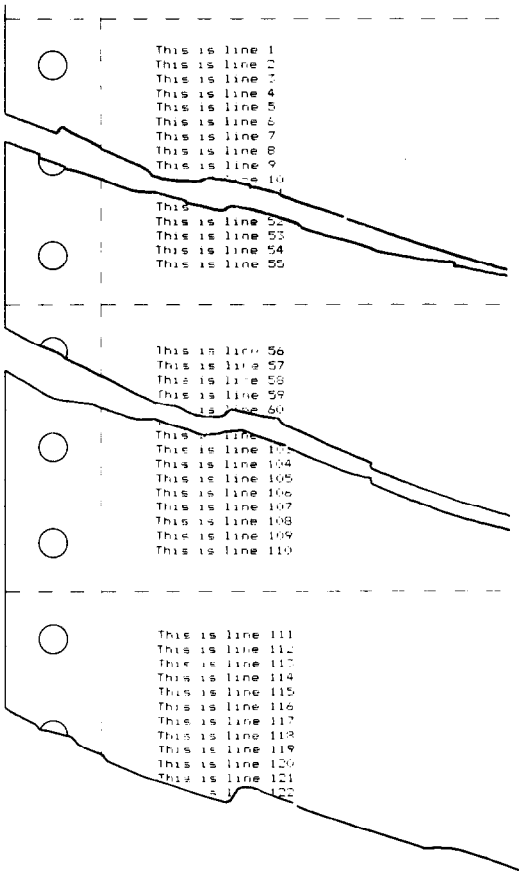
```
11 'Leave 6 blank lines at bottom of page.
12 LPRINT CHR$(27) "N" CHR$(6) ;
13 'Start top of page at line 6.
14 LPRINT CHR$(27) "R" CHR$(6) ;
55 LPRINT CHR$(27) "O" ; 'Clear top & bottom margins.
```

Now when you run the program with STAR mode SR-10/15 will skip the first six lines and the last six lines on each page. Always send a form feed after setting the top margin, or it will not work on the first page printed. That's because the top margin only takes effect after a form feed.

Line 14 sets the top margin, line 12 sets the bottom margin, and line 55 clears both margins when we are done.

SUMMARY

Control code	**Function**
CHR$(10)	Line feed
<ESC> CHR$(10)	Reverse line feed
CHR$(13)	Carriage return
<ESC> "A" CHR$(*n*)	Set line spacing to *n*/72 inch (for STAR mode)
<ESC> "3" CHR$(*n*)	Set line spacing to *n*/144 inch (for STAR mode)



< ESC > "0"	Set line spacing to 1/8 inch
< ESC > "1"	Set line spacing to 7/72 inch
< ESC > "2"	Set line spacing to 1/6 inch (for STAR mode)
< ESC > "J" CHR$(*n*)	One-time line feed of *n*/144 inch (for STAR mode)
< ESC > "j" CHR$(*n*)	One-time reverse line feed of *n*/144 inch (for STAR mode)
< ESC > "a" CHR$(*n*)	Advance the paper *n* lines
< ESC > "A" CHR$(*n*)	Define line spacing of *n*/72 inch (for IBM mode)
< ESC > "2"	Use < ESC > "A" definition (for IBM mode)
< ESC > "3" CHR$(*n*)	Set line spacing to *n*/216 inch (for IBM mode)
< ESC > "J"CHR$(*n*)	One-time line feed of *n*/216 inch (for IBM mode)

<ESC> "j"CHR$(*n*)	One-time reverse line feed of *n*/216 inch (for IBM mode)
CHR$(12)	Form feed
<ESC> CHR$(12)	Reverse form feed
<ESC> "C" CHR$(*n*)	Set page length to *n* lines
<ESC> "C" CHR$(0) CHR$(*n*)	Set page length to *n* inches
<ESC> "R" CHR$(*n*)	Set top margin; start printing on line *n* (for STAR mode)
<ESC> "r" CHR$(*n*)	Set top margin; start printing on line *n* (for IBM mode)
<ESC> "N" CHR$(*n*)	Set bottom margin; leave *n* lines blank
<ESC> "O"	Clear top and bottom margins

CHAPTER 7

FORMATTING YOUR OUTPUT

You have probably used the tab and margin features on a typewriter. They make it easier to format the text on a page. SR-10/15 also has tabs and margins that you can set. But it goes beyond the capabilities of a typewriter because besides having tabs that go across the page, called *horizontal tabs,* SR-10/15 has *vertical tabs* that go down the page.

In this chapter we will discover how to use:

- **Horizontal tabs**
- **Vertical tabs**
- **Left and right margins**

USING HORIZONTAL TABS

When you turn SR-10/15 on there are horizontal tabs set automatically every eight spaces. It's easy to use these tabs; you just send a CHR$(9) to SR-10/15 and the print head will move to the next tab position. CHR$(9) is the ASCII code <HT> for *horizontal tab.*

Try this one line program to demonstrate the use of the default horizontal tabs.

```
1Ø 'Tabs demo
2Ø LPRINT "one" CHR$(9) "two CHR$(9) "three" CHR$(9)
   "four"
```

Here's what will print:

```
one     two     three   four
```

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

CHR$(9) is a problem with some computers. Some BASICs convert CHR$(9) to a group of spaces that act like a sort of *pseudo-tab*. This is fine if the computer and the printer have the same tab settings, but it doesn't allow us to use our own tab settings on SR-10/15. We can "outsmart" these computer by adding 128 to the ASCII value that we use. Instead of using CHR$(9), use CHR$(137) for a tab command. Even this trick won't work for Apple II computers, for they use CHR$(9) for something else entirely. Apple users can get some help in Appendix J.

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

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

<ESC> "D" is the command to begin setting horizontal tabs. It must be followed by characters representing the positions that you want the tabs set. In our program we are setting tabs in columns 7, 14, and 21. The CHR$(0) at the end ends the string of tabs. In fact, any character that is not greater than the previous one will stop setting tabs. This means that you must put all your tab values in order, from least to greatest, or they won't all get set. (It also means that a CHR$(1) is just as good as a CHR$(0) for ending a group of tabs; some computers have trouble sending CHR$(0).)

When you run the program now it produces this:

```
one     two     three   four
```

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

■ A one-shot tab command

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

The solution is called a *one-time tab* and is <ESC> "b" CHR$(*n*). This command moves the print head *n* columns to the right. It has the same effect as sending *n* spaces to the printer.

Table 7-1
Horizontal tab commands

Function	Mode	Control code
Advance to next tab position	STAR	CHR$(9)
	IBM	CHR$(9)
Set tabs at *n1, n2,* etc.	STAR	<ESC> "D" CHR$(*n1*) CHR$(*n2*)...CHR$(0)
	IBM	<ESC> "D" CHR$(*n1*) CHR$(*n2*)...CHR$(0)
One-time tab of *n* spaces	STAR	<ESC> "b" CHR$(*n*)
	IBM	<ESC> "b" CHR$(*n*)

Note: If your computer does not support lowercase characters, use CHR$(98) for "b."

SETTING LEFT AND RIGHT MARGINS

SR-10/15's left and right margins work just like a typewriter—once they are set all the printing is done between them. The commands to set the margins are given in the following table:

Table 7-2
Left and right margin commands

Function	Mode	Control code
Set left margin at column *n*	STAR	<ESC> "M"CHR$(*n*)
	IBM	<ESC> "l" CHR$(*n*)
Set right margin at column *n*	STAR	<ESC> "Q" CHR$(*n*)
	IBM	<ESC> "Q" CHR$(*n*)

Try setting SR-10/15's margins with this program for STAR mode:

```
1Ø 'Demo margins.
2Ø GOSUB 7Ø
3Ø LPRINT CHR$(27) "M" CHR$(1Ø) ; 'Left margin = 1Ø.
4Ø LPRINT CHR$(27) "Q" CHR$(7Ø) ; 'Right margin = 7Ø.
5Ø GOSUB 7Ø
6Ø END
7Ø FOR I = 1 TO 8Ø
8Ø LPRINT "X" ;
9Ø NEXT I
1ØØ LPRINT
11Ø RETURN
```

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

Run the program. The results will look like this:

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
         xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
         xxxxxxxxxxxxxxxxxxxx
```

When you want to reset the margins to the default values, you have two choices. You can either turn the printer off and back on, or you can set margin values equal to the default values. This means that you should set a left margin of 0 and a right margin of 80 on SR-10 or 136 on SR-15

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

USING VERTICAL TABS

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

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

```
1Ø 'Demo vertical tabs.
2Ø LPRINT CHR$(11) "First tab."
3Ø LPRINT CHR$(11) "Second tab."
4Ø LPRINT CHR$(11) "Third tab."
5Ø LPRINT CHR$(11) "Fourth tab."
```

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

```
12 LPRINT CHR$(27) "P" CHR$(1Ø) ;
14 LPRINT CHR$(2Ø) CHR$(4Ø) CHR$(5Ø) CHR$(Ø) ;
```

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

```
First tab.



Second tab.



Third tab.



Fourth tab.
```

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

```
6Ø LPRINT CHR$(11) "Fifth tab."
```

Now when you run the program the first page looks just like before, but line 60 sends one more <VT> than there are tabs.

This doesn't confuse SR-10/15—it advances the paper to the *next* tab position which happens to be the first tab position on the next page. That's nice, isn't it?

■ A one-shot vertical tab command

There's a one-time vertical tab command that works just like the one-time horizontal tab command. It is <ESC> "a" CHR$(*n*), and it causes the paper to advance *n* lines. It doesn't change the settings of the vertical tabs.

Table 7-3
Vertical tab commands

Function	Mode	Control code
Advance paper to next tab position	STAR	CHR$(11)
	IBM	CHR$(11)
Set vertical tabs at *n1, n2,* etc.	STAR	<ESC> "P" CHR$(*n1*) CHR$(*n2*)...CHR$(0)
	IBM	<ESC> "B" CHR$(*n1*) CHR$(*n2*)...CHR$(0)
Advance paper *n* lines	STAR	<ESC> "a" CHR$(*n*)
	IBM	<ESC> "a" CHR$(*n*)

Note: If your computer does not support lowercase characters, use CHR$(97) for "a."

SUMMARY

Control code	Function
CHR$(9)	Horizontal tab
<ESC> "D" *n1 n2 n3* ...CHR$(0)	Set horizontal tabs
<ESC> "b" *n*	One-time horizontal tab of *n* spaces
<ESC> "M" *n*	Set left margin (for STAR mode)
<ESC> "l" *n*	Set left margin (for IBM mode)
<ESC> "Q" *n*	Set right margin
CHR$(11)	Vertical tab
<ESC> "P" *n1 n2 n3* ...CHR$(0)	Set vertical tabs (for STAR mode)
<ESC> "B" *n1 n2 n3* ...CHR$(0)	Set vertical tabs (for IBM mode)
<ESC> "a" *n*	One-time vertical tab of *n* lines

CHAPTER 8

SPECIAL FEATURES OF THE SR-10/15

In the previous chapters we have learned about several groups of control codes. In this chapter we will look at more control codes. These codes don't fit neatly into any of the groupings that we have studied, but they add a lot of capability to SR-10/15. So here goes.

Commands covered in this chapter include:

- **Bell**
- **Master reset**
- **Unidirectional printing**
- **Eighth bit control**
- **Block graphics**
- **International character sets**
- **Macro instruction**

■ Now hear this

You may have heard SR-10/15's *bell* if you have ever run out of paper. And you may have wondered why it's called a bell when it *beeps* instead of ringing! It's a long story that goes back to the early days of computers, when teletype machines were used for computer terminals. These mechanical marvels had a bell in them that could be heard for blocks. This bell was used to signal the operator that something needed attention. The code that the computer sent to the teletype machine to ring the bell was, reasonably enough, called a *bell code*. Well the name *bell code* is still with us, even if the bell has changed to a beeper, and a lot of people still call the beeper a bell, even if it doesn't sound like one. So with our trivia lesson out of the way, let's see how we can "ring the bell."

The code to sound SR-10/15's "bell" is CHR$(7), which is ASCII code 7 or < BEL > . Any time SR-10/15 receives this code it will sound the bell for a quarter of a second. This can be used

to remind an operator to change the paper or to make another adjustment to the printer. Note to Apple users: Entering a CHR$(7) will sound *Apple's* bell; the code will not be sent to SR-10/15.

You can try this by typing:

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

There are two other codes that affect the bell. One disables the bell, so that SR-10/15 will ignore a CHR$(7), and the other turns the bell back on. All three codes that affect the bell are shown in the following table.

Table 8-1
Bell commands

Function	Mode	Control code
Sound bell	STAR	CHR$(7)
	IBM	CHR$(7)
Disable bell	STAR	<ESC> "Y" 0
	IBM	<ESC> "y" 0
Enable bell	STAR	<ESC> "Y" 1
	IBM	<ESC> "y" 1

Note: If your computer does not support lowercase characters, use CHR$(121) for "y".

■ Initializing SR-10/15

Up to now when we wanted to reset SR-10/15 to the power on condition we have had to either turn the printer off and then on again, or to send the specific codes that reset the particular features. There is an easier way. The control code <ESC> "@" will reset all of SR-10/15's features to the power on condition (as determined by the DIP switches), with two exceptions. Those exceptions are that <ESC> "@" will not erase any characters that you have stored in SR-10/15's RAM memory (Chapter 9 tells you how to create your own characters), and it won't erase the macro if you have one stored in SR-10/15's RAM (this chapter will tell you how to create a macro).

■ Putting SR-10/15 to sleep

You know how to put SR-10/15 *off-line* with the On Line button. SR-10/15 has another *off-line* state that can be controlled from your computer. When you turn SR-10/15 *off-line* from your computer, SR-10/15 will ignore anything that you send it, except

for the code to go *on-line* again. CHR$(19) is the code to turn SR-10/15 off-line; CHR$(17) returns SR-10/15 to on-line status.

■ Printing to the bottom of the sheet

Sometimes when you are using sprocket paper you may want to print near the bottom of the last sheet. The paper-out detector usually stops SR-10/15 when you are about 3 inches from the bottom of the sheet. This is to notify you if you are running out of continuous paper.

SR-10/15 has the ability to print right to the bottom of the sheet. You can disable the paper-out detector so that it doesn't stop the printer. This will allow you to print to the end of the sheet, and even beyond if you are not careful. The codes to control the paper-out detector, along with the other codes that we have just learned are in the following table.

Table 8-2
Some miscellaneous commands

Function	Mode	Control code
Master reset	STAR	<ESC> "@"
	IBM	<ESC> "@"
Off line	STAR	CHR$(19)
	IBM	CHR$(19)
On-line	STAR	CHR$(17)
	IBM	CHR$(17)
Paper-out detector off	STAR	<ESC> "8"
	IBM	<ESC> "8"
Paper-out detector on	STAR	<ESC> "9"
	IBM	<ESC> "9"
Move print head back one space	STAR	CHR$(8)
	IBM	CHR$(8)
Delete last character sent	STAR	CHR$(127)
	IBM	CHR$(127)
Cancel text in print buffer	STAR	CHR$(24)
	IBM	CHR$(24)
Print "zero" with slash	STAR	<ESC> "\" 1
	IBM	<ESC> "\" 1
Print "zero" without slash	STAR	<ESC> "\" 0
	IBM	<ESC> "\" 0

■ Backspace, delete, and cancel text

Backspace (CHR$(8)) "backs up" the printhead so that you can print two characters right on top of each other. Each time SR-10/15 receives a backspace it moves the printhead one character to the left, instead of to the right. You can *strike over* multiple letters by sending more than one backspace code.

Delete (CHR$(127)) also "backs up" one character, but then it "erases" the previous character (it's erased from SR-10/15's buffer, not from the paper).

Cancel text (CHR$(24)) deletes all the text in the print buffer; that is, in the line before the delete text command. Since SR-10/15 prints one line of text at a time, only that line will be deleted.

The following program shows how these three codes work.

```
NEW
10 LPRINT "BACKSPACE DOES NOT";
20 LPRINT CHR$(8) CHR$(8) CHR$(8);
30 LPRINT "=== WORK"
40 LPRINT "DELETE DOES NOT";
50 LPRINT CHR$(127) CHR$(127) CHR$(127);
60 LPRINT "WORK"
70 LPRINT "CANCEL TEXT";
80 LPRINT CHR$(24);
90 LPRINT "DOES NOT PRINT"
```

Here is what this program will print:

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

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

In line 80, CHR$(24) deletes the words in line 70. The semicolon at the end of line 70 prevents a line feed from causing that line to print before SR-10/15 receives the CHR$(24) code. The text in line 90 prints as it normally would because it is after CHR$(24).

■ "Zero" printing

Sometimes, you want to print "zero" with slash to distinguish between "0" and "O". Your SR-10/15 can print either "0" or "Ø" as you wish.

■ Unidirectional printing

Unidirectional printing is a big word that means *printing in one direction only*. SR-10/15 normally prints when the printhead is moving in both directions. But once in a while you may have an application where you are more concerned about how the vertical lines align than with how fast it prints. SR-10/15 lets you make this choice. The table below shows the commands for controlling how SR-10/15 prints.

Table 8-3
Printing direction commands

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

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

```
10 'Demo unidirectional printing.
20 LPRINT CHR$(27) "A" CHR$(7) ; 'Line spacing = 7/72".
30 FOR I = 1 TO 10
40 LPRINT "¦"
50 NEXT I
60 LPRINT : LPRINT
70 LPRINT CHR$(27) "U" CHR$(1) ; 'Turn on unidirectional
   printing.
80 FOR I = 1 TO 10
90 LPRINT "¦"
100 NEXT I
110 LPRINT CHR$(12) CHR$(27) "@" ; 'Form feed, master
    reset.
```

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

Let's analyze the program. Line 20 sets the line spacing to 7/72 of an inch so that the characters that we print will touch top to bottom. Lines 30-50 print 10 vertical line characters. Then line 70 sets one-direction printing and the vertical lines are printed

again. Finally line 110 sends a form feed to advance the paper to the top of a new page, and then uses the master reset to restore SR-10/15 to the power-on condition.

You can also set SR-10/15 to print in one direction for one line only by using the command <ESC> "<". This command immediately moves the printhead to the left margin and then prints the remainder of the line from left to right.

■ The seven bit dilemma

Certain computers (most notably the Apple II) don't have the capability to send eight bits on their parallel interface. They can only send seven bits. This would make it impossible for these computers to use SR-10/15's block graphics characters and special symbols if Star's engineers hadn't thought of a solution. (All of these characters have ASCII codes greater than 127 which means that the eighth bit must be on to use them.) The solution lies in the three control codes given in the following table.

Table 8-4
Eight bit control commands

Function	Mode	Control code
Turn the eighth bit ON	STAR	<ESC> ">"
	IBM	<ESC> ">"
Turn the eighth bit OFF	STAR	<ESC> "="
	IBM	<ESC> "="
Accept the eighth bit "as is" from the computer	STAR	<ESC> "#"
	IBM	<ESC> "#"

■ Block graphics characters and special symbols

Besides the upper and lower case letters and symbols that we are by now familiar with, SR-10/15 has a whole different set of characters that are for special uses. These characters include block graphics characters for drawing forms and graphs, and special symbols for mathematical, engineering and professional uses. The following program will print out all of the graphics characters available in STAR mode.

```
1Ø 'Demo all block graphic characters.
2Ø WIDTH "LPT1:",255
3Ø LPRINT CHR$(27) "D" CHR$(1Ø) CHR$(2Ø) ;
4Ø LPRINT CHR$(3Ø) CHR$(40) CHR$(5Ø) CHR$(6Ø) ;
5Ø LPRINT CHR$(7Ø) CHR$(Ø) ; 'Set tabs.
6Ø FOR J = 16Ø TO 255 STEP 8
7Ø FOR I = J TO J + 7
8Ø LPRINT I "= " ;
9Ø LPRINT CHR$(I) ; 'Send graphic char.
1ØØ LPRINT CHR$(9) ; 'Tab.
11Ø NEXT I : LPRINT : NEXT J
```

Figure 8-1 shows what this program will print. If your chart doesn't look like this because it has regular letters and numbers instead of the special symbols, then your computer is only using seven bits. You can get the correct printout by adding these lines:

```
85 LPRINT CHR$(27) ">" ; 'Turn on 8th bit.
95 LPRINT CHR$(27) "=" ; 'Turn off 8th bit.
```

The special characters for IBM mode are included in two character sets. The character set you normally use is called character set #1. The special characters are printed out when you send ASCII codes 160-255 to the printer.

SR-10/15 also offers character set #2 which is almost the same as character set #1 except for the addition of ASCII codes 3-6, 21, and 128-159. Character set #2 is selected with <ESC> "6"; to go back to character set #1, use <ESC> "7".

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

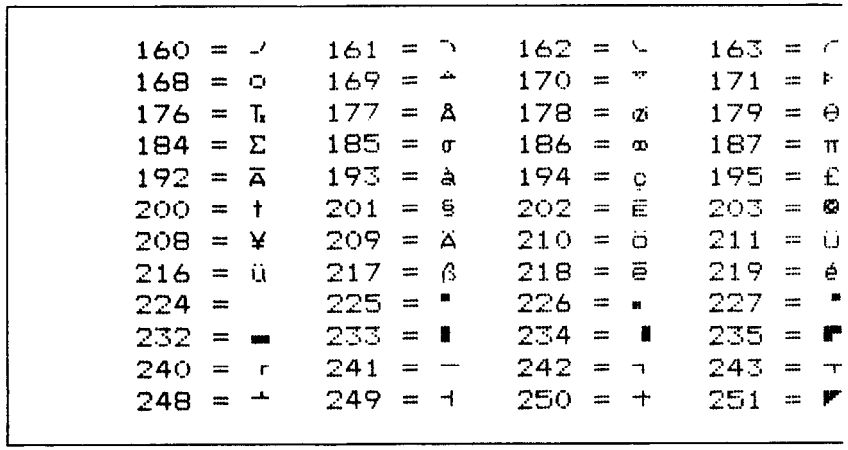


Figure 8-1.

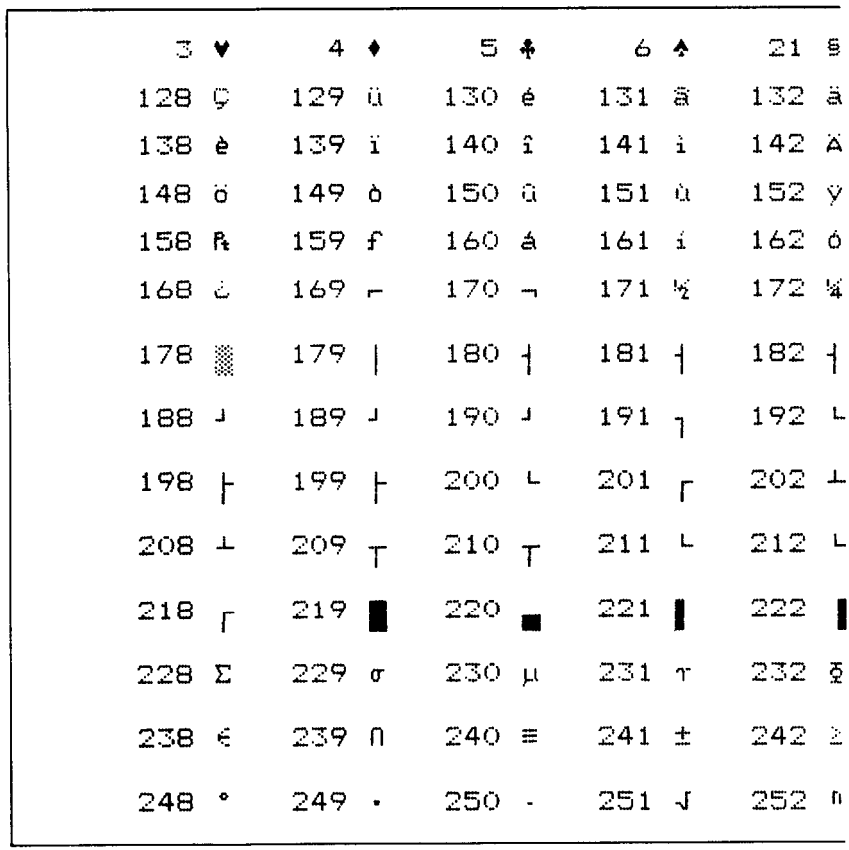


Figure 8-2.

164 = ↑	165 = ↓	166 = ←	167 = →
172 = ¼	173 = ¡	174 = ◆	175 = □
180 = ┤	181 = ╞	182 = Ω	183 = ü
188 = ±	189 = ↄ	190 = ×	191 = ÷
196 = ä	197 = µ	198 = °	199 = ›
204 = ¼	205 = ¤	206 = ½	207 = ‖
212 = ¢	213 = Ñ	214 = ä	215 = ö
220 = ú	221 = è	222 = ñ	223 = f
228 = ▪	229 = ▪	230 = ▪	231 = ▬
236 = ▀	237 = ▙	238 = ▟	239 = ■
244 = ⊦	245 = ⊦	246 = ⊦	247 = ⌟
252 = ◢	253 = ◥	254 = ◣	255 =

133 à	134 ä	135 ç	136 ê	137 ë
143 Ä	144 é	145 æ	146 Æ	147 ö
153 ö	154 Ü	155 ¢	156 £	157 ¥
163 ú	164 ñ	165 Ñ	166 ª	167 º
173 ¡	174 «	175 »	176 ⣿	177 ▒
183 ┐	184 ┐	185 ┤	186 │	187 ┐
193 ┴	194 ┬	195 ├	196 ─	197 ┼
203 ┬	204 ├	205 ─	206 ┼	207 ┴
213 ┌	214 ┌	215 ┼	216 ┼	217 ┘
223 ■	224 α	225 ß	226 Γ	227 π
233 Θ	234 Ω	235 δ	236 ∞	237 Ø
243 ≤	244 ⌠	245 ⌡	246 ÷	247 ≈
253 ²	254 ▪			

```
NEW
1Ø LPRINT CHR$(27) "Ø"
2Ø LPRINT CHR$(27) "6"
3Ø FOR J = 3 TO 6
4Ø LPRINT "  " J CHR$(J) "  ";
5Ø NEXT
6Ø LPRINT "  21 " CHR$(21)
7Ø LPRINT
8Ø FOR J = 128 TO 254 STEP 1Ø
9Ø FOR I = J TO J + 9
95 IF I > 254 THEN 11Ø
1ØØ LPRINT I CHR$(I) "  ";
11Ø NEXT I : LPRINT : LPRINT : NEXT J
```

Figure 8-2. shows what this program will print. If your chart doesn't look like this because it has regular letters and numbers instead of the special symbols, then your computer is only using seven bits. You can get the correct printout by changing line 100 to this:

```
1ØØ LPRINT I CHR$(27) ">" CHR$(I) CHR$(27) "="
    CHR$(9);
```

■ International character sets

Table 8-5
International character set commands

Country	Mode	Control code
U.S.A.	STAR	<ESC> "7" CHR$(0)
	IBM	<ESC> "R" CHR$(0)
France	STAR	<ESC> "7" CHR$(1)
	IBM	<ESC> "R" CHR$(1)
Germany	STAR	<ESC> "7" CHR$(2)
	IBM	<ESC> "R" CHR$(2)
England	STAR	<ESC> "7" CHR$(3)
	IBM	<ESC> "R" CHR$(3)
Denmark	STAR	<ESC> "7" CHR$(4)
	IBM	<ESC> "R" CHR$(4)
Sweden	STAR	<ESC> "7" CHR$(5)
	IBM	<ESC> "R" CHR$(5)
Italy	STAR	<ESC> "7" CHR$(6)
	IBM	<ESC> "R" CHR$(6)
Spain	STAR	<ESC> "7" CHR$(7)
	IBM	<ESC> "R" CHR$(7)

SR-10/15 is a multi-lingual printer for it can speak in eight languages! SR-10/15 changes languages by changing 11 characters that are different for the different languages. These sets of characters are called *international character sets.* The control codes to select the international character sets are given in Table 8-5.

The characters that change are shown beneath their ASCII code in Table 8-6.

Table 8-6
Internationad character sets

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

■ The macro control code

The last of our group of miscellaneous control codes is definitely not the least. It is a *user-defined* control code, called a *macro* control code. The term *macro* is from the jargonese *macro-instruction* which refers to an instruction that "calls," or uses a group of normal instructions. In computer programming macro-instructions (which are similar to subroutines) save programmers a lot of time and effort. SR-10/15's macro can save you a lot of time and effort also.

Here is how SR-10/15's macro works. You *define* your macro by telling SR-10/15 what normal control codes are to be included in the macro. Then you can use the macro any time that you want and SR-10/15 will do all the things that you included in the macro definition. You can include up to 16 codes in a single macro. You can even use the macro to store a frequently used word or phrase. There are two control codes for the macro: one to define it, and one to use it. They are given in the Table 8-7.

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

Table 8-7
Macro instruction commands

Function	Mode	Control code
Define macro	STAR	<ESC> "+"....*codes you include*...CHR$(30)
	IBM	<ESC> "+"....*codes you include*...CHR$(30)
Use macro	STAR	<ESC> "!"
	IBM	<ESC> "?"

```
1Ø LPRINT CHR$(27) "+";  ' START DEFINITION
   OF MACRO
2Ø LPRINT CHR$(18);      ' PICA
3Ø LPRINT CHR$(27) "WØ"; ' EXPANDED OFF
4Ø LPRINT CHR$(27) "F";  ' EMPHASIZED OFF
5Ø LPRINT CHR$(27) "H";  ' DOUBLE-STRIKE OFF
6Ø LPRINT CHR$(27) "-Ø"; ' UNDERLINE OFF
7Ø LPRINT CHR$(27) "T";  ' SUPER & SUBSCRIPTS
   OFF
8Ø LPRINT CHR$(3Ø);      ' END MACRO DEFINITION
```

As the comments in the program listing show this will define a macro that will reset all the print style functions. SR-10/15 will remember this macro until the power is turned off or until a new macro is defined. A macro can hold up to 16 bytes (characters) of information. The one that we defined contains thirteen.

Now that you have defined a macro, let's see how to use it. This program will print one line using several printing style features. Then it "calls" the macro in line 60. When line 70 prints the style is "plain vanilla" because the macro has reset it.

```
1Ø LPRINT CHR$(27) "-1"; ' UNDERLINE
2Ø LPRINT CHR$(27) "G";  ' DOUBLE- STRIKE
3Ø LPRINT CHR$(27) "W1"; ' EXPANDED
4Ø LPRINT "TESTING ONE, TWO, THREE"
5Ø LPRINT CHR$(27) "!";  ' USE THE MACRO
6Ø LPRINT "TESTING FOUR, FIVE, SIX"
```

If you are using with the IBM mode, change the line 50 as shown below.

```
5Ø LPRINT CHR$(27) "?";  ' USE THE MACRO
```

```
TESTING ONE, TWO, THREE
TESTING FOUR, FIVE, SIX
```

In this chapter we have learned many different commands that have many different uses. In the next chapter we will make up for this diversity—the whole chapter only covers three commands! But they are some of the most powerful that SR-10/15 offers. They give you the ability to create your own characters.

SUMMARY

Control code	Function
CHR$(7)	Bell
<ESC> "Y" 0	Disable bell (for STAR mode)
<ESC> "Y" 1	Enable bell (for STAR mode)
<ESC> "y" 0	Disable bell (for IBM mode)
<ESC> "y" 1	Enable bell (for IBM mode)
<ESC> "@"	Reset
CHR$(19)	Off-line
CHR$(17)	On-line
<ESC> "8"	Paper-out detector off
<ESC> "9"	Paper-out detector on
<ESC> "<"	Print in one direction for one line only
<ESC> "U" 1	Unidirectional printing
<ESC> "U" 0	Bidirectional printing
CHR$(8)	Backspace
CHR$(127)	Delete character
CHR$(24)	Cancel line
<ESC> "\" 1	Print "zero" with slash
<ESC> "\" 0	Print "zero" without slash
<ESC> ">"	Eighth bit on
<ESC> "="	Eighth bit off
<ESC> "#"	Eighth bit as-is
<ESC> "7" *n*	Select international character set (for STAR mode)
<ESC> "R" *n*	Select international character set (for STAR mode)
<ESC> "+" ..CHR$(30)	Define macro
<ESC> "!"	Use macro (for STAR mode)
<ESC> "?"	Use macro (for IBM mode)

CHAPTER 9

CREATING YOUR OWN CHARACTERS

In this chapter we'll cover:
- **Designing and printing your own characters**
- **Designing proportional characters**

In the previous four chapters of this manual you've learned how to control the SR-10/15 printer to give you dozens of different typefaces. By using various combinations of pitches, character weights, and font selections, you can create nearly any effect you want to in text. And with international character sets and the special text and graphics characters described in Chapter 8, you can print almost any character you can think of.

But if "almost any character" isn't good enough for you, then it's a good thing you have an SR-10/15 printer! With it you can actually create your own characters. As you'll see in this chapter, *download characters* can be used to print a logo, special characters for foreign languages, scientific and professional applications, or any other specific printing task.

DOT MATRIX PRINTING

In order to create download characters, you'll need some understanding of how dot matrix printers work. They're called "dot matrix" because each character is made up of a group of dots. Look closely at some printed characters produced by your SR-10/15 and you will see the dots. Figure 9-1 shows how the letter "C" is formed by printing 15 dots.

The printhead in SR-10/15 consists of nine thin wires stacked one atop the other. Figure 9-2 shows an enlarged schematic view of the front of the printhead, showing the ends of the wires and their relationship to the printed characters. As you can see, the capital letters use the top seven wires of the printhead, and the

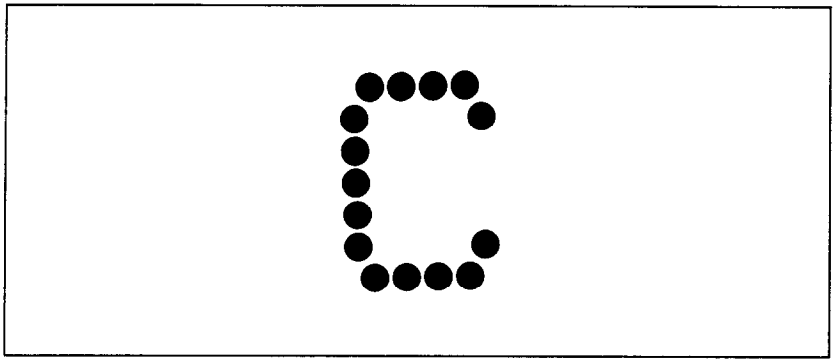


Figure 9-1. The letter "C" is created by printing 15 dots.

descenders (such as the lower case "p" shown) use the bottom seven pins. As the printhead moves across the page (in either direction—that's what is meant by bi-directional printing) it prints one column of dots at a time. Each time a dot is supposed to print an electromagnet inside the printhead causes the appropriate wire to strike the ribbon (making the SR-10/15 an *impact* printer).

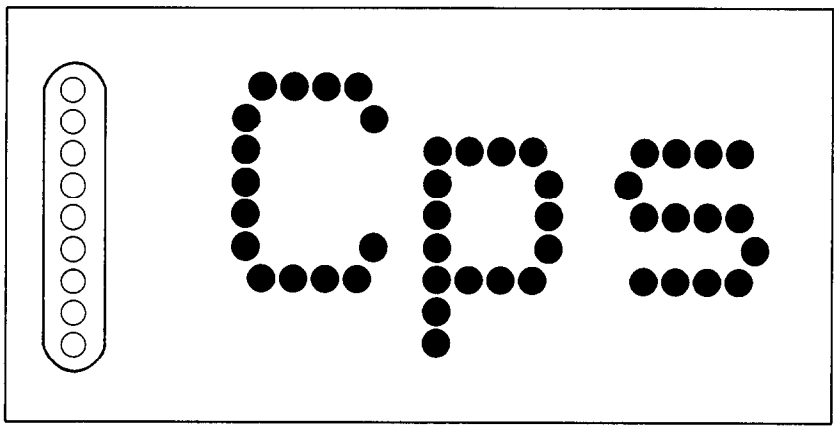


Figure 9-2. As the printhead moves across the page, each of the wires prints one row of dots.

THE PRINT MATRIX

All of the standard characters that the SR-10/15 prints are formed from patterns of dots that are permanently stored in the printer's *ROM* (read-only memory). This includes all of the standard ASCII characters, the block graphics and special characters, the international character sets, the NLQ characters and the italic characters.

But there is another area of memory in the SR-10/15 reserved for *user-defined* characters. These are characters that you design and *download* into SR-10/15. When download characters are defined they are stored in *RAM* (random access memory), which allows you to define or modify them at any time.

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

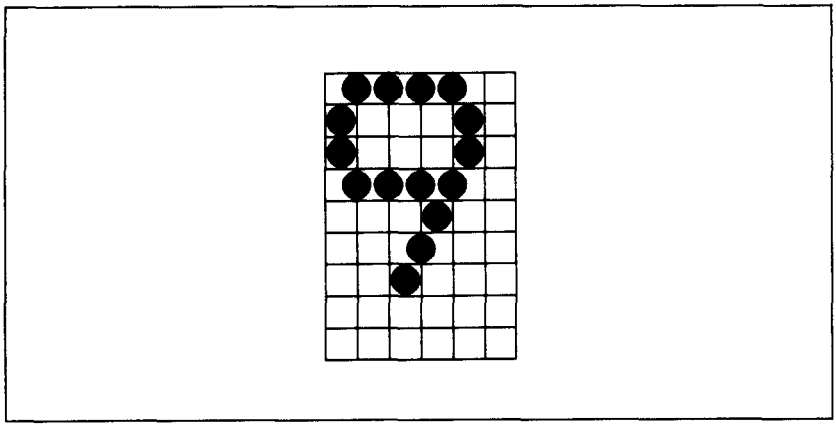


Figure 9-3. Dots can be inside boxes or straddle the vertical lines of the grid.

DEFINING YOUR OWN CHARACTERS

You've seen how the engineers at Star designed their characters by using a grid to lay out the dots. Now you can define characters exactly the same way. Make up some grids (photocopy Figure 9-4 if you wish) and get ready to be creative! (Just in case you are not feeling creative, and to make our explanations a little clearer, we'll be using a picture of a chemist's flask as an example of a download character. You can see how we've laid it out in

Figure 9-5. Later in this chapter we'll use this character to create a small graph.)

You'll notice that Figure 9-4 includes a lot of information around the grid. Don't be intimidated; we'll explain each item as we come to it in our discussion of defining and actually printing download characters. You may have noticed another difference between this grid and the one shown in Figure 9-3: it's only eight boxes high. Which leads us to...

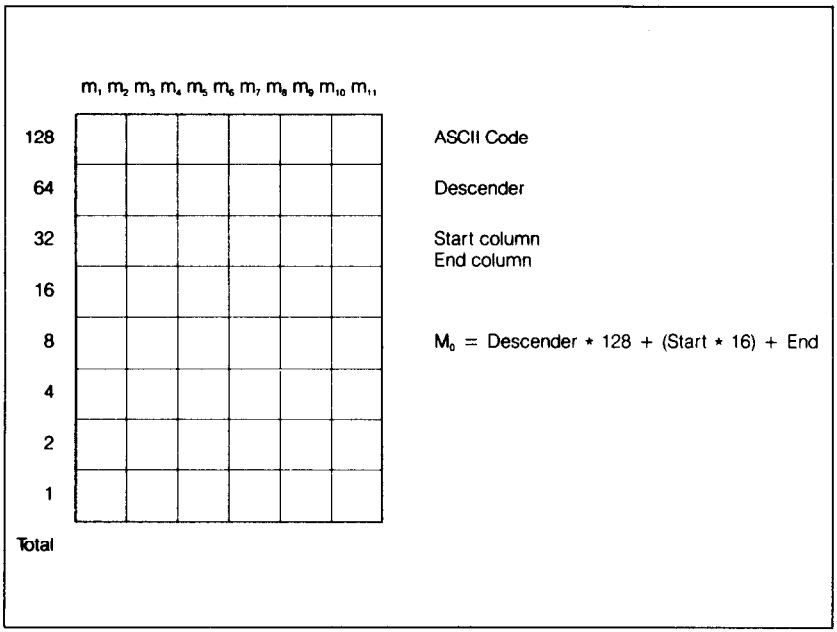


Figure 9-4. Use this grid (or one similar to it) to define your own characters.

■ Rule 1: Download characters are eight dots high

As you noticed in Figure 9-2, capital letters, most lowercase letters, and most special characters use only the top seven pins of the printhead. Download characters can go one better: they can use as many as eight of the nine wires in the print head. So our grid is eight dots high.

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

One bit in the download character definition command is used to tell SR-10/15 whether a character is to be treated as a descender or not. We'll get to the command in due time. For now, if your

character uses the top eight dots, write in a one next to the word "Descender" on the layout grid; if it uses the bottom eight dots, write in a zero. In our example, we'll want the bottom of the flask to line up with the baseline of the other characters, so it will *not* be a descender. As shown in Figure 9-5, we've written in a "1" on our grid.

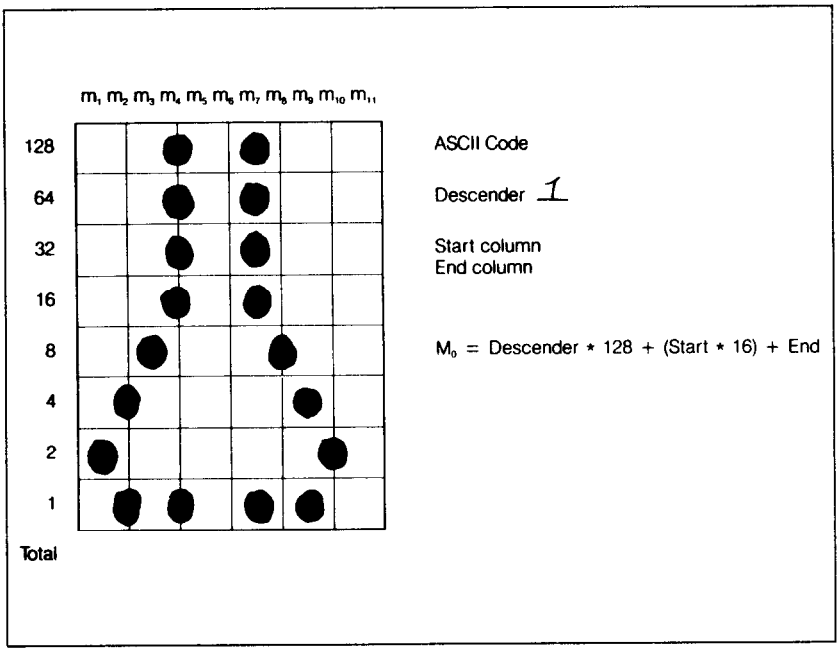


Figure 9-5. We've designed a character and decided that it would not be a descender, hence the "1" written in.

■ Rule 2: Dots cannot overlap

As you can see in Figure 9-5 our flask has a nearly continuous outline. But, you may ask, why not make it a *really* solid line and print all the intermediate dots, as shown in Figure 9-6? Because the dots that straddle the vertical lines in the grid actually overlap those inside the boxes. If we tried to print overlapping dots, the SR-10/15 printhead would have to slow down and back up to print both dots—not very efficient! To avoid this inefficiency, SR-10/15 will not allow you to define a character like Figure 9-6. (Actually, you can define it, but when it prints, SR-10/15 will leave out the overlapping dots, so that it would print like Figure 9-5.)

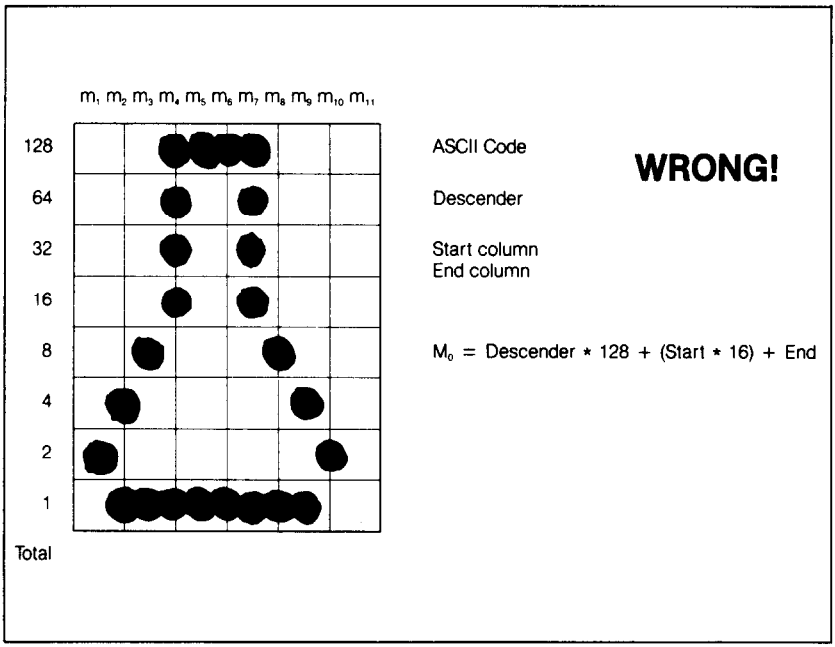


Figure 9-6. Dots cannot overlap; those in immediately adjacent "half columns" will be ignored when the character is printed.

■ Add up each column of dots

Now it's time to give our creative side a break and get down to some basic arithmetic. That's where the numbers down the left side of the grid come in. Notice that there is a number for each row of dots and that each number is twice the number below it. By making these numbers powers of two we can take any combination of dots in a vertical column and assign them a unique

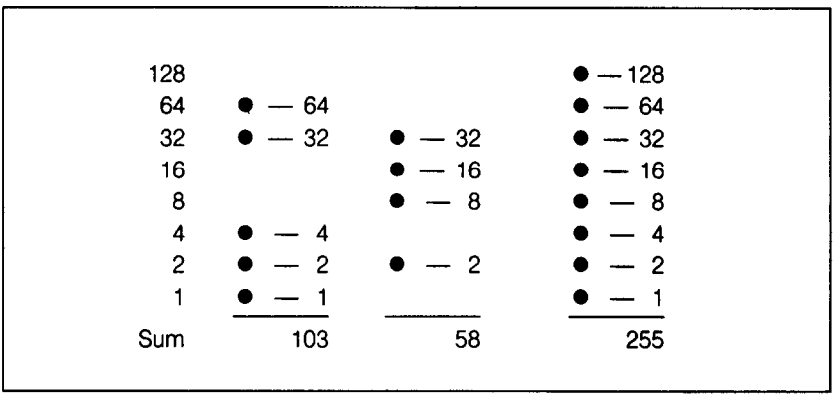


Figure 9-7. By adding the values of each dot in a column, you'll get a unique description for any combination of dots.

value. Some examples will make this clearer. As shown in Figure 9-7, if we add the numbers for the dots that print in a column, the sum will be a number in the range of 0 to 255. Each number from 0-255 represents a unique combination of dots.

So add up the values of the dots in each column using this system. In Figure 9-8 we've shown our grid with the sums of the columns filled in across the bottom (see if these agree with your answers!). Across the top of the grid you've probably noticed the cryptic labeling of each column: *m1, m2, m3,* etc. These labels correspond to the labels in the command syntax statement, which we'll get to shortly.

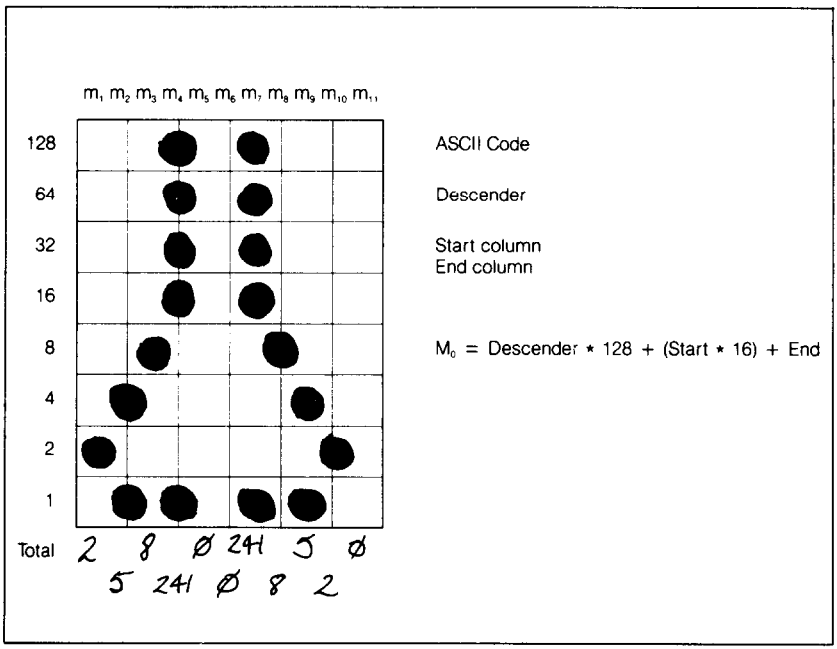


Figure 9-8. Add the values of the dots in each column and write the sum of each column at the bottom.

■ Assigning a value to your character

We've done a pretty thorough job of designing and describing a user-defined character. But the SR-10/15 has room for 240 download characters—how does it know which user-defined character we want to print? Exactly the same way it knows which standard character we want to print: every character is assigned a unique number.

The standard characters are assigned the ASCII codes—numbers from 0 to 255. For the download character sets you

can define any positions except the defined control code positions. This means that once a character is defined and assigned a value (and the download character set is selected), you can use that character on the printer the same way you would any standard character. You can send the character with the same ASCII value (for instance, if you had assigned your character a code of 66, it would print each time you sent a character "B" to the printer). You can also access the character from a BASIC program with the CHR$ function—in this case LPRINT CHR$(66) would print the character.

Except for the limitation that download characters must be avoid the defined control code positions, there are no rules or restrictions on the use of numbers. This means you can use whatever is most convenient for you—perhaps seldom-used keys can be replaced by more useful characters. In our example, we'll assign the flask a value of 160, which is the code for the character "ノ" or "á". A rather arbitrary selection, but SR-10/15 doesn't care!

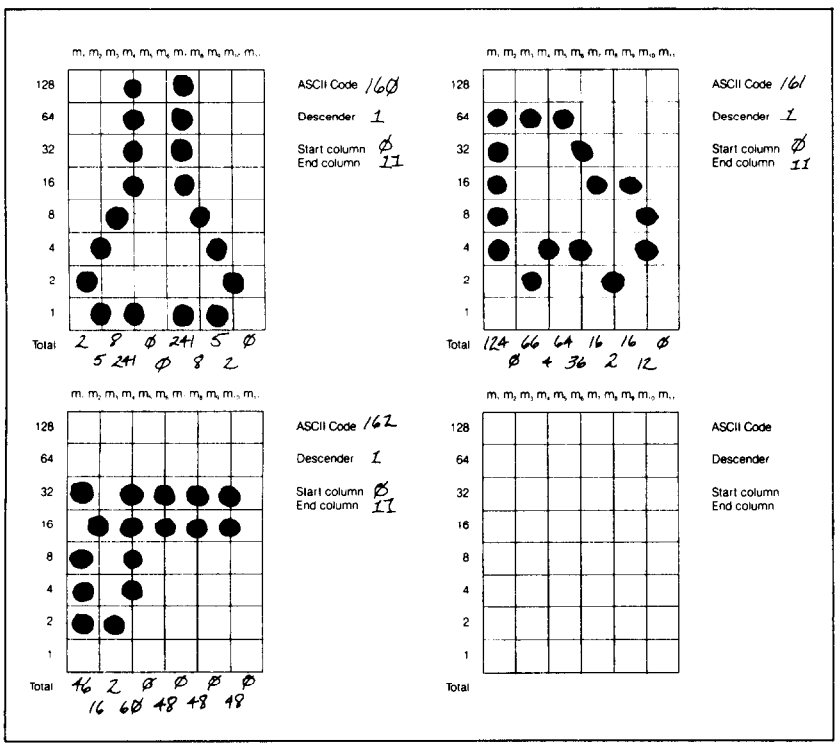


Figure 9-9. Character designs for the three graph symbols.

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

■ Download character definition command

You've read through a long explanation of download characters and we haven't even told you the command syntax yet! Now the wait is over. This is the most complex command in the SR-10/15 repertoire and now you've got the necessary knowledge to implement it. Here it is:

(For STAR mode)

```
<ESC> "*" 1 n1 n2 mØ m1 m2 m3 m4 m5 m6 m7 m8
   m9 m1Ø m11
```

(For IBM mode)

```
<ESC> "&" CHR$(Ø) n1 n2 mØ m1 m2 m3 m4 m5 m6
   m7 m8 m9 m1Ø m11
```

Like the other SR-10/15 commands, it starts with an < ESC > (CHR$(27)). The next character is an asterisk (*) (CHR$(42)) followed by 1, or an ampersand (&) (CHR$(38)) followed by a CHR$(0).

n1 and *n2* are used to specify the ASCII values of the characters you are defining. The reason that there are two bytes reserved for this is that SR-10/15 allows you to define many characters with just a single command. *n1* is used to specify the beginning of a range of characters to be defined; *n2* specifies the end of the range. For instance, if you wanted to change the appearance of the numerals from 0 to 9 (which have ASCII codes 48 through 57) for the STAR mode, the command would begin with < ESC > "*" CHR$(1) CHR$(48) CHR$(57)... Of course, you can also define individual characters by making *n1* and *n2* equal.

m0 is called the attribute byte, for it describes two attributes of the character we have designed: descender data and proportional width information. A byte consists of eigth bits. In the attribute byte, the first (high order) bit is used for the descender data, and the last seven bits are used for proportional widths.

We'll be discussing proportional character widths in detail later in this chapter; for now, we'll leave it at 11. The descender data was discussed earlier: to use the top eight pins, this bit should be 1; to use the bottom eight pins this bit should be 0. Figure 9-10 shows the bits of the attribute byte as we'll use them for our flask character. By now you've probably seen an easier way to determine the value of the attribute byte. Instead of translating everything to binary, merely assign the descender data a value of 128 (the value of the first bit) if you *dont't want* descenders, or 0 if you *want* descenders. Then just add the descender data to the proportional width. This way, it's simply a matter of adding two decimal numbers. (In our case, it's 128 + 11 = 139.)

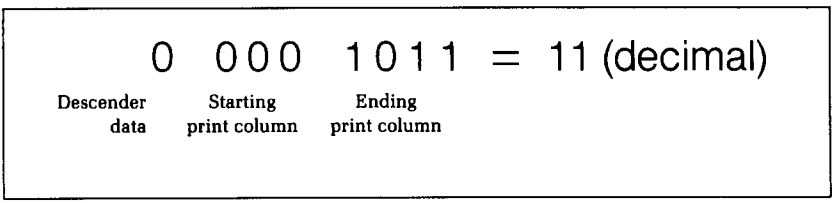


Figure 9-10. The attribute byte (*m0*) for our flask character.

You'll probably recognize *m1...m11* from the top of our layout grid. That's right, each column is described by one byte. Now we've got everything we need to download one character to the printer. The complete command for our flask character with the STAR mode is shown in Figure 9-11.

```
CHR$(27)CHR$(42)CHR$(1)CHR$(16Ø)CHR$(16Ø)CHR$(139)
 Escape     *        1       n1        n2        mØ
CHR$(2)CHR$(5)CHR$(8)CHR$(241)CHR$(Ø)CHR$(Ø)
   m1       m2       m3       m4       m5       m6
CHR$(241)CHR$(8)CHR$(5)CHR$(2)CHR$(Ø)
    m7       m8       m9       m1Ø      m11
```

Figure 9-11. This is the complete command to send our flask character to the SR-10/15 printer.

Now let's send the information to the printer. But, before you send the information, be sure that the DIP switch 1-5 is set off position. If not, set it correctly while the power is off, then, turn the power on again. The following program will send the character definitions for all three characters with the STAR mode to the printer. Enter the program and run it.

```
10 LPRINT CHR$(27) "*" CHR$(1) CHR$(160)
   CHR$(162);
20 FOR N = 160 TO 162
30 FOR M = 0 TO 11
40 READ MM
50 LPRINT CHR$(MM);
60 NEXT M
70 NEXT N
80 LPRINT
90 DATA 139,2,5,8,241,0,0,241,8,5,2,0
100 DATA 139,124,0,66,4,64,36,16,2,16,12,0
110 DATA 139,46,16,2,60,0,48,0,48,0,48,0
```

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

PRINTING DOWNLOAD CHARACTERS

You've now defined and sent three characters to SR-10/15. But how do you know that? If you try printing those characters now (type LPRINT CHR$(160) CHR$(161) CHR$(162)) you don't get a flask, car and gun. Instead you get... ノ ヽ ㄴ or áíó. That's because the download characters are stored in a different part of SR-10/15's memory. To tell it to look in download character RAM instead of standard character ROM it requires another command:

(For STAR mode)

<ESC> "$" *n*

(For IBM mode)

<ESC> "%" *n* 0

This command is used to select the download character set (if $n=1$) or to select the standard character set (if $n=0$). Let's try it out. Enter this command:

```
LPRINT CHR$(27) "$1" CHR$(160) CHR$(161) CHR$(162)
```

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

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

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

```
1Ø LPRINT CHR$(27) "$1"
2Ø FOR I=33 TO 126 : LPRINT CHR$(I); : NEXT I
3Ø FOR I=16Ø TO 254 : LPRINT CHR$(I); : NEXT I
4Ø LPRINT
5Ø LPRINT CHR$(27) "$Ø"
```

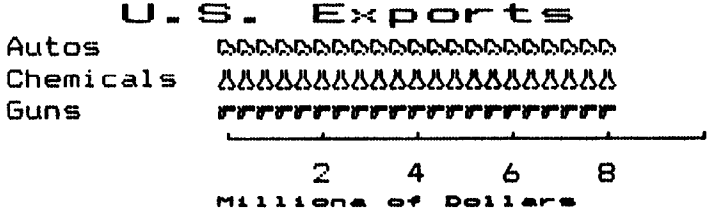
As you can see, in addition to the characters you have defined (they are the last ones on the printout), SR-10/15 also printed all characters. This makes it very easy to combine user-defined characters with regular text.

If SR-10/15 didn't have this feature, mixing download and standard characters would be rather inconvenient: every time you wanted to use a download character you would have to switch back and forth between character sets.

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

```
5 ESC$= CHR$(27) : TB$=CHR$(9)
1Ø LPRINT ESC$ "*1" CHR$(16Ø) CHR$(162);
2Ø FOR N=16Ø TO 162
3Ø FOR M=Ø TO 11
4Ø READ MM
5Ø LPRINT CHR$(MM);
6Ø NEXT M
7Ø NEXT N
8Ø LPRINT
9Ø DATA 139,2,5,8,241,Ø,Ø,241,8,5,2,Ø
1ØØ DATA 139,124,Ø,66,4,64,36,16,2,16,12,Ø
11Ø DATA 139,46,16,2,6Ø,Ø,48,Ø,48,Ø,48,Ø
12Ø LPRINT ESC$ "D" CHR$(11) CHR$(Ø)
13Ø LPRINT CHR$(14) "   U.S. Exports"
```

```
14Ø LPRINT ESC$ "$1";
15Ø LPRINT "Autos" TB$;
16Ø FOR I=.4 TO 9.3 STEP .4 : LPRINT CHR$(161); :
    NEXT
17Ø LPRINT
18Ø LPRINT "Chemicals" TB$;
19Ø FOR I=.4 TO 8.7 STEP .4 : LPRINT CHR$(16Ø); :
    NEXT
2ØØ LPRINT
21Ø LPRINT "Guns" TB$;
22Ø FOR I=.4 TO 1.4 STEP .4 : LPRINT CHR$(162); :
    NEXT
23Ø LPRINT
24Ø LPRINT ESC$ "$Ø";
25Ø LPRINT TB$ CHR$(246) CHR$(241) CHR$(241);
26Ø SCALE$ =
    CHR$(241)+CHR$(241)+CHR$(248)+CHR$(241)+CHR$(241)
27Ø FOR I=2 TO 8 STEP 2 : LPRINT SCALE$; : NEXT
28Ø LPRINT CHR$(241) CHR$(241) CHR$(247)
29Ø LPRINT TB$ "  ";
3ØØ FOR I=2 TO 8 STEP 2 : LPRINT "  " I; : NEXT
31Ø LPRINT
32Ø LPRINT TB$ ESC$ "SØ" "Millions of Dollars" ESC$
    "T"
33Ø END
```



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

ERASING DOWNLOAD CHARACTER DEFINITIONS

After you have defined a set of characters (a whole new alphabet, perhaps) you may want to go back to using *mostly* standard characters with a few new user-defined characters mixed in. Rather than turning SR-10/15 off (which erases all of the current settings, including download characters), you can send a command which will restore the default characters. This command copies all the characters from the standard character ROM into download RAM:

(For STAR mode)

<ESC> "*" Ø

(For IBM mode)

<ESC> ":" Ø Ø Ø

Since it will copy *all* characters into the download area, it will wipe out any characters that are already there. So it's important to send this command to the printer *before* you send any download characters you want to define.

DEFINING PROPORTIONAL CHARACTERS

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

Besides being able to specify the actual width of the character, SR-10/15 allows you to specify the position in the standard grid where the character will print. You must specify the dot column in which the printed character starts and the dot column in which the character ends. Why, you may ask, would you want to define a character this way instead of merely defining the overall width of the character? Because SR-10/15's proportional character definitions can also be used to print normal width characters, which are eleven dot columns wide. And by centering even the narrow characters in the complete grid (look at the "i" in Figure 9-12) they will look good even when you aren't printing them proportionally.

The command format for proportional character definition is exactly the same as you have learned; the only difference is the

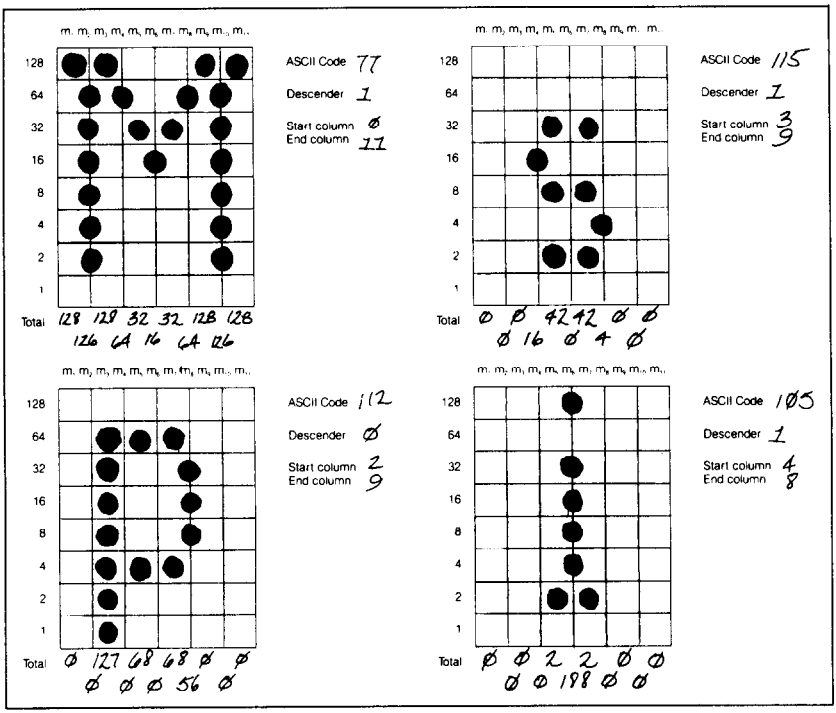


Figure 9-12. These download characters are defined as proportional characters.

attribute byte, *m0*. As you know, the first bit of *m0* is used to specify whether the character is a descender or not. The next three bits are used to specify the starting print column (acceptable values are 0 to 7). The last four bits specify the ending print column (acceptable values are 4 to 11). The minimum character width is five dots (so you could not, for instance, specify a starting column of 6 and an ending column of 8, even though those are both within the acceptable range). If you inadvertantly give an incorrect width value, however, SR-10/15 is forgiving: it will automatically revert to the default width of eleven dot columns.

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

The examples in Figure 9-12 show characters of different widths. These characters are defined in the program below; the output is shown in Figure 9-13.

```
1Ø FOR N=1 TO 4
2Ø READ N1
3Ø LPRINT CHR$(27) "*" CHR$(1) CHR$(N1) CHR$(N1);
4Ø FOR M=Ø TO 11
5Ø READ MM
6Ø LPRINT CHR$(MM);
7Ø NEXT M
8Ø NEXT N
9Ø LPRINT "     Mississippi"
1ØØ LPRINT
11Ø LPRINT "Standard characters without proportional
    spacing"
12Ø LPRINT
13Ø LPRINT
14Ø LPRINT CHR$(27) "$1" "     Mississippi"
15Ø LPRINT CHR$(27) "$Ø"
16Ø LPRINT "Download characters without proportional
    spacing"
17Ø LPRINT
18Ø LPRINT CHR$(27) "$1"
19Ø LPRINT CHR$(27) "p1" "     Mississippi"
2ØØ LPRINT CHR$(27) "pØ" CHR$(27) "$Ø"
21Ø LPRINT "Download characters with proportional
    spacing"
22Ø DATA
    77,139,128,126,128,64,32,16,32,64,128,126,128
23Ø DATA 1Ø5,2ØØ,Ø,Ø,Ø,Ø,2,188,2,Ø,Ø,Ø,Ø
24Ø DATA 112,41,Ø,Ø,127,Ø,68,Ø,68,56,Ø,Ø,Ø
25Ø DATA 115,185,Ø,Ø,Ø,16,42,Ø,42,4,Ø,Ø,Ø
```

```
     Missisippi
Standard characters without proportional spacing

     Mississippi
Download characters without proportional spacing

     Mississippi
Download characters with proportional spacing
```

Figure 9-13. This printout shows the same text, printed with the same download characters, in both normal and proportional widths.

One thing to remember about defining proportional characters: a character cannot be wider than the specified width. That seems obvious enough! For example, if you specify a width of 6 for a character (starting in column 1 and ending in column 6), the seventh through eleventh columns of dots (if you specified any) will not print. You must, however, send information (even if it is 0) for those columns when you defined a character; SR-10/15 expects eleven characters following the <ESC> "*" 1 *n1 n2 m0* or the <ESC> "&" CHR$(0) *n1 n2 m0* sequence.

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

Table 9-1
Download character commands

Function	Mode	Control code
Define download character	STAR	<ESC> * 1 *n1 n2 m0 m1...m11*
	IBM	<ESC> "&" CHR$(0) *n1 n2 m0 m1...m11*
Copy ROM to download RAM	STAR	<ESC> "*" 0
	IBM	<ESC> ":" 0 0 0
Download characters ON	STAR	<ESC> "$" 1
	IBM	<ESC> "%" 1 0
Download characters OFF	STAR	<ESC> "$" 0
	IBM	<ESC> "%" 0 0

■ Connecting characters

As we noted earlier, it's possible to connect proportional width characters. This can be useful for creating logos or other characters which are larger than one normal character. It also makes it possible to create connecting scripts, like handwritng. The trick to this is to specify the width in the attribute byte to be exactly the same as the number of columns of dots that the character (or partial character) occupies. And, if you change the vertical spacing to 7/72″ (use the <ESC> "1" command), you can make characters connect vertically. This allows you to make very large characters indeed!

In the program that follows, we've used this technique to create some large numbers. Each digit is actually made up of four characters—two horizontally by two vertically. This means, of course, that you must define and print four characters for each finished digit. We assigned the upper left quadrant of each digit to ASCII codes from 160 to 169, the upper right quadrant to codes 170 to 179, and so on. Figure 9-14 shows how one digit is defined, and Figure 9-15 shows the final output of our program.

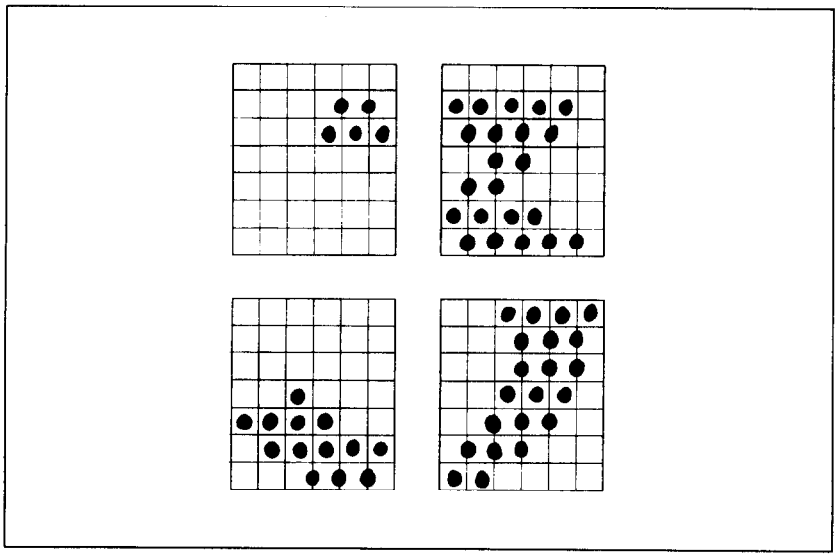


Figure 9-14. Each digit is made up of four individual characters.

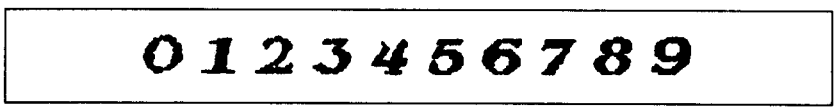


Figure 9-15. The output for characters like this must be carefully planned.

```
1Ø 'Program to define and print numerals
2Ø 'Each numeral is made up of 4 characters (2 wide
   x 2 high)
3Ø OPEN "LPT1:" AS #1 : WIDTH #1, 255
4Ø DOWN.CHAR.PROP$ = CHR$(27)+"p1"+CHR$(27)+"$1"
5Ø NOT.DOWN.CHAR.PROP$ = CHR$(27)+"pØ"+CHR$(27)+"$Ø"
6Ø LINE.7$ = CHR$(27)+"1" : LINE.12$ = CHR$(27)+"2"
7Ø PRINT #1, CHR$(27) "&1" CHR$(16Ø) CHR$(2ØØ);
8Ø FOR I = 16Ø TO 2ØØ
1ØØ READ MØ
11Ø PRINT #1, CHR$(MØ);
12Ø FOR S = 1 TO 11
```

```
13Ø READ MS
14Ø PRINT #1, CHR$(MS);
15Ø NEXT S
16Ø NEXT I
17Ø '
18Ø ASCII = 16Ø                    'START OF DOWNLOAD
    CHARACTERS
19Ø FOR NUM = Ø TO 9               'NUMERALS Ø TO 9
2ØØ NUMERAL.TOP$(NUM) = CHR$(ASCII + Ø) + CHR$(ASCII
    + 1)
21Ø NUMERAL.BOT$(NUM) = CHR$(ASCII + 2) + CHR$(ASCII
    + 3)
22Ø ASCII = ASCII + 4
23Ø NEXT NUM
24Ø BLANK$ = CHR$(2ØØ)
25Ø PRINT #1, DOWN.CHAR.PROP$; LINE.7$
26Ø FOR NUM = Ø TO 9
27Ø PRINT #1, NUMERAL.TOP$(NUM);BLANK$;
28Ø NEXT NUM
29Ø PRINT #1, CHR$(1Ø)
3ØØ FOR NUM = Ø TO 9
31Ø PRINT #1, NUMERAL.BOT$(NUM);BLANK$;
32Ø NEXT NUM
33Ø PRINT #1, NOT.DOWN.CHAR.PROP$; LINE.12$
34Ø 'ZERO
35Ø DATA 11, Ø, 6, 8, 22, 8, 52, 12Ø, 112, Ø, 64,
    128
36Ø DATA 11, 64, 128, 64, 128, 96, 16, 1ØØ, 3Ø, 4,
    3Ø, 4
37Ø DATA 11, 192, 48, 2ØØ, 48, 2Ø4, Ø, 6, Ø, 6, Ø, 6
38Ø DATA 11, Ø, 4, Ø, 12, Ø, 56, 192, 48, 192, 32,
    192
39Ø 'ONE
4ØØ DATA 11, Ø, Ø, Ø, Ø, Ø, 32, Ø, 32, Ø, 32, 126
41Ø DATA 9, 48, 78, 48, 78, 48, 64, Ø, Ø, Ø, Ø, Ø
42Ø DATA 11, 2, Ø, 2, Ø, 2, Ø, 2, 4, 1Ø, 244, 1Ø
43Ø DATA 9, 244, 1Ø, 244, 2, Ø, 2, Ø, 2, Ø, Ø, Ø
44Ø 'TWO
45Ø DATA 11, Ø, Ø, Ø, Ø, Ø, 48, 8, 112, Ø, 96, Ø
46Ø DATA 11, 192, Ø, 192, Ø, 98, 28, 98, 28, 32, 24,
    Ø
47Ø DATA 11, 2, Ø, 2, 4, 2, 4, 1Ø, 4, 1Ø, 2Ø, 2
48Ø DATA 11, 52, 2, 1ØØ, 13Ø, 68, 13Ø, 4, 1Ø, 4, 24,
    Ø
49Ø 'THREE
```

```
500 DATA 11, 0, 0, 0, 0, 0, 0, 32, 64, 32, 64, 32
510 DATA 11, 68, 42, 68., 58, 68, 50, 68, 34, 64, 2,
    0
520 DATA 11, 8, 0, 12, 0, 28, 2, 12, 2, 4, 2, 4
530 DATA 11, 2, 4, 2, 12, 144, 108, 144, 104, 144,
    96, 128
540 'FOUR
550 DATA 11, 0, 0, 0, 0, 0, 0, 2, 36, 30, 4, 8
560 DATA 11, 0, 0, 2, 4, 2, 28, 2, 60, 64, 48, 0
570 DATA 11, 0, 16, 32, 80, 160, 80, 160, 16, 32,
    18, 32
580 DATA 11, 30, 100, 154, 100, 154, 96, 146, 32,
    16, 96, 0
590 'FIVE
600 DATA 11, 0, 0, 0, 0, 2, 4, 42, 76, 50, 68, 34
610 DATA 10, 68, 34, 68, 34, 68, 34, 64, 34, 64, 0,
    0
620 DATA 10, 0, 4, 24, 166, 24, 134, 0, 2, 0, 2, 0
630 DATA 11, 2, 0, 6, 128, 12, 240, 12, 240, 8, 240,
    0
640 'SIX
650 DATA 11, 0, 6, 0, 14, 0, 30, 0, 58, 0, 102, 0
660 DATA 11, 70, 0, 70, 0, 70, 0, 98, 0, 112, 0, 96
670 DATA 11, 224, 16, 232, 16, 236, 16, 198, 0, 130,
    0, 2
680 DATA 11, 0, 6, 0, 14, 128, 124, 128, 120, 128,
    112, 0
690 'SEVEN
700 DATA 11, 0, 8, 16, 96, 16, 96, 16, 96, 16, 96,
    16
710 DATA 9, 98, 16, 102, 16, 108, 16, 96, 0, 64, 0,
    0
720 DATA 11, 0, 2, 0, 6, 0, 30, 0, 62, 0, 120, 128
730 DATA 9, 96, 128, 0, 0, 0, 0, 0, 0, 0, 0, 0
740 'EIGHT
750 DATA 11, 0, 0, 0, 0, 24, 36, 24, 102, 24, 102, 0
760 DATA 11, 194, 0, 194, 0, 198, 56, 68, 56, 68,
    56, 0
770 DATA 11, 48, 72, 52, 200, 54, 200, 6, 128, 2, 0,
    2
780 DATA 11, 0, 6, 128, 14, 240, 12, 240, 8, 112, 0,
    0
790 'NINE
800 DATA 11, 0, 0, 30, 32, 30, 96, 30, 96, 0, 192, 0
```

```
81Ø DATA 11, 192, Ø, 192, Ø, 194, 32, 222, 32, 94,
    32, 3Ø
82Ø DATA 11, 12, Ø, 28, Ø, 142, Ø, 198, Ø, 198, Ø,
    198
83Ø DATA 11, Ø, 2Ø6, Ø, 156, Ø, 248, Ø, 24Ø, Ø, 224,
    Ø
84Ø 'SPACE
85Ø DATA 11, Ø, Ø, Ø, Ø, Ø, Ø, Ø, Ø, Ø, Ø, Ø
```

SUMMARY

Control code	Function
<ESC> "*" 1 *n1 n2 m0 m1 ...m11*	Defines download character into RAM (for STAR mode)
<ESC> "&" CHR$(0) *n1 n2 m0 m1 ...m1*	Defines download character into RAM (for IBM mode)
<ESC> "*" 0	Copies fonts in ROM into download RAM (for STAR mode)
<ESC> ":" 0 0 0	Copies fonts in ROM into download ram (for IBM mode)
<ESC> "$" 1	Selects the download character set (for STAR mode)
<ESC> "%" 1 0	Selects the download character set (for IBM mode)
<ESC> "$" 0	Cancels download character set (for STAR mode)
<ESC> "%" 0 0	Cancels download character set (for IBM mode)

CHAPTER 10

PRINTING WITH DOT GRAPHICS

Subjects covered in this chapter include:

- **SR-10/15's bit image graphics capabilities**
- **Printing a pre-defined shape**
- **Plotting a calculated shape**
- **High resolution graphics**

In Chapter 9 you were introduced to a form of computer graphics; you were able to actually define characters dot by dot. In this chapter you'll learn to use the same principles to make SR-10/15 print whole pages of dot graphics! We'll show you how to use dot graphics to create "super download characters." In addition, you'll see how your SR-10/15 printer can be used as a graphic plotter. This can have some practical business applications as well as create some terrific computer art!

COMPARING DOT GRAPHICS WITH DOWNLOAD CHARACTERS

A good understanding of dot graphics requires an understanding of how dot matrix printers work; you may want to review the first few pages of Chapter 9. The principles for dot graphics are the same as those for download characters.

There are some differences in the way they are implemented however. While download commands can be used to define a character between four and eleven columns of dots wide, dot graphics commands can be used to define a shape as narrow as one column of dots wide or as wide as 3264 dots on an SR-15!

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

(if you're not sure which type of interface your computer has, check the appendix for your computer).

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

- If the graphic image to be printed is wider than 11 dots or higher than 8 dots
- If an image is to be printed just one time, as opposed to a frequently used "text" character
- If you want higher resolution (SR-10/15 can print as many as 240 dots per inch in dot graphics mode; text mode, which includes download characters, prints 60 dots per inch)

USING THE DOT GRAPHICS COMMANDS

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

<ESC> "K" *n1 n2 m1 m2* .....

Just like many of the other codes you have learned, the command starts with an escape sequence (< ESC > "K" in this case). But unlike SR-10/15's other codes there can be any number of graphics data bytes following the command. That's where *n1* and *n2* come in; they are used to tell SR-10/15 how many bytes of graphics data to expect.

■ Specifying the number of columns of dots

To figure the values of *n1* and *n2*, you'll need to figure out how wide your graphic image will be (remember that there are 60 columns of dots per inch in normal density). Then comes the fun part: converting one number (the number of columns of dots) into two! Why is it necessary to use two numbers to tell SR-10/15 the number of graphics codes to expect? Because the largest number we can send in one byte (that's what the BASIC CHR$() function sends: one byte) is 255. And with normal density graphics

it's possible to have a graphics image as wide as 480 dots on SR-10 or 816 dots on SR-15. So to figure out how many columns of graphics data to expect, SR-10/15 multiplies *n2* by 256 and adds the value of *n1*. If you divide the number of columns by 256, then *n2* is the quotient and *n1* is the remainder (why not let your computer figure it out for you: if the number of columns is assigned to variable X, then *n1*=X MOD 256 and *n2*=INT(X/256)). Table 10-1 might make things even easier.

Table 10-1
Calculating *n1* and *n2* for graphics

If the number of columns, x, ranges from:	then *n1* is:	and *n2* is:
1 to 255	x	0
256 to 511	x−256	1
512 to 767	x−512	2
768 to 1023	x−768	3
1024 to 1279	x−1024	4
1280 to 1535	x−1280	5
1536 to 1791	x−1536	6
1792 to 2047	x−1792	7
2048 to 2303	x−2048	8
2304 to 2559	x−2304	9
2560 to 2815	x−2560	10
2816 to 3071	x−2816	11
3072 to 3264	x−3072	12

■ Specifying the graphics data

Now that we've told SR-10/15 how much data to expect, we better figure out how to send that information! Just as you do with download characters, with dot graphics you have control over the firing of every single pin on SR-10/15's print head. In Figure 10-1, you can see that we've labeled each pin on the print head with a number, as we did with download characters (you should note one important difference: this time the *top* pin has the highest value; for download character definitions it is the bottom pin). And specifying pins to fire is done in the same way: to fire the second pin from the top, for instance, send a CHR$(64). Firing several pins at once is done in a similar fashion. For example, to print the first, third, and fourth dots, add their values (128 + 32 + 16) to send this total: CHR$(176). This is one byte of graphics data; it would replace *m1* in our format statement on page 104.

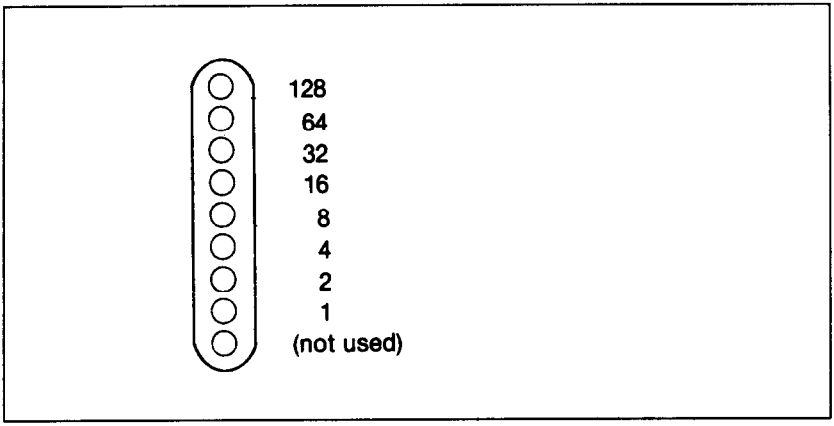


Figure 10-1. Starting with the most significant bit at the top, each pin of the print head is assigned a value which is a power of two. Note that for 7-bit computers, the top pin cannot be used.

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

```
1Ø 'Demo dot graphics.
2Ø PI = 3.14159
3Ø WID = 1ØØ
4Ø OPEN "LPT1:" AS #1 : WIDTH #1,255
5Ø PRINT #1,CHR$(27) "K" CHR$(WID MOD 256)
   CHR$(INT(WID/256)) ;
6Ø FOR I = Ø TO WID-1
7Ø PRINT #1,CHR$(2^INT((1+SIN(I*PI/32))*3.5+.5) ;
8Ø NEXT I
9Ø LPRINT
1ØØ CLOSE #1
```

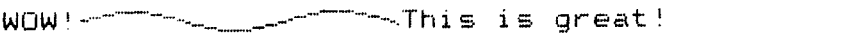
In line 50 we've selected normal density graphics and said that 100 characters of graphics data would follow. The loop between line 60 and 80 is repeated to plot 100 points along a curve. This is an example of plotting a very simple mathematical function (a sine wave) to create a design. Later in this chapter we'll show something more complex. The mathematical concepts (such as sine and pi) demonstrated here are not important; you don't have to be a math whiz to use SR-10/15's graphics.

■ Combining text and graphics

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

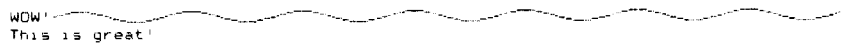
```
45 PRINT #1,"WOW!" ;
85 PRINT #1,"This is great!" ;
```

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



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

```
3Ø WID = 1ØØØ
```



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

As you can see, SR-10/15 printed graphics up to the end of the line, then ignored the rest of the graphics data and returned to normal text on the next line.

PRINTING A DESIGN OR LOGO

Since you control the firing of every pin, you can print nearly anything with SR-10/15 that you can draw (and probably better, if you're like most computer users!). This can be used for creating "computer art" or drawing maps. Or, as we'll show you here, you can use dot graphics to print your logo at the top of each letter you print.

Designing an image to print with dot graphics is much like designing download characters. The best way to start is to lay out your image on graph paper. Since you can print eight rows (seven with a 7-bit interface) of dots with each pass of the print head, draw a heavy horizontal line every eight rows on your graph paper. And it may be helpful to write the dot values (128, 64,

32, etc.) down the left side of each row. Then after you've filled in the "dots" that you want to print, it's time to get out the old calculator again! Just as you did with download characters, add up the values of each column of dots; this makes up one byte.

In the program below, we've taken the logo graphics information and put it into BASIC DATA statements. The program itself is short and simple. The loop starting at line 100 reads the data statements into a string array variable called LOGO$. In line 170 we change the line spacing to 8/72 inch so that the lines of graphics data will connect vertically. The actual printing is done in the loop between lines 180 and 210; line 190 sends the graphics control code to SR-10/15 and line 200 sends one line of graphics data.

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

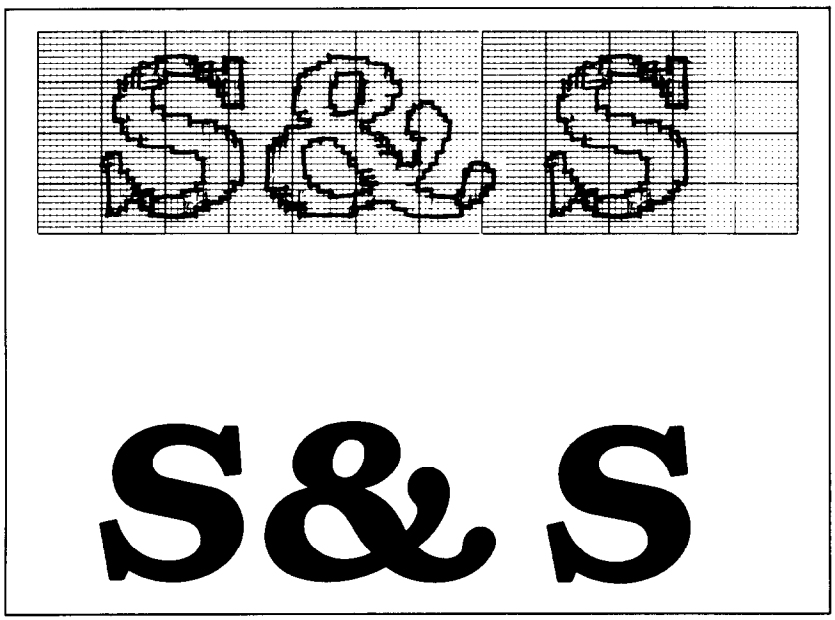


Figure 10-2. By laying out the logo on graph paper, you can calculate all of the graphics data.

```
1Ø 'Prints S&S logo.
2Ø LINE.8$ = CHR$(27)+CHR$(65)+CHR$(8)
3Ø 'Set line spacing to 1/6"
4Ø LINE.12$ = CHR$(27)+CHR$(5Ø)
5Ø 'Select dot graphics
```

```
6Ø GRAPHIC$ = CHR$(27)+CHR$(75)
7Ø DIM LOGO$(4)
8Ø WIDTH "LPT1:",255
9Ø ' READ DATA
1ØØ FOR ROW = 1 TO 4
11Ø FOR COLUMN = 1 TO 1ØØ
12Ø READ P
13Ø LOGO$(ROW) = LOGO$(ROW) + CHR$(P)
14Ø NEXT COLUMN
15Ø NEXT ROW
16Ø ' PRINT LOGO
17Ø LPRINT LINE.8$;
18Ø FOR ROW = 1 TO 4
19Ø LPRINT GRAPHIC$;CHR$(1ØØ);CHR$(Ø);
2ØØ LPRINT LOGO$(ROW)
21Ø NEXT ROW
22Ø LPRINT LINE.12$
23Ø 'ROW 1
24Ø DATA Ø,Ø,Ø,Ø,1,3,7,7,7,15
25Ø DATA 14,14,14,14,14,7,7,3,3,15
26Ø DATA 15,15,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
27Ø DATA Ø,1,3,3,7,7,15,14,14,14
28Ø DATA 14,15,7,7,7,3,Ø,Ø,Ø,Ø
29Ø DATA Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
3ØØ DATA Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
31Ø DATA Ø,Ø,Ø,Ø,1,3,7,7,7,15
32Ø DATA 14,14,14,14,14,7,7,3,3,15
33Ø DATA 15,15,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
34Ø 'ROW 2
35Ø DATA Ø,Ø,6Ø,255,255,255,255,255,143,15
36Ø DATA 7,7,7,7,3,3,3,131,193,241
37Ø DATA 24Ø,24Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,1
38Ø DATA 121,253,253,255,255,255,143,7,7,7
39Ø DATA 31,253,252,248,248,24Ø,192,Ø,7,15
4ØØ DATA 31,31,15,7,3,Ø,Ø,Ø,Ø,Ø
41Ø DATA Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
42Ø DATA Ø,Ø,6Ø,255,255,255,255,255,143,15
43Ø DATA 7,7,7,7,3,3,3,131,193,241
44Ø DATA 24Ø,24Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
45Ø 'ROW 3
46Ø DATA Ø,31,31,3,129,128,192,192,192,192
47Ø DATA 192,224,224,224,224,24Ø,255,255,255,255
48Ø DATA 255,127,Ø,Ø,Ø,Ø,63,127,255,255
49Ø DATA 255,255,193,128,128,128,128,192,224,24Ø
```

```
500 DATA 252,255,255,255,127,63,31,7,7,31
510 DATA 254,252,248,224,128,0,0,3,7,7
520 DATA 7,3,0,0,0,0,0,0,0,0
530 DATA 0,31,31,3,129,128,192,192,192,192
540 DATA 192,224,224,224,224,240,255,255,255,255
550 DATA 255,127,0,0,0,0,0,0,0,0
560 'ROW4
570 DATA 0,248,248,240,224,224,112,112,56,56
580 DATA 56,56,56,120,120,240,240,224,224,192
590 DATA 128,0,0,0,0,0,192,224,240,240
600 DATA 240,248,248,248,120,120,56,56,56,56
610 DATA 48,112,224,224,224,224,240,240,248,248
620 DATA 120,120,56,56,56,56,120,240,224,224
630 DATA 192,128,0,0,0,0,0,0,0,0
640 DATA 0,248,248,240,224,224,112,112,56,56
650 DATA 56,56,56,120,120,240,240,224,224,192
660 DATA 128,0,0,0,0,0,0,0,0,0
```

S&S

If you are using with the IBM mode, change the following lines to the program given above.

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

PLOTTING WITH SR-10/15

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

If designing and calculating dot graphics images by laying them out on graph paper seems too tedious to you, then let the computer do the work for you! With your computer doing the calculations

and SR-10/15 plotting the output, you can come up with some terrific business graphs, charts, and mathematical function plots.

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

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

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

```
10 ' General purpose for plotting program
20 '
30 'Set program constants.
40 MAXCOL% = 75          : MAXROW% = 14
50 DIM BIT%(MAXCOL%, MAXROW%)
60 MASK%(1) = 64         : MASK%(4) = 8
70 MASK%(2) = 32         : MASK%(5) = 4
80 MASK%(3) = 16         : MASK%(6) = 2
90 LX = 20               : LY = 20
100 LXFAC = 72/LX        : LYFAC = 87/LY
110 '
120 'Plot curve.
130 GOSUB 600
140 '
150 'Send bit image map to printer.
160 LPRINT CHR$(27) "A" CHR$(6)
170 FOR ROW% = 0 TO MAXROW%
180 A$ = ""
190 LPRINT CHR$(27) "K" CHR$(MAXCOL%) CHR$(0);
200 FOR COL% = 1 TO MAXCOL%
210 A$ = A$ + CHR$(BIT%(COL%,ROW%))
220 NEXT COL%
230 LPRINT A$ " "
240 NEXT ROW%
250 LPRINT CHR$(27) "2"
260 END
```

```
270 '
280 'Subroutine to draw a line from X1,Y1 to X2,Y2.
290 '
300 XL = X2 - X1          : YL = Y2 - Y1
310 NX = ABS(XL*LXFAC) : NY = ABS(YL*LYFAC)
320 IF NX < NY THEN NX = NY
330 NS% = INT(NX+1)
340 DX = XL/NS%           : DY = YL/NS%
350 FOR I% = 1 TO NS%
360 X1 = X1 + DX          : Y1 = Y1 + DY
370 GOSUB 400
380 NEXT I%
390 RETURN
400 '
410 'Subroutine to plot a point at X1,Y1.
420 '
430 XX = X1 * LXFAC       : YY = Y1 * LYFAC
440 COL% = INT(XX) + 1
450 ROW% = INT(YY/6)
460 XIT% = INT(YY - ROW% * 6) + 1
470 BIT%(COL%,ROW%) = BIT%(COL%,ROW%) OR MASK%(XIT%)
480 RETURN
600 '
610 ' Subroutine to plot a circle.
620 '
630 RAD = 9
640 X1 = 19               : Y1 = 10
650 FOR ANG% = 0 TO 360 STEP 10
660 RANG = ANG%*6.28/360
670 X2 = RAD*COS(RANG)+10  : Y2 = RAD*SIN(RANG)+10
680 GOSUB 270
690 NEXT ANG%
700 RETURN
```

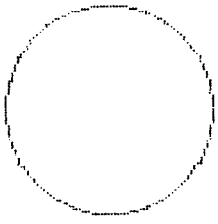
If you are using with the IBM mode, change the following lines to the program given above.

```
160 LPRINT CHR$(27) "A" CHR$(6) CHR$(27) "2"
250 LPRINT CHR$(27) "A" CHR$(12) CHR$(27) "2"
```

■ How the program works

In the program above, we've created an array called BIT%, which is dimensioned in line 50. You'll note that instead of using numeric constants to dimension the array, we used the variables MAXCOL% and MAXROW%. This way, if your computer has enough memory and you want to plot a larger image, all

you need to change are the values in line 40. The array MASK% contains the values of the dots. (In order to make this program run on the most computers, we're using only six pins for graphics. With many computers, you can use all eight available pins.) In lines 90 and 100 we've defined some other variables you'll be interested in: LX, LXFAC, LY, and LYFAC are used as scaling factors. By changing these values, you can change the size of your printed image or even distort it (you can, for example, make our circle print as an ellipse). Experiment a little bit!



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

What the program section starting at line 600 actually does is to calculate starting and ending points for a line (in our circle the "lines" are very short—sometimes the starting and ending points are the same). The coordinates of the starting point of the line are assigned to variables X1 and Y1. The line ends at point X2, Y2. When these coordinates have been calculated, a subroutine call is made to line 270. This subroutine calculates the coordinates of individual points along that line.

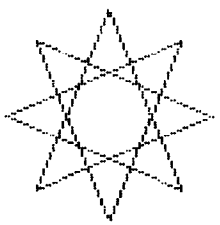
After these coordinates have been determined, the subroutine at line 400 is called. This routine turns "on" an individual dot in our array called BIT%. (Keep in mind that no printing has been done yet; the computer is still drawing the image on its "graph paper" in memory.) The way an individual dot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

When all the points have been plotted in memory, printing begins at line 150. We first set the line spacing to 6/72 inch using the <ESC> "A" command. This is so that there are no gaps between rows of dots. Then the loop from line 170 to line 240 prints the dot graphics image one line (which is six dots high)

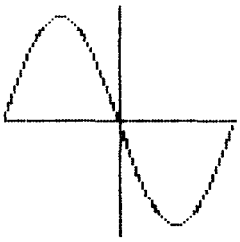
at a time. The variable A$ is used to build a string of all the columns of BIT% in a given row.

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

```
6ØØ '
61Ø 'Subroutine to plot a star.
62Ø '
63Ø RAD = 9
64Ø FOR ANG% = Ø TO 36Ø STEP 45
65Ø RANG = ANG% * 3.14159 / 18Ø
66Ø RANG2 = (ANG% + 135) * 3.14159 / 18Ø
67Ø X1 = RAD * COS(RANG) + 1Ø
68Ø Y1 = RAD * SIN(RANG) + 1Ø
69Ø X2 = RAD * COS(RANG2) + 1Ø
7ØØ Y2 = RAD * SIN(RANG2) + 1Ø
71Ø GOSUB 27Ø
72Ø NEXT ANG%
73Ø RETURN
```



```
6ØØ '
61Ø 'Subroutine to plot a sine wave.
62Ø '
63Ø X1 = Ø : Y1 = 1Ø : X2 = 2Ø : Y2 = 1Ø
64Ø GOSUB 27Ø
65Ø X1 = 1Ø : Y1 = Ø : X2 = 1Ø : Y2 = 2Ø
66Ø GOSUB 27Ø
67Ø X1 = Ø : Y1 = 1Ø
68Ø FOR X2 = Ø TO 2Ø STEP .2
69Ø Y2 = 1Ø - 9 * SIN(3.14159 * X2 / 1Ø) : GOSUB 27Ø
7ØØ NEXT X2
71Ø RETURN
```

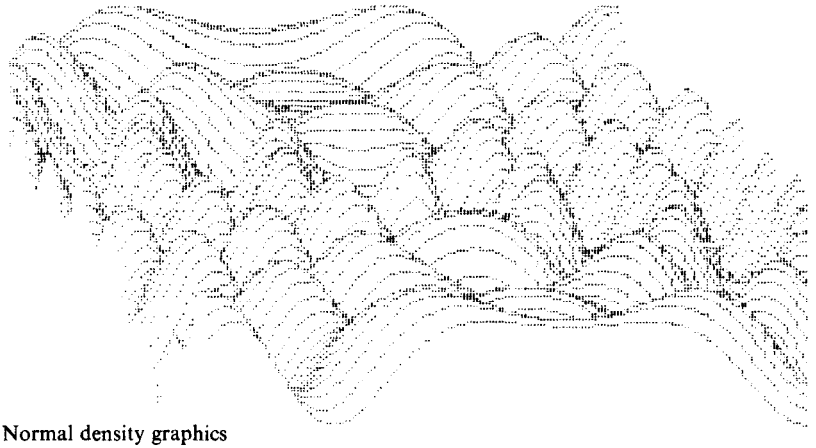


HIGH RESOLUTION GRAPHICS

Up until now all of the dot graphics printing we have done has been with SR-10/15's normal density mode. This can give you some pretty sharp images at great speed. Sometimes though, you may want to create an image with even higher resolution. SR-10/15 has seven graphics modes you can use; they're summarized in Table 10-2.

The command syntax for all of the commands is the same—just as you have learned it for the < ESC > "K" (normal density) command. The number of columns to be printed is *n1* + 256 * *n2*.

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

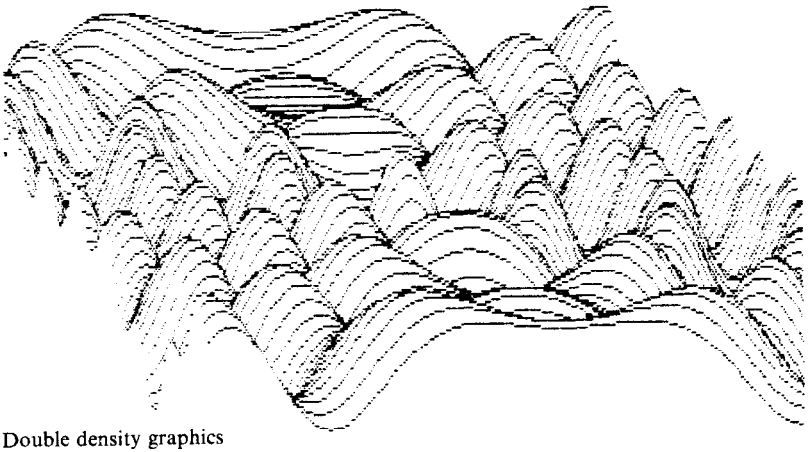


Normal density graphics

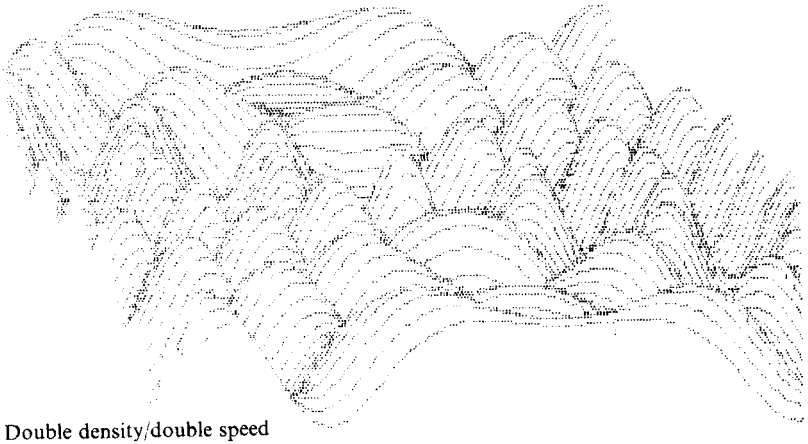
Table 10-2
Dot graphics commands

Function	Mode	Control code
Normal density (60 dots/inch)	STAR	<ESC> "K" *n1 n2 m1 m2* ..... or <ESC> "g" CHR$(0) *n1 n2 m1 m2* .....
	IBM	<ESC> "K" *n1 n2 m1 m2* ..... or <ESC> "*" CHR$(0) *n1 n2 m1 m2* .....
Double density (120 dots/inch)	STAR	<ESC> "L" *n1 n2 m1 m2* ..... or <ESC> "g" CHR$(1) *n1 n2 m1 m2* .....
	IBM	<ESC> "L" *n1 n2 m1 m2* ..... or <ESC> "*" CHR$(1) *n1 n2 m1 m2* .....
Double density with double Speed (120 dots/inch)	STAR	<ESC> "y" *n1 n2 m1 m2* ..... or <ESC> "g" CHR$(2) *n1 n2 m1 m2* .....
	IBM	<ESC> "Y" *n1 n2 m1 m2* ..... or <ESC> "*" CHR$(2) *n1 n2 m1 m2* .....
Quadruple density (240 dots/inch)	STAR	<ESC> "z" *n1 n2 m1 m2* ..... or <ESC> "g" CHR$(3) *n1 n2 m1 m2* .....
	IBM	<ESC> "Z" *n1 n2 m1 m2* ..... or <ESC> "*" CHR$(3) *n1 n2 m1 m2* .....
CRT graphics (80 dots/inch)	STAR	<ESC> "g" CHR$(4) *n1 n2 m1 m2* .....
	IBM	<ESC> "*" CHR$(4) *n1 n2 m1 m2* .....
Plotter graphics (72 dots/inch)	STAR	<ESC> "g" CHR$(5) *n1 n2 m1 m2* .....
	IBM	<ESC> "*" CHR$(5) *n1 n2 m1 m2* .....
CRT graphics type II (90 dots/inch)	STAR	<ESC> "g" CHR$(6) *n1 n2 m1 m2* .....
	IBM	<ESC> "*" CHR$(6) *n1 n2 m1 m2* .....

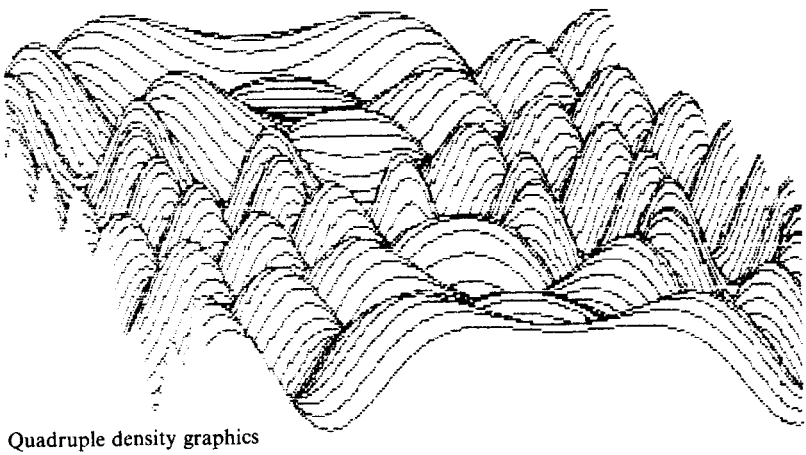
Note: If your computer does not support lowercase characters, use CHR$(103), CHR$(121), and CHR$(122) for "g", "y", and "z", respectively.



Double density graphics



Double density/double speed

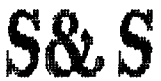


Quadruple density graphics

So if quadruple density looks so great, why not use it all the time? Let's try an experiment on your printer which will show just how the different density modes work. Using the "logo" program in this chapter, change line 60 to try each of the different modes. Just change the "CHR$(75)" to "g" + CHR$(1), "g" + CHR$(2), "g" + CHR$(3), "g" + CHR$(4), "g" + CHR$(5), "g" + CHR$(6) in turn for the STAR mode. (For the IBM mode, use "*" instead of "g".) Your print outs should look something like this:

Normal density graphics

Double density graphics



Double density/double speed

Quadruple density graphics

CRT graphics

Plotter graphics

CRT graphics type II

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

Star's engineers have given programmers a unique shortcut for program development though—double density double speed graphics. Although this mode requires just as much memory and computing time as double density, it prints at the same speed as normal density graphics. Amazing, you say? Well, it is—until you know the secret. Every other column of dots is ignored, so the output is actually the same as normal density graphics. The advantage is that you can write and debug your programs at double speed, then change to double density graphics for terrific output.

IF YOU HAVE PROBLEMS WITH BASIC

You may write some graphics programs that look just right in the listing, but the printouts aren't quite what you expected. A common problem is that the BASIC interpreter in your computer is inserting a few of its own codes. For instance, if your program generates a CHR$(13) as valid graphics data, BASIC may follow it with a CHR$(10). Another problem arises with certain computers that replace horizontal tabs (CHR$(9)) with a series of spaces (CHR$(32)). A possible solution to these problems is not to use the bottom dot (which has a value of 1). This way, you will never produce an odd number, hence, you will never have a CHR$(13) or CHR$(9). (This is why we used only six pins in our plotting program.)

That's one solution to one problem. You'll find more of each (with specific information for *your* computer) in the appropriate appendix.

SUMMARY

Control code	Function
<ESC> "K" *n1 n2 m1 m2...*	Print *n1* + 256 * *n2* columns of normal density graphics
<ESC> "L" *n1 n2 m1 m2...*	Print double density graphics
<ESC> "y" *n1 n2 m1 m2...*	Print double density graphics at double speed (for STAR mode)
<ESC> "Y" *n1 n2 m1 m2...*	Print double density graphics at double speed (for IBM mode)
<ESC> "z" *n1 n2 m1 m2...*	Print quadruple density graphics (for STAR mode)

<ESC> "Z" *n1 n2 m1 m2...* Print quadruple density graphics (for IBM mode)

<ESC> "g" *n0 n1 n2 m1 m2...*
Master graphics selection (for STAR mode)

<ESC> "*" *n0 n1 n2 m1 m2...*
Master graphics selection (for IBM mode)