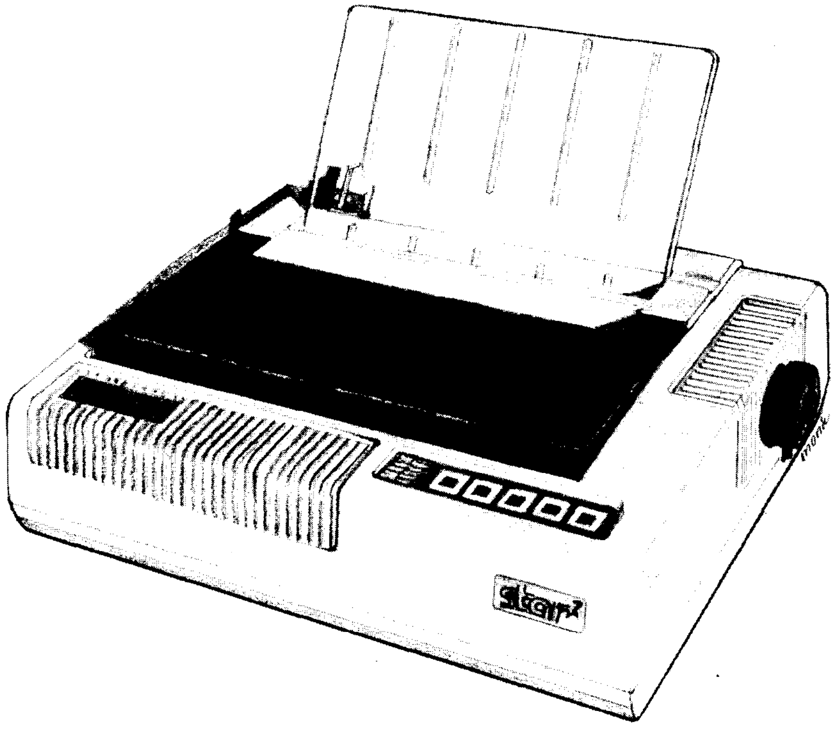
Radix

USER'S MANUAL



NOT INTENDED FOR SALE

Federal Communications Commission
Radio Frequency Interference
Statement

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

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

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

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

A note about the programs in this manual:

This manual contains several programs that help to demonstrate the versatility of the Radix printers. Star Micronics has made every effort to insure that the programs are functional and accurate. However, Star Micronics cannot guarantee their accuracy or suitability to any particular application.

Trademark Acknowledgement

Radix-10, Radix-15, grafstar: Star Micronics

Apple, Apple II, Apple II +, Apple IIe, Applesoft: Apple Computer Inc.

Compaq: Compaq Computer Corporation

CP/M: Digital Research

EasyWriter II: Information Unlimited Software, Inc.

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

Kaypro: Kaypro Computer Corporation

Microsoft BASIC: Microsoft Corporation

Osborne 1: Osborne Computer Corporation

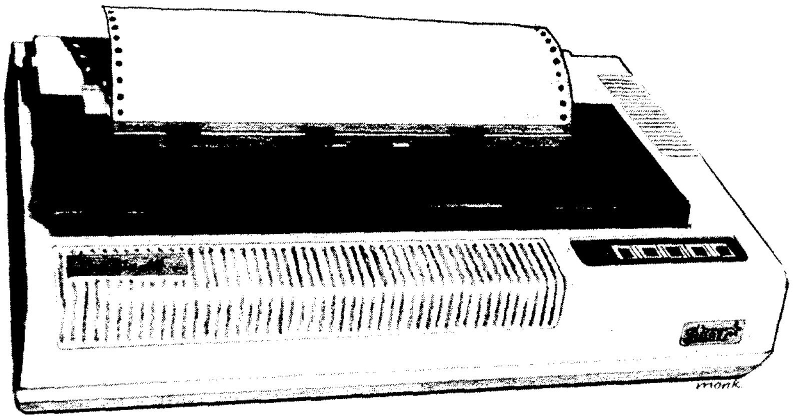
PeachText: Peachtree Software Incorporated, an MSA Company

SuperCalc: Sorcim Corporation

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

WordStar: MicroPro International Corporation

©Copyright 1984 Star Micronics, Inc.



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 *Radix*!

Right now, before you even start readying your Radix 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 Radix offers you, and
2. We'll show you how this manual can help you get the most from your Radix, 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 Radix is smart too: when printing blank spaces, Radix speeds up to a blistering 240 CPS!
16K Memory — Also called *print buffer*. Radix has a mind of its own! Buffer memory holds over 16,000 characters or about 8 printed pages, thus allowing your printer to accept information as fast as your computer can send it. Since computer speed is faster than printer speed, this feature frees your computer to do other things while your printer continues to print.
472 Characters — Allows printing in no less than nine different fonts or 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 column printing or other special applications, with a neater appearance, too.
Graphics — If you're designing your own, you'll be delighted at finding *three* 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 Interfaces — Both parallel and serial interface capabilities are built into Radix — there's nothing extra to buy.
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 cartridge, ready to snap into place; paper is machine-fed, not cranked into place manually. Easy is the word for Radix!

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 1 and 2 — "Getting to Know Your Radix" and "Getting Started with Radix" — as well as Appendix A, "Setting Up Radix." *Now* you can unpack the box and start putting things together.

When you're ready to connect your computer to your Radix, look at Appendices B through E for directions applying to your make of computer. Remember, Radix has both serial and parallel interfaces, so there's nothing extra to buy!

If you're not a programmer, you'll be most interested in Chapters 3, 4, and 5. They explain — in non-programmer's language — how to get the most from Radix using some of the most popular software packages on the market today. You won't need to know a word of BASIC!

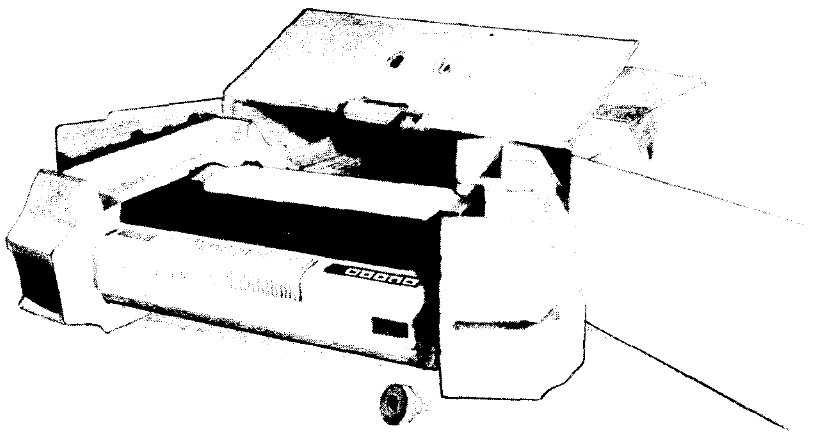
For you who wish to design your own characters, do your own plotting, your own infinite variety of dot graphic patterns and densities, you'll have a ball! For you, Chapters 7 through 12 are a must, and of course everybody should look at Chapter 14, which tells how to maintain your Radix for a long and carefree life.

In this manual there are plenty of example programs to demonstrate and show off all of Radix's features. There are even two utility programs included: one which allows you to design your own printing characters on your computer screen and one to set up Radix the way *you* want it with just a few keystrokes. Since many Radix 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 Radix work with their computer. And in the appendix, complete translations of the utility programs for several popular computers are included.

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

Table of Contents

Table of Tables



Chapter 1

Getting to Know Your Radix

The more you learn about Radix and its sophisticated features, old and new, the better Radix 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**

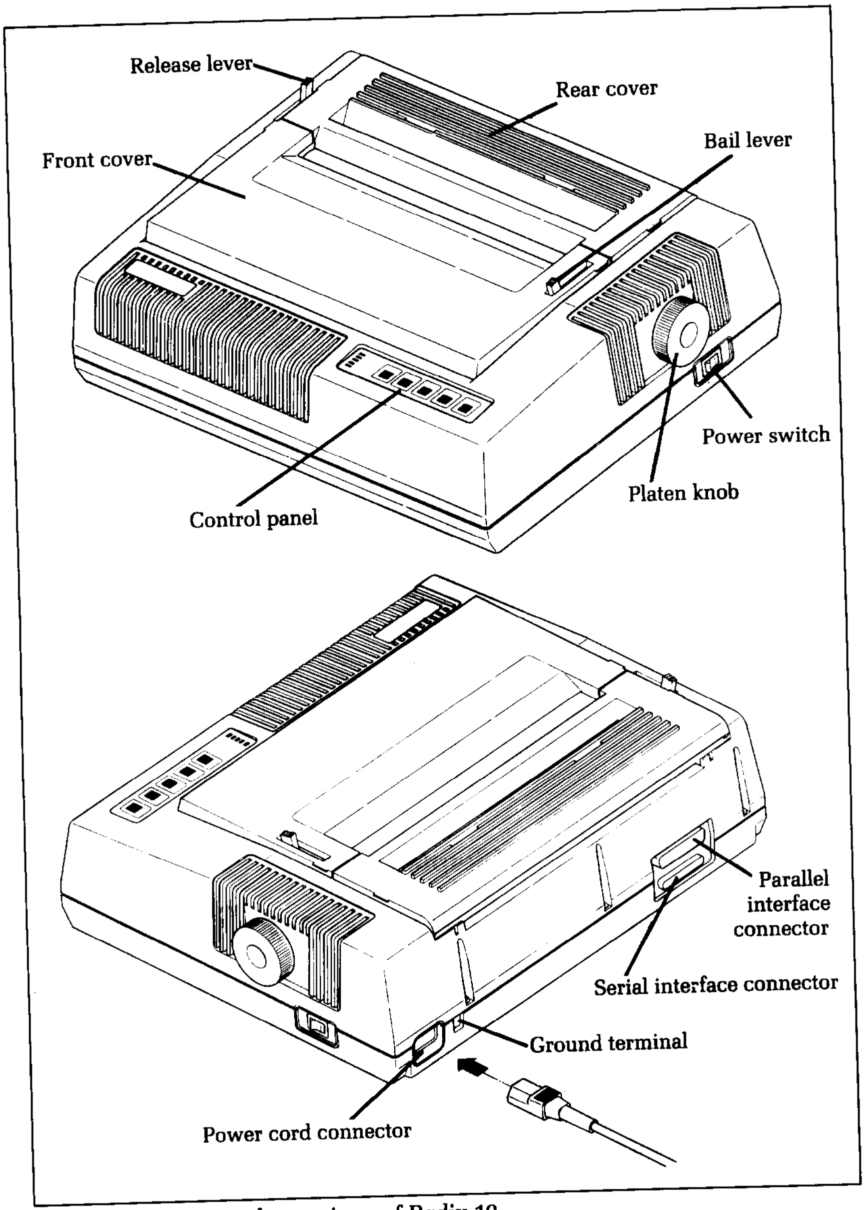


Figure 1-1. *Front and rear views of Radix-10.*

Components and Controls

First, the components. You saw most of these when you unpacked your printer. Now we'll give you a condensed run-

down on what they do. (For details on your initial set-up of Radix, with all components in place, see Appendix A.)
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. 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.
Parallel interface connector — Around on the back, this is the place where you connect your computer to Radix, so that they are able to communicate with each other. It's for computers that use parallel communications.
Serial interface connector — This interface allows you to connect your Radix with a computer that uses serial communications.

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:
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.

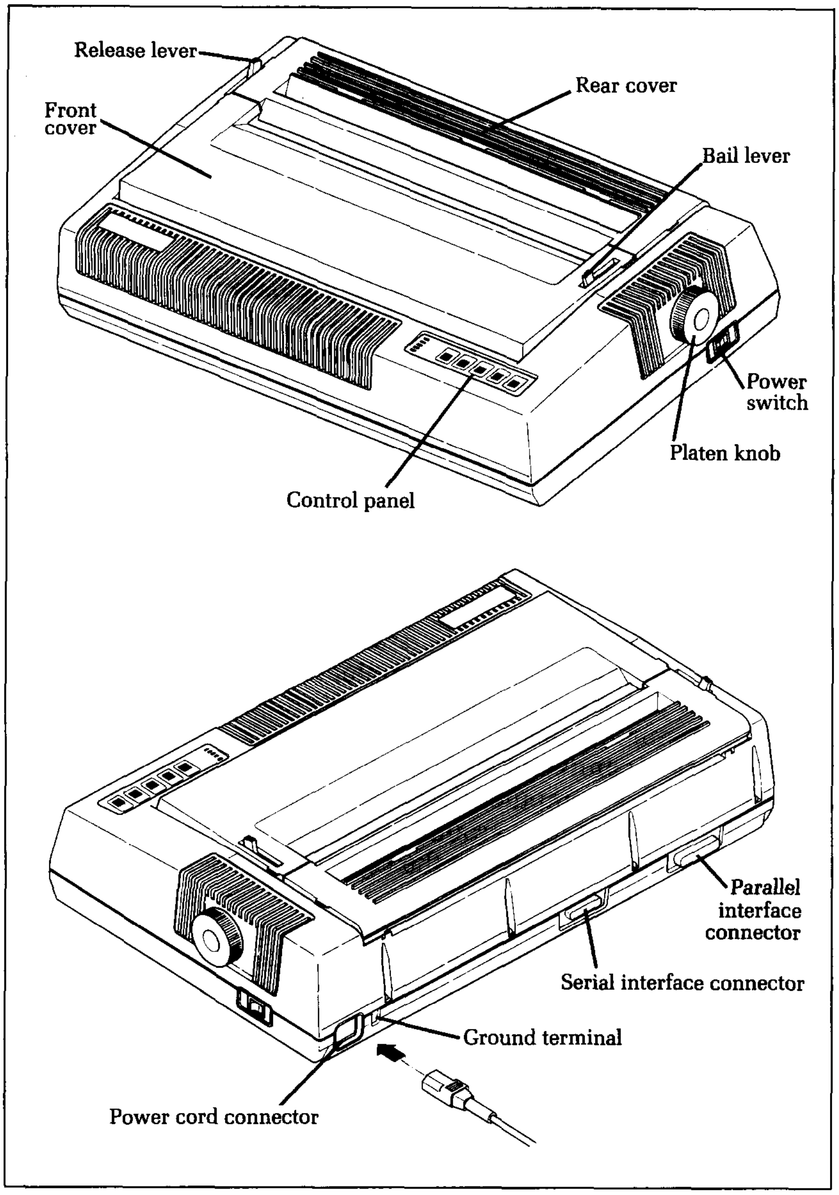


Figure 1-2. *Front and rear views of Radix-15.*

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.

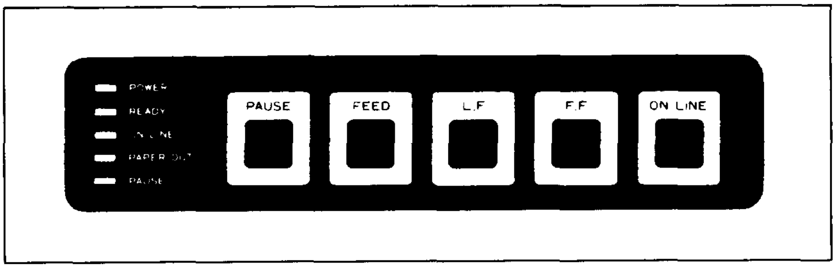


Figure 1-3. *Radix's controls.*

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 the 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 Radix to your particular brand of computer. But there are also switches to set the power-on default settings for print style, line spacing, and page size. See the appendix for a complete explanation.

Paper Selection and Loading

Now we'll look at paper. Your Radix 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 Radix-15), and paper thickness between .07 mm and .10 mm (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 1-4).

2. To set the margin, use the little metal guide (shown in Figure 1-5) in one of its 3 positions.
3. Put the release lever in the "set" position. This step is very important for proper sheet alignment.

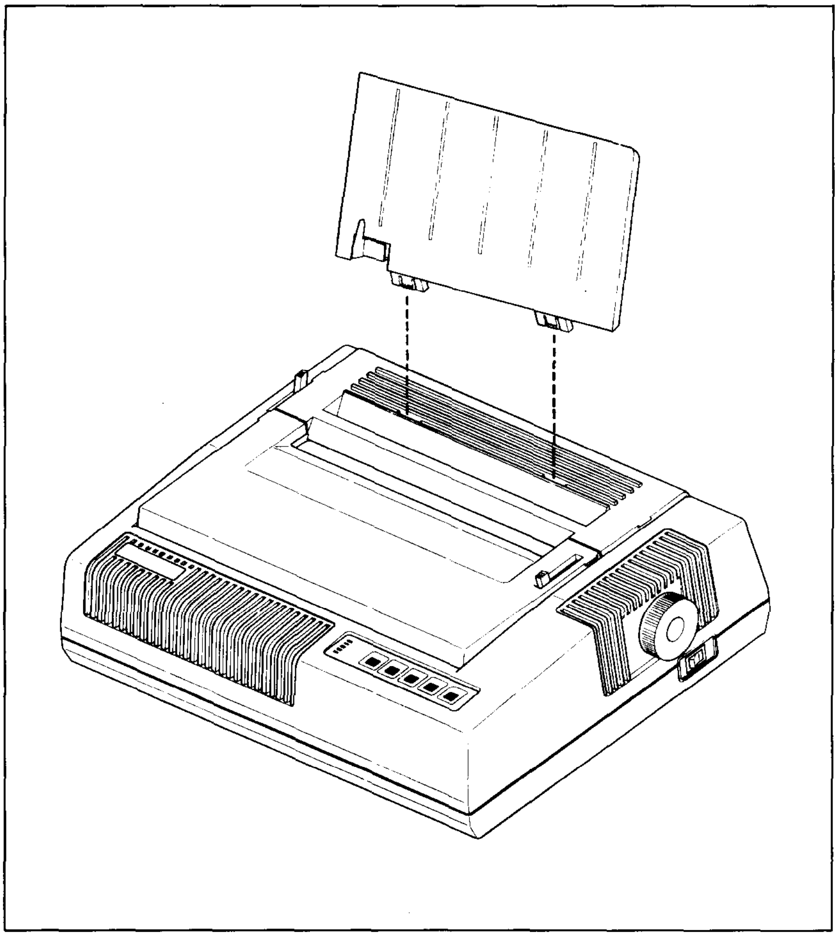


Figure 1-4. *Use the single sheet guide for loading cut paper.*

Table 1-1
Left margin on the single sheet guide

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

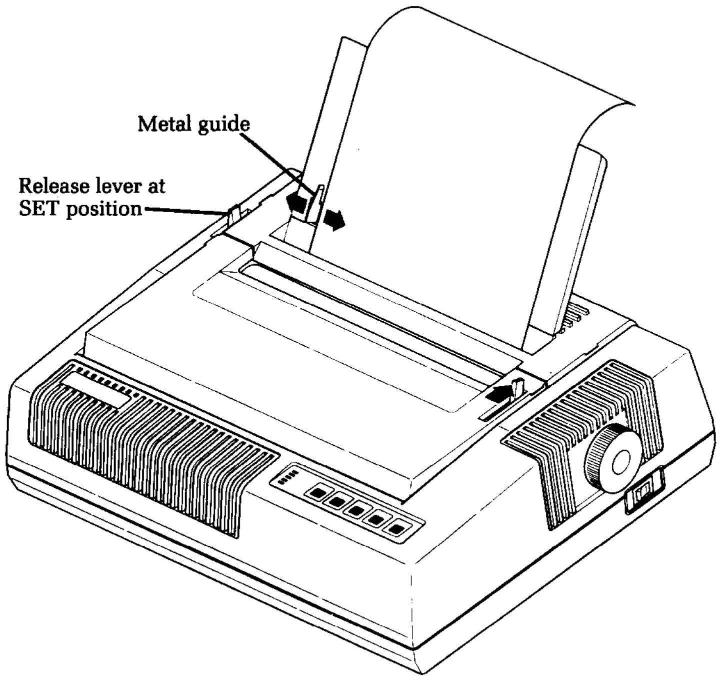


Figure 1-5. *The metal guide is used to align the left margin.*

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). Radix 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.

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.

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 1-6.

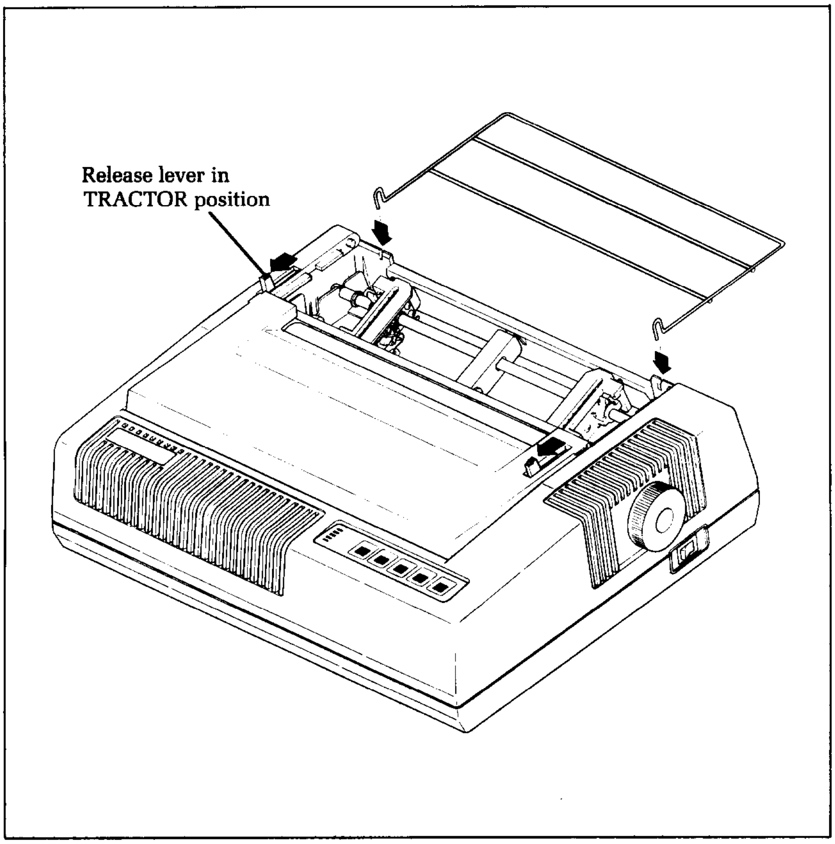


Figure 1-6. *The wire paper guide keeps continuous paper away from the cables.*

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 1-7).

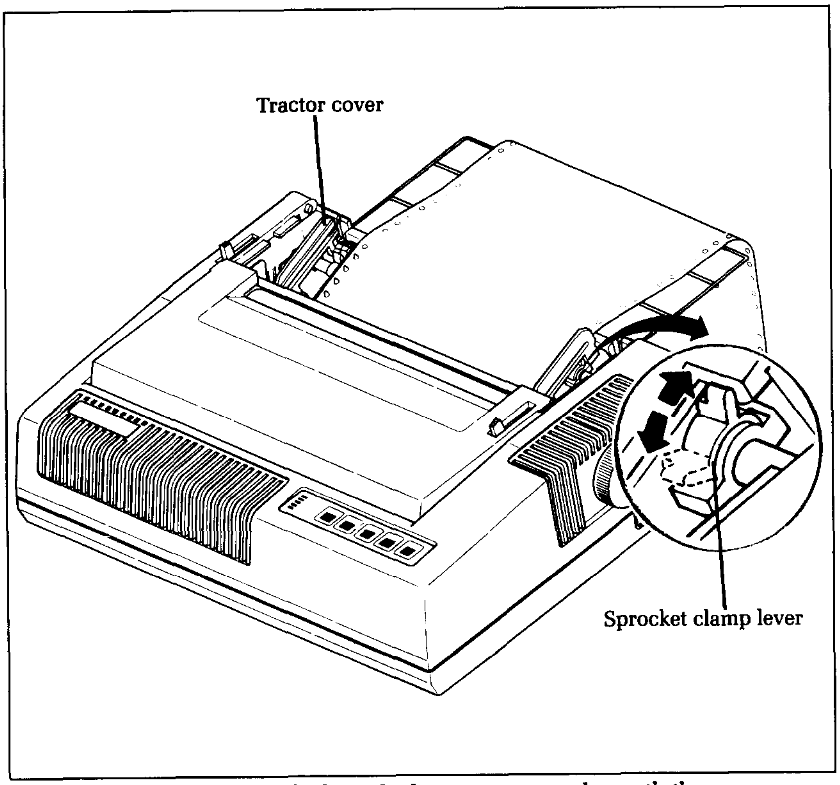


Figure 1-7. *The tractors, which guide the paper, are underneath the rear cover.*

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.
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 1-8).
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.

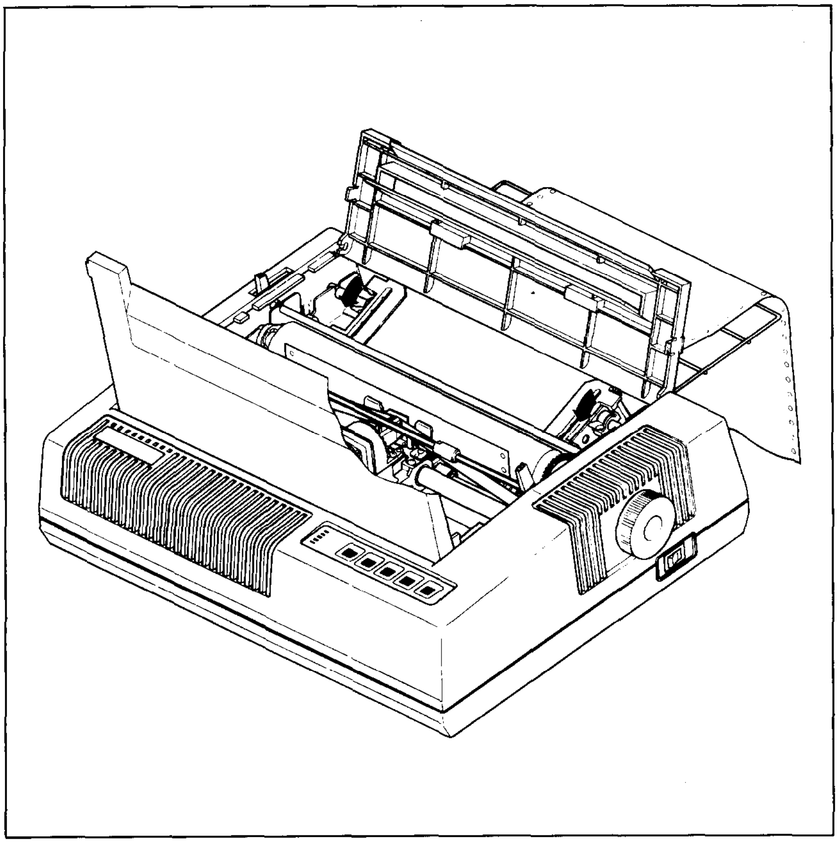


Figure 1-8. *With the tractors in place, you're ready to close the covers and advance the paper.*

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 Appendix A; replacing the ink ribbon *inside* the ribbon cartridge casing is described in Chapter 14 ("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 1-9. Pulling the adjustment lever towards you will widen the gap; pushing it away from you will narrow the gap.

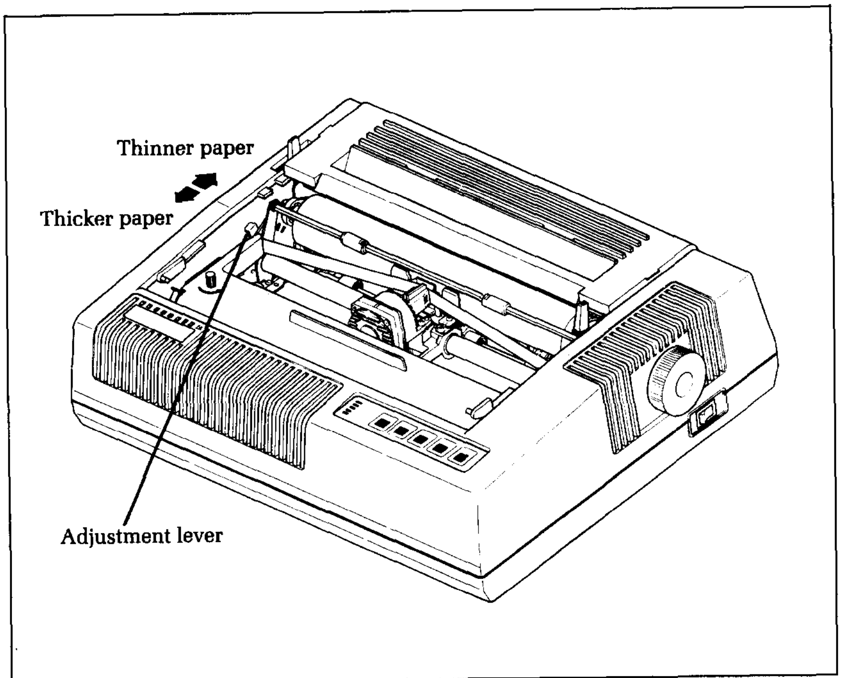


Figure 1-9. *The adjustment lever allows for different thicknesses of paper.*

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.

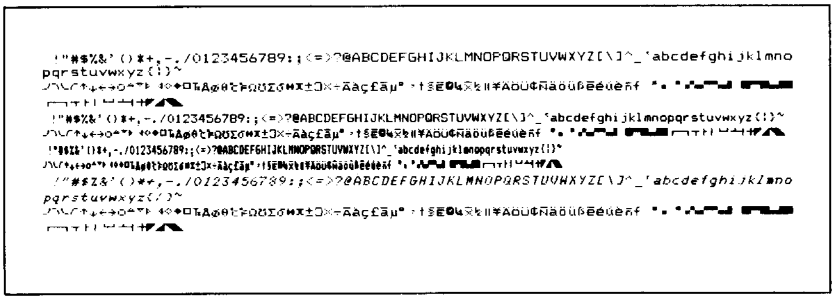
Self-Test

The "self-test" is a trial run of your beautiful new machine. Radix 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 Radix. 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 Radix to your computer! It's as simple as 1, 2, 3 . . .

1. Plug the printer's power cord into a 120 VAC 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).



```
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno
pqrstuvwxyz{|}~
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmno
pqrstuvwxyz{|}~
```

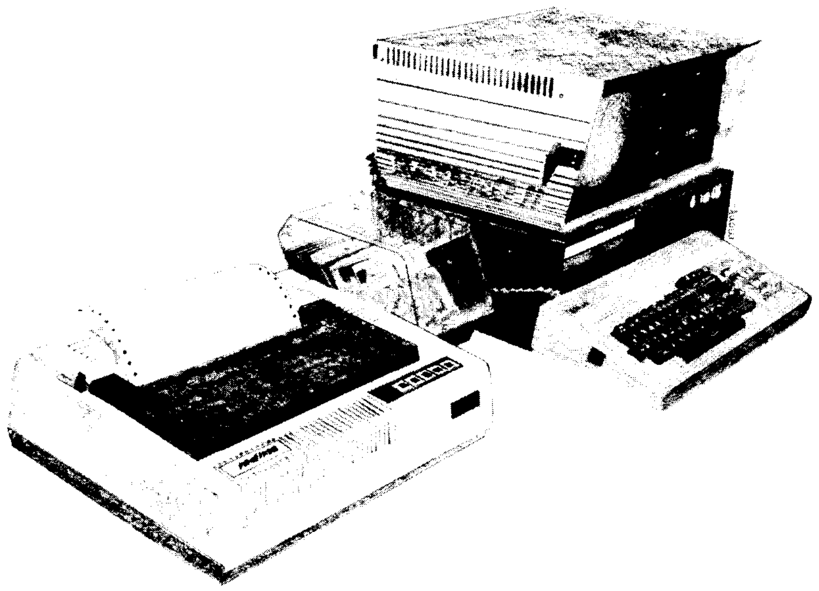
Figure 1-10. *Radix's self-test gives a preview of its capabilities.*

Some Tips for Smoother Operation

Here are some ideas that might save time and trouble with Radix.

- 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 2

Getting Started With Radix

In this chapter you'll learn about:

- **Connecting Radix to your computer**
- **Using Radix with commercial software**
- **ASCII codes**

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

But first you need to connect Radix to your computer. Figure 2-1 shows where the cable connects, but there's more that you need to know. Complete instructions for connecting Radix to many popular computers are given in the appendix. Find the appendix that covers your computer and follow the instructions for connecting Radix and for setting the DIP switches. If your computer isn't listed in the appendix, then ask your Star dealer which computer that is listed is most like yours. If none of the listed computers are similar to yours, then your Star dealer will give you advice on connecting Radix to your computer.

When everything is connected, come back here and we will check it out!

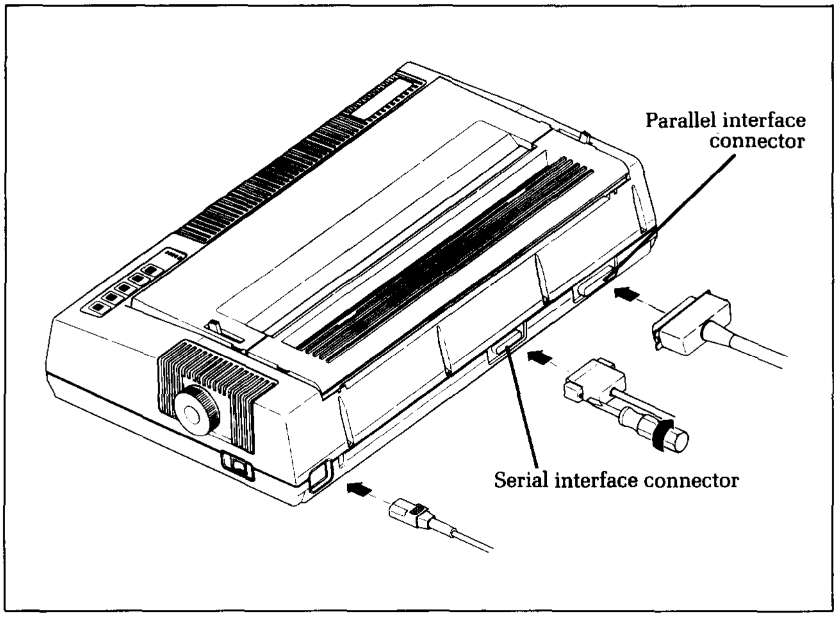


Figure 2-1. *Radix has both serial and parallel interfaces.*

Using Commercial Software

Many of you purchased Radix to use with commercial software. You made a good choice because Radix 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 Radix are: "TTY type printer with backspace", "IBM-dot matrix printer", "Centronics-type printer", "Dot matrix ASCII printer". Radix should work fine with any of these descriptions.

Some printer lists are not very clear, and may not include anything that you think describes Radix. If you can't decide which description best fits Radix, we recommend that you narrow the list to two or three choices (you can quickly eliminate all the daisy-wheel 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. Radix *can* do a "backspace". Radix *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 and Chapters 3 through 5 of this manual to see how to make it send information for Radix to print. This is all you need to know to use Radix as a regular printer. But Radix isn't just a regular printer. Radix has many capabilities that your commercial software isn't aware of. A little later we will see what it takes to use some of Radix's advanced features with commercial software.

First, some terminology

Radix 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 Radix. Both the computer and Radix 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 Radix'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 M 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 Radix'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 Radix 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 M 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 Radix 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 Radix's advanced features. It may help, however if we look at a couple of examples. In the next three chapters, we'll give examples of many of Radix's codes using many popular programs.

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 Radix'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.

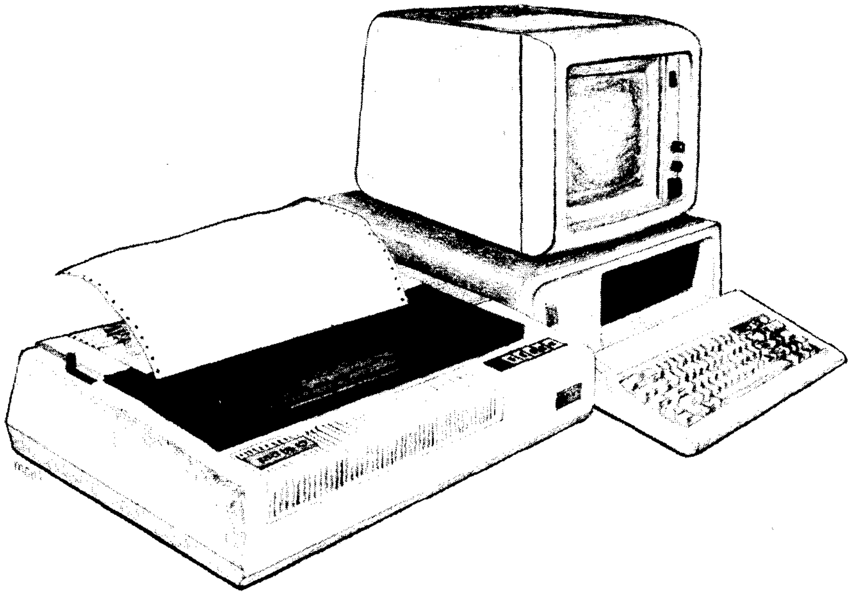
A typical *escape code sequence* starts with ⟨ESC⟩ which is followed by one or more codes. As an example, the escape code sequence to turn on italic print is:

```
⟨ESC⟩ "4"
```

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 Radix's features using the BASIC programming language in our examples. This is because it is easy to communicate with Radix 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 Radix, as you will see in the next chapters. Even if you don't know BASIC, you can learn how to use Radix'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 3

Word Processing with Radix

Not many word processing programs directly support the advanced features of printers like Radix. They usually provide a method for using a few of the more common print features such as boldface and underlining. But as you are probably beginning to see from this manual, Radix can do much more than that.

As a result, most word processing programs provide a way of sending special codes to a printer. The actual codes used (as well as the method of entering them) can be different. The theory behind these methods, however, is basically the same.

This chapter discusses four word processing programs most used by Radix owners. The programs also provide a variety of ways to enter the codes necessary to use the advanced features of Radix. These concepts can be applied to many other

programs besides those detailed here. The four programs are:

- **Easywriter II**
- **PeachText**
- **WordStar**
- **Scripsit**

If your word processing program is not included in this chapter, you should still study the different techniques used. Then, with the help of your program manual and the supporting chapters in this manual, you should be able to figure out how yours works.

General Concepts

Each word processing program has a way to get out of the standard text entry mode in order to accept the special printer function codes. PeachText uses an \OUT\ statement. Wordstar uses the CONTROL key in different ways to define the print function codes.

Easywriter II has a system function which allows you to define print pitches and special print functions for use with the ALT key. Scripsit has a similar process in which user-defined codes are used as recognition characters to select and cancel print functions.

Your word processing User Manual (if it supports this process) will have a section describing how to get out of the standard program. You will probably have to figure out on your own which codes are used. The general concepts and details of the four sample programs should be enough to help you be successful.

The escape code

Most of Radix's special print functions start with a code called the escape code. It can be entered in decimal or hexadecimal values, by an ASCII character, or by using the control keys on your keyboard. It depends on which program and which computer you are using.

This escape code tells the printer to interpret the values (or characters) following it as printer functions. The codes used to describe the functions are also entered in the same method as the escape code. In this chapter, we will show you the format each word processor uses as well as the general rules to correctly enter the function codes.

Chapter 2, "Getting Started with Radix," covers how to convert forms of ASCII codes. You should review Chapter 2, if you have not already done so, before working with the function codes.

The master reset code

There is one function code which turns off all the print functions currently being used by the printer. It is called the master reset code and resets the printer to its DIP switch settings. These print characteristics are the same as the ones used by the printer when it is first turned on.

The code sequence for master reset is ⟨ESC⟩ "@". By checking the ASCII equivalents in Appendix M, you can see that the decimal expression is 27 64. You'll see these numbers several times in this chapter.

Technically speaking, initializing the printer clears the print buffer and the form length, character pitch, character set, line feed pitch and international character set are all reset to the values defined by their respective DIP switch settings.

We suggest you get in the habit of using the master reset code in any document where you use function codes. If you do not, the printer will keep the characteristics most recently defined and print any following documents the same way.

You could turn Radix off each time (which also resets the default settings) but that would be hard on the printer circuits. Also, you'll save time and paper by letting the printer automatically reset with this code. (If you need more information on DIP switch settings for Radix, please refer to Appendix H.)

Using Near Letter Quality (NLQ)

With near letter quality, Radix prints more dots for each character than with the draft printing. This process results in a higher quality look to your text. Draft quality characters print much faster, so use them for your first drafts and use near letter quality for a professional looking finished manuscript.

The escape code sequence to turn the NLQ set on is ⟨ESC⟩ "B" 4 and the code sequence to select draft quality is ⟨ESC⟩ "B" 5. The decimal equivalents are 27 66 4 and 27 66 5, respectively.

Near letter quality printing can be printed in pica width (10 characters per inch) and underlined if you wish. It cannot, however, be mixed with Radix's other print widths, italics, superscripts, subscripts, double-strike, or emphasized printing.

Getting the most from your print choices

After working with Radix for a while, you may find that you want to add to or change some of the print functions we have described in this chapter.

We suggest you do three things. First, you should review Chapter 2 and Appendix K to become as familiar as possible with ASCII codes and the Radix function codes.

Second, read Chapter 7 which describes them in greater detail and shows examples of how they are used in BASIC programming. The functions will, for the most part, act the same in your word processing program. Understanding what's available and how they perform will help you use them correctly in your documents.

And third, follow the procedures in this chapter and your program User's Manual.

You may want to experiment with expanded text in combination with other print types. You can create some great-looking results with these functions. Also, try applying what you have learned in this chapter to your own work while it is still fresh in your mind. If you are unsure of any functions, review them first, then try some of your own samples.

Using Radix with Easywriter II

(*Note:* If you have not read the "General Concepts" section at the beginning of this chapter, you should do so before continuing.)

The Radix printer can be used with most of the standard print functions available with your Easywriter II word processing program. These functions require no special adjustments to the printer or your program. They include:

1. Printing from the Print List Form screen.
2. Setting margins, tabs and lines per inch in the ruler line of your document. (The pitch settings, however, should be adjusted to obtain maximum use. They will be discussed later in this chapter.)
3. Print settings in the System Parameter function which are either default settings or new settings edited by you.

You can also redefine print functions of Easywriter II to take advantage of many of the printing capabilities of Radix. You may already be familiar with reconfiguring the printer driver from Appendix B of your Easywriter II User's Manual. If not, don't be nervous; it's not as hard as it sounds. We will show you how to make changes in your program specifically to help you print with Radix.

By changing the pitch settings, you can use the document ruler line to print pica, elite and condensed width pitches. In addition, you can use the same theory to print in near letter quality pica.

The print control codes can be redefined to enhance the final product of your document. The boldface, underline, superscript and subscript functions require only a slight "recoding" of information in the printer driver. And we have some suggestions for changing the characteristics of the other print control codes to use italic, expanded, emphasized and italic-underline print. With these options, you will have even more printing flexibility with Radix.

Redefining pitch settings and print control codes

In order to change the settings used in the document ruler line and the print control codes, it is necessary to edit ASCII code decimal values in the System Functions portion of your Easywriter II program. (For more details on ASCII codes, please refer to Chapter 2.)

Your Radix printer is considered a Type B printer by the Easywriter II program. Before making any changes in the printer driver, you should first check to be sure the printer selection is set for printer Type B (Option 7 on the System Functions menu).

Then follow the instructions in Appendix B of your Easywriter II User's Manual to reconfigure Type B printers. To become more familiar with the reconfiguration process and its terms and to make the instructions in this section easier to understand, we suggest you read through Appendix B first.

In these next few paragraphs, we'll show you the ASCII decimal values we feel provide a good flexibility in printing with Radix. You should follow the instructions hands-on with your own Easywriter II program.

The changes you will make are for pitch settings and print control codes (also called font support). However, all the screens involved will be explained as you see them displayed.

From the System Functions Menu, choose Option 9 (Reconfigure Printer Type B) and the printer name will be displayed. Type over the present printer name as follows:

```
1.  Printer Name  [Radix Dot Matrix Printer    ]
```

Press RETURN and the Edit Global Sequences screen will be displayed. These codes control the print functions for form feed, line feed, margin settings and automatic justification. We do not recommend that you edit any of these codes.

Press RETURN and the Edit Pitch Table screen will be displayed. On this screen, you will enter the ASCII decimal values to define the print pitches. The first two fields in each line define the pitch range (which in this case are both the same number). They should be assigned as follows:

```
10 = Draft Pica      1 = Near Letter Quality Pica
12 = Draft Elite     2 = Master Reset Code
17 = Draft Condensed
```

On this screen, the column labeled "Sequence" is used to define the print functions in their ASCII decimal values. For these print pitches, we will use a combination of codes to turn near letter quality print on and off and to choose the function code for each pitch. (For more details on function codes, please refer to Appendix K.)

Follow the sample and enter the (italic) codes for lines 17-21.

```
17.  [10 ] [10 ]          [ 27  66   5  27  66  1  ]
18.  [12 ] [12 ]          [ 27  66   5  27  66  2  ]
19.  [17 ] [17 ]          [ 27  66   5  27  66  3  ]
20.  [1  ] [1  ]          [ 27  66   4             ]
21.  [2  ] [2  ]          [ 27  64                 ]
22.  [120] [120]          [                        ]
23.  [120] [120]          [                        ]
24.  [120] [120]          [                        ]
25.  [120] [120]          [                        ]
26.  [120] [120]          [                        ]
```

The codes 120 in lines 22 through 26 can be changed to reflect more pitch settings. We recommend that, until you are more familiar with using special function codes, you use just the five we have defined.

When you have finished, press RETURN. You will be transferred to the Edit Line Spacing screen. Do not change these codes.

They define how many lines per inch the printer uses. Press RETURN to transfer to the Edit Font Support screen.

Change all of the entry fields to Option 2 (Control Code Support) on the Edit Font Support screen. Also, make changes in the other fields to look like the figure shown below. Enter the (italic) codes for lines 41-50.

```
41.  Bold/Shadow Face Support   [2]
42.  Single Underline Support   [2]   Using Character [95 ]
43.  Double Underline Support   [2]   Using Character [0  ]
44.  Overstrike Support         [2]   Using Character from
45.  Special (Color) Support    [2]   System Parameters
46.  Sub/Superscript Support    [2]
47.  Will underline retain font (Y) or be normal font (N)?
   [N]
49.  Start double underline [      ]
50.  After double underline [      ]
```

When you have finished, press RETURN and the Edit Font Sequences screen will be displayed. Here you will define print control codes for use in your documents. As with the pitch settings, ASCII decimal values are used that correspond to the print function assigned to each control key. Table 3-1 shows the current control function, the print function we will assign to it and the keyboard keys used.

Table 3-1
Easywriter II control keys

Easywriter II function	New function	Keys used
Boldface	Boldface	ALT & B
Shadow	Italic	ALT & S
Underline	Underline	ALT & _
Double	Expanded	ALT & =
Overstrike	Emphasized	ALT & O
Special	Italic Underlined	ALT & *
Subscript	Subscript	ALT & D
Superscript	Superscript	ALT & U

Enter the (italic) codes for lines 51-66.

51.	Normal to Bold	[*27*	*71*	*27*	*69*		]
52.	Bold to Normal	[*27*	*72*	*27*	*70*		]
53.	Normal to Shadow	[*27*	*52*				]
54.	Shadow to Normal	[*27*	*53*				]
55.	Normal to Underline	[*27*	*45*	*1*			]
56.	Underline to Normal	[*27*	*45*	*0*			]
57.	Normal to Double	[*27*	*87*	*1*			]
58.	Double to Normal	[*27*	*87*	*0*			]
59.	Normal to Overstrike	[*27*	*69*				]
60.	Overstrike to Normal	[*27*	*70*				]
61.	Normal to Special	[*27*	*52*	*27*	*45*	*1*	]
62.	Special to Normal	[*27*	*53*	*27*	*45*	*0*	]
63.	Normal to Subscript	[*27*	*83*	*1*			]
64.	Subscript to Normal	[*27*	*84*				]
65.	Normal to Superscript	[*27*	*83*	*0*			]
66.	Superscript to Normal	[*27*	*84*				]

When you have finished, press RETURN. You're done! You will be transferred out of the Reconfigure Type B Printer function and back to the System Functions Menu.

A sample printout with Easywriter II

Let's look at a short example to demonstrate how pitch settings and print control keys can be used in a document. The example below shows the use of expanded and italic prints used in combination with condensed and pica pitch settings. Use your Easywriter II program hands-on and type the example below.

```
SUBJECT: ORDERING STATIONERY SUPPLIES
I would like to place an order for stationery supplies
from your mail order catalog. Enclosed is my order form
and a check for $247.67. Please process this order as
soon as possible. Thank you.
```

With the cursor under the "S" in "SUBJECT", set the print pitch in the ruler line to condensed width pitch. Name the ruler line "condensd" (without the quotes) and change the character pitch to 17 and the line spacing to 6. To make the subject title expanded, use the ALT and = keys (in the line mode) to highlight the line.

Now, change the pitch setting in the next line to pica by setting a new ruler line: Ruler Name - pica; Character Pitch - 10. Use the print control key S (for italic) to highlight the second sentence in the paragraph. Move the cursor to the "E" in "Enclosed" and (in the sentence mode) use the ALT and S keys to highlight the sentence. (You'll have to press S twice to get the .67.)

At the end of the document, reinitialize the printer to its default settings with a new ruler line using the Master Reset code. Ruler Name - reset; Character Pitch - 2.

Print the document. Your printout should look like this:

```
SUBJECT:  ORDERING STATIONERY SUPPLIES

  I would like to place an order for stationery supplies from your
mail order catalog.  Enclosed is my order form and a check for
$247.67.  Please process this order as soon as possible.  Thank
you.
```

The subject title will print in expanded condensed characters which are twice the width as standard condensed characters. The sentence in the paragraph is printed in italic pica print. The last ruler line will reinitialize the printer. (See the general concepts section of this chapter for more details on master reset.) This is just one example, however, you should be able to apply most of the function codes to the setup used here.

Redefining your own print pitches

If you want to define a new print pitch (Edit Pitch Table), be sure to start the sequence with the code 27 66 5. This code tells the printer to turn off near letter quality print. Then enter your function code to choose the print you want. By not using the function code 27 66 5 first, the printer will continue to print near letter quality.

For example, if you found yourself frequently using italic print for large blocks of text in pica width pitch, you can combine italic and pica pitch to define italic pica and use it in the ruler line of your document. The ASCII code sequence would be 27 66 5 27 66 1 27 52 which would print italic pica pitch.

Redefining your own print control keys

The ASCII codes to redefine the print control keys (Edit Font Sequences) are pretty straight forward. There are individual ASCII decimal values to turn on and off different prints. You want to affect that aspect but not the print pitch itself. Leave that for your document ruler line. Remember, all the codes can be found in Appendix K of this manual.

Also, keep in mind that print control keys can be combined in your document such as boldface and underline. Easywriter II uses three methods of highlighting on the display screen. It highlights, underlines and shows reverse image characters. You cannot combine print control functions that use the same method of highlighting.

For example, in our definitions, underline and expanded prints are both displayed as underlined on the screen. Whichever function you use last will cancel out any previous modes.

Using Radix with PeachText

(***Note:*** If you have not read the "General Concepts" section at the beginning of this chapter, you should do so before continuing.)

Radix can be used with PeachText for a wide range of different print functions. Radix automatically supports many of the standard printing capabilities as well as the method of sending special codes to use all of its printing features. As a result, you can really customize the final look of your documents.

With Radix, you can perform all the following print functions without making any special changes to your PeachText program.

1. Print documents from the Text Edit screen.
2. Print documents from the Print Status screen.
3. Use print commands and recognition characters in your document.
4. Select and print variable information for merge letters, etc.

All of these functions are fully described in the PeachText User's Manual and will not be discussed in this chapter. You should refer to the manual if you need help in successfully performing any of these functions.

With the print capabilities described above, however, some individual functions will not work without using special codes. They include changing lines per inch and characters per inch, using superscript and subscript recognition characters, backspacing, and using horizontal and vertical tabs commands.

Entering special function codes

When you want to enter a code to perform a special print function, you must first enter the PeachText command, \OUT. Then, enter the decimal values relating to the ASCII code for each function you want performed. The statement is closed by using the \ symbol.

Each value must be separated by a comma. For example, a valid statement would be \OUT27,69\. If the function code you want is expressed in more than one ASCII character, you must use a comma between each decimal value. To select elite pitch, for example, the ASCII code is ⟨ESC⟩ "B" 2. The PeachText statement would read \OUT27,66,2\.

Different print combinations can be combined within one statement. In these cases, you need to use the escape code for each function. For example, to select double-strike and emphasized print at once, the PeachText statement is \OUT27,71,27,69\ where 27,71 selects double-strike print and 27,69 selects emphasized print.

When you enter these codes in your document, they will appear on the display screen, but they will not print out on paper. The characters do not take up any hard space when printed. Your first character of text will actually print in the first space where the command begins (on the screen).

When you start using these codes on your own, you should turn to Appendix K in this manual to look up the printer function you want. To enter the codes into PeachText, simply use the decimal values in the format previously described. Let's take an example.

Say you want to change to near letter quality characters. The function code as shown in Appendix K is ⟨ESC⟩ "B" 4. The decimal equivalent is 27 66 4. Your PeachText command would be entered as \OUT27,66,4\. By following this format, you can use any of the function codes applicable to Radix as described in Appendix K.

Table 3-2 references the most commonly used print functions and the PeachText statements you should enter in your document.

Using boldface print

There are several different ways to highlight text as boldface print. The easiest way is to use the boldface function with the PeachText recognition character (@) as described in the PeachText User's Manual. You can also use function codes to select double-strike, emphasized or combine both to highlight in boldface fashion.

Each type will give you a slightly different look. You may want to experiment with them and use the one you like the best. Once you have chosen the method you feel is best for your needs, try to stick with it as much as possible. Switching back and forth will be confusing and ultimately detract from the look of your documents.

Table 3-2
PeachText print functions

Print Function	Select Code	Deselect Code
Pica Pitch	\OUT27,66,1\	–
Elite Pitch	\OUT27,66,2\	–
Condensed Pitch	\OUT27,66,3\	–
Near Letter Quality	\OUT27,66,4\	\OUT27,66,5\
Double-Strike	\OUT27,71\	\OUT27,72\
Emphasized	\OUT27,69\	\OUT27,70\
Italic Print	\OUT27,52\	\OUT27,53\
Single-Line Expanded	\OUT27,14\	see note below
Continuous Expanded	\OUT27,87,1\	\OUT27,87,0\
Underline	\OUT27,45,1\	\OUT27,45,0\
Superscripts	\OUT27,83,0\	\OUT27,84\
Subscripts	\OUT27,83,1\	\OUT27,84\

Note: A carriage return will automatically turn off single-line expanded text.

Underlining with Radix

There are two different ways to underline text. The easiest way is to use the underline function with the PeachText recognition character (_) as described in the PeachText User's Manual. With Radix, this will print as a series of dashes under the text (as opposed to a solid line).

The special printer function codes can also be used to select and cancel underlining. As shown in the Table 3-2, the code \OUT27,45,1\ selects underlining and \OUT27,45,0\ cancels it. With these codes, Radix will print a solid line under the text (including spaces) rather than dashes.

The choice is yours as to which you use depending on your application. The flexibility, however, is nice to have.

A sample printout with PeachText

Let's look at a short example to demonstrate how function codes are used in a PeachText document. The following example shows how italic and italic-underline print are used. Use your PeachText program hands-on and type the example exactly as you see it below.

```
SUBJECT: \OUT27,52\ORDERING STATIONERY
SUPPLIES\OUT27,53\
I would like to place an order for stationery supplies
from your mail order catalog.
\OUT27,52,27,45,1\Enclosed is my order form and a
check for $247.67.\OUT27,53,27,45,0\ Please process
this order as soon as possible. Thank you. \OUT27,64\
```

In the subject title of this example, the first code \OUT27,52\ selects italic print; the second code \OUT27,53\ cancels it. In the paragraph of the example, the first function code combines italic with underline print. The decimal values 27,52 select italic and 27,45,1 select underline.

Also in the paragraph, the function code \OUT27, 53,27,45,0\ cancels italic-underline print. The decimal values 27,53 cancel italic and 27,45,0 cancel underline.

The final function code \OUT27,64\ resets the default settings on the printer. Now, print the document. Your printout should look like this:

SUBJECT: *ORDERING STATIONERY SUPPLIES*

I would like to place an order for stationery supplies from your mail order catalog. <u>*Enclosed is my order form and a check for $247.67.*</u> Please process this order as soon as possible. Thank you.

This is just one example, but you should be able to apply most of the function codes to the setup used here.

Storing documents with function codes

When you create combinations of function codes you like, store them as skeleton documents. To access the documents later, go to the Main Menu and use the copy document function (CO) to duplicate the skeleton document. Then rename the new document. This will save you time and reduce your chance of entering incorrect codes.

Using Radix with WordStar

(*Note:* If you have not read the "General Concepts" section at the beginning of this chapter, you should do so before continuing.)

Radix supports many of the standard WordStar printing capabilities without requiring any changes. You can:

1. Print documents from the No-File Menu.
2. Use the dot commands except for lines per inch, characters per inch and microjustification.
3. Print boldface, underline, double-strike, strikeout, superscript and subscript characters as well as use print pause.
4. Select and print variable information for merge letters, etc.

User-defined print commands

There are several CONTROL-P (^P) commands that automatically work with Radix and require no changes. They include:

^PS Underscore	^PB Boldface
^PD Double-strike	^PX Strikeout
^PT Superscript	^PV Subscript
^PC Print Pause	

It is also possible to define the ^PA (alternate pitch) command to change the print pitch of your document. The WordStar User's Manual fully describes the use of these ^P functions. You should refer to your manual if you need help with them.

There are four alternative ^P codes that can be defined during the installation of your WordStar program to perform other printer functions. They are ^PQ, ^PW, ^PE, and ^PR. The process of defining ^P commands is called "patching" and is a fairly complicated process. Once you have successfully defined these codes, they are inserted in your text exactly like the other ^P commands. If you wish to use them, refer to the WordStar User's Manual for instructions or contact your dealer for assistance.

Perhaps the most useful user-defined ^P command is ^PE. If you define this as an escape (ASCII code 27), you can then access nearly all of Radix's advanced features. Without this patch, you cannot place an escape in the WordStar document and subsequently, you are limited to using WordStar's repertoire of print functions. A shame when you have a powerful Radix!

Using Radix with Scripsit

(*Note:* If you have not read the "General Concepts" section at the beginning of this chapter, you should do so before continuing.)

Radix can be used with most of the Scripsit print functions. You can use most of the basic functions on the Open Document Options screen as well as on the Print Text Options screen. Two minor adjustments you need to make. On the Open Document Options screen the printer type is LPN4. Also, DIP switch C-4 must be OFF to cause a line feed at the end of each line.

There are some print functions, however, that do not work without redefining the printer driver or by continually changing the DIP switch settings. That could be an awkward, time-consuming task.

Instead, we recommend the User Print Code Facility to define several recognition characters and control your print functions. With the use of the CLEAR key and each recognition character, Scripsit can perform any print function you define.

Print Pause and Form Feed automatically work with Radix. Double underscore only works with daisy wheel printers and strikeout does not work without redefining the printer driver. (Since strikeout is seldom used, we do not suggest you try it.)

Boldface, underline, superscripts and subscripts do not work with the recognition characters described in the Scripsit User's Manual but can be defined in the Print Code Facility along with some other print functions (emphasized, italic and expanded).

The print pitch settings on the Open Document Options screen will not work with Radix. Therefore, we will show you how to use recognition characters to define pica, elite, condensed and near letter quality pitches.

Defining user print codes

Your first step is to enter the Edit Printer Control screen. If you are not familiar with this function, review your Scripsit User's Manual for details. Each character on this screen will be assigned an ASCII decimal value corresponding to the function you want. If you have not read Chapter 2 in this manual, we suggest you do so before continuing.

The decimal equivalents of each function can be found in Appendix K. A space should be left between each decimal value. The sample below shows each code as it should be entered into the Edit Printer Control screen. Type the (italic) codes as they appear here.

```
Code  Units Sequence:up to 11 codes will be counted
                                         Comments
0     0—   27 64 ——————————————————————  mstr reset
1     0—   27 66 5 27 66 1 ————————————  pica
2     0—   27 66 5 27 66 2 ————————————  elite
3     0—   27 66 5 27 66 3 ————————————  condensed
4     0—   27 66 4 ————————————————————  near lettr
5     0—   27 71 27 69 ————————————————  bold on
6     0—   27 72 27 70 ————————————————  bold off
7     0—   27 69 ——————————————————————  emphasz on
8     0—   27 70 ——————————————————————  emphas off
9     0—   27 45 1 ————————————————————  under on
      Press <ENTER> to edit next screen
!     0—   27 45 0 ————————————————————  under off
"     0—   27 52 ——————————————————————  italic on
#     0—   27 53 ——————————————————————  italic off
$     0—   27 87 1 ————————————————————  expand on
%     0—   27 87 0 ————————————————————  expand off
&     0—   27 83 0 ————————————————————  supscrp on
'     0—   27 83 1 ————————————————————  subscrp on
(     0—   27 84 ——————————————————————  script off
)     0—   ————————————————————————————    —————
@     0—   ————————————————————————————    —————
      Press <ENTER> to return to System Setup menu
```

When you have finished entering the codes, press ENTER to return to the System Setup menu. Then, press BREAK to return to the Main menu.

Each function code is pretty straight forward and they are all included in Appendix K of this manual. But there is a certain way to enter them.

Say you want to enter the codes for double-strike print. The function code as shown in Appendix K is <ESC> "G". The decimal equivalent is 27 71. Your Scripsit command would be entered as 27 71. By following this format, you can use any of the function codes applicable to Radix as described in Appendix K.

The function codes for pica, elite and condensed width pitches are expressed a bit differently than the rest. If you want to request one of these three prints and you are changing from Near Letter Quality (NLQ), the first codes 27 66 5 will turn NLQ off before printing draft pica, elite or condensed 27 66 1 (2 or 3).

Notice that boldface is actually a combination of function codes for defining double-strike and emphasized print. If you want to use just double-strike as boldface, you can delete the codes 27 69 and 27 70 from sequences 5 and 6.

A sample printout with Scripsit

Let's look at a short example to demonstrate how pitch settings and print control keys can be used in a document. The example below shows the use of boldface and italic prints used in combination with pica and elite pitch settings. Use your Scripsit program hands-on and type the example below. (To get the "@" symbol, use the CLEAR key on your keyboard).

```
@1SUBJECT: @5ORDERING STATIONERY SUPPLIES@6
@2I would like to place an order for stationery supplies
from your mail order catalog. @''Enclosed is my order
form and a check for $247.67.@# Please process this
order as soon as possible. Thank you.
@0
```

The first recognition character @1 will start pica width pitch. The second character @5 will turn on boldface print until @6 turns it off. At the beginning of the paragraph, @2 starts elite pitch. The second sentence is bracketed by the recognition characters @" and @# which will turn italic print on and off. The last code @0 is the master reset code which will reinitialize the printer to its default characteristics.

Now print the document. Your printout should look like this:

```
SUBJECT:  ORDERING STATIONERY SUPPLIES

  I would like to place an order for stationery supplies from your
mail order catalog.  Enclosed is my order form and a check for
$247.67.  Please process this order as soon as possible.  Thank
you.
```

The subject title prints in boldface pica. The entire paragraph prints in elite pitch but with the second sentence as italic characters. And although you do not see it happen, the printer is reset by the code @0. (See the general concepts section of this chapter for more details on Master Reset.)

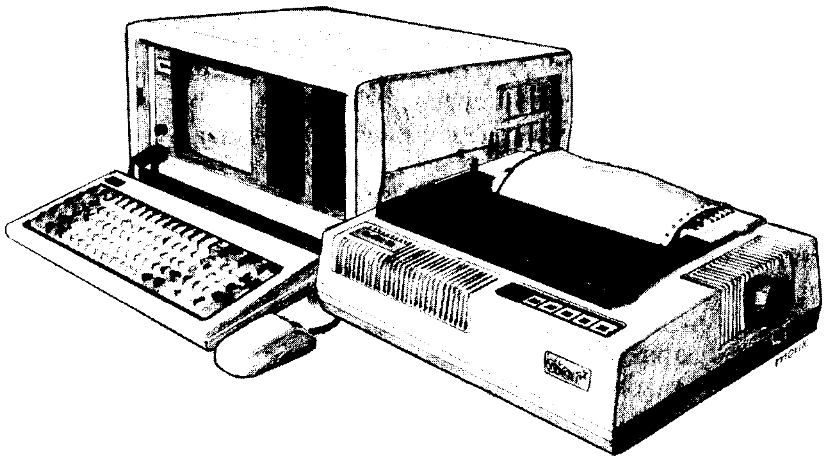
This is just one example, but you should be able to apply most of the function codes to the setup used here. A note about underlining: with Scripsit, Radix will underline the spaces designated for the left margins of each line. You will have to enter the control codes at the beginning and end of each line to turn underlining on and off.

Redefining your own print codes

If you want to define a new print pitch, be sure to start the sequence with the code 27 66 5. This code tells the printer to turn off near letter quality print. Then enter your function code to choose the pitch you want. Otherwise, if you change pitches in a document from near letter quality to draft, the function code alone (in the printer driver) will continue to print near letter quality.

For example, if you found yourself frequently using italic print for large blocks of text in pica pitch, you can combine italic and pica pitch to define italic pica and use it in the ruler line of your document. The ASCII code sequence would be 27 66 5 27 66 1 27 52 which would print italic pica pitch. Remember, you can enter up to 11 codes for each User Print Code.

Also, keep in mind that recognition characters can be combined in your document to print, for instance, italic and underline. Simply enter the two codes before the text. With italic-underline, the codes would be @"@9 to turn it on and @#@! to turn it off.



Chapter 4

Using Radix With Spreadsheet Programs

Radix is a good printer to use with spreadsheet programs because its capabilities match the requirements for printing spreadsheets. It can print large spreadsheets fast and it can also print good looking final reports with the Near Letter Quality (NLQ) character set.

We will look at how to use Radix with three popular spreadsheet programs:

- **VisiCalc**
- **SuperCalc**
- **Lotus 1-2-3**

Using Radix With VisiCalc

VisiCalc was the first spreadsheet program. Although it is now widely imitated, it is still one of the most popular. The descriptions here are for VisiCalc as implemented on the IBM-PC; it may work slightly differently on your computer.

VisiCalc spreadsheets are printed with the */Print* command. The */Print* command prints the area of the spreadsheet between the active cell, at the upper left, and a cell that you enter in the command, at the lower right.

First, move the active cell to the cell in the upper left corner of the area that you want to print. Then, to start the print command, enter **/P**, and then **P** to direct the output to the printer. Now you must specify the lower right corner of the part of the worksheet that you want to print, and press return to start printing.

This system works fine if the area of the worksheet that you want to print all fits on one page. But if you want to print a larger area and break it into pages, then you must figure out the different areas that you want to print and use separate */Print* commands to print them, moving the active cell to the upper left cell of each area before you print it. You also must use the printer controls to advance the paper to the next page so that you don't print over the perforations on sprocket feed paper.

Radix gives you another way to print large spreadsheets. You can change the width of the characters that Radix prints, and thus print more characters per line. The "*Setup* option allows you to send function codes to the printer from VisiCalc. To change the character pitch to condensed, enter **"^CO⟨RET⟩**. The following table shows what the shorthand that VisiCalc uses when sending function codes to the printer. The ^ character (the caret on the

Table 4-1
VisiCalc control codes

^C	This marks the next character as a control code. For example, if you want to send control-O to the printer, you must enter ^CO.
^E	This sends an escape to the printer. Since escape is so widely used, they made a special code for it.
^R	This sends a carriage return to the printer.
^L	This sends a line feed to the printer. You can use this to put blank lines between sections of the worksheet as you print it out.
^H	This code says to treat the next two characters as hexadecimal digits. For example, if you entered ^HOF, then VisiCalc would send hex 0F (decimal 15, or control-O) to the printer.
^^	This sends one caret character (^) to the printer.

keyboard) signifies the beginning of one of the special codes to VisiCalc.

With these codes you can send any of Radix's function codes through VisiCalc. If you select a different printing style than normal, you will have to enter it before printing each section of the spreadsheet because VisiCalc resets the printer as it starts each */Print* command.

The most common codes are to change the print pitch, so the following table shows how many print columns will fit on a page with the various print pitches possible, and the codes to use to get them. If you have a Radix-15, but are using 8½ inch wide paper, use the values for a Radix-10.

Table 4-2
Print columns on a page with VisiCalc

Pitch	Radix-10	Radix-15	Setup codes
Pica	80	136	^EB^CA
Elite	96	163	^EB^CB
Condensed	136	233	^EB^CC
Pica Expanded	40	68	^EB^CA ^EW^CA
Elite Expanded	48	81	^EB^CB ^EW^CA
Condensed Expanded	68	116	^EB^CC ^EW^CA
NLQ on	80	136	^EB^CD
NLQ off			^EB^CE

One more thing that you may wish to do is to switch to NLQ printing for a final report. To switch to NLQ, enter "**^EB^CD.** This will turn on NLQ printing. As you can see, you can use any of Radix's features with VisiCalc, just by entering the proper codes in the "*Setup* option to the */Print* command.

Using Radix with SuperCalc

SuperCalc is a popular spreadsheet program. It has a lot of flexibility and can utilize many of Radix's advanced features.

The */Output* command is used to print SuperCalc spreadsheets. This command allows you great variation in the way you print your spreadsheets.

The simplest way to print a spreadsheet is to enter **/O** to start the */Output* command, type **D** to print the spreadsheet as it is displayed, and then type **ALL,** to specify printing the entire spreadsheet, and finally press **P** to direct the output to the printer.

If your spreadsheet is too wide to fit onto a single sheet of paper, then SuperCalc will automatically split the worksheet into strips (you'll see how to tell SuperCalc how wide your printer is in a moment). First, SuperCalc will print as much of the spreadsheet as will fit on a page, and then it will print additional page-wide strips until the entire worksheet is printed.

To make this automatic system of dividing the spreadsheet into strips work you need to tell SuperCalc how many character columns that you want to print on each page. (Note that we are now talking about character columns, and not spreadsheet columns.) The */Output* command has a Setup option that, among other things, allows you to specify the number of character columns that will fit on a line. To use this option, enter **/O D ALL, S.** This will present you with a menu of setup choices. The selection that we are interested in is "W = Change page width". Select this option and enter the appropriate number of columns for the printing pitch and paper width that you want to use. SuperCalc will remember this setting until you exit the program. (You can make a setting permanent by using the INSTALL program that comes with SuperCalc.)

Table 4-3
Print columns on a page with SuperCalc

Pitch	Radix-10	Radix-15	Setup codes
Pica	80	136	⟨ESC⟩ B Ctrl-A
Elite	96	163	⟨ESC⟩ B Ctrl-B
Condensed	136	233	⟨ESC⟩ B Ctrl-C
Pica Expanded	40	68	⟨ESC⟩ B Ctrl-A ⟨ESC⟩ W Ctrl-A
Elite Expanded	48	81	⟨ESC⟩ B Ctrl-B ⟨ESC⟩ W Ctrl-A
Condensed Expanded	68	116	⟨ESC⟩ B Ctrl-C ⟨ESC⟩ W Ctrl-A
NLQ on	80	136	⟨ESC⟩ B Ctrl-D
NLQ off			⟨ESC⟩ B Ctrl-E

Sending control codes from SuperCalc

The Setup option of the */Output* command also allows you to send control codes to the printer. The menu item "S = Manual

setup codes" lets you send any type of code to the printer that you wish. When you select "S", a prompt appears that says "Enter codes (CR when done:)." You can then enter any codes that you wish, and when you are done you press return to signal the end of the setup codes. Control codes are sent in the normal manner, by holding down the control key and pressing a letter key. On some computers the ⟨ESC⟩ key will not send an escape code (ASCII 27). If this is a problem, try using control-[. Many times this will send an escape code to the printer.
As an example, to turn on NLQ printing, enter ⟨**ESC**⟩ **B control-D return,** and then press **P** to start printing the report. Table 4-3 shows the codes required to change the print width to various sizes.

Lotus 1-2-3

Lotus is one of the new integrated software packages that includes a spreadsheet, a database manager and graphics. We will see how to print Lotus 1-2-3 spreadsheets in this chapter.

Lotus 1-2-3 uses the */Print* command to print spreadsheets. When you enter **/P,** a menu appears that presents you with a number of choices. Lotus 1-2-3 gives you a lot of flexibility in printing spreadsheets through this menu, but the only thing you *have* to do is to define a range to print. All the other items have default values that make getting started easy.

If you do change several of the things listed on the */Print* menu, Lotus 1-2-3 will remember the selections that you have made and use them each time you print the spreadsheet. They are even saved with the spreadsheet so that they will be the same the next time that you use the spreadsheet.

You can specify the range to print in all the normal ways: by pointing, by typing the cell addresses of the endpoints, by entering a range name, or by using the F3 key to point to a range name.

After you have specified a range to print, and changing any of the other options that you wish, begin to print the spreadsheet by selecting the Go option. Lotus 1-2-3 will split the spreadsheet into sections to fit onto pages if it won't all fit on one page.

Let's look at some of the other options on the */Print* menu, and see how they add to the flexibility of printing spreadsheets.

The *Line* option advances the paper one line. Use this to put space between different sections of your spreadsheets when you print them. The *Page* option advances the paper to the top of a new page. Use this option to start on a new page.

Selecting the *Align* option tells Lotus 1-2-3 that you have moved the paper to the top of a new page. Use this option after using the F.F. button to move the paper or after inserting a new single sheet of paper.

The *Clear* option allows you to clear any or all of the other options that you have selected. The *Quit* option ends the */Print* command and returns you to Ready Mode.

Selecting *Options* from the */Print* menu presents you with some additional page format selections.

You can add *Headers* or *Footers* to each page of your output. A header is a line that prints at the top of each page, while a footer is a line that prints at the bottom of each page.

Lotus 1-2-3 has three characters that perform special functions when they are included in a header or a footer. You can include sequential page numbers on each page by including the # character where you want the page number to print (For example: Page #).

The current date will be printed if you include the @ character in a header or footer. (For example: As of @.)

You can direct sections of headers and footers to the left, right, or center by using the | character. Each header or footer is divided into three sections; Left, center, and right. The | character shows the limits of these sections. So to print a header with the date to the left, a title in the center, and a page number to the right, the header might look like this:

```
@|Spreadsheet Title|Page #
```

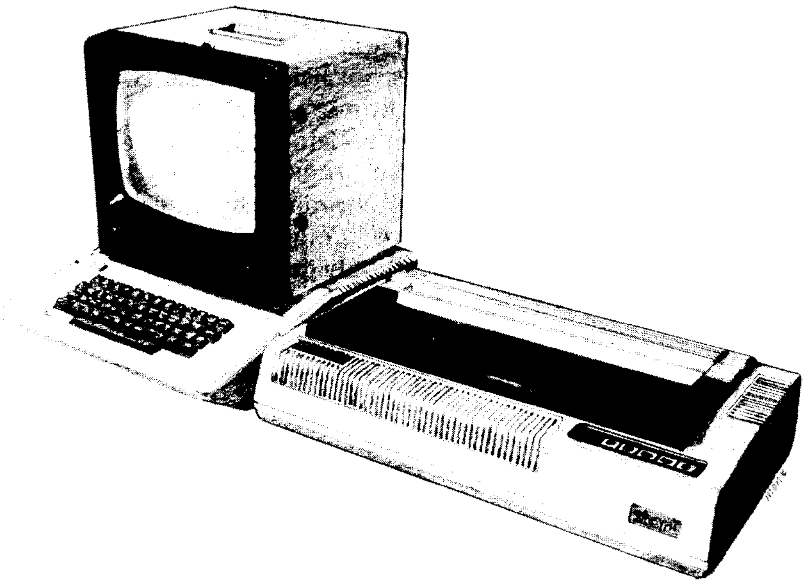
And, on January 12, 1984, the results might look like this:

```
          12-Jan-84    Spreadsheet Title      Page 1
```

Another of the selections under *Options* is *Setup*. This selection allows you to create a *setup string* that will be sent to the printer before each section of a spreadsheet is printed. You can include non-printing codes in the setup string by using a backslash (\) followed by a three digit number that consists of the decimal ASCII value for the code that you wish to send (with leading zeros if required). For example, to print a worksheet in condensed print, use the setup string \015. This sends ASCII 15 which is the code for condensed printing. The following table shows how many character columns will fit with different printing widths, and the setup string to get each width.

Table 4-4
Print columns on a page with Lotus 1-2-3

Pitch	Radix-10	Radix-15	Setup codes
Pica	80	136	\027B\001
Elite	96	163	\027B\002
Condensed	136	233	\027B\003
Pica Expanded	40	68	\027B\001 \027W\001
Elite Expanded	48	81	\027B\002 \027W\001
Condensed Expanded	68	116	\027B\003 \027W\001
NLQ on	80	136	\027B\004
NLQ off			\027B\005



Chapter 5

Using Radix With Graphics Programs

There are now many business graphics programs on the market. Radix is a good printer to use with these programs because of its advanced graphics capabilities. We will look at two graphics programs in some detail. If you have a different graphics program, then hopefully you can use some of the concepts in the programs that we cover to help you with your program.

In this chapter you'll learn how to use Radix with:

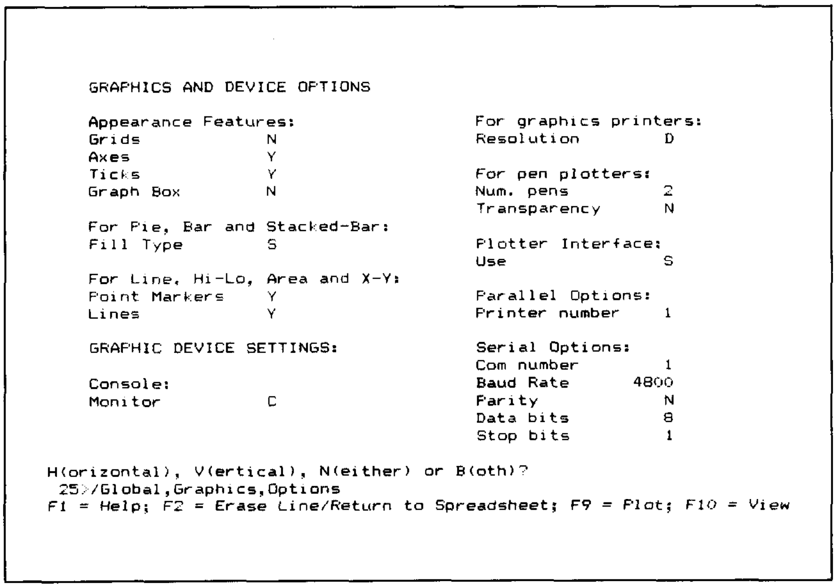
- **SuperCalc^3**
- **BPS Business Graphics**

SuperCalc^3

SuperCalc^3 can produce 7 different kinds of graphs. Using information contained on the worksheet you can create and print a wide variety of graphs.

Before you start printing graphs you must tell SuperCalc^3 what kind of printer you are using. This is done with the STARTUP program that is furnished on the SuperCalc^3 disk. To use this, type STARTUP at the A) prompt and follow the on-screen directions. If the Star Radix printer is not listed, please consult with your Star dealer for an alternate printer selection.

Once you have started the SuperCalc^3 program there are still some things that you must tell the program about your printer. Use the *|Global Graphics Options* Command by entering **/G G O**. You will see a menu like that shown in Figure 5-1. There are many things on this screen but we need only concern ourselves with a few.



```
    GRAPHICS AND DEVICE OPTIONS

    Appearance Features:                  For graphics printers:
    Grids              N                  Resolution          D
    Axes               Y
    Ticks              Y                  For pen plotters:
    Graph Box          N                  Num. pens           2
                                          Transparency        N
    For Pie, Bar and Stacked-Bar:
    Fill Type          S                  Plotter Interface:
                                          Use                 S
    For Line, Hi-Lo, Area and X-Y:
    Point Markers      Y                  Parallel Options:
    Lines              Y                  Printer number      1

    GRAPHIC DEVICE SETTINGS:              Serial Options:
                                          Com number          1
    Console:                              Baud Rate        4800
    Monitor            C                  Parity              N
                                          Data bits           8
                                          Stop bits           1

H(orizontal), V(ertical), N(either) or B(oth)?
 25>/Global,Graphics,Options
F1 = Help; F2 = Erase Line/Return to Spreadsheet; F9 = Plot; F10 = View
```

Figure 5-1. *SuperCalc^3* /Global Graphics Options menu.

The first selection to make is at the top of the second column. Use the Tab key to quickly move to the "Resolution:" entry. This choice affects the density and quality of the graphs that you print. There are four choices, single through quad density, but if you did not find the Radix listed in the STARTUP program only single and double will work. We suggest that you select double until you are more familiar with the program.

The next thing that you must specify is the "Plotter Interface" that you are using. Move to this selection and enter P if you are using the parallel interface on the Radix, or S if you are using the serial interface.

Your entry of S or P will affect your next entry also. Depending on which you have selected, a different section of the menu below will be highlighted. Here you need to specify which printer number the Radix is. Generally, printers are connected as printer number 1, so if you don't know any differently, enter 1. Then, if you are using a serial interface you must enter the values that reflect the way that you set the DIP switches on Radix. If these settings are not consistent with the way the switches are set then it will not work.

Once these selections are made, enter **/G G S** to save your selections to disk so that you won't have to make these selections every time that you want to print a graph.

Now you are ready to print a graph. Create a graph using the */View* command, and when it is in the form that you like, press the F9 key to print the graph on Radix.

By using the */Global Graphics Fonts* command you can change the style of letters that are on your graphs. SuperCalc^3 has eight different fonts available and you can use any or all of them on any graph.

You can change the size and placement of the graph on the page by using the */Global Graphics Layout* command. This command allows you to put up to four graphs on a page.

BPS Business Graphics

BPS Business Graphics is a program that changes groups of numbers into graphs. It is a very flexible program and can produce many different kinds of graphs.

BPS Business Graphics prints graphs by first creating the graph on the screen and then making a copy of the screen on the printer. This means that your computer must have the ability to do graphics on the screen.

Before you can print a graph you must have the proper *device driver* installed. Use the INSTALL DEVICES B: command with the device installation disk in drive B:. If you do not find RADIX in the list of available devices, please ask your Star dealer for an alternate printer selection. You only need to do this installation once;

BPS Business Graphics will use the same device driver until you install another one.

Now, when you have your points loaded and are ready to print a graph, use the following commands:

```
DRAW (Plus the appropriate arguments)
SET OUTPUT UNIT PARALLEL
WRITE SCREEN (Plus the appropriate device name)
```

Your graph will first be displayed on the screen and then printed on Radix.

There are several ways to vary the way graphs are printed. Obviously you can use the many features of BPS Business Graphics to add elements to your graph. But you can also change the quality of the way the graph is printed.

One way to change the printing is to use the SET FILL CYCLE command. This will change the pattern of each element of the graph as it is drawn. In a pie graph it will change the pattern in each section of the graph.

Another way to improve the quality of the printed graph is to use the DET DEVICE CONSOLE HIBW command. This command tells the program that you have a high resolution monitor. BPS Business Graphics then plots in its highest quality, which may not look very good on your screen if you don't have a high resolution monitor. But when the image is printed it will be in the highest possible quality.

Making pie graphs round

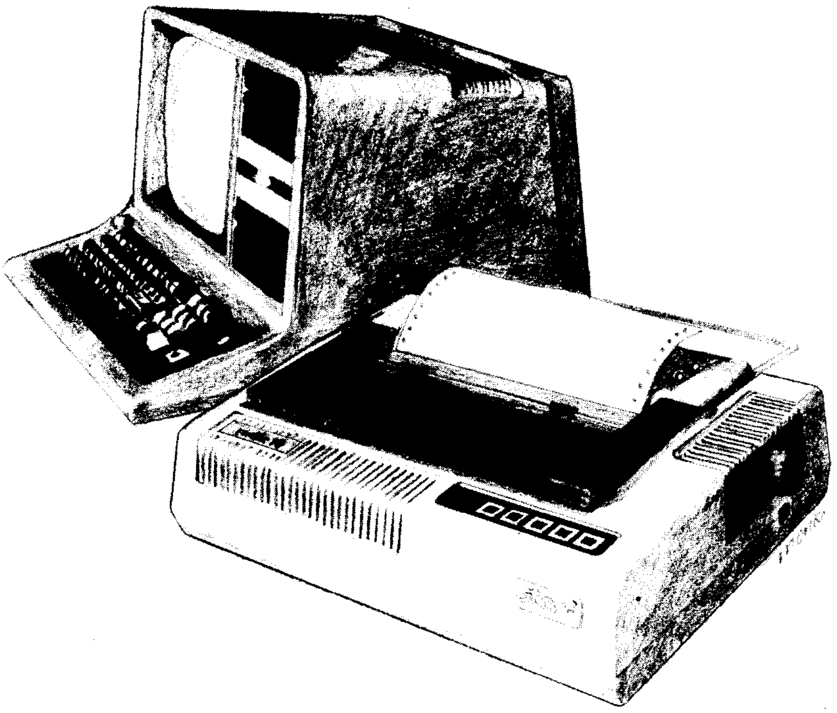
BPS Business Graphics uses what it calls a *rounding factor* to make pie graphs round. Many output devices produce oval pie graphs because their *aspect ratios* don't match. The aspect ratio of an object is the ratio of its height to its width. The rounding factor is the third argument to the DRAW PIE command, so with a rounding factor of 1, the command looks like this: DRAW PIE 0,0,1.

The following table shows the rounding factors to use with the different console modes.

BPS Business Graphics can create many types of graphs, and we have just touched the surface of its capabilities, but with these hints you should be able to get started printing graphs quickly.

Table 5-1
Rounding factors for console modes

Console Mode	Rounding Factor
COLOR	1.21
RGB	1.21
TV	0.61
HIBW	2.43
LOBW	1.21



Chapter 6

Controlling Radix With BASIC

Throughout the rest of this book we will be teaching you how to use Radix's features using the BASIC programming language in our examples. It is easy to communicate with Radix 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 Radix, as we have already seen.

Subjects covered in this chapter include:

- **Listing BASIC programs on the printer**
- **Printing from BASIC**
- **CHR$ function**
- **Problem codes**

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 and utility programs can be adapted to run in any version of BASIC. In this chapter and in the appendix for your computer, 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 Radix. Now we are ready to jump into the world of programming with Radix. 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. Again take a few moments to look at the appendix that relates to your computer. We'll continue when you have it all figured out.

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

```
NEW
10 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 2 as one way to express ASCII codes. We are going to use it a lot in communicating with Radix. Radix uses many of the ASCII codes 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$(82)
RUN
```

That should print an "R" for Radix. If you check the chart in Appendix I you will see that 82 is the ASCII code for "R".

Control codes

Radix 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 Radix's bell. We will learn more about it in Chapter 10. We just wanted to illustrate a code that causes Radix 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 Radix'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 italic print is:

```
<ESC> CHR$(52)
```

In a program, this would look like this:

```
NEW
10 LPRINT CHR$(27) CHR$(52);
20 LPRINT "TESTING"
RUN
```

Try this program. It will print the word TESTING in italic.

Some of you fast students may have noticed that CHR$(52) is the same as "4". That's right, the program will work just as well if line 10 is changed like this:

```
10 LPRINT CHR$(27) "4";
```

That's just another form of the same ASCII code, and it's all the same to Radix.

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)
10 LPRINT ESC$ "4";
```

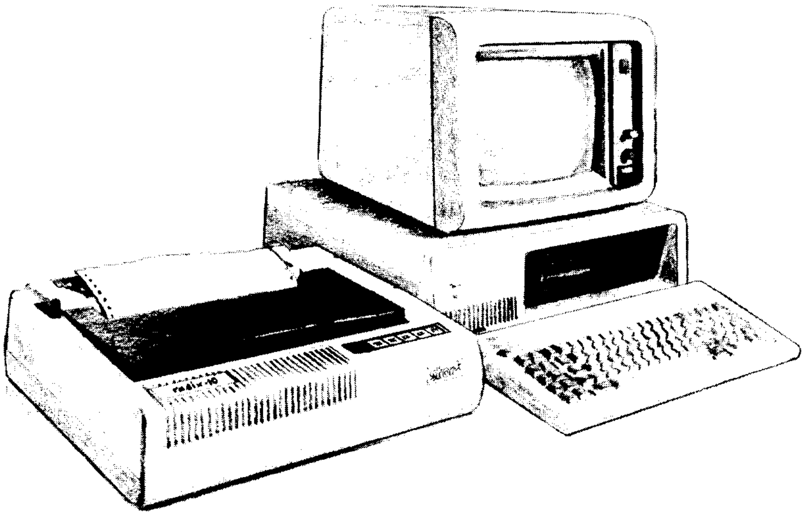
Turn your printer off and back on now, or you will be printing in italic for quite a while!

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. Once again we refer you to the appendix about your computer, where some more specific information awaits.

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



Chapter 7

Printing Text With Radix

Beginning with this chapter we will be exploring all the features of Radix.

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 Radix's features. Just use the same ASCII codes as we do in our examples.

If your computer doesn't use Microsoft BASIC, look in the appendix to see what changes you need to make for your BASIC. The appendix tells you how to change the short example programs, and gives complete listings of the longer programs, already converted for your computer.

You have already printed a few lines on your Radix 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

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

Near Letter Quality characters

Radix'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 A-4 off), Radix prints draft quality characters. This is adequate for most work and it prints fastest. But for the final printout, try NLQ. The program below shows how.

```
10 'Demo near letter quality character set.
20 LPRINT CHR$(27) "B" CHR$(4)  ; 'Select NLQ.
30 LPRINT "This line shows Radix's NEAR LETTER QUALITY!"
40 LPRINT CHR$(27) "B" CHR$(5)  ; 'Select draft.
50 LPRINT "This line shows Radix's standard print."
```

In this program, line 20 selects NLQ characters with the ⟨ESC⟩ "B" CHR$(4) command. Line 30 prints a sample before line 40 switches Radix back to draft printing with an ⟨ESC⟩ "B" CHR$(5). When you run the program you should get this:

```
This line shows Radix's NEAR LETTER QUALITY!
This line shows Radix's standard print.
```

Table 7-1
Near letter quality commands

Function	Control code
Near letter quality ON	〈ESC〉 "B" CHR$(4)
Near letter quality OFF	〈ESC〉 "B" CHR$(5)

Italic printing

Italic letters are letters that are slanted to the right. Radix can print all of its letters except NLQ characters in *italic* as well as the roman (standard) letters you are accustomed to. Italics can be used to give extra emphasis to certain words. The command codes to turn italic on and off are shown in Table 7-2.

Table 7-2
Italic commands

Function	Control code
Italic ON	〈ESC〉 "4"
Italic OFF	〈ESC〉 "5"

Use this program 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 Radix print all styles of printing in both roman and italic, but it can underline them too. The control codes are shown in Table 7-3.

Table 7-3
Underline commands

Function	**Control code**
Underline ON	⟨ESC⟩ "-" CHR$(1)
Underline OFF	⟨ESC⟩ "-" CHR$(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 continued. Therefore, BASIC didn't send a carriage return and line feed at the end of those lines. We just did this to illustrate that all these control codes can be used in the middle of a line. It's easy to underline or *italicize* only part of a line.

Superscripts and subscripts

Radix 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 7-4 has the codes for using superscripts and subscripts.

Table 7-4
Superscript and subscript commands

Function	**Control code**
Superscript ON	⟨ESC⟩ "S" CHR$(0)
Subscript ON	⟨ESC⟩ "S" CHR$(1)
Super & subscript OFF	⟨ESC⟩ "T"

Try this program to see them work:

```
1Ø 'Demo subscripts and superscripts.
2Ø LPRINT "Look! " ;
3Ø LPRINT CHR$(27) "S" CHR$(Ø) ; 'Superscript on.
4Ø LPRINT "Superscripts " ;
5Ø LPRINT CHR$(27) "T" ; 'Cancel superscripts.
6Ø LPRINT "& " ;
7Ø LPRINT CHR$(27) "S" CHR$(1) ; 'Subscripts on.
8Ø LPRINT "subscripts " ;
9Ø LPRINT CHR$(27) "T" ; 'Cancel subscripts.
1ØØ 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, Radix prints 10 characters per inch. This is called *pica* pitch because it's the same spacing as a standard pica typewriter.

Radix 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.

You tell Radix which pitch you want to use with the ⟨ESC⟩ "B" command. The table below shows three options of this command.

Table 7-5
Print pitch commands

Pitch	Characters/inch	Control code
Pica	10	⟨ESC⟩ "B" CHR$(1) or CHR$(18)
Elite	12	⟨ESC⟩ "B" CHR$(2)
Condensed	17	⟨ESC⟩ "B" CHR$(3) or CHR$(15)

Let's see how these three pitches look. Try this program:

```
10 'Demo all pitches.
20 LPRINT CHR$(27) "B" CHR$(3) ; 'Select condensed
   pitch.
30 LPRINT "This line is CONDENSED pitch."
40 LPRINT CHR$(27) "B" CHR$(2) ; 'Select elite pitch.
50 LPRINT "This line is ELITE pitch."
60 LPRINT CHR$(27) "B" CHR$(1) ; 'Select pica pitch.
70 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 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 Radix to elite pitch and line 50 prints a line in elite pitch. Line 60 resets Radix to pica pitch and line 70 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 Radix's three 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:

```
1Ø 'Demo permanent expanded mode.
2Ø LPRINT CHR$(27) "W" CHR$(1) ; 'Expanded mode on
   permanently.
3Ø LPRINT "Permanent expanded"
4Ø LPRINT "mode stays on until"
5Ø LPRINT "it is ";
6Ø LPRINT CHR$(27) "W" CHR$(Ø) ; 'Expanded mode off.
7Ø 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 7-6
Expanded print commands

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

By combining expanded print with the three pitches, Radix has six 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$(27) "W" CHR$(Ø) ; 'Permanent expanded
   mode off.
1ØØ LPRINT "This is UNEXPANDED PICA pitch (default)."
```

Here's what you should get from this program:

```
This line is EXPANDED CONDENSED pitch.
This is EXPANDED ELITE.
This is EXPANDED PICA.
This is UNEXPANDED PICA pitch (default).
```

Making Radix Print Darker

Radix 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. Radix 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 7-7
Print emphasis commands

Function	Control code
Double-strike ON	⟨ESC⟩ "G"
Double-strike OFF	⟨ESC⟩ "H"
Emphasized ON	⟨ESC⟩ "E"
Emphasized OFF	⟨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 Radix 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 Radix 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 Radix's many different printing modes individually. Now let's see how we can combine these modes for even more printing effects. Condensed, italic, double-strike, underlined subscripts are something that you are probably just itching to print!

There are 336 "theoretical" combinations of the modes that we have learned. Of these, a mere 114 will work! (Some combinations, like expanded superscripts, just don't work.) Instead of trying to list all the combinations that work, we have a program that prints a chart showing all the combinations. There is a sample of each of the 114 possible combinations on the chart. (The dots just indicate the few combinations that *don't* work.) Enter the following program and run it to make your own chart.

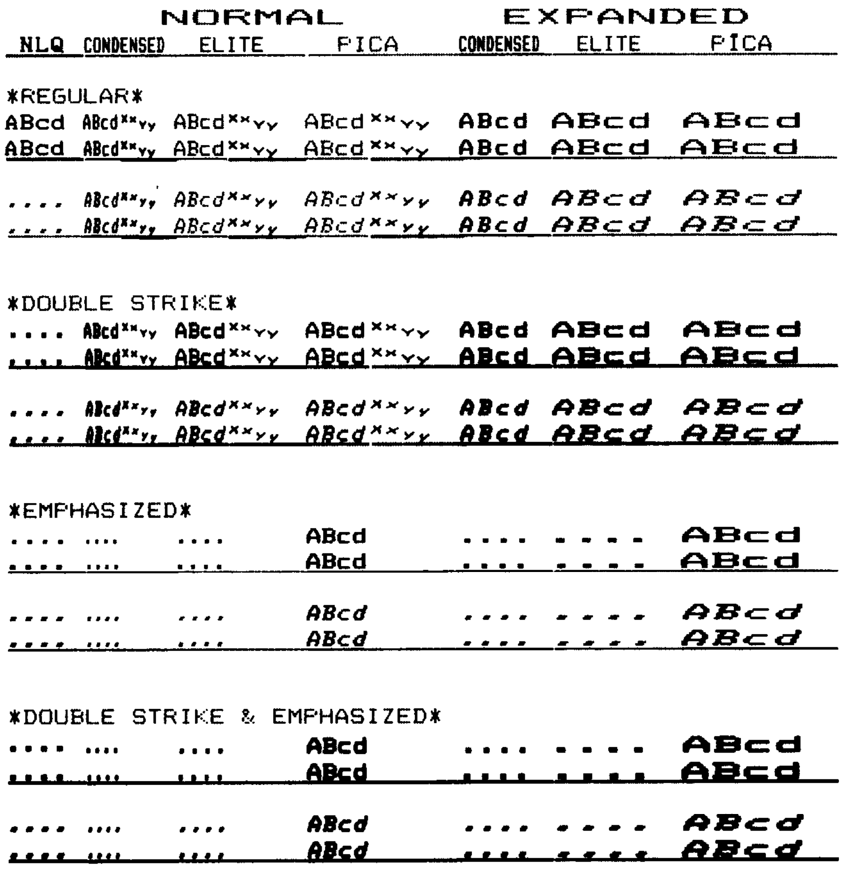
```
10 'Prints a chart of all RADIX print styles.
20 WIDTH "LPT1:", 255 'Cancel auto CR & LF after 80
   chars.
30 '
```

```
40 'Initialize constants.
50 ITALIC$ = CHR$(27) + "4"
60 ROMAN$  = CHR$(27) + "5"
70     EXPANDED$   = CHR$(27) + "W" + CHR$(1)
80 NOT.EXPANDED$   = CHR$(27) + "W" + CHR$(0)
90     PICA$       = CHR$(27) + "B" + CHR$(1)
100     ELITE$      = CHR$(27) + "B" + CHR$(2)
110     CONDENSED$ = CHR$(27) + "B" + CHR$(3)
120     NLQ$        = CHR$(27) + "B" + CHR$(4)
130 NOT.NLQ$        = CHR$(27) + "B" + CHR$(5)
140     EMPHASIZED$     = CHR$(27) + "E"
150 NOT.EMPHASIZED$     = CHR$(27) + "F"
160     DOUBLE.STRIKE$ = CHR$(27) + "G"
170 NOT.DOUBLE.STRIKE$ = CHR$(27) + "H"
180     UNDERLINED$     = CHR$(27) + "-" + CHR$(1)
190 NOT.UNDERLINED$     = CHR$(27) + "-" + CHR$(0)
200     SUPERSCRIPT$    = CHR$(27) + "S" + CHR$(0)
210     SUBSCRIPT$      = CHR$(27) + "S" + CHR$(1)
220 NOT.SCRIPTED$       = CHR$(27) + "T"
230 RESET.ALL$ = NOT.EMPHASIZED$ + NOT.UNDERLINED$ +
   NOT.DOUBLE.STRIKE$
240 RESET.ALL$ = RESET.ALL$ + ROMAN$ + PICA$ +
   NOT.EXPANDED$ + NOT.NLQ$
250 TRUE = 1 :FALSE = 0
260 '
270 '
280 'Print heading.
290 LPRINT RESET.ALL$
300 LPRINT EXPANDED$ "     NORMAL     EXPANDED   "
310 LPRINT RESET.ALL$;
320 LPRINT UNDERLINED$ ;
330 LPRINT NLQ$        " NLQ " NOT.NLQ$ ;
340 LPRINT CONDENSED$ "CONDENSED " ;
350 LPRINT ELITE$      "  ELITE    " ;
360 LPRINT PICA$       "  PICA     " ;
370 LPRINT CONDENSED$ "CONDENSED " ;
380 LPRINT ELITE$      "  ELITE    " ;
390 LPRINT PICA$       "  PICA     "
400 LPRINT RESET.ALL$
410 LPRINT "*REGULAR*"
420 GOSUB 540 'Print four lines regular.
430 LPRINT  "*DOUBLE STRIKE*"
440 LPRINT DOUBLE.STRIKE$;
450 DS.OR.EMP = TRUE
```

```
460 GOSUB 540 'Print four lines double strike.
470 LPRINT "*EMPHASIZED*"
480 EMPHASIZED = TRUE
490 GOSUB 540 'Print four lines emphasized.
500 LPRINT "*DOUBLE STRIKE & EMPHASIZED*"
510 LPRINT DOUBLE.STRIKE$ EMPHASIZED$;
520 GOSUB 540 'Print double strike & emphasized.
530 END
540 '
550 'Subroutine to print four lines.
560 'Each shows NLQ, also the three different pitches
570 'are shown in normal and expanded.
580 'Roman, roman underlined, italic, and italic
   underlined.
590 '
600 ITALICS    = FALSE      :LPRINT ROMAN$ ;
610 UNDERLINED = FALSE      :LPRINT NOT.UNDERLINED$ ;
620 EXPANDED   = FALSE      :LPRINT NOT.EXPANDED$ ;
630 PICA       = FALSE
640 '
650 'Produce a line in four different pitches.
660 IF EXPANDED THEN 720
670 IF ITALICS THEN LPRINT ".... " ; : GOTO 720
680 IF DS.OR.EMP THEN LPRINT ".... " ; : GOTO 720
690 LPRINT NLQ$ ; : NLQ = TRUE
700 GOSUB 860                   'Print near-letter-
   quality.
710 LPRINT NOT.NLQ$ ; : NLQ = FALSE
720 LPRINT CONDENSED$ ;
730 GOSUB 860                   'Print condensed.
740 LPRINT ELITE$ ;
750 GOSUB 860                   'Print elite.
760 LPRINT PICA$ ; : PICA = TRUE
770 GOSUB 860                   'Print pica.
780 '
790 'See what has just been done and prepare for next
   line.
800 IF EXPANDED = TRUE THEN LPRINT :GOTO 820
810 LPRINT EXPANDED$; :EXPANDED = TRUE :GOTO 630
820 IF UNDERLINED = TRUE THEN LPRINT :GOTO 840
830 LPRINT UNDERLINED$;  :UNDERLINED = TRUE :GOTO 620
840 IF ITALICS = TRUE THEN LPRINT RESET.ALL$ :RETURN
850 LPRINT ITALIC$;  :ITALICS = TRUE  :GOTO 610
860 '
```

```
870 'Print a small sample showing upper case, lower
   case.
880 'Also show subscripts and superscripts if
   appropriate.
890 '
900 BLANK$ = STRING$(6,32)   :FOUR.DOT$ = "...."
910 IF EMPHASIZED = FALSE THEN LPRINT "ABcd";  :GOTO 970
920 IF PICA = FALSE THEN LPRINT FOUR.DOT$;     :GOTO 940
930 LPRINT EMPHASIZED$ "ABcd" ;
940 IF EXPANDED = TRUE THEN LPRINT " ";        :ELSE
   LPRINT BLANK$;
950 RETURN
960 '
970 'Handle non-emphasized cases.
980 IF EXPANDED OR NLQ THEN LPRINT " ";         :RETURN
990 LPRINT SUPERSCRIPT$; "Xx";
1000 LPRINT SUBSCRIPT$;   "Yy  ";
1010 LPRINT NOT.SCRIPTED$;
1020 RETURN
```

Here is the chart it produces:

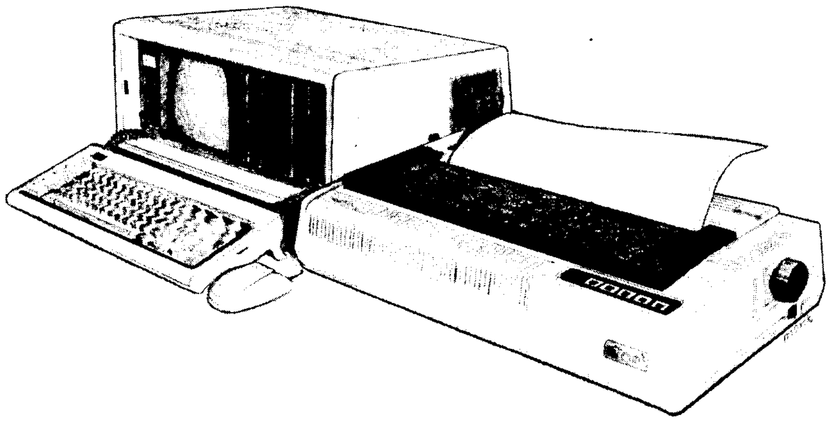


```
              NORMAL                          EXPANDED
 NLQ  CONDENSED  ELITE      PICA        CONDENSED  ELITE   PICA

*REGULAR*
ABcd  ABcdxxyy  ABcdxxyy  ABcdxxyy      ABcd     ABcd    ABcd
ABcd  ABcdxxyy  ABcdxxyy  ABcdxxyy      ABcd     ABcd    ABcd

....  ABcdxxyy  ABcdxxyy  ABcdxxyy      ABcd     ABcd    ABcd
....  ABcdxxyy  ABcdxxyy  ABcdxxyy      ABcd     ABcd    ABcd

*DOUBLE STRIKE*
....  ABcdxxyy  ABcdxxyy  ABcdxxyy      ABcd     ABcd    ABcd
....  ABcdxxyy  ABcdxxyy  ABcdxxyy      ABcd     ABcd    ABcd

....  ABcdxxyy  ABcdxxyy  ABcdxxyy      ABcd     ABcd    ABcd
....  ABcdxxyy  ABcdxxyy  ABcdxxyy      ABcd     ABcd    ABcd

*EMPHASIZED*
....  ....      ....      ABcd          ....     ....    ABcd
....  ....      ....      ABcd          ....     ....    ABcd

....  ....      ....      ABcd          ....     ....    ABcd
....  ....      ....      ABcd          ....     ....    ABcd

*DOUBLE STRIKE & EMPHASIZED*
....  ....      ....      ABcd          ....     ....    ABcd
....  ....      ....      ABcd          ....     ....    ABcd

....  ....      ....      ABcd          ....     ....    ABcd
....  ....      ....      ABcd          ....     ....    ABcd
```

Summary

Control code	Function
〈ESC〉 "B" CHR$(4)	Near letter quality on
〈ESC〉 "B" CHR$(5)	Near letter quality off
〈ESC〉 "4"	Italic on
〈ESC〉 "5"	Italic off
〈ESC〉 "–" CHR$(1)	Underline on
〈ESC〉 "–" CHR$(0)	Underline off
〈ESC〉 "S" CHR$(0)	Superscript on

〈ESC〉 "S" CHR$(1)	Subscript on
〈ESC〉 "T"	Super & subscript off
〈ESC〉 "B" CHR$(1)	Sets pica pitch
〈ESC〉 "B" CHR$(2)	Sets elite pitch
〈ESC〉 "B" CHR$(3)	Sets condensed pitch
CHR$(18)	Sets pica pitch
CHR$(15)	Sets condensed pitch
CHR$(14)	One line expanded
〈ESC〉 CHR$(14)	One line expanded
CHR$(20)	One line expanded off
〈ESC〉 "W" CHR$(1)	Expanded on
〈ESC〉 "W" CHR$(0)	Expanded off
〈ESC〉 "G"	Double-strike on
〈ESC〉 "H"	Double-strike off
〈ESC〉 "E"	Emphasized on
〈ESC〉 "F"	Emphasized off



Chapter 8

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 escape 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 C-4 is off; see below).

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

Now to add a little confusion—most (but not all) versions of BASIC add a line feed (CHR$(10)) to every carriage return (CHR$(13)) that they send. If your version of BASIC doesn't do this, then you should turn DIP switch C-4 on so that Radix will add the line feed for you. When you have DIP switch C-4 on the printer will do the same thing when it receives a carriage return as it does when it receives a line feed.

If you find that your printer double spaces when it should single space, then you probably need to turn DIP switch C-4 off.

Reverse line feeds

Your Radix 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 8-1
Line feed commands

Function	ASCII code	Control code
Return print head to left margin	⟨CR⟩	CHR$(13)
Advance paper one line	⟨LF⟩	CHR$(10)
Reverse paper one line		⟨ESC⟩CHR$(10)

Changing Line Spacing

When you turn Radix on the line spacing is set to 6 lines per inch (or 8 lines per inch if DIP switch A-5 is off). This is fine for most printing applications, but sometimes you may want something different. Radix makes it easy to set the line spacing to whatever value you want.

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

```
10 'Demo variable line spacing.
20 OPEN "LPT1:" AS #1 : WIDTH #1,255
30 FOR I = 1 TO 25
40 'Set line spacing.
50 PRINT #1,CHR$(27) "A" CHR$(I) ;
60 LPRINT "RADIX line spacing set to " I
70 NEXT I
80 LPRINT "Line spacing is set to 1/6 inch (normal)."
90 'Set line spacing to 1/6 inch (normal).
100 LPRINT CHR$(27)  "2" ;
110 CLOSE #1
```

The printout is shown on the next page.

In this program, notice that we're sending codes to the printer a different way. In addition to the LPRINT statements for character strings, we've opened the printer as a random file in line 20. This method, which works with most versions of Microsoft BASIC, allows us to send codes that would otherwise be "problem codes" (such as CHR$(13), which BASIC automatically follows with a CHR$(10)). The codes are sent with the PRINT #1 statement in line 50. Unfortunately, this method doesn't work for all computers, but the appendix for your computer shows some other ways to send "problem codes."

Line 50 changes the line spacing. The command ⟨ESC⟩ "A" CHR$(n) changes the line spacing to n/72 of an inch. The loop that is started in line 30 increases the value of n (the variable I in the program) each time it is executed. So the line spacing increases as the program continues. Finally, the ⟨ESC⟩ "2" in line 100 resets the line spacing to 6 lines per inch. This is a shortcut that is the same as ⟨ESC⟩ "A" CHR$(12).

You may wonder why they picked 1/72 of an inch as the increment for the line spacing command. There's a good reason: the

```
RADIX line spacing set to 1
RADIX line spacing set to 2
RADIX line spacing set to 3
RADIX line spacing set to 4
RADIX line spacing set to 5
RADIX line spacing set to 6
RADIX line spacing set to 7
RADIX line spacing set to 8
RADIX line spacing set to 9
RADIX line spacing set to 10
RADIX line spacing set to 11
RADIX line spacing set to 12
RADIX line spacing set to 13
RADIX line spacing set to 14
RADIX line spacing set to 15
RADIX line spacing set to 16
RADIX line spacing set to 17
RADIX line spacing set to 18
RADIX line spacing set to 19
RADIX line spacing set to 20
RADIX line spacing set to 21
RADIX line spacing set to 22
RADIX line spacing set to 23
RADIX line spacing set to 24
RADIX line spacing set to 25

Line spacing is set to 1/6 inch (normal).
```

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.

The 〈ESC〉 "3" CHR$(n) command sets the line spacing in increments of 1/144 inch. Change line 50 in your program so it is like this:

```
5Ø PRINT #1,CHR$(27) "3" CHR$(I) ;
```

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

```
[illegible]
RADIX line spacing set to 10
RADIX line spacing set to 11
RADIX line spacing set to 12
RADIX line spacing set to 13
RADIX line spacing set to 14
RADIX line spacing set to 15
RADIX line spacing set to 16
RADIX line spacing set to 17
RADIX line spacing set to 18
RADIX line spacing set to 19
RADIX line spacing set to 20
RADIX line spacing set to 21
RADIX line spacing set to 22
RADIX line spacing set to 23
RADIX line spacing set to 24
RADIX line spacing set to 25
Line spacing is set to 1/6 inch (normal).
```

The program works just the same as before, but the line spacings are just half what they were. This is because 〈ESC〉 "3" CHR$(n) sets the line spacing to n/144 inch.

Table 8-2 shows all the line spacing commands, including several "shortcut" commands for commonly used line spacings.

Let's take a look at the last two commands in the table, which give a one-time line feed (or reverse line feed) of n/144 inch. The 〈ESC〉 "J" CHR$(n) command does not *change* the setting of the line spacing, but it does cause the printer to make one line feed of n/144 inch. 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 1ØØ/144 inch.
```

Table 8-2
Line spacing commands

Function	Control code
Set line spacing to *n*/72 inch	⟨ESC⟩ "A" CHR$(*n*)
Set line spacing to n/144 inch	⟨ESC⟩ "3" CHR$(n)
Set line spacing to 1/8 inch	⟨ESC⟩ "0"
Set line spacing to 7/72 inch	⟨ESC⟩ "1"
Set line spacing to 1/6 inch	⟨ESC⟩ "2"
One-time line feed of *n*/144 inch	⟨ESC⟩ "J" CHR$(*n*)
One-time reverse line feed of n/144 inch	⟨ESC⟩ "j" CHR$(n)
Advance paper *n* lines	⟨ESC⟩ "a" CHR$(n)

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

```
5Ø LPRINT CHR$(27) "J" CHR$(1ØØ) ;
6Ø LPRINT "Line number 3."
7Ø LPRINT "Line number 4."
```

Here is what Radix will produce:

```
Line number 1.
Line number 2.




Line number 3.
Line number 4.
```

The ⟨ESC⟩ "J" CHR$(100) in line 50 changes the line spacing to 100/144 for one line only. The rest of the lines are printed with the normal line spacing. Notice that both line 30 and line 50 end with semicolons. This prevents the normal line feed from occurring.

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

```
4Ø 'One time reverse line feed 1ØØ/144 inch.
5Ø LPRINT CHR$(27) "j" CHR$(1ØØ) ;
```

```
Line number 3.
Line number 4.

Line number 1.
Line number 2.
```

The value of *n* in all four commands (〈ESC〉 "A", 〈ESC〉 "3", 〈ESC〉 "J", and 〈ESC〉 "j") can range from 0 to 255. A value of 0 means that there is *no* line spacing. This allows you to print multiple lines in the same position on the page. This is useful when you want to overprint graphics and text.

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 〈ESC〉 "a" CHR$(*n*) command does just that. This command advances the paper *n* lines (using whatever the current line spacing is) without moving the printhead. Change lines 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

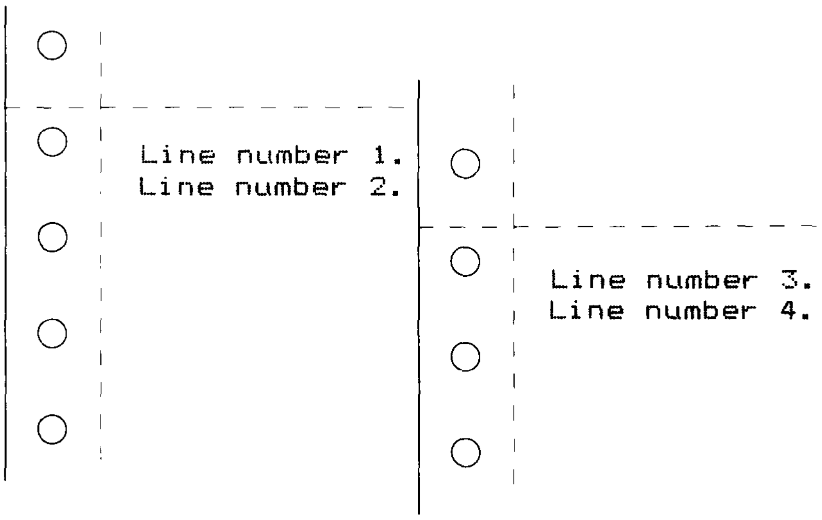
We have seen how to control the spacing between lines on a page. Radix also has commands that control the placement of printing on the page, and even adjust for different size pages.

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:

```
4Ø 'Form feed
5Ø 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 1. 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$(12) is a problem code for the TRS-80. To send a form feed command to Radix you must add 128 to it making it CHR$(140). Use CHR$(140) where we use CHR$(12) in these programs.

Reverse form feed

Just as Radix 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 8-3
Form feed commands

Function	ASCII code	Control code
Advance paper to top of next page	⟨FF⟩	CHR$(12)
Reverse paper to top of current page		⟨ESC⟩ CHR$(12)

Changing the Page Length

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

Table 8-4
Form length commands

Function	Control code
Set the page length to *n* lines	⟨ESC⟩ "C" CHR$(*n*)
Set the page length to *n* inches	⟨ESC⟩ "C" CHR$(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.

```
10 'Demo variable form lengths.
20 LPRINT CHR$(27) "C" CHR$(0) CHR$(7) ; 'Form length 7
   inches.
30 LPRINT "Pay to the order of:"
40 LPRINT CHR$(12) ; 'Form feed.
50 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 to 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 Radix has a solution to this predicament. Radix 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 8-5
Top and bottom margin commands

Function	**Control code**
Set top margin	⟨ESC⟩ "R" CHR$(n)
Set bottom margin	⟨ESC⟩ "N" CHR$(n)
Clear top and bottom margins	⟨ESC⟩ "O"

In both cases the value of *n* tells Radix how many lines to skip, although there is a slight difference in the usage. When you set the top margin with ⟨ESC⟩ "R" CHR$(*n*), the value of *n* tells Radix what line to start printing on. When you set the bottom margin with ⟨ESC⟩ "N" CHR$(*n*), the value of *n* tells Radix how many blank lines should be left at the bottom of the page.

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

```
10 'Demo top and bottom margins.
20 LPRINT CHR$(12) ; 'Form feed.
30 FOR I = 1 TO 150
40 LPRINT "This is line" I
50 NEXT I
60 LPRINT CHR$(12) ; 'Form feed.
```

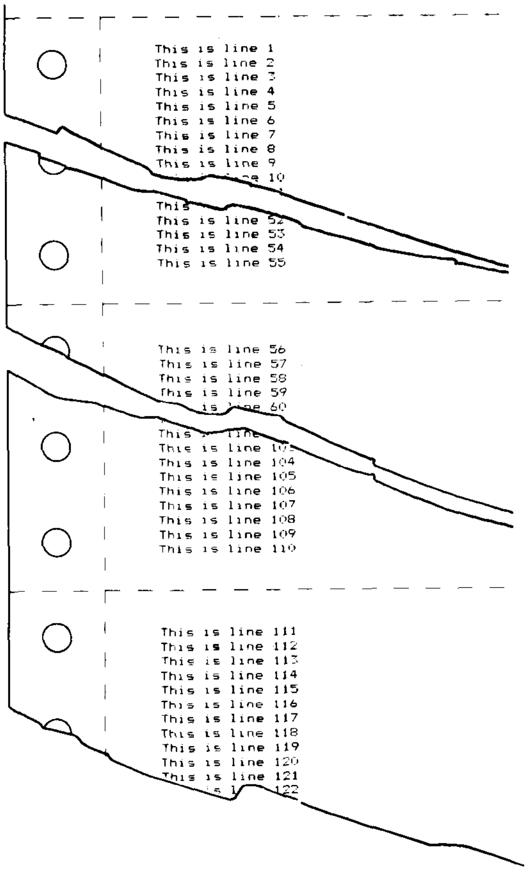
When you run this program it will print 150 lines right down the page and across the perforations. When it's done line 60 sends a form feed to advance the paper to the top of the next page. Look at the lines that have printed near the perforations. Separate the sheets and see if any of the lines have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

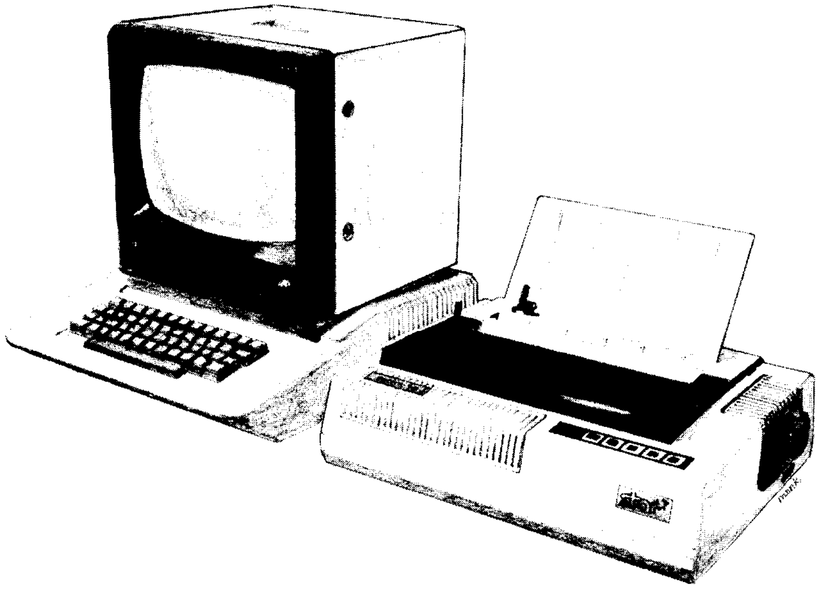


```
This is line 1
This is line 2
This is line 3
This is line 4
This is line 5
This is line 6
This is line 7
This is line 8
This is line 9
This is line 52
This is line 53
This is line 54
This is line 55

This is line 56
This is line 57
This is line 58
This is line 59
This is line 104
This is line 105
This is line 106
This is line 107
This is line 108
This is line 109
This is line 110

This is line 111
This is line 112
This is line 113
This is line 114
This is line 115
This is line 116
This is line 117
This is line 118
This is line 119
This is line 120
This is line 121
```

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
⟨ESC⟩ "3" CHR$(n)	Set line spacing to n/144 inch
⟨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

〈ESC〉 "J" CHR$(n)	One-time line feed of *n*/144 inch
〈ESC〉 "j" CHR$(n)	One-time reverse line feed of *n*/144 inch
〈ESC〉 "a" CHR$(n)	Advance the paper *n* lines
CHR$(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$(n)	Set top margin; start printing on line *n*
〈ESC〉 "N" CHR$(n)	Set bottom margin; leave *n* lines blank
〈ESC〉 "O"	Clear top and bottom margins



Chapter 9

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. Radix 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*, Radix 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 Radix on there are horizontal tabs set automatically every ten spaces. If you start counting at column 1 they are at columns 10, 20, 30, 40, etc. It's easy to use these tabs; you just send a CHR$(9) to Radix and the print head will move to the next tab position. CHR$(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 Radix. We can "outsmart" these computers 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 C.

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

```
15 LPRINT CHR$(27) "D" CHR$(8) CHR$(16) CHR$(24) 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 8, 16, and 24. 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.

If you set tabs in one pitch, such as pica, and then change the pitch, say to elite, the tab settings will also change. If, for example, the tabs are set every eight spaces, when you change pitch they will still be set every eight spaces, but the spaces will be a different width.

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 9-1
Horizontal tab commands

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

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

Setting Left and Right Margins

Radix'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 9-2
Left and right margin commands

Function	**Control code**
Set left margin at column n	⟨ESC⟩ "M" CHR$(n)
Set right margin at column n	⟨ESC⟩ "Q" CHR$(n)

Try setting Radix's margins with this program:

```
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 61 characters between the margins. (There's room for 61 (instead of 60) characters because you can print in both the first and last column that you name.)

Run the program. The results will look like this:

```
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
         XXXXXXXXXXXXXXXXXXX
```

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 1 and a right margin of 80 on Radix-10 or 136 on Radix-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 default vertical tab settings are every six lines. If you send a CHR$(11), which is the ASCII code for ⟨VT⟩, before we have set up tabs it will advance the paper to one of these preset tabs. 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."
```

The CHR$(11) in each line advances the paper to the next vertical tab. The lines should be spaced evenly, six lines apart.

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. 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.

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 Radix—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?

```
First tab.









Second tab.


















Third tab.









Fourth tab.
```

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.

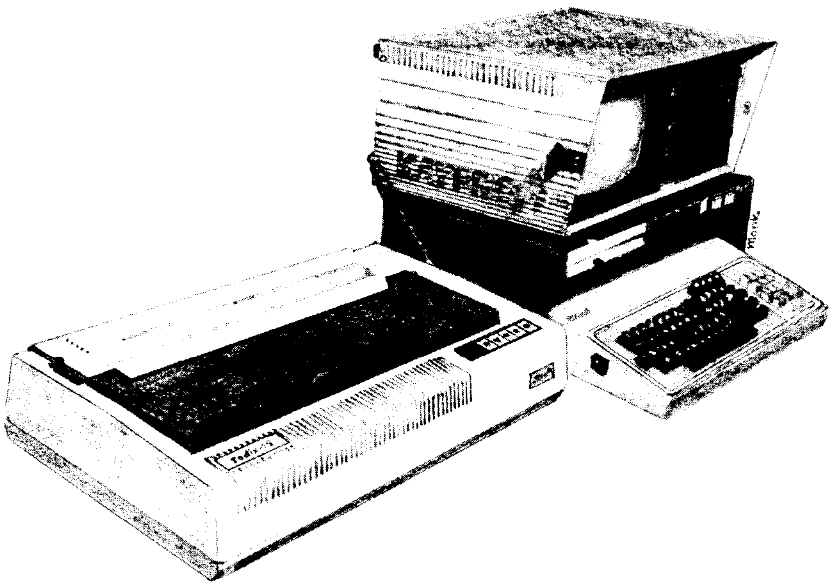
Table 9-3
Vertical tab commands

Function	**Control code**
Advance paper to next tab position	CHR$(11)
Set vertical tabs at *n1*, *n2*, etc.	⟨ESC⟩ "P" CHR$(*n1*) CHR$(*n2*)...CHR$(0)
Advance paper *n* lines	⟨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
⟨ESC⟩ "Q" *n*	Set right margin
CHR$(11)	Vertical tab
⟨ESC⟩ "P" *n1 n2 n3* . . . CHR$(0)	Set vertical tabs
⟨ESC⟩ "a" *n*	One-time vertical tab of *n* lines



Chapter 10

Special Features of the Radix Printer

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 Radix. 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 Radix'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 Radix's "bell" is CHR$(7), which is ASCII code 7 or ⟨BEL⟩. Any time Radix receives this code it will sound the bell for a quarter of a second. This can be used to remind an operator to change the paper or to make another adjustment to the printer. Note to Apple users: Entering a CHR$(7) will sound *Apple's* bell; the code will not be sent to Radix.

You can try this by typing:

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

There are two other codes that affect the bell. One disables the bell, so that Radix 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 10-1
Bell commands

Function	**Control code**
Sound bell	CHR$(7)
Disable bell	⟨ESC⟩ "Y" CHR$(0)
Enable bell	⟨ESC⟩ "Y" CHR$(1)

Initializing Radix

Up to now when we wanted to reset Radix 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 Radix'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 Radix's RAM memory (Chapter 11 tells you how to create your own characters), and it won't erase the macro if you have one stored in Radix's RAM (this chapter will tell you how to create a macro).

Putting Radix to sleep

You know how to put Radix *off-line* with the On Line button. Radix has another *off-line* state that can be controlled from your computer. When you turn Radix *off-line* from your computer, Radix will ignore anything that you send it, except for the code to go *on-line* again. CHR$(19) is the code to turn Radix off-line; CHR$(17) returns Radix 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 Radix 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.

Radix 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 10-2
Some miscellaneous commands

Function	**Control code**
Master reset	⟨ESC⟩ "@"
Off-line	CHR$(19)
On-line	CHR$(17)
Paper-out detector off	⟨ESC⟩ "8"
Paper-out detector on	⟨ESC⟩ "9"
Move print head back one space	CHR$(8)
Delete last character sent	CHR$(127)

Backspace and delete

Backspace (CHR$(8)) "backs up" the printhead so that you can print two characters right on top of each other. Each time Radix 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 Radix's buffer, not from the paper).

The following program shows how these two codes work.

```
1Ø 'Demo backspace and delete codes.
2Ø LPRINT "Backspace does not" ;
3Ø LPRINT CHR$(8) CHR$(8) CHR$(8) ; 'Three backspaces.
4Ø LPRINT "=== work."
5Ø LPRINT "Delete does not" ;
6Ø LPRINT CHR$(127) CHR$(127) CHR$(127) ; 'Three
   deletes.
7Ø LPRINT "work."
```

Here is what this program will print:

```
Backspace does not work.
Delete does work.
```

The backspace codes in line 30 move the printhead a total of three spaces to the left so that the first part of line 40 will overprint the word "not". The delete codes in line 60 "erase" the three letters in the word "not" so that it doesn't even print.

Unidirectional printing

Unidirectional printing is a big word that means *printing in one direction only*. Radix 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. Radix lets you make this choice. The table below shows the commands for controlling how Radix prints.

Table 10-3
Printing direction commands

Function	**Control code**
Print in one direction	⟨ESC⟩ "U" CHR$(1)
Print in both directions	⟨ESC⟩ "U" CHR$(0)

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

```
1Ø 'Demo unidirectional printing.
2Ø LPRINT CHR$(27) "A" CHR$(7) ; 'Line spacing = 7/72".
3Ø FOR I = 1 TO 1Ø
4Ø LPRINT "¦"
5Ø NEXT I
6Ø LPRINT : LPRINT
7Ø LPRINT CHR$(27) "U" CHR$(1) ; 'Turn on unidirectional
   printing.
8Ø FOR I = 1 TO 1Ø
9Ø LPRINT "¦"
1ØØ NEXT I
11Ø LPRINT CHR$(12) CHR$(27) "@" ; 'Form feed, master
   reset.
```

Here is what you will get. The top line is printed bidirec-

tionally, and the bottom is printed unidirectionally. 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 Radix to the power-on condition.

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 Radix'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 10-4
Eighth bit control commands

Function	**Control code**
Turn the eighth bit ON	〈ESC〉 "〉"
Turn the eighth bit OFF	〈ESC〉 "="
Accept the eighth bit "as is" from the computer	〈ESC〉 "#"

Block graphics characters and special symbols

Besides the upper and lower case letters and symbols that we are by now familiar with, Radix 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.

```
1Ø 'Prints all block graphic characters.
2Ø WIDTH "LPT1:",255
3Ø FOR J = 16Ø TO 255 STEP 8
4Ø FOR I = J TO J + 7
```

```
5Ø LPRINT I "= " ;
6Ø LPRINT CHR$(I) ; 'Send graphic char.
7Ø LPRINT CHR$(9) ; 'Tab.
8Ø NEXT I : LPRINT : NEXT J
```

Figure 10-1 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 (unless you have set DIP switch C-3 on by mistake). You can get the correct printout by adding these lines:

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

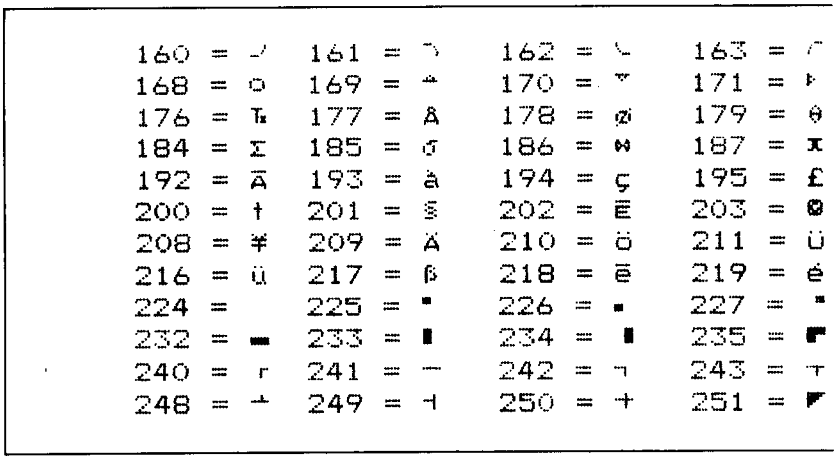
So how are all of these strange characters used? Here is a short program that demonstrates how the graphics characters can be combined to create figures.

```
1Ø 'Draws a figure with block graphic chars.
2Ø LPRINT CHR$(27) "A" CHR$(6) ; 'Set line spacing
   to 6/72".
3Ø LPRINT CHR$(235) CHR$(231) CHR$(231) CHR$(236)
4Ø LPRINT CHR$(233) CHR$(163) CHR$(161) CHR$(234)
5Ø LPRINT CHR$(233) CHR$(162) CHR$(16Ø) CHR$(234)
6Ø LPRINT CHR$(237) CHR$(232) CHR$(232) CHR$(238)
7Ø LPRINT CHR$(27) "2" ; 'Restore 1/6" line spacing.
```

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

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

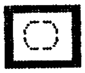
In this program line 20 sets the line spacing to 6 dots which is the height of the graphics characters. Then lines 30-60 print the



```
160 = ⌟  161 = ⌝  162 = ⌞  163 = ⌜
168 = ○  169 = ⁂  170 = ⁂  171 = ⊦
176 = Ŧ  177 = Å  178 = ø  179 = θ
184 = Σ  185 = σ  186 = ∞  187 = π
192 = Ā  193 = à  194 = ç  195 = £
200 = †  201 = §  202 = Ē  203 = ©
208 = ¥  209 = Ä  210 = ö  211 = ü
216 = ü  217 = ß  218 = ē  219 = é
224 =    225 = ▘  226 = ▖  227 = ▝
232 = ▬  233 = ▌  234 = ▐  235 = ▛
240 = ┌  241 = ─  242 = ┐  243 = ┬
248 = ┴  249 = ┤  250 = ┼  251 = ▜
```

Figure 10-1.

figure, and line 70 resets the line spacing to 1/6 inch. Here is what this program prints:



International character sets

Radix is a multi-lingual printer for it can speak in eight languages! Radix changes languages by changing 11 characters that are different for the different languages. These sets of characters

Table 10-5
International character set commands

Country	**Control code**
U.S.A.	⟨ESC⟩ "7" CHR$(0)
England	⟨ESC⟩ "7" CHR$(1)
Germany	⟨ESC⟩ "7" CHR$(2)
Denmark	⟨ESC⟩ "7" CHR$(3)
France	⟨ESC⟩ "7" CHR$(4)
Sweden	⟨ESC⟩ "7" CHR$(5)
Italy	⟨ESC⟩ "7" CHR$(6)
Spain	⟨ESC⟩ "7" CHR$(7)

```
164  =  ↑      165  =  ↓      166  =  ←      167  =  →
172  =  ◂      173  =  ◇      174  =  ◆      175  =  □
180  =  ち      181  =  ド      182  =  Ω      183  =  ඊ
188  =  ±      189  =  ⊃      190  =  ×      191  =  ÷
196  =  ā      197  =  µ      198  =  ▫      199  =  ˀ
204  =  ↳      205  =  ẍ      206  =  ⱪ      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  =
```

are called *international character sets*. The control codes to select the international character sets are given in Table 10-5.

The characters that change are shown beneath their ASCII code in Table 10-6.

Table 10-6
International character sets

Country	35	64	91	92	93	94	96	123	124	125	126
U.S.A.	#	@	[	\	]	^	`	{	¦	}	~
England	£	@	[	\	]	^	`	{	¦	}	~
Germany	#	§	Ä	Ö	Ü	^	`	ä	ö	ü	β
Denmark	#	@	Æ	Ø	Å	^	`	æ	ø	å	~
France	£	à	°	ç	§	^	`	é	ù	è	¨
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. Radix's macro can save you a lot of time and effort also.

Here is how Radix's macro works. You *define* your macro by telling Radix what normal control codes are to be included in the macro. Then you can use the macro any time that you want and Radix 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 below.

Table 10-7
Macro instruction commands

Function	**Control code**
Define macro	⟨ESC⟩ "+" . . . codes you include . . . CHR$(30)
Use macro	⟨ESC⟩ "!"

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.

```
1Ø 'Defines a macro that will reset RADIX to normal.
2Ø LPRINT CHR$(27) "+" ; 'Start macro definition.
3Ø LPRINT CHR$(18) ; 'Select pica pitch.
4Ø LPRINT CHR$(27) "W" CHR$(Ø) ; 'Expanded off.
5Ø LPRINT CHR$(27) "F" ; 'Emphasized off.
6Ø LPRINT CHR$(27) "H" ; 'Double-strike off.
7Ø LPRINT CHR$(27) "-" CHR$(Ø) ; 'Underline off.
8Ø LPRINT CHR$(27) "T" ; 'Super & subscripts off.
9Ø LPRINT CHR$(27) "5" ; 'Select roman character set.
1ØØ 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. Radix 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 fifteen.

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.

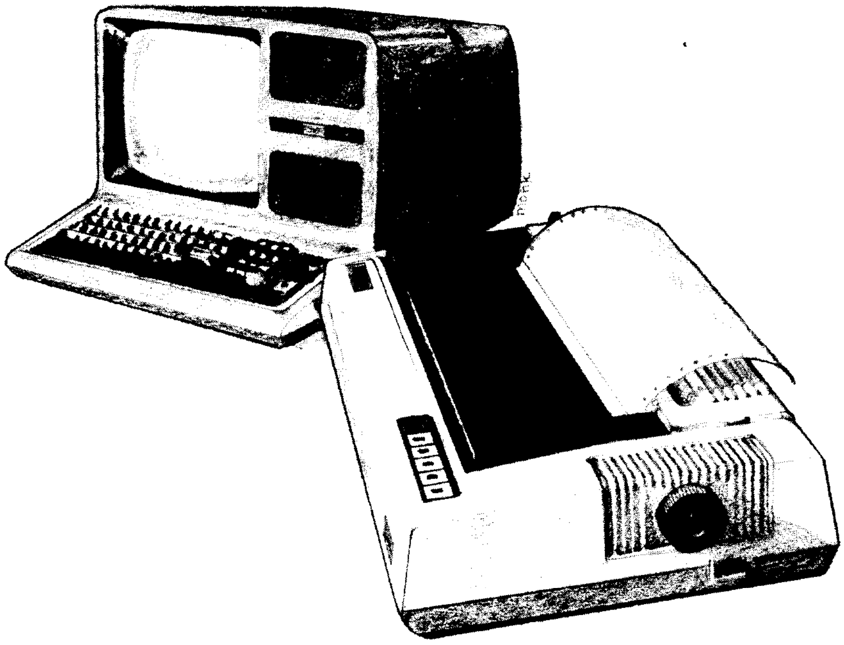
```
10 'Uses macro to reset RADIX to normal.
20 LPRINT CHR$(27) "4" ; 'Italic.
30 LPRINT CHR$(27) "G" ; 'Double-strike.
40 LPRINT CHR$(27) "W" CHR$(1) ; 'Expanded.
50 LPRINT "This line is special."
60 LPRINT CHR$(27) "!" ; 'Use the macro.
70 LPRINT "This line is normal printing."
```

```
This line is special.
This line is normal printing.
```

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 Radix offers. They give you the ability to create your own characters.

Summary

Control code	**Function**
CHR$(7)	Bell
⟨ESC⟩ "Y" CHR$(0)	Disable bell
⟨ESC⟩ "Y" CHR$(1)	Enable bell
⟨ESC⟩ "@"	Reset
CHR$(19)	Off-line
CHR$(17)	On-line
⟨ESC⟩ "8"	Paper-out detector off
⟨ESC⟩ "9"	Paper-out detector on
⟨ESC⟩ "U" CHR$(1)	Unidirectional printing
⟨ESC⟩ "U" CHR$(0)	Bidirectional printing
CHR$(8)	Backspace
CHR$(127)	Delete
⟨ESC⟩ "⟩"	Eighth bit on
⟨ESC⟩ "="	Eighth bit off
⟨ESC⟩ "#"	Eighth bit as-is
⟨ESC⟩ "7" n	Select international character set
⟨ESC⟩ "+" . . . CHR$(30)	Define macro
⟨ESC⟩ "!"	Use macro



Chapter 11

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 Radix 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 10, 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 a Radix 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 Radix and you will see the dots. Figure 11-1 shows how the letter "C" is formed by printing 15 dots.

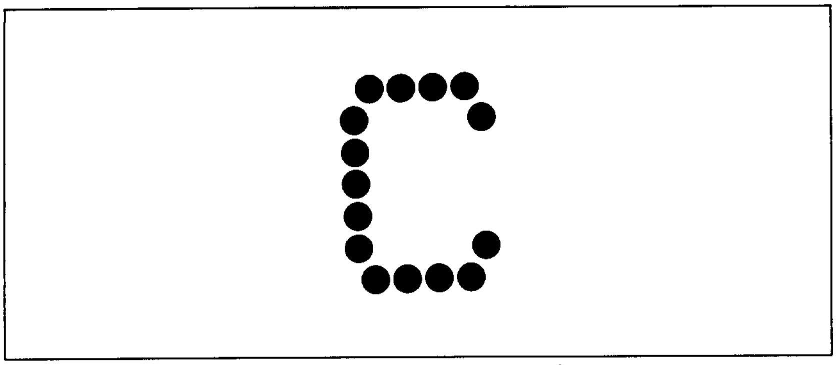


Figure 11-1. *The letter "C" is created by printing 15 dots.*

The printhead in Radix consists of nine thin wires stacked one atop the other. Figure 11-2 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 descenders (such as the lower case "g" 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 Radix an *impact* printer).

The Print Matrix

All of the standard characters that Radix 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 Radix reserved for

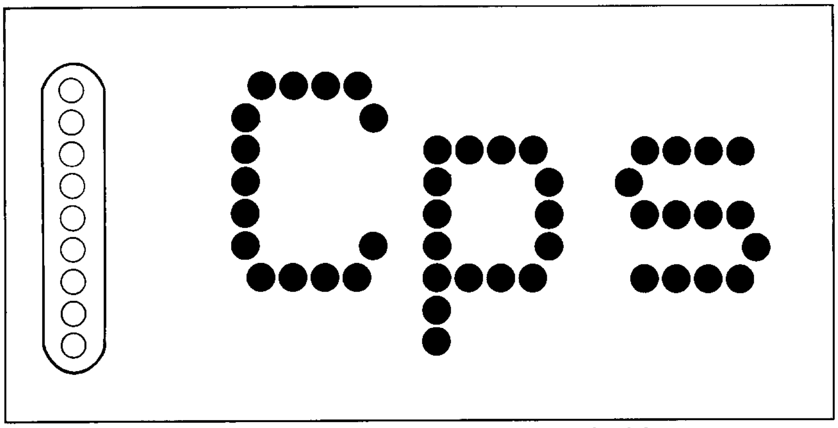


Figure 11-2. *As the printhead moves across the page, each of the wires prints one row of dots.*

user-defined characters. These are characters that you design and *download* into Radix. 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 11-3. 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 J.

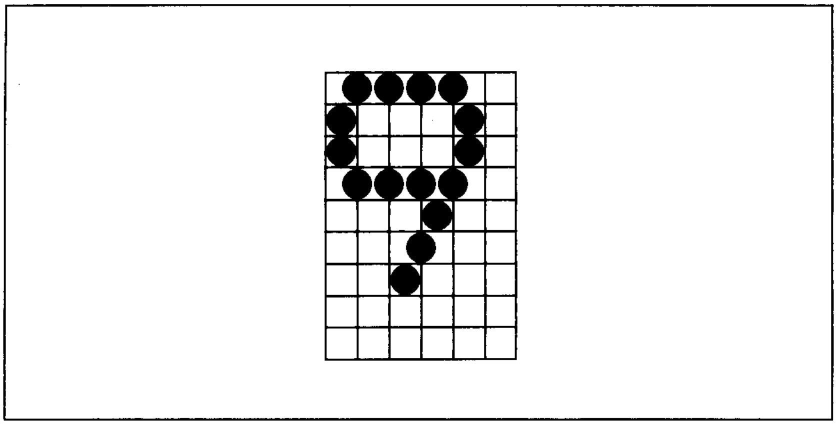


Figure 11-3. *Dots can be inside boxes or straddle the vertical lines of the grid.*

Defining Your Own Characters

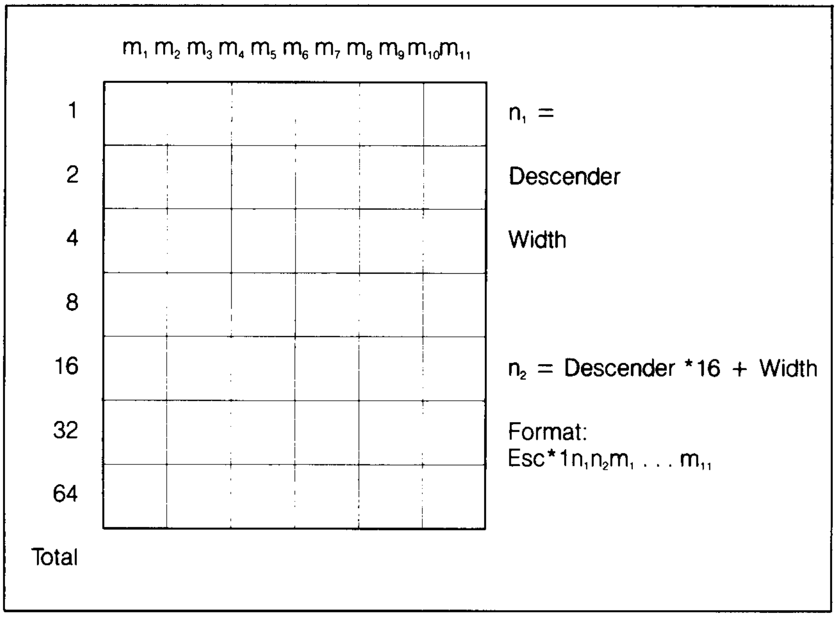
You've seen how the engineers at Star designed their characters by using a grid to lay out the dots. Now you can define characters exactly the same way. Make up some grids (photocopy Figure 11-4 if you wish) and get ready to be creative! (Just in case you are not feeling creative, and to make our explanations a little clearer, we'll be using a "bullet" as an example of a download character. You can see how we've laid it out in Figure 11-5. You'll find this useful for highlighting a list of items, as we have done at the beginning of each chapter in this manual.

You'll notice that Figure 11-4 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 11-3: it's only seven boxes high. Which leads us to. . .

Rule 1: Download characters are seven dots high

As you noticed in Figure 11-2, capital letters, most lowercase letters, and most special characters use only the top seven pins of the printhead. This is also the standard for download characters, so our grid is only seven dots high.

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



m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8 m_9 m_{10} m_{11}
1
2
4
8
16
32
64
Total
n_1 =
Descender
Width
n_2 = Descender *16 + Width
Format:
Esc*1$n_1n_2m_1$ . . . m_{11}

Figure 11-4. *Use this grid (or one similar to it) to define your own characters.*

below the baseline of the rest of the characters).

One bit in the download character definition command is used to tell Radix 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 seven dots, write in a zero next to the word "Descender" on the layout grid; if it uses the bottom seven dots, write in a one. In our example, we'll want the bottom of the bullet to line up with the baseline of the other characters, so it will *not* be a descender. As shown in Figure 11-5, we've written in a "0" on our grid.

Rule 2: Dots cannot overlap

As you can see in Figure 11-5 our bullet will print fairly solid. But, you may ask, why not make it *really* solid and print all the intermediate dots, as shown in Figure 11-6? Because the dots that straddle the vertical lines in the grid actually overlap those inside the boxes. If we tried to print overlapping dots, Radix's print head would have to slow down and back up to print both dots—not very efficient! To avoid this inefficiency, Radix will not allow you to define a character like Figure 11-6. (Actually, you can define it, but

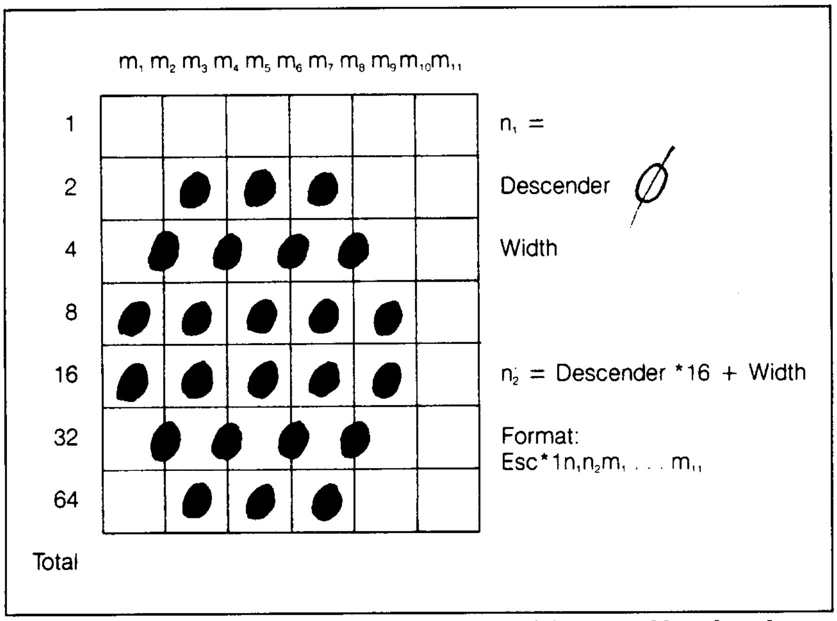


Figure 11-5. *We've designed a character and decided that it would not be a descender, hence the "0" written in.*

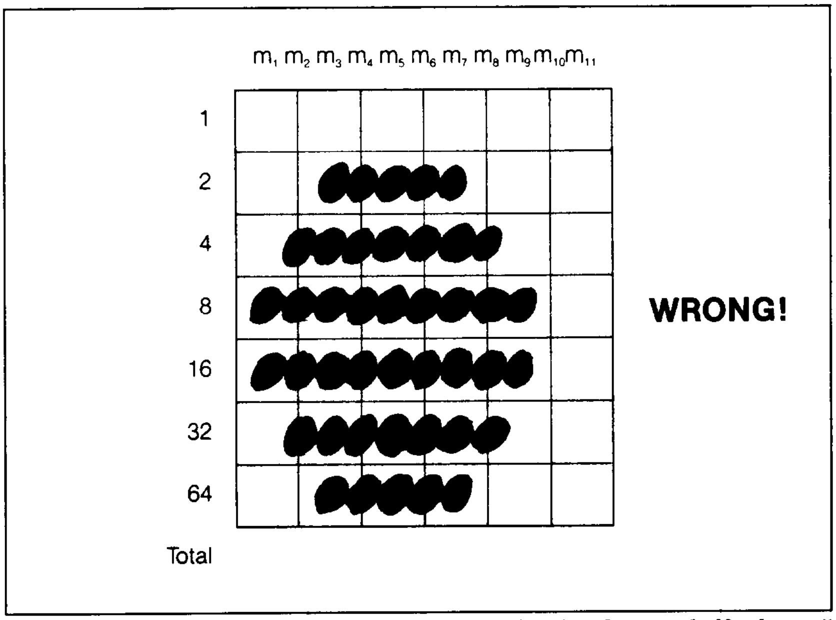


Figure 11-6. *Dots cannot overlap; those in immediately adjacent "half columns" will be ignored when the character is printed.*

when it prints, Radix will leave out the overlapping dots, so that it would print like Figure 11-5.)

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 previous number. By making these numbers powers of two we can take any combination of dots in a vertical column and assign them a unique value. Some examples will make this clearer. As shown in Figure 11-7, 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 127. Each number from 0-127 represents a unique combination of dots.

So add up the values of the dots in each column using this system. This way it takes one number to describe each column of dots. In Figure 11-8 we've shown our grid with the sums of the columns filled in across the bottom (see if these agree with your

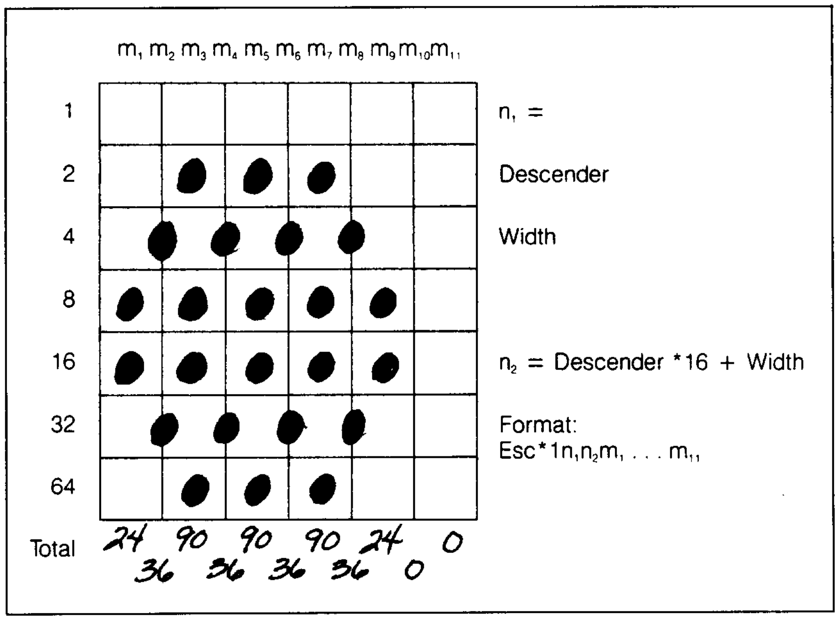
1		● – 1	● – 1
2	● – 2	● – 2	● – 2
4		● – 4	● – 4
8	● – 8		● – 8
16			● – 16
32	● – 32		● – 32
64		● – 64	● – 64
Sum	42	71	127

Figure 11-7. *By adding the values of each dot in a column, you'll get a unique description for any combination of dots.*

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.

Assigning a value to your character

We've done a pretty thorough job of designing and describing a user-defined character. But the Radix has room for 189



m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8 m_9 m_{10} m_{11}
1
2
4
8
16
32
64
Total
24 90 90 90 24 0
36 36 36 36 0
n_1 =
Descender
Width
n_2 = Descender *16 + Width
Format:
Esc*1$n_1$$n_2$$m_1$ . . . m_{11}

Figure 11-8. *Add the values of the dots in each column and write the sum of each column at the bottom.*

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 there are two banks of characters that can be defined: values from 33 to 126 and 160 to 254. 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 assigned values in the range of 33 to 126 or 160 to 254, 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 bullet a value of 43, which is the ASCII value for the "+" character. This way, when we want to print a bullet, all we have to do is send the printer a +.

To make our demonstration of download characters more complete, we've designed two more characters. To avoid confusion between the letter "O" and zero, we have created a slashed zero to replace Radix's zero (ASCII 48). And, since some people prefer the "lb" abbreviation for pound, we've replaced Radix's "#" symbol (ASCII 35) with a "lb." 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 Radix repertoire and now you've got the necessary knowledge to implement it. Here it is:

```
<ESC> "*" CHR$(1) n1n2m1m2m3m4m5m6m7m8m9m10m11
```

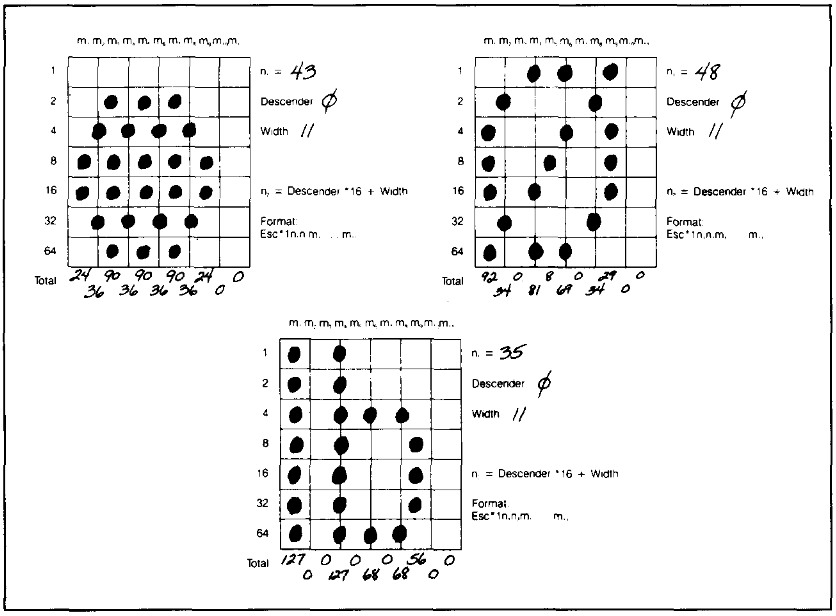


Figure 11-9. *Character designs for our three characters.*

Like the other Radix commands, it starts with an ⟨ESC⟩ (CHR$(27)). The next character is an asterisk (*), which is CHR$(42), followed by a CHR$(1).

n1 is the value we assign to the character—in the case of the bullet it is CHR$(43).

n2 is called the *attribute byte*, for it describes two attributes of the character we have designed: descender data and proportional width information. A byte consists of eight bits. In the attribute byte, the first three (high order) bits are unused, the fourth bit is used for the descender data, and the last four 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 seven pins, this bit should be 0; to use the bottom seven pins this bit should be 1. Figure 11-10 shows the bits of the attribute byte as we'll use them for our bullet character. Since the descender data is 0, the value of the byte is equal to the value of the proportional data—11. 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 16 (the value of the fourth bit) if you *want* descenders, or 0 if you *don't 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 0 + 11 = 11.)

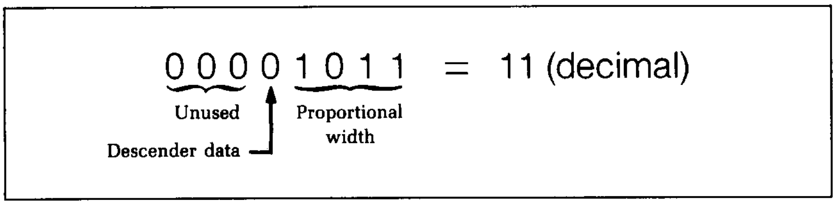


Figure 11-10. *The attribute byte (n2) for our bullet character.*

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 bullet character is shown in Figure 11-11.

Now let's send the information to the printer. The following program will send the character definitions for all three characters to the printer. Enter the program and run it.

CHR$(27)	CHR$(42)	CHR$(1)	CHR$(43)	CHR$(11)	
Escape	*	1	n_1	n_2	
CHR$(24)	CHR$(36)	CHR$(90)	CHR$(36)	CHR$(90)	CHR$(36)
M_1	M_2	M_3	M_4	M_5	M_6
CHR$(90)	CHR$(36)	CHR$(24)	CHR$(0)	CHR$(0)	
M_7	M_8	M_9	M_{10}	M_{11}	

Figure 11-11. *This is the complete command to send our bullet character to the Radix printer.*

```
1Ø 'Downloads symbols.
2Ø OPEN "LPT1:" AS #1 : WIDTH #1,255
3Ø FOR I = 1 TO 3 'Do three character downloads.
4Ø PRINT #1,CHR$(27) "*" CHR$(1) ; 'Begin char download.
5Ø READ N1$,N2
6Ø PRINT #1,N1$ CHR$(N2) ; 'Send char code, and
   attribute.
7Ø FOR M = 1 TO 11 'Send 11 bytes of download per char.
8Ø READ D
9Ø PRINT #1,CHR$(D) ;
1ØØ NEXT M
11Ø NEXT I
12Ø CLOSE #1
13Ø LPRINT
14Ø DATA "+",11,24,36,9Ø,36,9Ø,36,9Ø,36,24,Ø,Ø
15Ø DATA "Ø",11,92,34,Ø,81,8,69,Ø,34,29,Ø,Ø
16Ø DATA "#",11,127,Ø,Ø,127,Ø,68,Ø,68,56,Ø,Ø
```

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 the Radix.

But how do you know that? If you try printing those characters now (type LPRINT " + 0#") you don't get a bullet, slashed zero and "lb." Instead you get . . . + 0#. That's because the download characters are stored in a different part of Radix's memory. To tell it to look in download character RAM instead of standard character ROM it requires another command:

```
<ESC> "$" CHR$(n)
```

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$(27) "$" CHR$(1) "+0#"
```

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

```
●Ølb
```

(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 'Print all RAM characters.
2Ø LPRINT CHR$(27) "$" CHR$(1) ; 'Select download
   characters.
3Ø FOR I = 33 TO 126 : LPRINT CHR$(I) ; : NEXT I
4Ø FOR I = 16Ø TO 254 : LPRINT CHR$(I) ; : NEXT I
5Ø LPRINT
6Ø LPRINT CHR$(27) "$" CHR$(Ø) ; 'Select ROM characters.
```

Nope! Just three characters in the download set. This is inconvenient for a couple of reasons. First, every time you wanted to use a download character you would have to switch back and forth between character sets. Knowing that you wouldn't want to do that, Radix won't even allow it. Standard characters and download characters cannot be mixed in a line. If you want to use download characters, the command should appear at the beginning of the line. All subsequent characters (even on following lines) are printed with the download set until you return to the standard characters with an ⟨ESC⟩ "$" CHR$(0). (Note that the ⟨ESC⟩ "$" CHR$(1) command can be in the middle of a line, and that *entire line* will be printed with the download characters. Likewise, if you select the standard character set anywhere in a line, the *entire line* will be printed with the standard characters. Conflicting commands within a line can cause unpredictable results.)

So does that mean that in order to print something meaningful with our special symbols we have to define an entire alphabet? Fear not. The engineers at Star have made it an easy task to use mostly standard characters with just a few special characters thrown in. This command copies all the characters from the standard character ROM into download RAM:

```
⟨ESC⟩ "*" CHR$(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. With that in mind, add this line to the program we used to send the characters to Radix:

```
25 PRINT #1, CHR$(27) "*" CHR$(Ø) ; 'Copy ROM to RAM.
```

Now try the download printout test program again. Your results should look like Figure 11-12. You probably noticed that our printout test includes the characters with ASCII values from 160 to 254, but nothing prints. The ⟨ESC⟩ "*" CHR$(0) command copies only the standard ASCII characters (those in the range of 33 to 126) to download RAM; it does not copy any block graphics characters.

To demonstrate how to use these characters, let's use this character set with a word processing program to print a grocery ad. Just as you learned in Chapter 3, send the printer control codes to select download characters (27 36 1) followed by this text:

```
 !"lb$%&'()*●,-./Ø123456789:;<=>?@ABC
DEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefgh
ijklmnopqrstuvwxyz{|}~
```

Figure 11-12. *Printout of the download character set, into which all the standard characters have been copied, and the #, +, 0 have been changed.*

```
Today's Specials
+ Oranges 10 # / $1.ØØ
+ Ocean Perch $1.9Ø/#
```

Your output should look like this:

```
Today's Specials
● Oranges 1Ø lb / $1.ØØ
● Ocean Perch $1.9Ø/lb
```

Just a sampling of Radix's download capabilities! As you can see, it's no problem to define characters in BASIC (or another language) and use them with a word processor or other application.

Note that we didn't have to re-enter the download characters, since they were already sent to the printer with the previous 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.

Table 11-1
Download character definition commands

Function	**Control code**
Define download character	⟨ESC⟩ "*" CHR$(1) *n1 n2 m1 . . . m11*
Copy ROM to download RAM	⟨ESC⟩ "*" CHR$(0)

Proportional Characters

Up until now, all the characters that your Radix has printed have been of a fixed width—either 10, 12, or 17 (or 5, 6 or 8.5 in expanded mode) characters per inch. Whichever pitch you select, all the characters are the same width. You'll notice though, that in typeset books, such as this one, each character has a slightly different width. For instance, the "i" is quite narrow, and the "W" is very wide. This is more pleasing to the eye and easier to read.

So, if you're going to go to the trouble of designing your own download characters for Radix, you might as well make them pleasing to the eye! Proportional download characters allow you to do just that. As you'll remember from our initial discussion of download character definition, part of the attribute byte is for proportional width data. We skipped over that, with the promise of describing it later. Well now is the time!

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 4 to 11 dots wide. This means that characters can be as narrow as one-third the normal width. The examples in Figure 11-13 show characters of different widths. These characters are defined in the program that follows.

```
10 'Downloads proportional characters into RAM.
20 OPEN "LPT1:" AS #1 : WIDTH #1,255
30 FOR C = 1 TO 4
40 READ C$,CODE
50 PRINT #1,CHR$(27) "*" CHR$(1) C$ CHR$(CODE) ;
60 FOR I = 1 TO 11
70 READ BITS
80 PRINT #1,CHR$(BITS) ;
90 NEXT I
100 NEXT C
110 CLOSE #1
120 'Print a sample.
130 LPRINT "                  Mississippi"
140 LPRINT
150 LPRINT "ROM char set, normal spacing."
160 LPRINT
```

```
17Ø LPRINT
18Ø 'Select RAM set, normal spacing.
19Ø LPRINT CHR$(27) "$" CHR$(1) ;
2ØØ LPRINT "                    Mississippi"
21Ø 'Cancel RAM set, normal spacing.
22Ø LPRINT CHR$(27) "$" CHR$(Ø)
23Ø LPRINT "RAM char set, normal spacing."
24Ø LPRINT
25Ø LPRINT
26Ø 'Select RAM set, proportional spacing.
27Ø LPRINT CHR$(27) "X" CHR$(1) ;
28Ø LPRINT "                    Mississippi"
29Ø 'Cancel RAM set, proportional spacing.
3ØØ LPRINT CHR$(27) "X" CHR$(Ø)
31Ø LPRINT "RAM char set, proportional spacing."
32Ø END
33Ø DATA "M",11,1,126,1,2,4,8,4,2,1,126,1
34Ø DATA "i",4,64,61,64,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
35Ø DATA "p",23,127,Ø,17,Ø,17,14,Ø,Ø,Ø,Ø,Ø
36Ø DATA "s",6,8,84,Ø,84,32,Ø,Ø,Ø,Ø,Ø,Ø
```

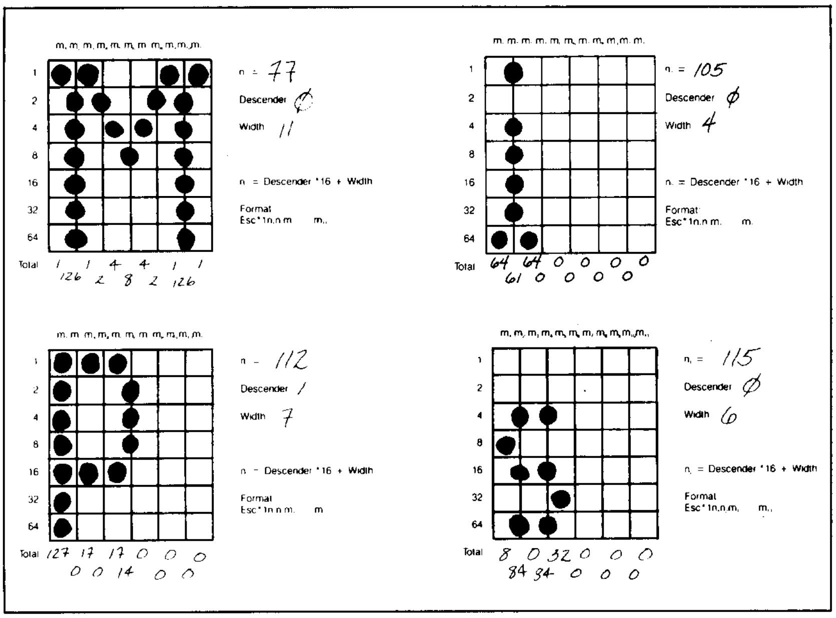

n = 77
Descender Ø
Width 11
n = Descender *16 + Width
Format Esc*1n,n m ... m
Total 1 1 4 4 1 1 / 126 2 8 2 126
n = 105
Descender Ø
Width 4
Total 64 64 0 0 0 0 / 61 0 0 0 0
n = 112
Descender 1
Width 7
Total 127 17 17 0 0 0 / 0 0 14 0 0
n = 115
Descender Ø
Width 6
Total 8 0 32 0 0 0 / 84 84 0 0 0

Figure 11-13. *These download characters are defined as proportional characters.*

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, 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 define a character; Radix expects eleven characters following the ⟨ESC⟩ "*" CHR$(1) *n1 n2* sequence.

In most cases, the width you select should actually be at least 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).

Printing proportional characters

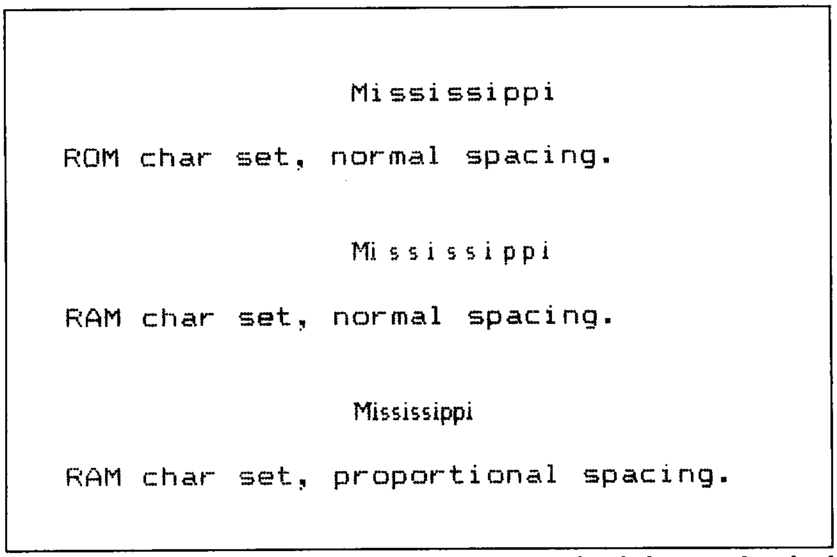
Printing with proportional download characters is much like using normal width download characters: one command is used to select the download set or the standard character set. Here's the command:

```
⟨ESC⟩ "X" CHR$(n)
```

If *n* is 1, then the download character set is selected, and proportional widths are used. If *n* is 0, the standard character set is selected.

It should be noted that it is possible to use the same character definitions for either normal width or proportional download characters (if a valid proportional width is included in the attribute byte). The only difference is the way they are accessed: ⟨ESC⟩ "$" CHR$(1) for normal width or ⟨ESC⟩ "X" CHR$(1) for proportional width. The two commands work independently of each other, so that ⟨ESC⟩ "$" CHR$(0) will *not* turn off proportional download characters, and ⟨ESC⟩ "X" CHR$(0) will *not* turn off normal width download characters. If you have selected both normal and proportional download characters, proportional will print until you send the printer an ⟨ESC⟩ "X" CHR$(0). The printer will then continue to print with normal width download characters (rather than returning to the standard character set) until you send an ⟨ESC⟩ "$" CHR$(0).

This can lead to confusion if you have accidentally specified both types of download characters.



```
Mississippi

ROM char set, normal spacing.

Mi s s i s s i p p i

RAM char set, normal spacing.

Mississippi

RAM char set, proportional spacing.
```

Figure 11-14. *This printout shows the same text, printed with the same download characters, in both normal and proportional widths.*

Table 11-2
Download character printing commands

Function	**Control code**
Normal download characters ON	〈ESC〉 "$" CHR$(1)
Normal download characters OFF	〈ESC〉 "$" CHR$(0)
Proportional download characters ON	〈ESC〉 "X" CHR$(1)
Proportional download characters OFF	〈ESC〉 "X" CHR$(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 handwriting. 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 11-15 shows how one digit is defined, and Figure 11-16 shows the final output of our program.

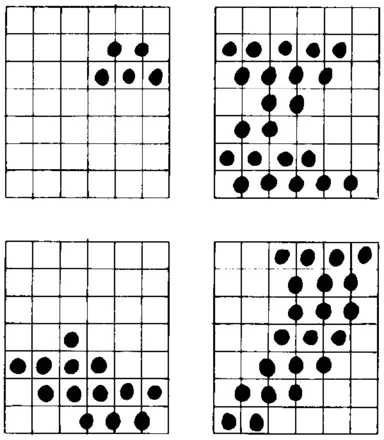


Figure 11-15. *Each digit is made up of four individual characters.*

```
10 'Program to define and print BIG numerals.
20 'Each numeral is made up of four characters,
30 'two wide, and two high.
40 'A blank is also defined.
50 '
60 'Download the 41 special characters.
70 OPEN "LPT1:" AS #1 : WIDTH #1,255
80 FOR N1 = 160 TO 200 'N1 is the char code.
90 PRINT #1,CHR$(27) "*" CHR$(1) ;
100 PRINT #1,CHR$(N1);
110 READ N2
120 PRINT #1,CHR$(N2);
```

```
130 FOR S = 1 TO 11
140 READ MS
150 PRINT #1,CHR$(MS);
160 NEXT S
170 NEXT N1
180 CLOSE #1
190 BLANK$ = CHR$(200)
200 '
210 'Print the BIG numerals.
220 LPRINT
230 LPRINT CHR$(27) "X" CHR$(1) ; 'Select RAM chars.
240 LPRINT CHR$(27) "1" ; '7/72" line spacing.
250 'Print the top half of the numerals.
260 FOR NUM = 0 TO 9
270 LPRINT CHR$(NUM*4+160) CHR$(NUM*4+161) BLANK$ ;
280 NEXT NUM
290 LPRINT
300 'Print the bottom half of the numerals.
310 FOR NUM = 0 TO 9
320 LPRINT CHR$(NUM*4+162) CHR$(NUM*4+163) BLANK$ ;
330 NEXT NUM
340 LPRINT CHR$(27) "X" CHR$(0) ; 'Deselect RAM.
350 LPRINT CHR$(27) "2" '1/6" line spacing (normal).
360 'ZERO
370 DATA 11,0,96,16,104,16,44,30,14,0,2,1
380 DATA 11,2,1,2,1,6,8,38,88,32,88,32
390 DATA 11,3,12,19,12,51,0,96,0,96,0,96
400 DATA 11,0,32,0,48,0,28,3,12,3,4,3
410 'ONE
420 DATA 11,0,0,0,0,0,4,0,4,0,4,126
430 DATA 9,12,114,12,114,12,2,0,0,0,0,0
440 DATA 11,64,0,64,0,64,0,64,32,80,47,80
450 DATA 9,47,80,47,64,0,64,0,64,0,0,0
460 ' TWO
470 DATA 11,0,0,0,0,0,12,16,14,0,6,0
480 DATA 11,3,0,3,0,70,56,70,56,4,24,0
490 DATA 11,64,0,64,32,64,32,80,32,80,40,64
500 DATA 11,44,64,38,65,34,65,32,80,32,88,0
510 ' THREE
520 DATA 11,0,0,0,0,0,0,4,2,4,2,4
530 DATA 11,34,84,34,92,34,76,34,68,2,64,0
540 DATA 11,16,0,48,0,56,64,48,64,32,64,32
550 DATA 11,64,32,64,48,9,54,9,22,9,6,1
560 ' FOUR
```

```
570 DATA 11,Ø,Ø,Ø,Ø,Ø,Ø,64,36,88,32,16
580 DATA 11,Ø,Ø,64,32,64,56,64,6Ø,2,12,Ø
590 DATA 11,Ø,8,4,1Ø,5,1Ø,5,8,4,72,4
600 DATA 11,88,38,89,38,89,6,73,4,8,6,Ø
610 ' FIVE
620 DATA 11,Ø,Ø,Ø,Ø,64,32,84,5Ø,76,34,68
630 DATA 1Ø,34,68,34,68,34,68,2,68,2,Ø,Ø
640 DATA 1Ø,Ø,32,24,1Ø1,24,97,Ø,64,Ø,64,Ø
650 DATA 11,64,Ø,96,1,48,15,48,15,16,15,Ø
660 ' SIX
670 DATA 11,Ø,96,Ø,112,Ø,12Ø,Ø,92,Ø,1Ø2,Ø
680 DATA 11,98,Ø,98,Ø,98,Ø,7Ø,Ø,14,Ø,6
690 DATA 11,7,8,23,8,55,8,99,Ø,65,Ø,64
700 DATA 11,Ø,96,Ø,112,1,62,1,3Ø,1,14,Ø
710 ' SEVEN
720 DATA 11,Ø,16,8,6,8,6,8,6,8,6,8
730 DATA 9,7Ø,8,1Ø2,8,54,8,6,Ø,2,Ø,Ø
740 DATA 11,Ø,64,Ø,96,Ø,12Ø,Ø,124,Ø,3Ø,1
750 DATA 9,6,1,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
760 ' EIGHT
770 DATA 11,Ø,Ø,Ø,Ø,24,36,24,1Ø2,24,1Ø2,Ø
780 DATA 11,67,Ø,67,Ø,99,28,34,28,34,28,Ø
790 DATA 11,12,18,44,19,1Ø8,19,96,1,64,Ø,64
800 DATA 11,Ø,96,1,112,15,48,15,16,14,Ø,Ø
810 ' NINE
820 DATA 11,Ø,Ø,12Ø,4,12Ø,6,12Ø,6,Ø,3,Ø
830 DATA 11,3,Ø,3,Ø,67,4,123,4,122,4,12Ø
840 DATA 11,48,Ø,56,Ø,113,Ø,99,Ø,99,Ø,99
850 DATA 11,Ø,115,Ø,57,Ø,31,Ø,15,Ø,7,Ø
860 ' SPACE
870 DATA 11,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
```

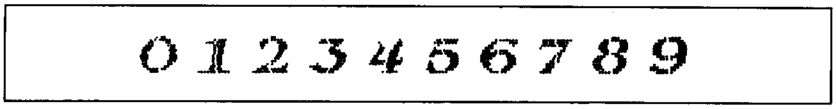


Figure 11-16. *The output for characters like this must be carefully planned.*

Mixing Print Modes with Download Characters

It's possible to get even more printing effects by combining download characters with the various print modes available with Radix. Most of the commands that you learned in Chapter 7 work with normal width download characters as well as standard characters. A few of them will work with proportional download characters as well. Table 11-3 summarizes the various print modes and their compatibility with download characters.

Table 11-3
Mixing download characters with various print modes

	Normal width (Escape $)	**Proportional (Escape X)**
Standard Characters	Yes	Yes
Italic	-	-
NLQ Characters	-	-
Pica	Yes	Yes
Elite	Yes	-
Condensed	Yes	-
Expanded	Yes	-
Double-strike	Yes	-
Emphasized	Yes	-
Underline	Yes	Yes
Super/subscript	Yes	-

A Utility Program

If you've followed along this far you've probably become pretty proficient at designing download characters. And even the addition is getting easier! But this is a good computer application—Computer Aided Design (CAD) for download characters. The program below allows you to design and edit characters on the screen. You can make changes (no erasing!) until it's the way you like it, and then the program makes the necessary calculations and sends the character to Radix.

As you can see, at 205 lines this is quite a long program! However, if you want to use the full capabilities of Radix's download characters, you'll really appreciate it.

Instructions for using DLEDIT

The program screen is shown in Figure 11-17. Above the main grid (where you actually place the dots) there are two informational lines.

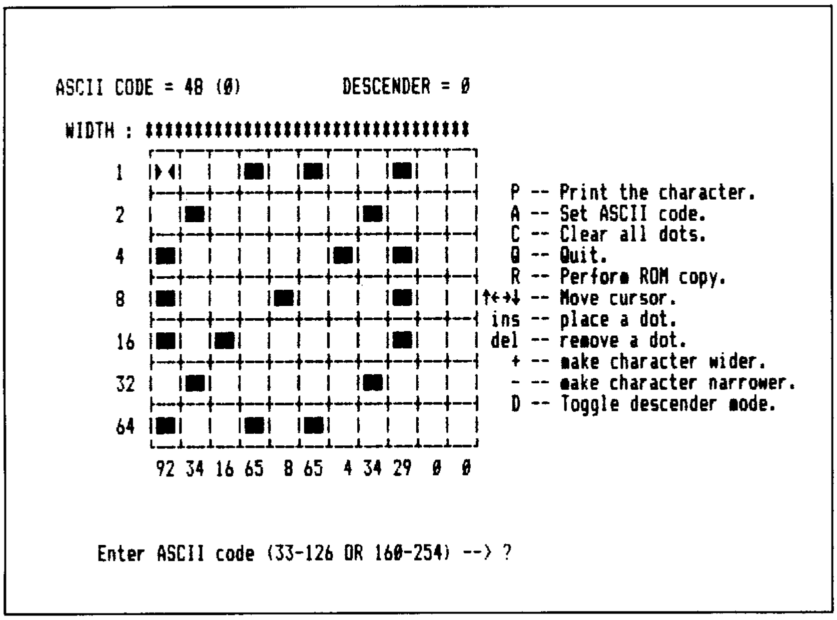
The first line tells the ASCII code of the character being edited (and in parentheses, the normal character for that code). The next field in the first line tells whether the character being edited is a descender or not (a "1" indicates that it is; "0" means that it is not).

The second status line shows the proportional width of the character being defined. The asterisks extend over the columns of dots to indicate the actual width when the character is printed using the ⟨ESC⟩ "X" command.

Below the layout grid is the prompt line. This will appear only when you need to enter information, such as the ASCII code of the character you wish to define.

To the right of the layout grid is the command menu. All of the valid commands are defined here; if you press any other key, the computer will beep and no action will be taken. Below, each command is defined in greater detail.

P - Print the character. This command takes the character that is currently on the screen and prints it in condensed, elite, pica, expanded pica, and proportional widths so you can see how it looks. In addition, it prints the complete character set in both normal and proportional widths. At the end of the printout is the data statement necessary to download this character through a BASIC program.



```
ASCII CODE = 48 (0)          DESCENDER = 0

 WIDTH : ************************************
         ┌──┬──┬──┬──┬──┬──┬──┬──┬──┬──┬──┐
      1  │▶◀│  │  │■■│  │■■│  │  │■■│  │  │
         ├──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┤   P -- Print the character.
      2  │  │■■│  │  │  │  │  │■■│  │  │  │   A -- Set ASCII code.
         ├──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┤   C -- Clear all dots.
      4  │■■│  │  │  │  │  │■■│  │■■│  │  │   Q -- Quit.
         ├──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┤   R -- Perform ROM copy.
      8  │■■│  │  │  │■■│  │  │  │■■│  │  │↑←→↓ -- Move cursor.
         ├──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┤ ins -- place a dot.
     16  │■■│  │■■│  │  │  │  │  │■■│  │  │ del -- remove a dot.
         ├──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┤   + -- make character wider.
     32  │  │■■│  │  │  │  │  │■■│  │  │  │   - -- make character narrower.
         ├──┼──┼──┼──┼──┼──┼──┼──┼──┼──┼──┤   D -- Toggle descender mode.
     64  │■■│  │  │■■│  │■■│  │  │  │  │  │
         └──┴──┴──┴──┴──┴──┴──┴──┴──┴──┴──┘
          92 34 16 65  8 65  4 34 29  0  0


    Enter ASCII code (33-126 OR 160-254) --> ?
```

Figure 11-17. *DLEDIT screen display shows ASCII code and character layout.*

A - Set ASCII code. To change the ASCII code (which is shown in the first status line), press "A." You will then be prompted for the code you want to use.
C - Clear all dots. Press "C" to get a clean screen.
Q - Quit. "Q" closes all files and ends the program.
R - Perform ROM copy. The ROM character set will be copied to download RAM immediately.
↑ ← → ↓ - Move cursor. The arrow keys are used to move the cursor around the grid.
Ins - Insert. The insert key places a dot at the current cursor location.
Del - Delete. The delete key deletes a dot from the current cursor location.
+ - Wider. Use the "+" key to increase the proportional width, which is indicated by the row of asterisks above the grid. The maximum width is 11 columns.
– - Narrower. Use the "–" key to decrease the proportional width. The minimum width is four columns.
D - Descender. This command toggles the descender flag, which is shown in the first status line. If it is equal to zero, the top seven pins of the printhead are used; if it is equal to 1, the bottom seven pins are used to create a descender character.

Enjoy the program!

```
10 'Program to allow editing down-load characters.
20 'for the RADIX printer.
30 '
40 'Initialization.
50 DIM Z(8,12),MM(11)
60 AS=33
70 CS$=CHR$(16)+CHR$(17):SC$=STRING$(2,219)
80 RAMNML$ = CHR$(27) + "$" + CHR$(1)
90 RAMNMLOFF$ = CHR$(27) + "$" + CHR$(0)
100 RAMPRO$ = CHR$(27) + "X" + CHR$(1)
110 RAMPROOFF$ = CHR$(27) + "X" + CHR$(0)
120 OPEN "LPT1:" AS #2 : WIDTH #2,255
130 LPRINT CHR$(27) "@" ; : WIDTH "LPT1:",255
140 GOSUB 1930
150 '
160 'Main loop.
170 A$=INKEY$:IF A$="" THEN 170
180 B$ = LEFT$(A$,1)
190 IF B$ = CHR$(0) THEN 290
```

```
200 IF A$ = "+" THEN GOSUB 1060 : GOTO 370 'Wider.
210 IF A$ = "-" THEN GOSUB 1090 : GOTO 370 'Narrower.
220 IF A$ = "D" OR A$ = "d" THEN GOSUB 1120 : GOTO 370
230 IF A$="Q" OR A$="q" THEN GOSUB 380 : END
240 IF A$="P" OR A$="p" THEN GOSUB 1360 : GOTO 370
250 IF A$="C" OR A$="c" THEN GOSUB 1930 : GOTO 370
260 IF A$="A" OR A$="a" THEN GOSUB 1720 : GOTO 370
270 IF A$="R" OR A$="r" THEN GOSUB 1980 : GOTO 370
280 BEEP:GOTO 370
290 B$=RIGHT$(A$,1)
300 IF B$=CHR$(75) THEN GOSUB 910:GOTO 370 'Left.
310 IF B$=CHR$(77) THEN GOSUB 930:GOTO 370 'Right.
320 IF B$=CHR$(80) THEN GOSUB 950:GOTO 370 'Down.
330 IF B$=CHR$(72) THEN GOSUB 970:GOTO 370 'Up.
340 IF B$=CHR$(82) THEN GOSUB 990:GOTO 370 'Insert.
350 IF B$=CHR$(83) THEN GOSUB 1030:GOTO 370 'Delete.
360 BEEP
370 GOTO 170
380 COLOR 7,0 : CLS
390 CLOSE #1,#2
400 RETURN
410 '
420 ' Subroutine to paint screen.
430 CLS
440 GOSUB 1820
450 '
460 'Draw grid.
470 P1 = 1 : M$ = CHR$(179) + STRING$(2,32)
480 N$ = STRING$(2,196) + CHR$(197)
490 L$ = STRING$(2,196) + CHR$(193)
500 LOCATE 4,10:PRINT CHR$(218);CHR$(196);
510 FOR I=1 TO 10
520 PRINT CHR$(196) CHR$(194) CHR$(196) ; : NEXT I
530 PRINT CHR$(196) CHR$(191) : LOCATE 5,10
540 FOR K=1 TO 12 : PRINT M$; : NEXT K : PRINT
550 FOR J=1 TO 6:LOCATE 5+P1,10:P1=P1+1:PRINT CHR$(195);
560 FOR K=1 TO 10:PRINT N$;:NEXT K
570 PRINT CHR$(196) CHR$(196) CHR$(180)
580 LOCATE 5+P1,10 : P1=P1+1
590 FOR K=1 TO 12:PRINT M$;:NEXT K
600 PRINT:NEXT J:LOCATE 18,10:PRINT CHR$(192);
610 FOR I=1 TO 10:PRINT L$;:NEXT I
620 PRINT CHR$(196);CHR$(196);CHR$(217)
```

```
63Ø FOR I=Ø TO 6:LOCATE  5+I*2,6:PRINT 2^I;:NEXT I
64Ø '
65Ø 'Put in dots.
66Ø FOR H = 1 TO 11 : FOR J = 1 TO 7 : Z(J,H) = Ø
7ØØ NEXT J : NEXT H
71Ø FOR H = 1 TO 11 : GOSUB 12ØØ : NEXT H
72Ø X=1:Y=1:G=1:H=1
73Ø GOSUB 13ØØ
74Ø '
75Ø 'Paint menu.
76Ø LOCATE 6,47 : PRINT "P -- Print the character."
77Ø LOCATE 7,47 : PRINT "A -- Set ASCII code."
78Ø LOCATE 8,47 : PRINT "C -- Clear all dots."
79Ø LOCATE 9,47 : PRINT "Q -- Quit."
8ØØ LOCATE 1Ø,47 : PRINT "R -- Perform ROM copy."
81Ø LOCATE 11,44 : PRINT CHR$(24) CHR$(27) CHR$(26)
   CHR$(25) ;
82Ø PRINT " -- Move cursor."
83Ø LOCATE 12,45:PRINT "ins -- place a dot.";
84Ø LOCATE 13,45:PRINT "del -- remove a dot.";
85Ø LOCATE 14,47 : PRINT "+ -- make character wider." ;
86Ø LOCATE 15,47 : PRINT "- -- make character narrower."
   ;
87Ø LOCATE 16,47 : PRINT "D -- Toggle descender mode." ;
88Ø RETURN
89Ø '
9ØØ 'Edit subroutines.
91Ø GOSUB 124Ø:Y=Y-3:H=H-1:IF Y<1 THEN BEEP:Y=1:H=1
92Ø GOSUB 13ØØ:RETURN
93Ø GOSUB 124Ø:Y=Y+3:H=H+1:IF Y>31 THEN BEEP:Y=31:H=11
94Ø GOSUB 13ØØ:RETURN
95Ø GOSUB 124Ø:X=X+2:G=G+1:IF X>13 THEN BEEP:X=13:G=7
96Ø GOSUB 13ØØ:RETURN
97Ø GOSUB 124Ø:X=X-2:G=G-1:IF X<1 THEN BEEP:X=1:G=1
98Ø GOSUB 13ØØ:RETURN
99Ø IF Z(G,H-1)=1 OR Z(G,H+1)=1 THEN BEEP:RETURN
1ØØØ Z(G,H) = 1 : COLOR 31,1
1Ø1Ø LOCATE X+4,Y+1Ø : PRINT SC$ ; : COLOR 7,Ø
1Ø2Ø GOSUB 115Ø : RETURN
1Ø3Ø Z(G,H)=Ø : COLOR 7,Ø
1Ø4Ø LOCATE X+4,Y+1Ø : PRINT CS$ ; : COLOR 7,Ø
```

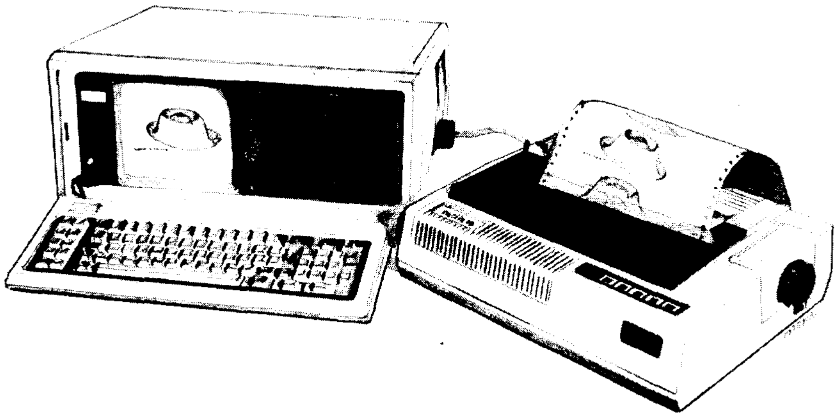
```
1050 GOSUB 1150 : RETURN
1060 IF PROWID = 11 THEN BEEP : RETURN
1070 PROWID = PROWID + 1
1080 GOSUB 1820 : RETURN
1090 IF PROWID = 4 THEN BEEP : RETURN
1100 PROWID = PROWID - 1
1110 GOSUB 1820 : RETURN
1120 IF DESC = 1 THEN DESC = 0 : GOTO 1140
1130 DESC = 1
1140 GOSUB 1820 : RETURN
1150 '
1160 'Subroutine to calculate a column value & print it.
1170 MM(H) = 0 : FOR J=1 TO 7
1180 MM(H)=MM(H)+Z(J,H)*2^(J-1)
1190 NEXT J : GOSUB 1200 : RETURN
1200 '
1210 'Subroutine to print a column value.
1220 LOCATE 19,7+H*3 : PRINT RIGHT$("  "+STR$(MM(H)),3)
   ;
1230 RETURN
1240 '
1250 'Subroutine to remove the cursor.
1260 LOCATE X+4,Y+10
1270 IF Z(G,H) = 0 THEN PRINT "  " ;
1280 IF Z(G,H) = 1 THEN COLOR 7,0 : PRINT SC$ ;
1290 RETURN
1300 '
1310 'Subroutine to place the cursor.
1320 LOCATE X+4,Y+10
1330 IF Z(G,H)=1 THEN COLOR 31,1 : PRINT SC$ ; : COLOR
   7,0
1340 IF Z(G,H)=0 THEN COLOR 7,0 : PRINT CS$ ;
1350 RETURN
1360 '
1370 'Subroutine to print current character.
1380 GOSUB 2050
1390 LPRINT "ASCII code =" AS : LPRINT
1400 PRINT #2,REC$ ; 'Download the character.
1410 LPRINT CHR$(27) "B" CHR$(3) "Condensed"
1420 LPRINT RAMNML$ STRING$(21,AS)
1430 LPRINT RAMNMLOFF$
1440 LPRINT CHR$(27) "B" CHR$(2) "Elite"
1450 LPRINT RAMNML$ STRING$(15,AS)
1460 LPRINT RAMNMLOFF$
```

```
1470 LPRINT CHR$(27) "B" CHR$(1) "Pica"
1480 LPRINT RAMNML$ STRING$(12,AS)
1490 LPRINT RAMNMLOFF$
1500 LPRINT CHR$(27) "W" CHR$(1) "Expanded"
1510 LPRINT RAMNML$ STRING$(6,AS)
1520 LPRINT RAMNMLOFF$ CHR$(27) "W" CHR$(0)
1530 LPRINT "Character set (normal width)"
1540 LPRINT RAMNML$;
1550 FOR I=33 TO 126 : LPRINT CHR$(I); : NEXT : LPRINT
1560 FOR I=160 TO 254 : LPRINT CHR$(I); : NEXT : LPRINT
1570 LPRINT RAMNMLOFF$
1580 LPRINT "Proportional"
1590 LPRINT RAMPRO$ STRING$(15,AS)
1600 LPRINT RAMPROOFF$
1610 LPRINT "Character set (proportional)"
1620 LPRINT RAMPRO$;
1630 FOR I=33 TO 126 : LPRINT CHR$(I); : NEXT : LPRINT
1640 FOR I=160 TO 254 : LPRINT CHR$(I); : NEXT : LPRINT
1650 LPRINT RAMPROOFF$
1660 LPRINT : LPRINT : LPRINT
1670 LPRINT "Use this data statement to download this
   character."
1680 GOSUB 2050 : LPRINT "DATA 27" ;
1690 FOR I = 2 TO LEN(REC$)
1700 LPRINT "," STR$(ASC(MID$(REC$,I,1))) ;
1710 NEXT I : LPRINT : LPRINT : LPRINT : LPRINT : RETURN
1720 '
1730 'Subroutine to input desired character code.
1740 LOCATE 23,5
1750 INPUT "Enter ASCII code (33-126 OR 160-254) --> " ;
   AS
1760 GOSUB 2010
1770 IF AS < 33 OR AS > 254 THEN BEEP : GOTO 1740
1780 IF AS < 160 AND AS > 126 THEN BEEP : GOTO 1740
1810 GOSUB 1820 : RETURN
1820 '
1830 'Subroutine to display header.
1840 LOCATE 1,1 : PRINT "ASCII CODE =" AS ;
1850 PRINT "(" CHR$(AS AND &H7F) ;
1860 IF AS > 127 THEN PRINT " + 128" ;
1870 PRINT ")        " ;
1880 LOCATE 1,30 : PRINT "DESCENDER =" DESC ;
```

```
1900 LOCATE 3,10 : PRINT STRING$(33, " ") ;
1910 LOCATE 3,2 : PRINT "WIDTH : " STRING$(PROWID*3,
   "*") ;
1920 RETURN
1930 '
1940 'Subroutine to clear current character.
1950 PROWID = 11 : DESC = 0
1960 FOR H = 1 TO 11 : MM(H) = 0 : NEXT H
1970 GOSUB 410 : RETURN
1980 '
1990 'Subroutine to perform a ROM copy.
2000 LPRINT CHR$(27) "*" CHR$(0) ; : RETURN
2010 '
2020 'Subroutine to erase query message.
2030 LOCATE 23,5 :PRINT STRING$(70," ") ;
2040 RETURN
2050 '
2060 'Subroutine to build command string.
2070 REC$ = CHR$(27) + "*" + CHR$(1)
2080 REC$ = REC$ + CHR$(AS) + CHR$(DESC*16 + PROWID)
2090 FOR I = 1 TO 11 : REC$ = REC$ + CHR$(MM(I)) : NEXT
   I
2100 RETURN
```

Summary

Control code	Function
⟨ESC⟩ "*" CHR$(1) *n1 n2 m1 . . . m11*	Defines download character into RAM
⟨ESC⟩ "*" CHR$(0)	Copies fonts in ROM into download RAM
⟨ESC⟩ "X" CHR$(1)	Selects the download character set and uses proportional spacing
⟨ESC⟩ "X" CHR$(0)	Cancels proportional download character set
⟨ESC⟩ "$" CHR$(1)	Selects the download character set and uses normal spacing
⟨ESC⟩ "$" CHR$(0)	Cancels normal download character set



Chapter 12

Printing With Dot Graphics

Subjects covered in this chapter include:

- **Radix's bit image graphics capabilities**
- **Printing a pre-defined shape**
- **Plotting a calculated shape**
- **High resolution graphics**

In Chapter 11 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 Radix 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 Radix printer can be used as a graphics plotter. This can have some practical business applications as well as create some terrific computer art!

Comparing Dot Graphics with Download Characters

A good understanding of dot graphics requires an understanding of how dot matrix printers work; you may want to review the first few pages of Chapter 11. 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 a Radix-15!

There is no "descender data" with dot graphics; graphics images are always printed with the top seven or eight pins of the print head, depending on whether you have a 7-bit or 8-bit interface (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 7 dots
- If an image is to be printed just one time, as opposed to a frequently used "text" character
- If you want higher resolution (Radix 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 Radix'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 Radix 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 Radix 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 Radix-10 or 816 dots on Radix-15. So to figure out how many columns of graphics data to expect, Radix multiplies *n2* by 256 and adds the value of *n1* to the product. If you divide the number of columns by 256, then *n2* is the quotient and *n1* is the remainder (why not let your computer figure it out for you: if the number of columns is assigned to variable X, then N1 = X MOD 256 and N2 = INT(X/256)). Table 12-1 might make things even easier.

Table 12-1
Calculating n1 and n2

If the number of columns, x, ranges from:	**then *n1* is:**	**and *n2* is:**
1 to 255	x	0
256 to 511	x - 256	1
512 to 767	x - 512	2
768 to 1023	x - 768	3
1024 to 1279	x - 1024	4
1280 to 1535	x - 1280	5
1536 to 1791	x - 1536	6
1792 to 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 Radix 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 Radix's print head. In Figure 12-1, 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 140.

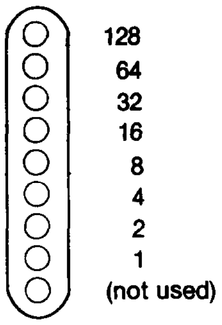


Figure 12-1. *Starting with the most significant bit at the top, each pin of the print head is assigned a value which is a power of two. Note that for 7-bit computers, the top pin has a value of 64, and the bottom two pins are unused.*

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

```
1Ø 'Demo bit 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
```

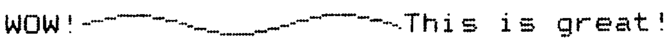
In line 50 we've selected normal density graphics and said that 100 characters of graphics data would follow. The loop between lines 60 and 80 is repeated to plot 100 points along a curve. This is an example of plotting a very simple mathematical function (a sine wave) to create a design. Later in this 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 Radix'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!" ;
```

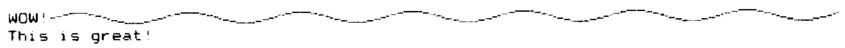
Now if you run the program you should get a printout that looks like this:



```
WOW!                    This is great!
```

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

```
30 WID = 1000
```



```
WOW!
This is great!
```

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

As you can see, Radix 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 Radix 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 Radix and line 200 sends one line of graphics data.

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

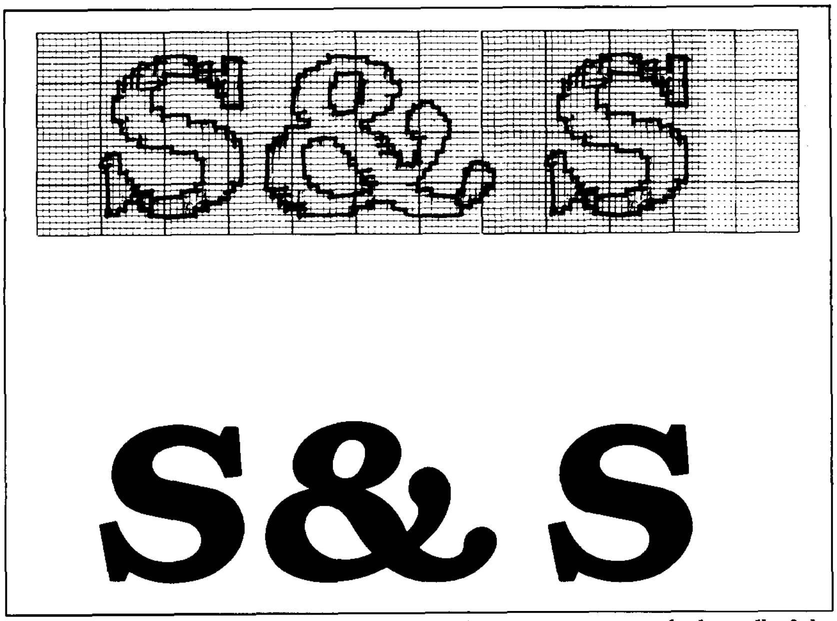


Figure 12-2. *By laying out the logo on graph paper, you can calculate all of the graphics data.*

```
10 'Prints S&S logo.
20 LINE.8$ = CHR$(27)+CHR$(65)+CHR$(8)
30 'Set line spacing to 1/6"
40 LINE.12$ = CHR$(27)+CHR$(50)
50 'Select dot graphics
60 GRAPHIC$ = CHR$(27)+CHR$(75)
70 DIM LOGO$(4)
80 WIDTH "LPT1:",255
90 ' READ DATA
100 FOR ROW = 1 TO 4
110 FOR COLUMN = 1 TO 100
120 READ P
130 LOGO$(ROW) = LOGO$(ROW) + CHR$(P)
140 NEXT COLUMN
150 NEXT ROW
160 ' PRINT LOGO
170 LPRINT LINE.8$;
180 FOR ROW = 1 TO 4
190 LPRINT GRAPHIC$;CHR$(100);CHR$(0);
200 LPRINT LOGO$(ROW)
210 NEXT ROW
```

```
220 LPRINT LINE.12$
230 'ROW 1
240 DATA 0,0,0,0,1,3,7,7,7,15
250 DATA 14,14,14,14,14,7,7,3,3,15
260 DATA 15,15,0,0,0,0,0,0,0,0
270 DATA 0,1,3,3,7,7,15,14,14,14
280 DATA 14,15,7,7,7,3,0,0,0,0
290 DATA 0,0,0,0,0,0,0,0,0,0
300 DATA 0,0,0,0,0,0,0,0,0,0
310 DATA 0,0,0,0,1,3,7,7,7,15
320 DATA 14,14,14,14,14,7,7,3,3,15
330 DATA 15,15,0,0,0,0,0,0,0,0
340 ' ROW 2
350 DATA 0,0,60,255,255,255,255,255,143,15
360 DATA 7,7,7,7,3,3,3,131,193,241
370 DATA 240,240,0,0,0,0,0,0,0,1
380 DATA 121,253,253,255,255,255,143,7,7,7
390 DATA 31,253,252,248,248,240,192,0,7,15
400 DATA 31,31,15,7,3,0,0,0,0,0
410 DATA 0,0,0,0,0,0,0,0,0,0
420 DATA 0,0,60,255,255,255,255,255,143,15
430 DATA 7,7,7,7,3,3,3,131,193,241
440 DATA 240,240,0,0,0,0,0,0,0,0
450 'ROW 3
460 DATA 0,31,31,3,129,128,192,192,192,192
470 DATA 192,224,224,224,224,240,255,255,255,255
480 DATA 255,127,0,0,0,0,63,127,255,255
490 DATA 255,255,193,128,128,128,128,192,224,240
500 DATA 252,255,255,255,127,63,31,7,7,31
510 DATA 254,252,248,224,128,0,0,3,7,7
520 DATA 7,3,0,0,0,0,0,0,0,0
530 DATA 0,31,31,3,129,128,192,192,192,192
540 DATA 192,224,224,224,224,240,255,255,255,255
550 DATA 255,127,0,0,0,0,0,0,0,0
560 'ROW 4
570 DATA 0,248,248,240,224,224,112,112,56,56
580 DATA 56,56,56,120,120,240,240,224,224,192
590 DATA 128,0,0,0,0,0,192,224,240,240
600 DATA 240,248,248,248,120,120,56,56,56,56
610 DATA 48,112,224,224,224,224,240,240,248,248
620 DATA 120,120,56,56,56,56,120,240,224,224
630 DATA 192,128,0,0,0,0,0,0,0,0
640 DATA 0,248,248,240,224,224,112,112,56,56
```

```
65Ø DATA 56,56,56,12Ø,12Ø,24Ø,24Ø,224,224,192
66Ø DATA 128,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø
```

Plotting with Radix

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 Radix 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.

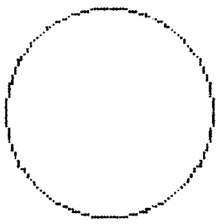
```
1Ø ' General purpose RADIX plotting program.
2Ø '
3Ø 'Set program constants.
4Ø MAXCOL% = 75       : MAXROW% = 14
```

```
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
```

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.

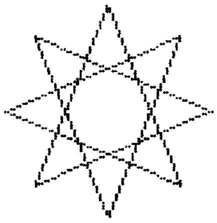
What the program section starting at line 600 actually does is to calculate starting and ending points for a line (in our circle the "lines" are very short—sometimes the starting and ending points are the same). The coordinates of the starting point of the line are assigned to variables X1 and Y1. The 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.

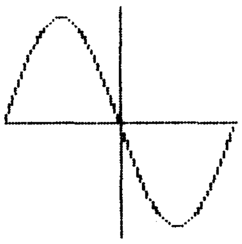
When all the points have been plotted in memory, printing begins at line 150. We first set the line spacing to 6/72 inch using the ⟨ESC⟩ "A" command. This is so that there are no gaps between rows of dots. Then the loop from line 170 to line 240 prints the dot graphics image one line (which is six dots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



```
6Ø '
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
```



Using Radix for business graphics

You don't have to be a mathematician, scientist, or computer hacker/artist to use Radix's graphics capabilities. It can be used for business graphics too—line graphs, bar charts, pie charts, and more! There are many commercially available graphics programs that support Radix's graphics. And, of course, you can write your own. To get you started, we've written a program that prints a pie chart. Here it is:

```
1Ø 'Program to print a piechart on the RADIX.
```

```
20 '
30 'Initialize program constants.
40 ESC$ = CHR$(27)      : LF$ =CHR$(10)
50 FF$ = CHR$(12)       : VTAB$ = CHR$(11)
60 REVFF$ = ESC$ + FF$
70 'Emphasized & expanded modes.
80 TITLE.ON$ = ESC$ + "E" + ESC$ + "W" + CHR$(1)
90 TITLE.OFF$ = ESC$ + "F" + ESC$ + "W" + CHR$(0)
100 OPEN "LPT1:" AS #1 : WIDTH #1,255
110 DIM BIT%(190,36),A$(36),PCT%(25)
120 DIM TEXT$(48),PIECETEXT$(25)
130 MASK%(1) = 64      : MASK%(4) = 8
140 MASK%(2) = 32      : MASK%(5) = 4
150 MASK%(3) = 16      : MASK%(6) = 2
160 LX = 20            : LY = 20
170 LXFAC = 190/LX     : LYFAC = 216/LY
180 FOR I= 0 TO 48
190 TEXT$(I) = SPACE$(79)
200 NEXT I
210 GOSUB 1040
220 '
230 ' Plot curve
240 RAD = 9
250 X1 = 19            : Y1 = 10
270 FOR ANG% = 0 TO 360 STEP 12
280 RANG = ANG%*6.28/360
290 X2 = RAD*COS(RANG)+10  : Y2 = RAD*SIN(RANG)+10
300 GOSUB 640
310 NEXT ANG%
320 FOR PIECE% = 1 TO NUMBER.PIECES%
330 X1 = 10            : Y1 = 10
340 TOTAL.PCT%=TOTAL.PCT%+PCT%(PIECE%)
350 ANG%=360*TOTAL.PCT%*.01
360 RANG = ANG%*6.28/360
370 X2 = RAD*COS(RANG)+10  : Y2 = RAD*SIN(RANG)+10
380 GOSUB 640
390 GOSUB 870
400 NEXT PIECE%
410 '
420 'Send chart title to printer.
440 LPRINT ESC$ "A" CHR$(6) REVFF$ VTAB$ ;
450 LPRINT TITLE.ON$ SPACE$(20-LEN(TITLE$)/2) ;
```

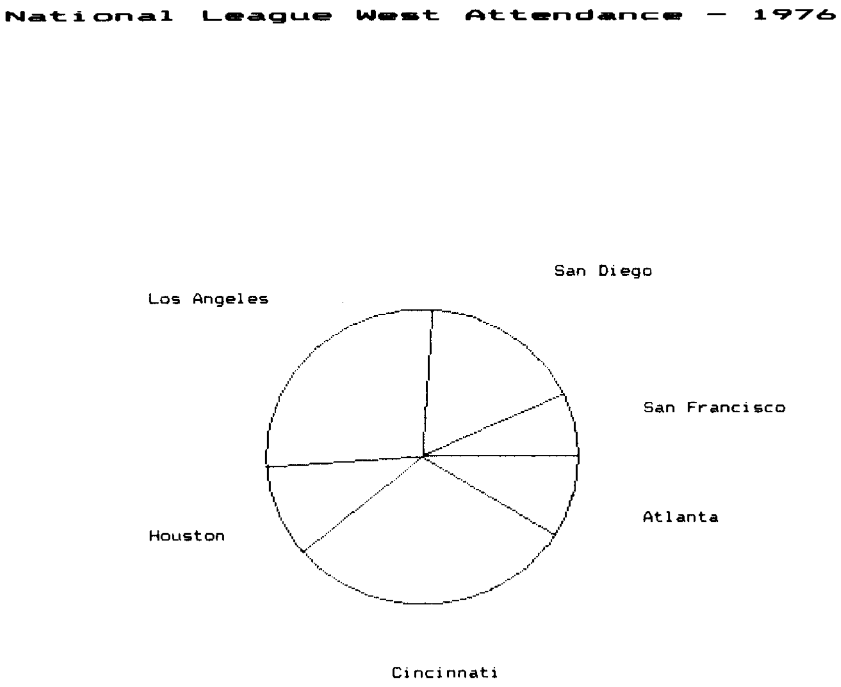
```
460 LPRINT TITLE$ TITLE.OFF$
470 LPRINT VTAB$ VTAB$ ;
480 FOR I = 0 TO 48
490 LPRINT TEXT$(I) : NEXT I
500 '
510 'Send bit image map to printer.
520 LPRINT REVFF$ VTAB$ VTAB$ VTAB$ ;
530 LPRINT LF$ LF$ LF$ LF$ LF$ LF$
540 FOR ROW% = 0 TO 35
550 LPRINT "                        " ;
560 LPRINT ESC$ "K" CHR$(190) CHR$(0) ;
570 FOR COL% = 1 TO 190
580 PRINT#1, CHR$(BIT%(COL%,ROW%)) ; : NEXT
590 PRINT#1, LF$
600 PRINT CHR$(176) CHR$(176);
610 NEXT ROW%
620 LPRINT ESC$ "2" FF$
630 END
640 '
650 'Subroutine to draw a line from X1,Y1 to X2,Y2.
660 '
670 XL = X2 - X1         : YL = Y2 - Y1
680 NX = ABS(XL*LXFAC)  : NY = ABS(YL*LYFAC)
690 IF NX < NY THEN NX = NY
700 NS% = INT(NX+1)
710 DX = XL/NS%          : DY = YL/NS%
720 FOR I% = 1 TO NS%
730 X1 = X1 + DX         : Y1 = Y1 + DY
740 GOSUB 780
750 NEXT I%
760 PRINT CHR$(29) CHR$(205) CHR$(205) CHR$(175);
770 RETURN
780 '
790 'Subroutine to plot a point at X1,Y1.
800 '
810 XX = X1 * LXFAC     : YY = Y1 * LYFAC
820 COL% = INT(XX) + 1
830 ROW% = INT(YY/6)
840 XIT% = INT(YY - ROW% * 6)+1
850 BIT%(COL%,ROW%) = BIT%(COL%,ROW%) OR MASK%(XIT%)
860 RETURN
870 '
880 'Subroutine to arrange field descriptions.
890 '
```

```
900 MIDANG%=(ANG%+PREVANG%)/2
910 RANG = MIDANG%*6.28/360
920 X3 = INT(24*SIN(RANG)+.5) : Y3 = INT(20*COS(RANG))
930 X4 = 24 + X3              : Y4 = 42 + Y3
940 IF (MIDANG% > 70 AND MIDANG% < 110) THEN 990
950 IF (MIDANG% > 250 AND MIDANG% < 290) THEN 990
960 IF MIDANG%>270 OR MIDANG%<90 THEN 1010
970 MID$(TEXT$(X4),Y4-LEN(PIECETEXT$(PIECE%)))
   =PIECETEXT$(PIECE%)
980 GOTO 1020
990 MID$(TEXT$(X4),Y4-LEN(PIECETEXT$(PIECE%))/2)
   =PIECETEXT$(PIECE%)
1000 GOTO 1020
1010 MID$(TEXT$(X4),Y4) = PIECETEXT$(PIECE%)
1020 PREVANG%=ANG%
1030 RETURN
1040 '
1050 'Subroutine to query user for data.
1060 '
1070 CLS: PRINT : PRINT : PRINT :
1080 INPUT "ENTER TITLE FOR CHART: ",TITLE$
1090 IF LEN(TITLE$) <= 40 THEN 1110
1100 PRINT "TITLE TOO LONG - 40 CHAR. MAX" : GOTO 1080
1110 AMT.SOFAR%=0       : AMT.LEFT%=100
1120 FOR I=1 TO 24
1130 CLS
1140 PRINT "                    ENTER PARAMETERS FOR
   PIECHART"
1150 PRINT "                         TOTAL SO FAR :   ";
1160 PRINT USING "###";AMT.SOFAR%
1170 PRINT "                         TOTAL REMAINING: ";
1180 PRINT USING "###";AMT.LEFT%
1190 PRINT :PRINT :PRINT :PRINT
1200 INPUT "ENTER PERCENTAGE FOR FIELD:   ",PCT%(I)
1210 IF PCT%(I)>AMT.LEFT% OR PCT%(I)=0 THEN
   PCT%(I)=AMT.LEFT%
1220 AMT.LEFT%=AMT.LEFT%-PCT%(I)
1230 AMT.SOFAR%=AMT.SOFAR%+PCT%(I)
1240 PRINT :PRINT
1250 INPUT "ENTER DESCRIPTION OF FIELD:
   ",PIECETEXT$(I)
1260 IF LEN(PIECETEXT$(I))<16 THEN 1280
1270 PRINT "FIELD TOO LONG - 15 CHAR. MAX": GOTO 1250
1280 IF AMT.LEFT%=0 THEN 1300
```

```
129Ø NEXT I
13ØØ NUMBER.PIECES%=I
131Ø IF NUMBER.PIECES%=1 THEN 111Ø
132Ø CLS
133Ø RETURN
```

You should recognize many sections of code from the plotting program. We've expanded on that program framework to include routines for inputting data to be graphed and placing labels next to the pie chart. We've used a feature of Radix to simplify programming and speed up the program: a reverse form feed. The program calculates locations and prints all of the labels. When the labels are done, a reverse form feed to the top of the sheet prepares Radix for the graphics data.

The output from our program is shown below.



High Resolution Graphics

Up until now all of the dot graphics printing we have done has been with Radix'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. Radix has four graphics modes you can use; they're summarized in Table 12-2.

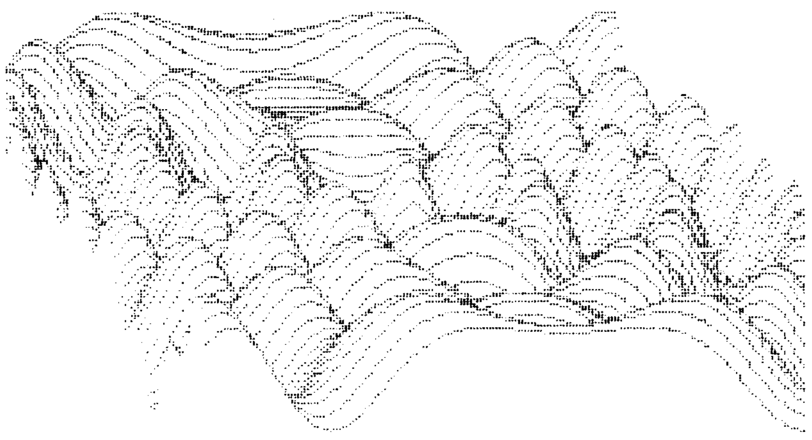
Table 12-2
Dot graphics commands

Function	**Control code**
Normal density (60 dots/inch)	⟨ESC⟩ "K" *n1 n2 m1 m2* . . .
Double density (120 dots/inch)	⟨ESC⟩ "L" *n1 n2 m1 m2* . . .
Double density/double speed	⟨ESC⟩ "y" *n1 n2 m1 m2* . . .
Quadruple density (240 dots/inch)	⟨ESC⟩ "z" *n1 n2 m1 m2* . . .

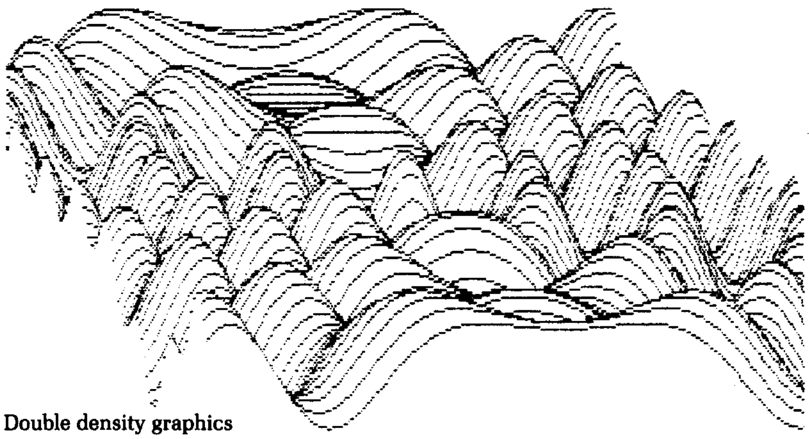
Note: If your computer does not support lowercase characters, use CHR$(121) and CHR$(122) for "y" and "z", respectively.

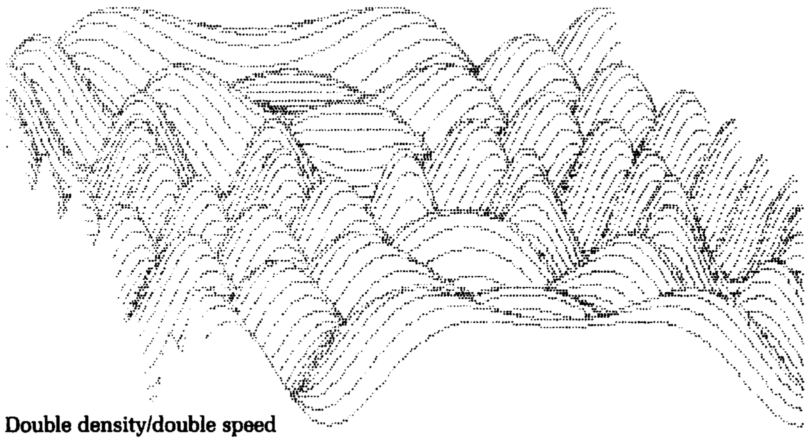
The command syntax for all of the commands is the same—just as you have learned it for the ⟨ESC⟩ "K" (normal density) command. The number of columns to be printed is *n1* + 256∗*n2*.

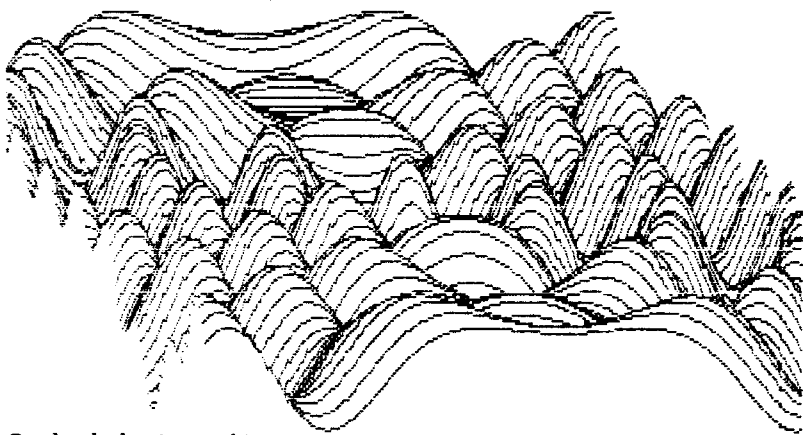
So what do these different modes do? On the following pages are actual size reproductions of printouts of the same image printed in each of the four different graphics modes. They were all printed using the plotting program in this chapter (with a rather complex set of formulas starting at line 600!).



Normal density graphics


Double density graphics


Double density/double speed


Quadruple density graphics

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 first program in this chapter, change line 50 to try each of the different modes. Just change the "K" to "L", "y", and "z" in turn. Your printouts should look something like this:

⟨ESC⟩"L"

⟨ESC⟩"y"

⟨ESC⟩"z"

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.

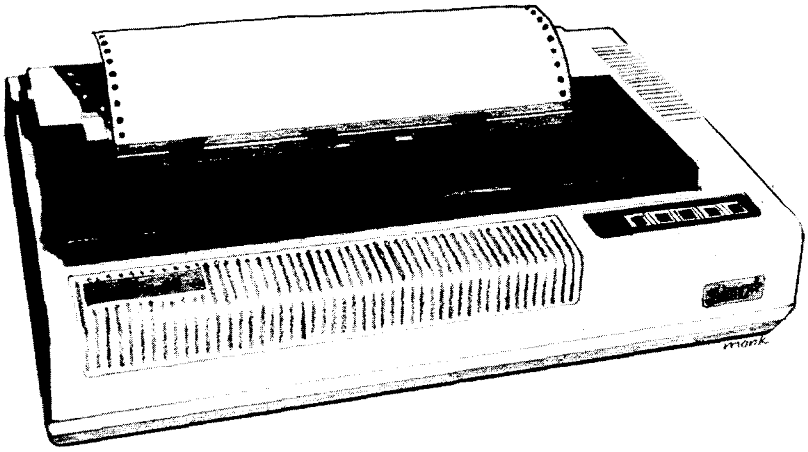
If You Have Problems with BASIC

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 to not 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
⟨ESC⟩ "z" *n1 n2 m1 m2* . . .	Print quadruple density graphics



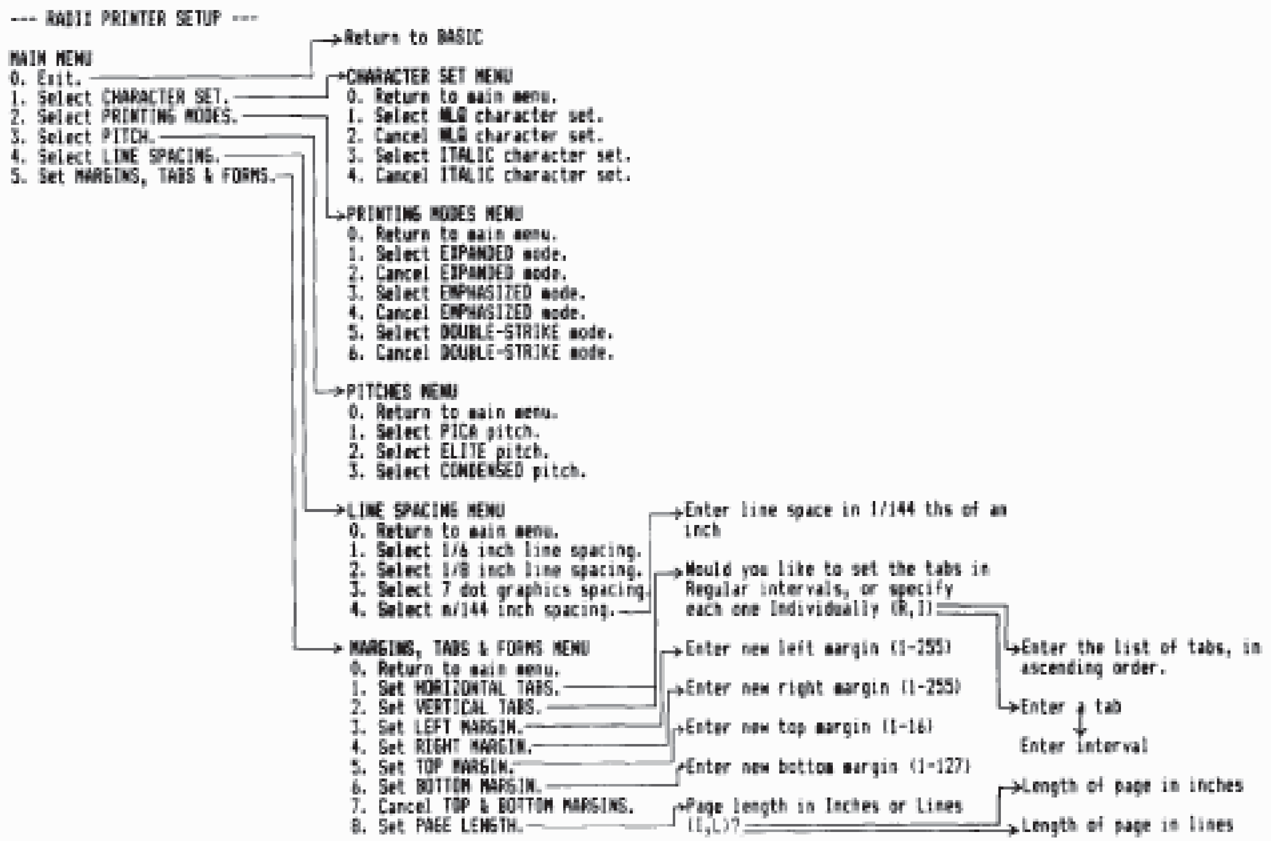
Chapter 13

Putting Radix to Work For You

If you've followed us this far, you've learned a lot about your Radix printer—how to use its myriad of type styles, sizes, line spacing options, character sets, margins, tabs, and more. Perhaps you've even created some download characters (maybe using the utility program in Chapter 11).

Now, as your reward (as if the knowledge of how to use all these features wasn't enough!) for reading this entire manual, we have one more utility program for you. With this program you can set many of Radix's print parameters with just a few keystrokes. No more writing a short program each time you want to change the print style to NLQ, for example. All you will need to do is type "RUN ⟨return⟩ 1100" and it's done—the program is completely menu-driven.

Table 13-1
Menus of Radix setup program



```
--- RADIX PRINTER SETUP ---

MAIN MENU                                 ->Return to BASIC
0. Exit.
1. Select CHARACTER SET.                  ->CHARACTER SET MENU
2. Select PRINTING MODES.                   0. Return to main menu.
3. Select PITCH.                            1. Select NLQ character set.
4. Select LINE SPACING.                     2. Cancel NLQ character set.
5. Set MARGINS, TABS & FORMS.               3. Select ITALIC character set.
                                            4. Cancel ITALIC character set.

                                          ->PRINTING MODES MENU
                                            0. Return to main menu.
                                            1. Select EXPANDED mode.
                                            2. Cancel EXPANDED mode.
                                            3. Select EMPHASIZED mode.
                                            4. Cancel EMPHASIZED mode.
                                            5. Select DOUBLE-STRIKE mode.
                                            6. Cancel DOUBLE-STRIKE mode.

                                          ->PITCHES MENU
                                            0. Return to main menu.
                                            1. Select PICA pitch.
                                            2. Select ELITE pitch.
                                            3. Select CONDENSED pitch.

                                          ->LINE SPACING MENU                     ->Enter line space in 1/144 ths of an
                                            0. Return to main menu.                 inch
                                            1. Select 1/6 inch line spacing.
                                            2. Select 1/8 inch line spacing.      ->Would you like to set the tabs in
                                            3. Select 7 dot graphics spacing.       Regular intervals, or specify
                                            4. Select n/144 inch spacing.           each one Individually (R,I)

                                          ->MARGINS, TABS & FORMS MENU            ->Enter new left margin (1-255)    ->Enter the list of tabs, in
                                            0. Return to main menu.                                                    ascending order.
                                            1. Set HORIZONTAL TABS.               ->Enter new right margin (1-255)
                                            2. Set VERTICAL TABS.                                                    ->Enter # tab
                                            3. Set LEFT MARGIN.                   ->Enter new top margin (1-16)
                                            4. Set RIGHT MARGIN.                                                       Enter interval
                                            5. Set TOP MARGIN.                    ->Enter new bottom margin (1-127)
                                            6. Set BOTTOM MARGIN.                                                    ->Length of page in inches
                                            7. Cancel TOP & BOTTOM MARGINS.       ->Page length in Inches or Lines
                                            8. Set PAGE LENGTH.                     (I,L)?                           ->Length of page in lines
```

It may take a while to enter it, but we think that in the long run, this program will save you time when you want to set margins or tabs or any of Radix's other advanced features. Enjoy!

```
10 'Program to setup RADIX printer as directed.
20 '
30 'Initialize.
40 ESC$ = CHR$(27) : TB = 25 : DIM TBS(256)
50 OPEN "lpt1:" AS #1 : WIDTH #1, 255 : KEY OFF
60 '
70 'Display MAIN menu.
80 TITLE$ = "MAIN MENU"
90 GOSUB 2290
100 PRINT TAB(TB) "0. Exit."
110 PRINT TAB(TB) "1. Select CHARACTER SET."
120 PRINT TAB(TB) "2. Select PRINTING MODES."
130 PRINT TAB(TB) "3. Select PITCH."
140 PRINT TAB(TB) "4. Select LINE SPACING."
150 PRINT TAB(TB) "5. Set MARGINS, TABS & FORMS."
160 GOSUB 2380
170 IF S<0 OR S>5 THEN BEEP : GOTO 160
180 IF S = 0 THEN CLOSE #1 : CLS : END
190 ON S GOSUB 210,480,350,1240,640
200 GOTO 60
210 '
220 'Subroutine to display CHARACTER SET menu.
230 TITLE$ = "CHARACTER SET MENU"
240 GOSUB 2290
250 PRINT TAB(TB) "0. Return to main menu."
260 PRINT TAB(TB) "1. Select NLQ character set."
270 PRINT TAB(TB) "2. Cancel NLQ character set."
280 PRINT TAB(TB) "3. Select ITALIC character set."
290 PRINT TAB(TB) "4. Cancel ITALIC character set."
300 GOSUB 2380
310 IF S<0 OR S>4 THEN BEEP : GOTO 300
320 IF S = 0 THEN RETURN
330 ON S GOSUB 1180,1210,1590,1620
340 GOTO 210
350 '
360 'Subroutine to display PITCHES menu.
370 TITLE$ = "PITCHES MENU"
380 GOSUB 2290
```

```
390 PRINT TAB(TB) "0. Return to main menu."
400 PRINT TAB(TB) "1. Select PICA pitch."
410 PRINT TAB(TB) "2. Select ELITE pitch."
420 PRINT TAB(TB) "3. Select CONDENSED pitch."
430 GOSUB 2380
440 IF S<0 OR S>3 THEN BEEP : GOTO 430
450 IF S = 0 THEN RETURN
460 ON S GOSUB 820,850,880
470 GOTO 350
480 '
490 'Subroutine to display PRINTING MODES menu.
500 TITLE$ = "PRINTING MODES MENU"
510 GOSUB 2290
520 PRINT TAB(TB) "0. Return to main menu."
530 PRINT TAB(TB) "1. Select EXPANDED mode."
540 PRINT TAB(TB) "2. Cancel EXPANDED mode."
550 PRINT TAB(TB) "3. Select EMPHASIZED mode."
560 PRINT TAB(TB) "4. Cancel EMPHASIZED mode."
570 PRINT TAB(TB) "5. Select DOUBLE-STRIKE mode."
580 PRINT TAB(TB) "6. Cancel DOUBLE-STRIKE mode."
590 GOSUB 2380
600 IF S<0 OR S>6 THEN BEEP : GOTO 590
610 IF S = 0 THEN RETURN
620 ON S GOSUB 1530,1560,2170,2200,2230,2260
630 GOTO 480
640 '
650 'Subroutine to display MARGINS, TABS & FORMS menu.
660 TITLE$ = "MARGINS, TABS & FORMS MENU"
670 GOSUB 2290
680 PRINT TAB(TB) "0. Return to main menu."
690 PRINT TAB(TB) "1. Set HORIZONTAL TABS."
700 PRINT TAB(TB) "2. Set VERTICAL TABS."
710 PRINT TAB(TB) "3. Set LEFT MARGIN."
720 PRINT TAB(TB) "4. Set RIGHT MARGIN."
730 PRINT TAB(TB) "5. Set TOP MARGIN."
740 PRINT TAB(TB) "6. Set BOTTOM MARGIN."
750 PRINT TAB(TB) "7. Cancel TOP & BOTTOM MARGINS."
760 PRINT TAB(TB) "8. Set PAGE LENGTH."
770 GOSUB 2380
780 IF S<0 OR S>8 THEN BEEP : GOTO 770
790 IF S = 0 THEN RETURN
800 ON S GOSUB 1820,2130,910,970,1030,1090,1150,1650
810 GOTO 640
```

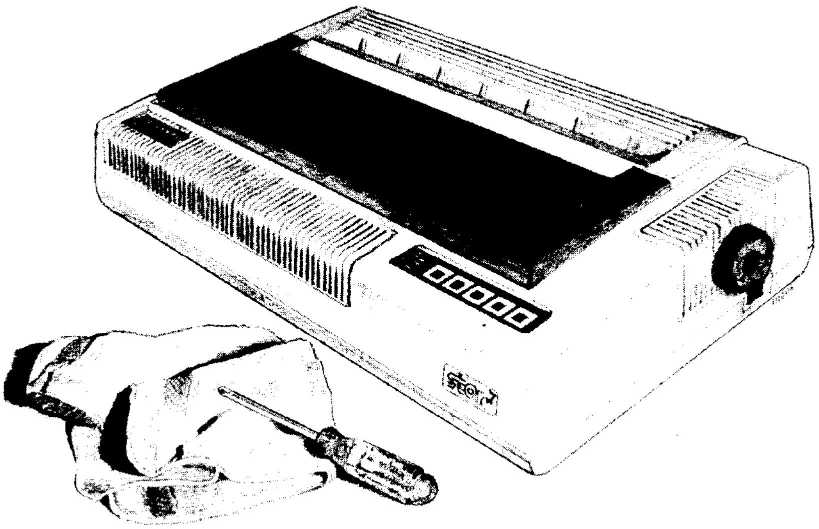
```
820 '
830 'Subroutine to select PICA pitch.
840 S$ = ESC$ + "B" + CHR$(1) : GOSUB 2460 : RETURN
850 '
860 'Subroutine to select ELITE pitch.
870 S$ = ESC$ + "B" + CHR$(2) : GOSUB 2460 : RETURN
880 '
890 'Subroutine to select CONDENSED pitch.
900 S$ = ESC$ + "B" + CHR$(3) : GOSUB 2460 : RETURN
910 '
920 'Subroutine to set LEFT MARGIN.
930 GOSUB 2500
940 INPUT "Enter new left margin (1-255)" ; X
950 IF X < 1 OR X > 255 THEN BEEP : GOTO 930
960 S$ = ESC$ + "M" + CHR$(X) : GOSUB 2460 : RETURN
970 '
980 'Subroutine to set right MARGIN
990 GOSUB 2500
1000 INPUT "Enter new right margin (1-255)" ; X
1010 IF X < 1 OR X > 255 THEN BEEP : GOTO 990
1020 S$ = ESC$ + "Q" + CHR$(X) : GOSUB 2460 : RETURN
1030 '
1040 'Subroutine to set TOP MARGIN.
1050 GOSUB 2500
1060 INPUT "Enter new top margin (1-16)" ; X
1070 IF X < 1 OR X > 16 THEN BEEP : GOTO 1050
1080 S$ = ESC$ + "R" + CHR$(X) : GOSUB 2460 : RETURN
1090 '
1100 'Subroutine to set BOTTOM MARGIN.
1110 GOSUB 2500
1120 INPUT "Enter new bottom margin (1-127)" ; X
1130 IF X < 1 OR X > 127 THEN BEEP : GOTO 1110
1140 S$ = ESC$ + "N" + CHR$(X) : GOSUB 2460 : RETURN
1150 '
1160 'Subroutine to CANCEL TOP & BOTTOM MARGINS.
1170 S$ = ESC$ + "O" : GOSUB 2460 : RETURN
1180 '
1190 'Subroutine to select NLQ character set.
1200 S$ = ESC$ + "B" + CHR$(4) : GOSUB 2460 : RETURN
1210 '
1220 'Subroutine to cancel NLQ character set.
1230 S$ = ESC$ + "B" + CHR$(5) : GOSUB 2460 : RETURN
1240 '
```

```
1250 'Subroutine to select LINE SPACING.
1260 TITLE$ = "LINE SPACING MENU"
1270 GOSUB 2290
1280 PRINT TAB(TB) "0. Return to main menu."
1290 PRINT TAB(TB) "1. Select 1/6 inch line spacing."
1300 PRINT TAB(TB) "2. Select 1/8 inch line spacing."
1310 PRINT TAB(TB) "3. Select 7 dot graphics spacing."
1320 PRINT TAB(TB) "4. Select n/144 inch spacing."
1330 GOSUB 2380
1340 IF S<0 OR S>4 THEN BEEP : GOTO 1330
1350 IF S = 0 THEN RETURN
1360 ON S GOSUB 1380,1410,1440,1470
1370 GOTO 1240
1380 '
1390 'Subroutine to select 1/6 inch line spacing.
1400 S$ = ESC$ + "2" : GOSUB 2460 : RETURN
1410 '
1420 'Subroutine to select 1/8 inch line spacing.
1430 S$ = ESC$ + "0" : GOSUB 2460 : RETURN
1440 '
1450 'Subroutine to select 7 dot graphics spacing.
1460 S$ = ESC$ + "1" : GOSUB 2460 : RETURN
1470 '
1480 'Subroutine to select n/144 inch line spacing.
1490 GOSUB 2500
1500 INPUT "Enter line space in 1/144 ths of an inch"; X
1510 IF X < 0 OR X > 255 THEN BEEP : GOTO 1490
1520 S$ = ESC$ + "3" + CHR$(X) : GOSUB 2460 : RETURN
1530 '
1540 'Subroutine to select EXPANDED print.
1550 S$ = ESC$ + "W" + CHR$(1) : GOSUB 2460 : RETURN
1560 '
1570 'Subroutine to cancel EXPANDED printing.
1580 S$ = ESC$ + "W" + CHR$(0) : GOSUB 2460 : RETURN
1590 '
1600 'Subroutine to select ITALIC character set.
1610 S$ = ESC$ + "4" : GOSUB 2460 : RETURN
1620 '
1630 'Subroutine to cancel ITALIC character set.
1640 S$ = ESC$ + "5" : GOSUB 2460 : RETURN
1650 '
1660 'Subroutine to set PAGE LENGTH.
1670 GOSUB 2500
```

```
1680 PRINT "Page length in Inches or Lines (I,L)?"
1690 PRINT TAB(TB) ;
1700 A$ = INKEY$ : IF A$ = "" THEN 1700
1710 IF A$ = "I" OR A$ ="i" THEN 1740
1720 IF A$ = "L" OR A$ ="l" THEN 1780
1730 BEEP : GOTO 1700
1740 INPUT "Length of page in inches (1-32)" ; X
1750 IF X < 1 OR X > 32 THEN BEEP : GOTO 1670
1760 S$ = ESC$ + "C" + CHR$(0) + CHR$(X)
1770 GOSUB 2460 : RETURN
1780 INPUT "Length of page in lines (1-127)" ; X
1790 IF X < 1 OR X > 127 THEN BEEP : GOTO 1670
1800 S$ = ESC$ + "C" + CHR$(X)
1810 GOSUB 2460 : RETURN
1820 '
1830 'Subroutine to set HORIZONTAL TABS.
1840 S$ = ESC$ + "D" : MAX = 255 : GOSUB 1850 : RETURN
1850 '
1860 'Subroutine to set tabs, either horiz or vert.
1870 GOSUB 2500
1880 PRINT "Would you like to set the tabs in"
1890 PRINT TAB(TB) "Regular intervals, or specify"
1900 PRINT TAB(TB) "each one Individually (R,I)"
1910 A$ = INKEY$ : IF A$ = "" THEN 1910
1920 IF A$ = "R" OR A$ = "r" THEN 2070
1930 IF A$ = "I" OR A$ = "i" THEN 1950
1940 BEEP : GOTO 1850
1950 PRINT : I = 2 : TBS(1) = -1
1960 PRINT TAB(TB) "Enter the list of tabs, in"
1970 PRINT TAB(TB) "ascending order. No more than" MAX
   "."
1980 PRINT TAB(TB) : INPUT "Enter a tab" ; TBS(I)
1990 IF TBS(I) < 0 OR TBS(I) > 255 THEN 1940
2000 IF TBS(I) = 0 THEN I = 1 : GOTO 2040
2010 IF TBS(I) <= TBS(I-1) THEN 1940
2020 I = I + 1 : IF I > MAX THEN 1940
2030 GOTO 1980
2040 I = I + 1
2050 S$ = S$ + CHR$(TBS(I)) : IF TBS(I) <> 0 THEN 2040
2060 S$ = S$ + CHR$(0) : GOSUB 2460 : RETURN
2070 PRINT : PRINT TAB(TB) ; : INPUT "Enter interval" ;
   X
2080 IF X < 0 OR X > 255 THEN 1940
2090 FOR I = 1 TO 255 STEP X
```

```
2100 MAX = MAX - 1 : IF MAX = 0 THEN 2120
2110 S$ = S$ + CHR$(I) : NEXT I
2120 S$ = S$ + CHR$(0) : GOSUB 2460 : RETURN
2130 '
2140 'Subroutine to set VERTICAL TABS.
2150 S$ = ESC$ + "P" : MAX = 20 : GOSUB 1850
2160 RETURN
2170 '
2180 'Subroutine to select EMPHASIZED printing.
2190 S$ = ESC$ + "E" : GOSUB 2460 : RETURN
2200 '
2210 'Subroutine to cancel EMPHASIZED printing.
2220 S$ = ESC$ + "F" : GOSUB 2460 : RETURN
2230 '
2240 'Subroutine to select DOUBLE-STRIKE printing.
2250 S$ = ESC$ + "G" : GOSUB 2460 : RETURN
2260 '
2270 'Subroutine to cancel DOUBLE-STRIKE printing.
2280 S$ = ESC$ + "H" : GOSUB 2460 : RETURN
2290 '
2300 'Subroutine to print a menu title.
2310 CLS
2320 PRINT : PRINT : PRINT
2330 PRINT TAB(27) "--- RADIX PRINTER SETUP ---"
2340 PRINT
2350 PRINT TAB((80-LEN(TITLE$))/2) TITLE$
2360 PRINT : PRINT
2370 RETURN
2380 '
2390 'Subroutine to input menu selection.
2400 LOCATE 20,18 : PRINT "Enter selection or press P
   for print sample."
2410 C$ = INKEY$ : IF C$ = "" THEN 2410
2415 IF C$ = "P" OR C$ = "p" THEN GOSUB 3000 : GOTO 2380
2420 IF C$ < "0" OR C$ > "9" THEN BEEP : GOTO 2410
2430 S = VAL(C$)
2440 LOCATE 20,18 : PRINT STRING$(50," ")
2450 RETURN
2460 '
2470 'Subroutine to output command string.
2480 PRINT #1, S$ ;
2490 RETURN
2500 '
```

```
2510 'Subroutine to clear screen & position cursor.
2520 CLS : LOCATE 10,TB : RETURN
3000 '
3010 ' Subroutine to print sample
3020 FOR I = 1 TO 4 : FOR J = 33 TO 126
3030 PRINT #1, CHR$(J);
3040 NEXT : PRINT #1, CHR$(10) : NEXT
3050 RETURN
```



Chapter 14

Basic Maintenance

As almost any good mechanic will tell you, dust and heat are prime enemies of any mechanism, and Radix is no exception. The best maintenance is *preventive*. So, to start with, we hope you've found a clean, dust-free location with a comfortable temperature range for both you and your computer/printer system. Appendix A gives you further tips on locating Radix.

Cleaning Radix

The second rule for long life is *periodic cleaning*. Both inside and outside of the case and covers respond gratefully to periodic cleaning with a damp rag and alcohol. Do this whenever the case appears to be getting dirty, always being careful to avoid dripping alcohol on the printer mechanism.

To remove dust and paper lint from inside the tractor and printer areas, it's best to use a soft brush, *but*, be very, very careful not to bend or injure any electronic parts or wiring, as they are vulnerable to a heavy-handed touch.

Besides the periodic cleanings, the only other maintenance you'll likely encounter will be changing the ink ribbon cartridge, replacing a blown fuse, or replacement of the print head after a long period of use.

Replacing the Ink Ribbon

When the printing gets too faint for comfortable reading, it's time for a new ink ribbon. By far the most convenient way is to simply replace the entire ribbon cartridge (Appendix A describes this procedure). After all, that's the purpose of the cartridge: to save time and messing with dirty ribbons.

It is possible, however, to buy a replacement ribbon and insert it yourself inside the original cartridge casing. The procedure for inserting a new ribbon into the old cartridge (not recommended for non-mechanical types!) is as follows.

1. First, obtain from your Radix dealer the correct type of ribbon "sub-cassette" (*not* spool-type ribbons used with some other printers).
2. Remove the ribbon cartridge from the printer by holding both ends and pulling straight up from the holder springs. (Refer to Appendix A for illustrations of installing ribbon cartridge.)
3. Pry open the cartridge cover with a thin-bladed screwdriver. Arrows in Figure 14-1 show the numerous slots for inserting a screwdriver.
4. Press hard against the end of the idler gear holder to make a gap between it and the ribbon drive gear, and remove the old ink ribbon sub-cassette. See Figure 14-2.
5. Clean out any dirt from inside and around the cartridge and around the ribbon drive gear.

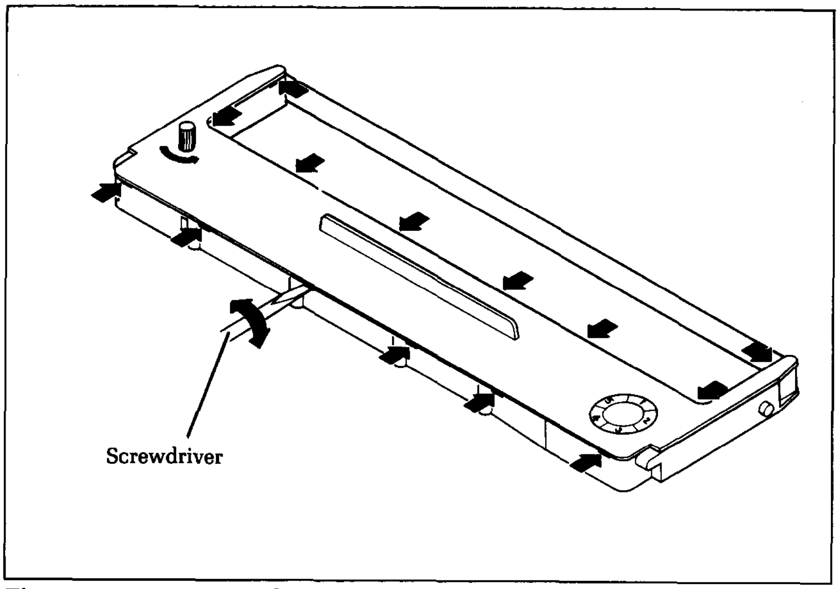


Figure 14-1. *Use a screwdriver to pry open the cartridge.*

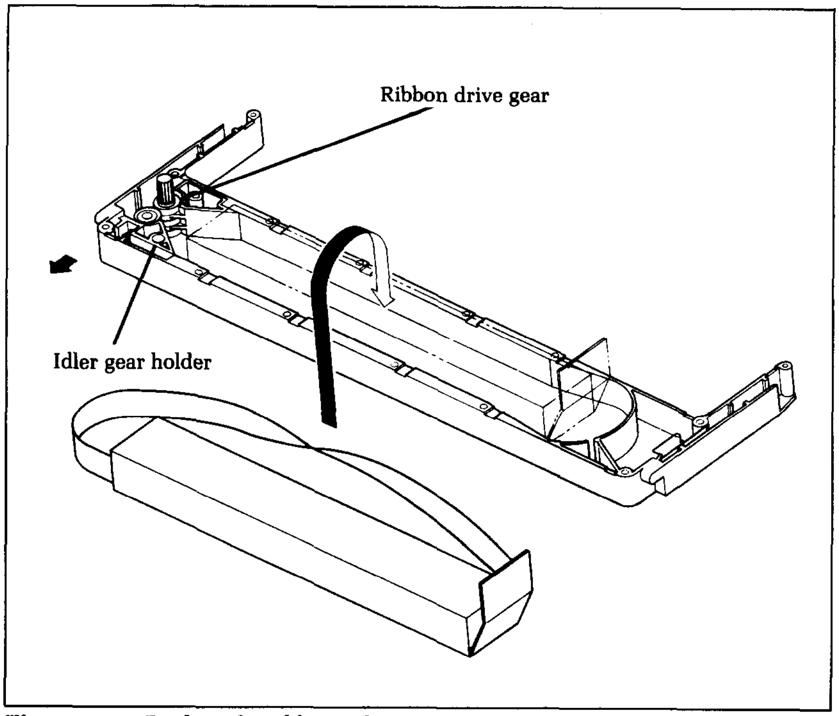


Figure 14-2. *Replace the ribbon sub-cassette.*

6. Remove the wrapping from the new ribbon sub-cassette, remove the adhesive tape attached to the joint, and insert the sub-cassette into the ribbon cassette as shown in Figure 14-2.
7. Pull out the ink ribbon and set it according to the directions shown by the arrow in Figure 14-3. It's easy for the ribbon to get twisted somewhere along its pathway. Don't let it happen!

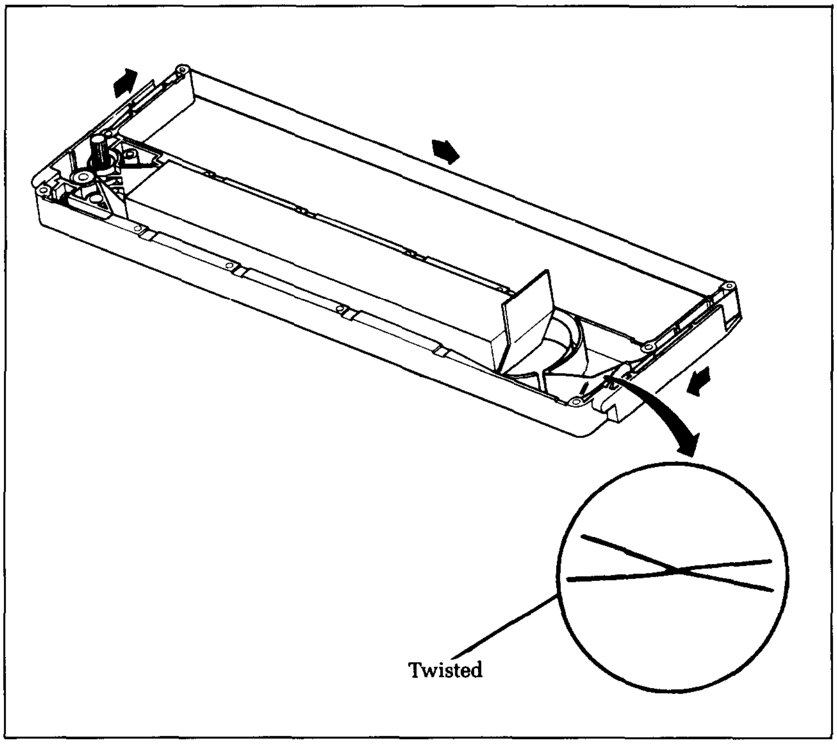


Figure 14-3. *Make sure that the ribbon is not twisted when you thread it through its path.*

8. Firmly pull the idler gear towards you and guide the ribbon between the idler gear and the ribbon drive gear.
9. Remove both top and bottom of the ribbon sub-cassette.
10. Replace the ribbon cartridge top cover.
11. When you've completed the installation, mark the correct number on the silver label stuck on the right-hand side of the cartridge cover. This number indicates the number of times the ribbon has been replaced. Five replacements is the maximum, after which you should buy a complete new cartridge.

Replacing a Fuse

How can you tell when you've blown a fuse? Well, when the printer won't operate and the power lamp on the control panel isn't lit, even though you're sure that the power switch is *on* and the printer is plugged in — it's likely a blown fuse.

To check the primary fuse, you start by turning the power switch *off* and unplugging the power cord.

Warning: There is an extreme shock hazard inside Radix. To avoid serious injury, it is important that the power cord is disconnected.

Next, remove the upper case, shown in Figure 14-4, by pulling off the platen knob.

Caution: Don't twist or turn the platen knob; pull it *straight* off.

Then remove the fastening screws along the back side. Lift the back edge of the cover and at the same time, pull it slightly forward to release the front of the case. Lift it all the way off, being careful not to pull the wires which connect the cover to the case.

When the case is off, check Figure 13-5 for location of the primary fuse, which you'll find held by its clamps close to the power switch. The fuse is a commonly used type, with a metal strip suspended in a glass and metal case. If the strip is broken, the fuse is blown. Replace this fuse with a 3A/125V slow-blow type fuse (Bell 5MT3 or equivalent). Now reassemble Radix and test-run it. If the printer still isn't working, call on your Radix dealer/service center for help.

Replacing the Print Head

The dot matrix print head has a remarkably long life, printing perhaps 100,000,000 characters before it wears out. You'll know when that happens when the printout is too faint for your taste even after replacing the ink ribbon or cartridge.

Warning: The print head gets hot during operation, so let it cool off for awhile, if necessary, to avoid burning your fingers.

To replace the print head, start by turning the power switch *off* and unplugging the power cord.

Then, in sequence:

1. Remove the front cover and the ribbon cartridge.

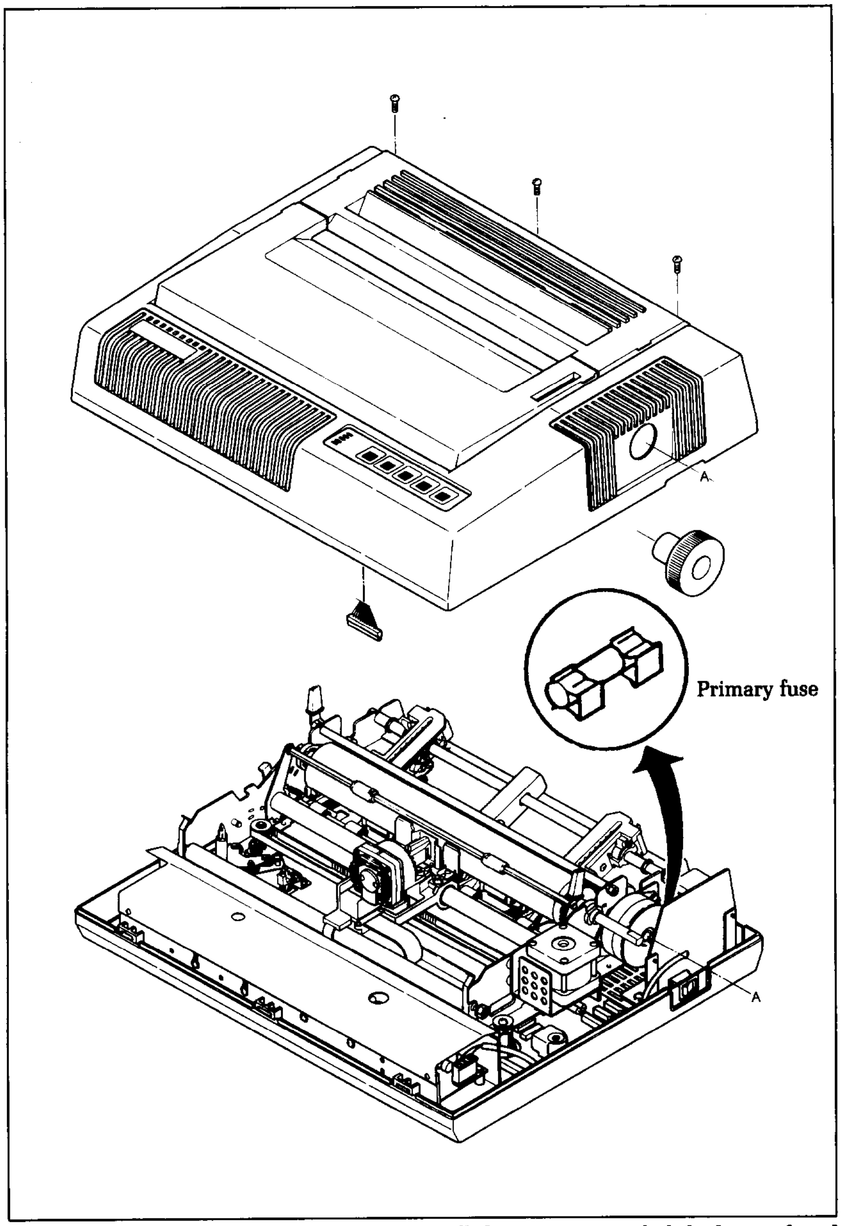


Figure 14-4. *After removing the screws, pull the upper case slightly forward and lift it off the printer. The primary fuse is located near the power switch.*

2. Remove the two screws and washers fastening the print head.
3. While holding the print head, pull off the head cable connector from the print head.
4. Insert the head cable connector to a new print head and fasten with the same two screws and washers.

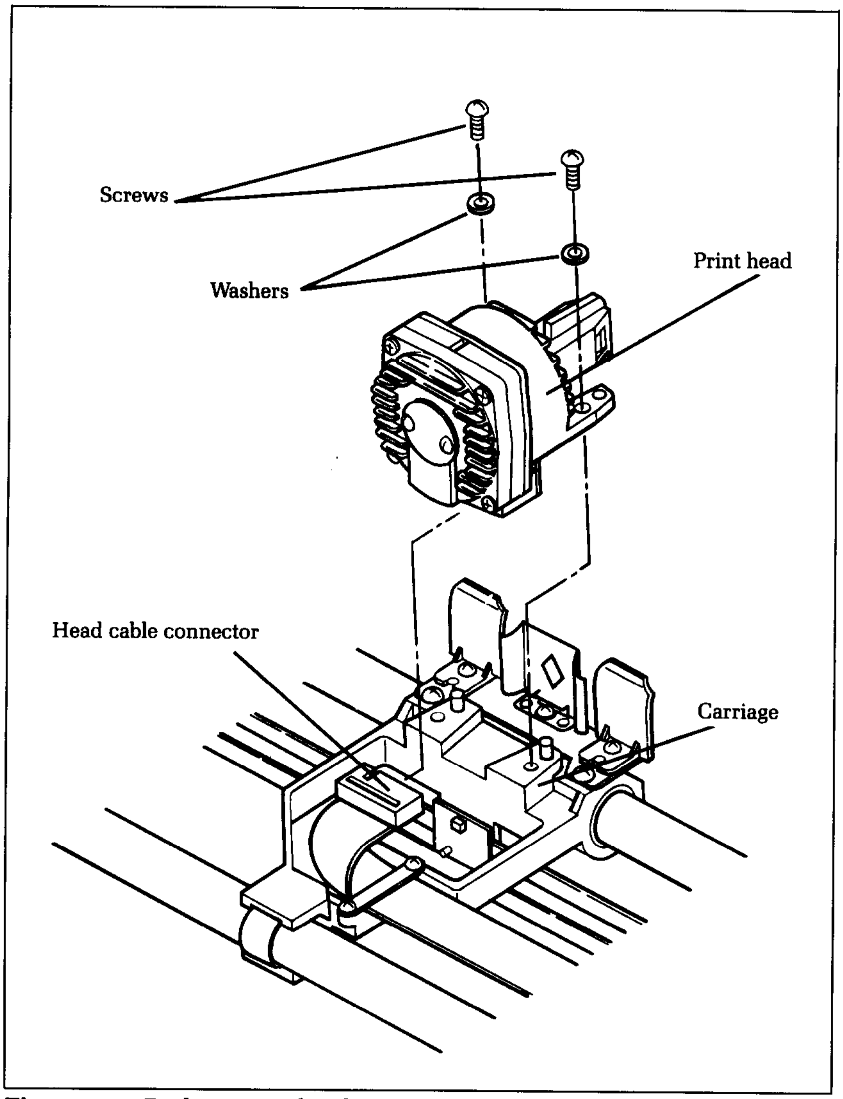


Figure 14-5. *Replacement of Radix's print head is simple.*

5. Apply "screw lock," (an adhesive available at hardware stores) to the heads of the screws.

Be absolutely sure that you've made a good solid connection between the print head and its cable connector, or it could cause problems.

Appendix

Appendix A

Setting Up Radix

In this appendix, we'll show you how to unpack your new Radix 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 120 volts AC.
  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 A-1. There should be exactly 9 items. One important item is the printer's warranty and registration card. Now is the time to fill it in and mail it. It's a good warranty, and you'll like the protection it gives you.

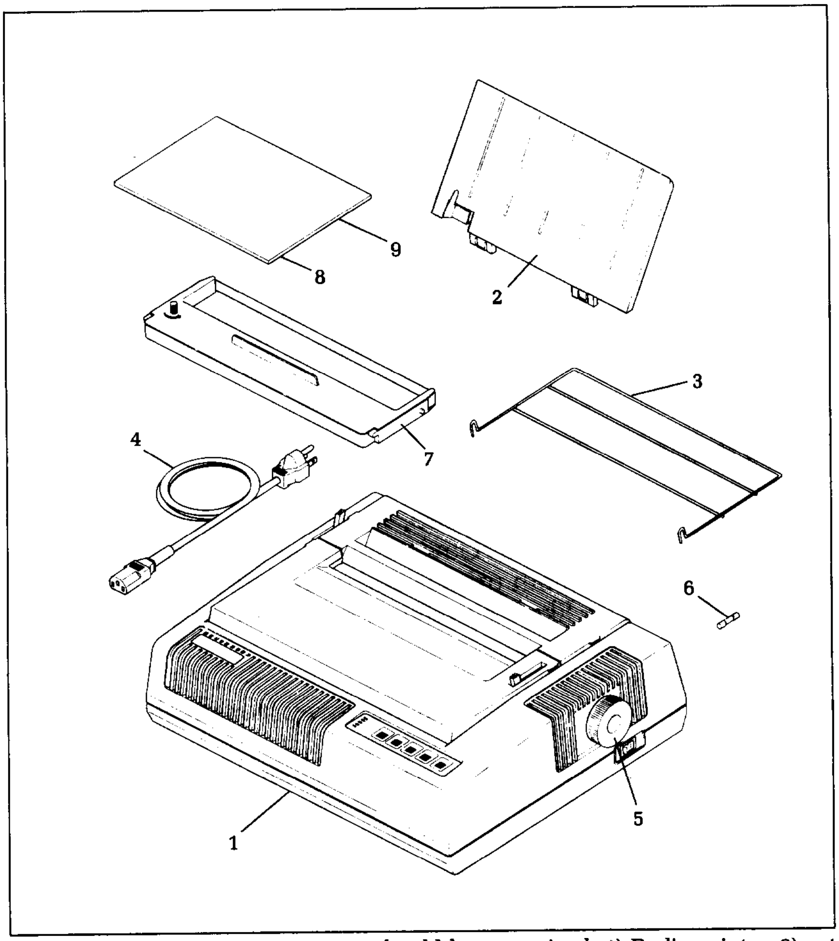


Figure A-1. *Inside the carton you should have received: 1) Radix printer, 2) cut sheet guide, 3) continuous paper guide, 4) power cord, 5) platen knob, 6) spare fuse, 7) ribbon cartridge, 8) this user's manual, and 9) warranty registration card.*

Let's move on to 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 Radix 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.

Radix 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 A-2 illustrates the proper position and movement for both removal and replacement of the covers.

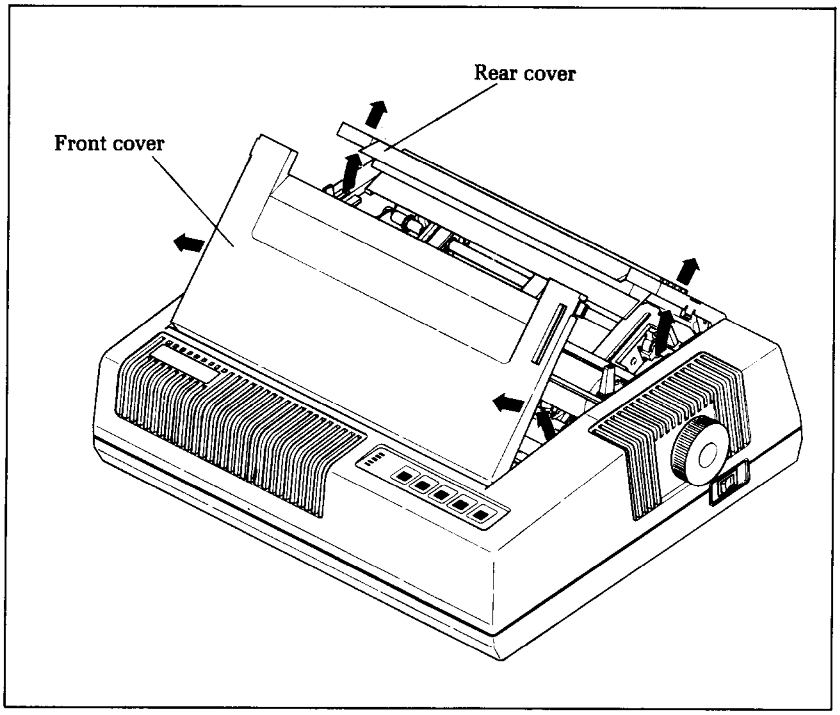


Figure A-2. *Remove the printer covers by tilting them up to about 45°, then lifting straight up.*

Removing packing and shipping screws

There are three (on a Radix-10) or four (on a Radix-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 A-3.

Next, remove the front cover, and remove the large flat piece of cardboard packing which protects the print head, per Figure A-4.

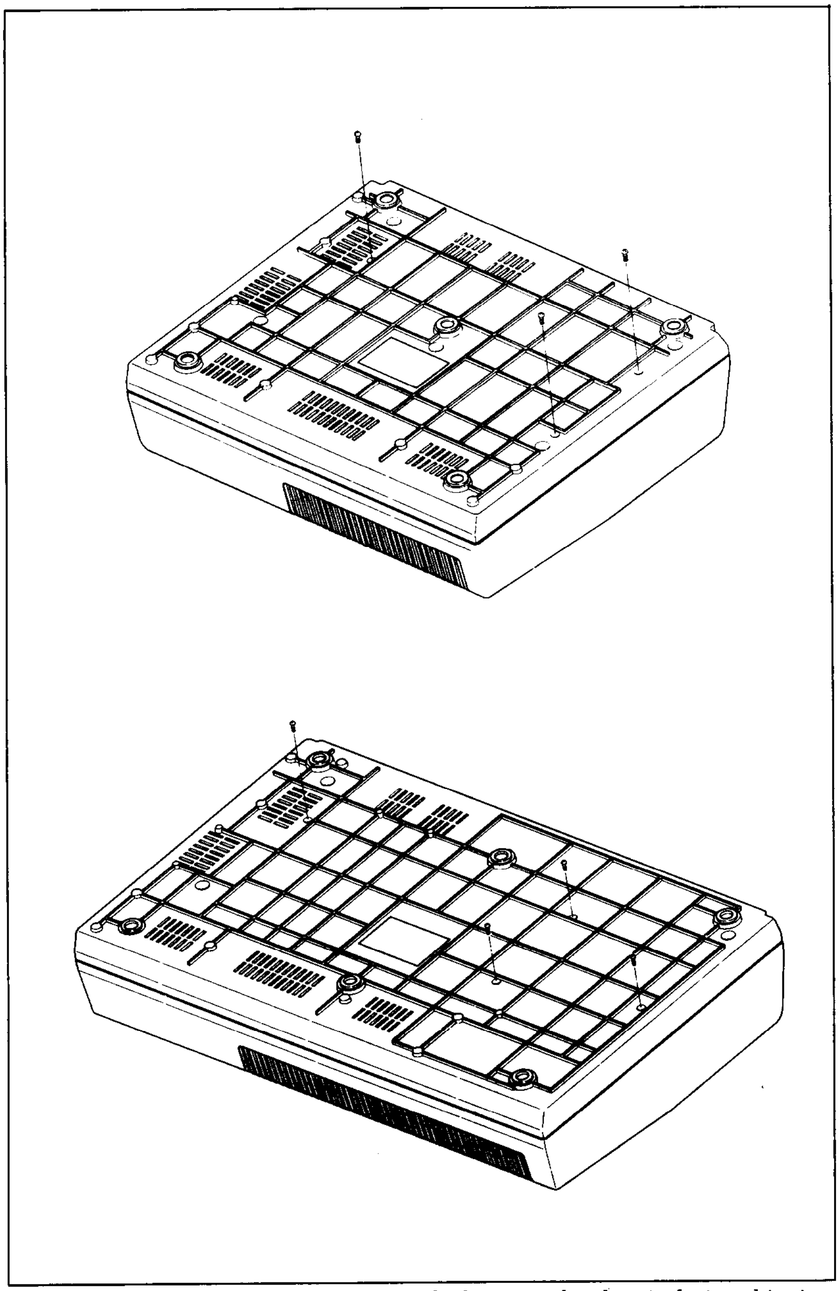


Figure A-3. *Radix-10 has three screws which secure the chassis during shipping; Radix-15 has four. They should be removed before use.*

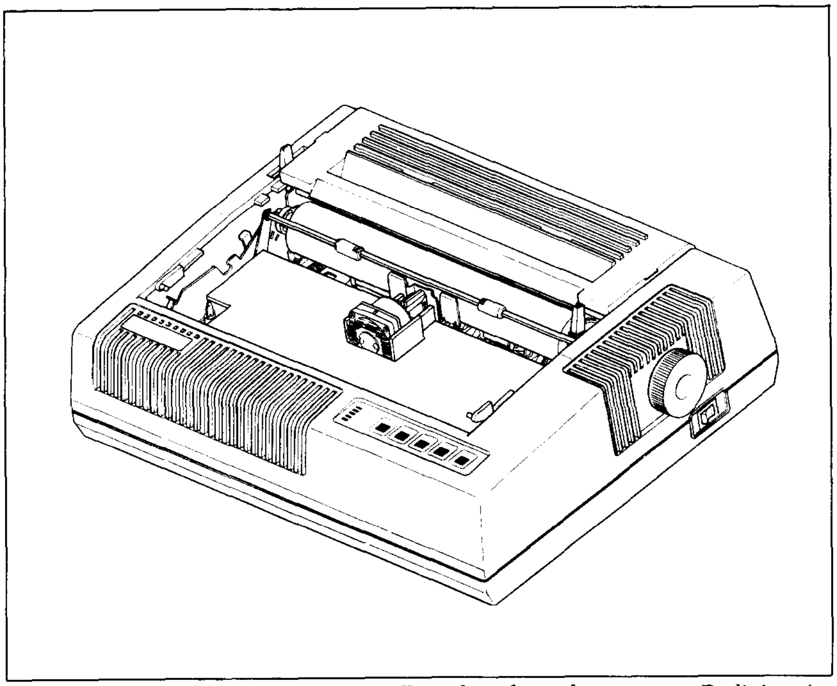


Figure A-4. *Remove the piece of cardboard packing that protects Radix's print head.*

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. (You *did* fill in that warranty card, didn't you?)

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 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.

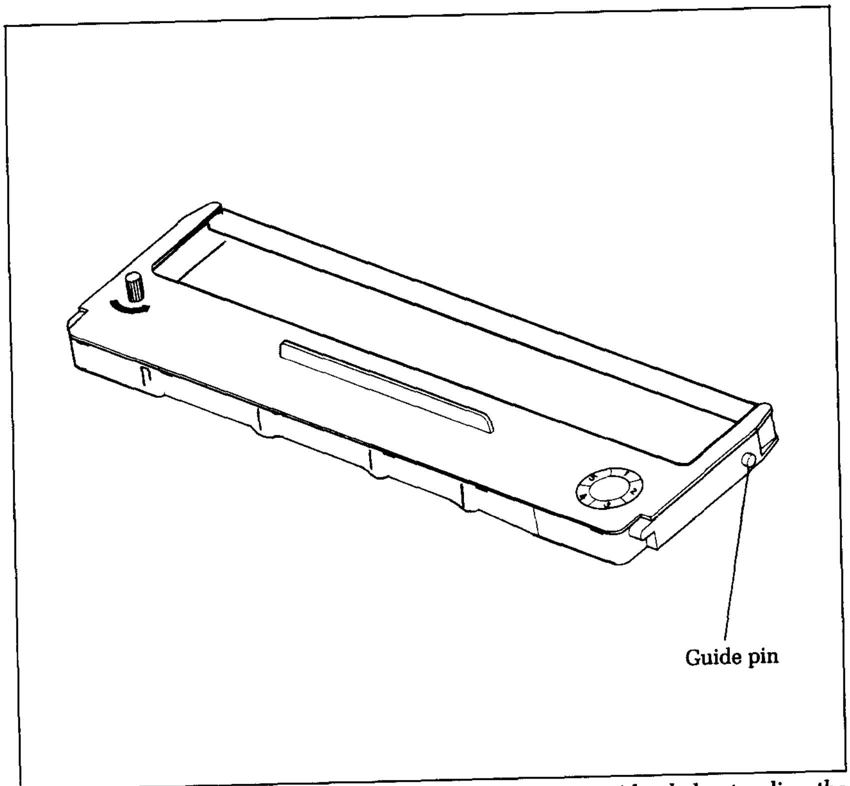


Figure A-5. *A guide pin on each side of the ribbon cartridge helps to align the cartridge during installation.*

3. Note the position of the guide pins on the cartridge as shown in Figure A-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 A-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.
5. Now thread the ribbon carefully between the print head and the ribbon guide next to the platen. (Take a good look at

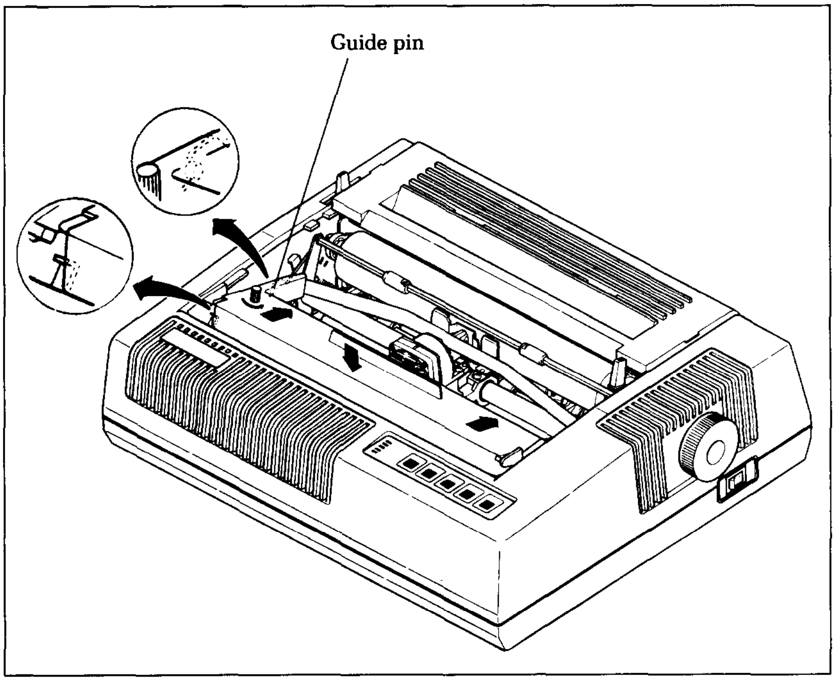


Figure A-6. *Tilt the ribbon cartridge in until the guide pins meet the hooks in the printer frame, then lower the front edge until the holder springs hold it in place.*

Figure A-7.) You might want to use a ball point pen to lightly press the ribbon guide against the platen (rubber 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 1, Radix 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!

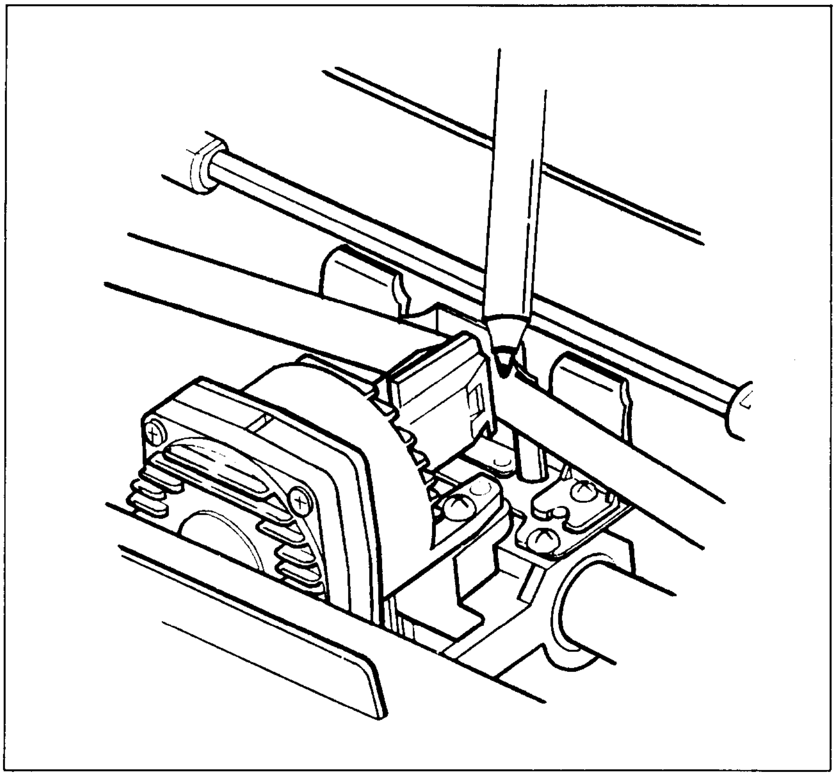


Figure A-7. *Use a ball point pen to place the ribbon between the print head and the ribbon guide. It's important that the ribbon is centered vertically between the print head and the ribbon guide.*

Connecting Radix to Your Computer

To complete the installation, you'll need to connect Radix to your computer. In appendices B through E, we've described this procedure, including specific guidelines for making connections ("interfacing") with several of the most popular computers used by Radix owners.

Then, in Chapter 1, you'll learn how to load paper (here's where you'll use the paper guides) and operate Radix.

Appendix B

IBM Personal Computer and Compaq Computer

Both the IBM Personal Computer and the Compaq computer function the same when connected to Radix. We will discuss the IBM-PC, knowing that all we say works just as well for the Compaq.

Connecting Radix to an IBM

Radix can connect to either a serial or a parallel interface in the IBM-PC or IBM-XT computers. IBM calls a parallel interface a "Parallel Printer Adapter," and they call a serial interface an "Asynchronous Communications Adapter."

You only need a cable to connect Radix to your IBM-PC. Your Radix dealer can furnish this cable, or you can use a standard IBM-PC parallel printer cable for the parallel interface.

Connecting with the parallel interface

We recommend that you set the DIP switches in Radix as shown below when connecting it to an IBM-PC parallel interface.

Connecting to the serial interface

The IBM-PC expects its printer to be connected to the parallel interface. If you are using the serial interface, then you will need to instruct your computer to send information to the serial interface instead of to the parallel interface. This is done with the MODE command. You must use the following two commands each time you turn on your computer.

```
MODE COM1:48,N,8,1,P
MODE LPT1:=COM1:
```

The first line sets up the asynchronous adapter to match the

Table B-1
Recommended DIP switch settings for IBM-PC

Switch	Setting	Function
A-1	ON	11 inch page size
A-2	ON	Normal print density
A-3	ON	10 CPI pitch
A-4	ON	Normal characters
A-5	ON	1/6 inch line feed
A-6	ON	U.S.A. Character set
A-7	ON	
A-8	ON	
C-1	ON	Paper-out detector active
C-2	OFF	Parallel interface
C-3	OFF	8-bit interface
C-4	OFF	No auto line feed

Table B-2
IBM-PC parallel cable

Radix			IBM-PC Parallel	
Pin No.	**Function**		**Pin No.**	**Function**
1	STROBE	————	1	STROBE
2	D1	————	2	D0
3	D2	————	3	D1
4	D3	————	4	D2
5	D4	————	5	D3
6	D5	————	6	D4
7	D6	————	7	D5
8	D7	————	8	D6
9	D8	————	9	D7
10	ACK	————	10	ACK
11	BUSY	————	11	BUSY
12	PAPER END	————	12	PAPER END
13	SELECTED	————	13	SELECT
16	GROUND	————	18-25	GROUND
31	RESET	————	16	RESET
32	ERROR	————	15	ERROR

settings of DIP switch B in Radix. The second re-directs printer output to the serial port. The switches on DIP switch B must be set as shown below to use this MODE command. (The IBM-DOS manual tells you how to create a different MODE command for different DIP switch settings.) You can put these two MODE commands into a file named AUTOEXEC.BAT and it will execute automatically each time you start your computer.

Table B-3
Serial switch settings

Switch	Setting	Function
B-1	OFF	1 stop bit
B-2	OFF	8 data bits
B-3	OFF	No parity
B-4	ON	Serial busy, 1 block mode
B-5	OFF	
B-6	either	Parity
B-7	ON	4800 baud
B-8	OFF	
B-9	ON	
B-10	either	Not used

The serial cable shown below will work with DIP switch B set as shown above to connect Radix to a serial interface on the IBM.

Table B-4
IBM-PC serial cable

Radix			IBM-PC	
Pin No.	**Function**		**Pin No.**	**Function**
2	TRANSMIT DATA	———	3	RECEIVE DATA
3	RECEIVE DATA	———	2	TRANSMIT DATA
4	REQUEST TO SEND	———	5	CLEAR TO SEND
5	CLEAR TO SEND	———	4	REQUEST TO SEND
7	SIGNAL GROUND	———	7	SIGNAL GROUND
8	CARRIER DETECT	———	4	REQUEST TO SEND
20	DATA TERMINAL READY	———	6	DATA SET READY

BASIC programming

All the programs in this book are written in the BASIC used by the IBM-PC. That makes it easy to do the things that we show you. But when you start writing your own programs there are several things that you should know.

IBM BASIC defaults to a printer width of 80. This means that it will automatically insert a carriage return and line feed after every 80 characters. If you want to print lines longer than 80 characters you will need to change the width of the printer. If you set the printer width to 255, then the IBM will *never* insert a line feed and carriage return, unless you start a new line. (This is what you want usually.) To set the width of the printer to 255, use this statement:

```
1ØØ WIDTH "LPT1:", 255
```

IBM BASIC has one other little trick that will mess up your graphics if you let it. IBM BASIC is very insistent about adding a line feed to a carriage return. This is fine if you are printing text, but if an ASCII 13 pops up in the middle of your graphics printout, IBM BASIC will *still* add a line feed to it. This will put strange things in the middle of your graphics, and leave you with extra characters at the end of your line.

There is an easy way to avoid this problem. You just open the printer as a random file. The following program shows how this is done.

```
1Ø OPEN "LPT1:" AS #1          ' RANDOM ACCESS
2Ø WIDTH #1, 255               ' SET WIDTH TO 255
3Ø PRINT #1, "TESTING"         ' PRINT A LINE
4Ø PRINT #1, CHR$(1Ø)          ' ADD YOUR OWN LF
```

Listing programs

To list programs on Radix, make sure the program is in the IBM's memory and use the LLIST command. This directs the listing to the printer instead of the screen.

Printing Graphics Screens

Version 2.0 of the IBM DOS has a program called GRAPHICS that allows you to print a graphics display screen. The program as IBM created it is, however, not compatible with Star printers. But all that is required to make it work is to change two bytes of the program. This can easily be done with the DEBUG program that comes with IBM DOS. (Even if you have never used DEBUG before we will lead you through it.)

The first step is to create a diskette with DOS, GRAPHICS.COM and DEBUG.COM on it (it doesn't matter if there are other things on it too). We will leave it to you to create this diskette. Look in your computer's manual if you have trouble. Be sure that this is *not* your original DOS diskette.

With this diskette in drive A, follow the script below. The things that you are to type are shown in *italic type*. The messages that will appear on your screen are shown in regular type. With two exceptions, every number should appear on your screen exactly as it does in this script. The two exceptions are the four digit numbers before the colons (0921: in the script). They may be different on your computer. The symbol ⟨enter⟩ means to press the enter key.

```
A>DEBUG GRAPHICS.COM <enter>
-E 169 <enter>
Ø921:Ø169  18.1Ø <enter>
-E 25Ø <enter>
Ø921:Ø25Ø  24.18 <enter>
-W <enter>
Writing Ø315 bytes
-Q <enter>

A>
```

To use this program, type GRAPHICS at the A⟩ prompt *before* you create a graphics image on the screen. Then when you want to print a graphics image, press shift-PrtSc and the image will be copied from the screen to the printer. For more information on the GRAPHICS program refer to your DOS manual.

Program Listings

There are no program listings given here for the IBM-PC because all the programs in the book are written for the IBM-PC.

Appendix C

Apple II Computers

Apple II computers require an interface board (mounted inside the Apple II) and a cable to run Radix. Star recommends that you use the **grafstar**™ interface for the Apple II, II+, and IIe. It comes complete with a cable and is easily installed. A unique feature of the **grafstar**™ makes it possible to do some fancy dot graphics programming.

You can, of course, use many of the available parallel interface boards for the Apple II, and an appropriate cable.

Setting the Switches

We recommend that you set the DIP switches in Radix as shown below when connecting it to an Apple II. Since you'll be using the parallel interface, the settings of switch B have no effect.

Table C-1
Recommended DIP switch settings for Apple

Switch	Setting	Function
A-1	ON	11 inch page size
A-2	ON	Normal print density
A-3	ON	10 CPI pitch
A-4	ON	Normal characters
A-5	ON	1/6 inch line feed
A-6	ON	U.S.A. Character set
A-7	ON	
A-8	ON	
C-1	ON	Paper-out detector active
C-2	OFF	Parallel interface
C-3	ON	7-bit interface
C-4	OFF	No auto line feed

Table C-2
Apple parallel cable

Radix			**Apple Board**	
Pin No.	**Function**		**Pin No.**	**Function**
25	SIG GND	————	1	SIG GND
26	SIG GND	————	2	SIG GND
27	SIG GND	————	3	SIG GND
1	STROBE	————	4	STROBE
28	SIG GND	————	5	N/C
2	DATA1	————	6	DATA1
3	DATA2	————	7	DATA2
4	DATA3	————	8	DATA3
5	DATA4	————	9	DATA4
6	DATA5	————	10	DATA5
7	DATA6	————	11	DATA6
8	DATA7	————	12	DATA7
9	DATA8	————	13	DATA8
10	ACK	————	14	ACK
29	SIG GND	————	15	SIG GND

Applesoft BASIC

The Apple II computer, using Applesoft BASIC, does not have different types of PRINT statements for the screen and printer. You must add commands to your programs that direct the output of the PRINT statements to the printer. To direct output to the printer (with the interface board in slot #1) you must use the PR# 1 command. Depending on the version of Applesoft BASIC that you are using this command can take various forms. It is usually one of the following:

```
1Ø PR# 1
or
1Ø PRINT "<Ctrl-D>PR#1"
or
1Ø PRINT CHR$(4) "PR#1"
```

To return output to the screen, the command is PR# 0, in the same form that works for PR# 1.

To allow line lengths longer than the Apple II usually uses you must add the following statement to your programs:

```
2Ø PRINT CHR$(9) "255N"
```

This allows lines of any length to be sent to the printer and is especially important for dot graphics. (The number 255 in the BASIC statement above could be replaced by any number from 0 to 255 and would set the line length to that value.)

Two codes are a particular problem on the Apple II: CHR$(7) and CHR$(9). The computer will not send these codes to Radix. Try to avoid using these in dot graphics programs.

The Apple II computer uses CHR$(9) as a printer initialization code. It won't send it on to the printer. There is a way to bypass this problem, however. You can change the printer initialization code to a value other than CHR$(9) like this:

```
PR#1
PRINT CHR$(9); CHR$(1)
```

This makes CHR$(1) the printer initialization code (and transfers the problems to *that* code) and allows you to use Radix's tabs.

There is one more way to sneak problem codes past the Apple II's operating system and that's to poke the codes directly to the output port. To send ASCII code 9, for example, you could do this:

```
1ØØ N = 9
11Ø IF PEEK(496Ø1)>127 THEN 11Ø
12Ø POKE 49296,N
```

Line 110 checks the printer's status, and when it's okay, line 120 pokes the code to the printer.

Listing programs

To make a listing of your BASIC programs on Radix from your Apple II computer you must take the following steps:

1. Be sure that the program that you wish to list is in the memory of the Apple II.
2. Direct the output to the printer by typing PR#1.
3. Type LIST to start the listing.
4. When the listing is finished, type PR#0 to redirect the output to the screen.

Program Listings

Following are program listings in Applesoft BASIC for the main utility programs used in the tutorial section of this book.

Download character editing utility

```
10  DIM Z(8,12),MM(11)
11 PF$ =  CHR$ (27) + "X" +  CHR$ (0)
12 PN$ =  CHR$ (27) + "X" +  CHR$ (1)
13 NF$ =  CHR$ (27) + "$" +  CHR$ (0)
14 NR$ =  CHR$ (27) + "$" +  CHR$ (1)
15 CS$ = "*":SC$ = "@"
16 BEEP$ =  CHR$ (7)
18 AS = 33:PP$ = "$":ESC$ =  CHR$ (27)
20  GOSUB 1910
260  REM
265  FOR I = 1 TO 11:MM(I) = 0: NEXT I
270  VTAB 3: HTAB 6: PRINT CS$;
275  VTAB 23: HTAB 1
280  GET A$
290  IF A$ = "J" THEN  GOSUB 390: GOTO 370
300  IF A$ = "K" THEN  GOSUB 410: GOTO 370
310  IF A$ = "M" THEN  GOSUB 430: GOTO 370
320  IF A$ = "I" THEN  GOSUB 450: GOTO 370
330  IF A$ =  CHR$ (13) THEN  GOSUB 470: GOTO 370
340  IF A$ =  CHR$ (32) THEN  GOSUB 490: GOTO 370
350  IF A$ =  CHR$ (27) THEN  HOME : END
360  IF A$ = "+" THEN  GOSUB 3000: GOTO 370
362  IF A$ = "-" THEN  GOSUB 3100: GOTO 370
363  IF A$ = "A" THEN  GOSUB 3500: GOTO 370
364  IF A$ = "D" THEN  GOSUB 3200: GOTO 370
365  IF A$ = "P" THEN  GOSUB 3300: GOTO 370
366  IF A$ = "C" THEN  GOSUB 1910: GOTO 260
367  IF A$ = "R" THEN  GOSUB 3700: GOTO 370
370  GOTO 280
```

```
380  RETURN
390  GOSUB 1000:Y = Y - 2:H = H - 1: IF Y < 1 THEN
   PRINT  CHR$ (7);:Y = 1:H = 1
400  GOSUB 1050: RETURN
410  GOSUB 1000:Y = Y + 2:H = H + 1: IF Y > 21 THEN
   PRINT  CHR$ (7);:Y = 21:H = 11
420  GOSUB 1050: RETURN
430  GOSUB 1000:X = X + 2:G = G + 1: IF X > 13 THEN
   PRINT  CHR$ (7);:X = 13:G = 7
440  GOSUB 1050: RETURN
450  GOSUB 1000:X = X - 2:G = G - 1: IF X < 1 THEN
   PRINT  CHR$ (7);:X = 1:G = 1
460  GOSUB 1050: RETURN
470  IF Z(G,H - 1) = 1 OR Z(G,H + 1) = 1 THEN  PRINT
   CHR$ (7);: RETURN
480 Z(G,H) = 1: INVERSE : VTAB X + 2: HTAB Y + 5: PRINT
   SC$;: NORMAL : GOSUB 4000: RETURN
490 Z(G,H) = 0: NORMAL : VTAB X + 2: HTAB Y + 5: PRINT
   CS$;: GOSUB 4000: RETURN
900 X = 1:Y = 1:G = 1:H = 1
901  HOME
902  FOR I = 2 TO 16 STEP 2: VTAB I: HTAB 5: FOR J = 1
   TO 23: PRINT "-";: NEXT J: PRINT : NEXT I
904  FOR J = 3 TO 16 STEP 2: VTAB J: FOR I = 5 TO 27
   STEP 2: HTAB I: PRINT "!";: NEXT I: PRINT : NEXT J
905 K = 1: VTAB 1: HTAB 5
906  FOR K = 1 TO 11: PRINT K;" ";: NEXT K
907 K = 0
908  FOR V = 3 TO 15 STEP 2: VTAB V: HTAB 2: PRINT 2 ^
   K:K = K + 1: NEXT V
909  VTAB 17: FOR I = 1 TO 11: HTAB 4 + I * 2: PRINT
   "0";: NEXT I
910  VTAB 1: HTAB 30: PRINT "CURSOR"
912  VTAB 2: HTAB 29: PRINT "MOVEMENT"
914  VTAB 3: HTAB 29: PRINT "<I> UP"
916  VTAB 4: HTAB 29: PRINT "<M> DOWN"
918  VTAB 5: HTAB 29: PRINT "<J> LEFT"
920  VTAB 6: HTAB 29: PRINT "<K> RIGHT"
922  VTAB 7: HTAB 29: PRINT "<RET> INSERT"
924  VTAB 8: HTAB 29: PRINT "<SPACE> DEL"
926  VTAB 9: HTAB 29: PRINT "<A> ASCII"
928  VTAB 10: HTAB 29: PRINT "<P> PRINT"
930  VTAB 11: HTAB 29: PRINT "<C> CLEAR"
932  VTAB 12: HTAB 29: PRINT "<R> COPY ROM"
```

```
934  VTAB 13: HTAB 29: PRINT "<+> WIDER"
936  VTAB 14: HTAB 29: PRINT "<-> NARROWER"
938  VTAB 15: HTAB 29: PRINT "<D> DESCENDER"
940  VTAB 16: HTAB 29: PRINT "<ESC> EXIT"
950  FOR I = 1 TO 11: FOR J = 1 TO 7:Z(J,I) = 0: NEXT J:
   NEXT I
960  RETURN
1000  IF Z(G,H) = 0 THEN  VTAB X + 2: HTAB Y + 5: PRINT
   " ";
1010  IF Z(G,H) = 1 THEN  VTAB X + 2: HTAB Y + 5: PRINT
   SC$;
1015  VTAB 23: HTAB 1
1020  RETURN
1050  IF Z(G,H) = 1 THEN  INVERSE : VTAB X + 2: HTAB Y +
   5: PRINT CS$;: NORMAL
1060  IF Z(G,H) = 0 THEN  NORMAL : VTAB X + 2: HTAB Y +
   5: PRINT CS$;: NORMAL
1065  VTAB 23: HTAB 1
1070  RETURN
1910  REM  CLEAR CURRENT CHARACTER
1920 PW% = 11:DS = 0
1930  FOR H = 1 TO 11:MM(H) = 0: NEXT H
1935  GOSUB 900
1940  GOSUB 2200:  RETURN
2080  REM  BUILD COMMAND STRING
2085 RC$ = ESC$ + "*" +  CHR$ (1)
2090 RC$ = RC$ +  CHR$ (AS) +  CHR$ (DS * 16 + PW%)
2095  FOR I = 1 TO 11:RC$ = RC$ +  CHR$ (MM(I)): NEXT I
2096  RETURN
2200  REM
2210  VTAB 20: HTAB 1: PRINT "ASCII CODE = ";AS;
2220  PRINT "("; CHR$ (AS);")";
2230  VTAB 20: HTAB 25: PRINT "DESCENDER= ";DS;
2250  FOR I = 8 TO 19: VTAB 22: HTAB I: PRINT " ";: NEXT
   I
2260  VTAB 22: HTAB 1: PRINT "WIDTH: ";: FOR I = 1 TO
   PW%: PRINT "*";: NEXT I
2270  VTAB 23: HTAB 1
2280  RETURN
3000  REM WIDER
3010  IF PW% = 11 THEN  PRINT BEEP$;: RETURN
3020 PW% = PW% + 1
3030  GOSUB 2200
```

```
3040  RETURN
3100  REM NARROWER
3110  IF PW% = 4 THEN  PRINT BEEP$;: RETURN
3120 PW% = PW% - 1
3130  GOSUB 2200
3140  RETURN
3200  REM DESCENDER
3210 DS =  ABS (1 - DS)
3220  GOSUB 2200: RETURN
3300  REM  PRINT
3310  GOSUB 2080
3320  PR# 1
3325  PRINT  CHR$ (9);"255N"
3327  PRINT  CHR$ (27);"@"
3330  PRINT "ASCII CODE = ";AS: PRINT
3335  PRINT RC$
3345  PRINT  CHR$ (15);"CONDENSED"
3350  PRINT NR$: FOR I = 1 TO 21: PRINT  CHR$ (AS);:
   NEXT I: PRINT
3355  PRINT NF$
3360  PRINT  CHR$ (27); "B"; CHR$ (2); "ELITE"
3365  PRINT NR$: FOR I = 1 TO 15: PRINT  CHR$ (AS);:
   NEXT I: PRINT
3370  PRINT NF$
3375  PRINT  CHR$ (27);"B"; CHR$ (1);"PICA"
3378  PRINT NR$: FOR I = 1 TO 12: PRINT  CHR$ (AS);:
   NEXT I: PRINT
3379  PRINT NF$
3380  PRINT  CHR$ (27);"W"; CHR$ (1);"EXPANDED"
3384  PRINT NR$;: FOR I = 1 TO 6: PRINT  CHR$ (AS);:
   NEXT I
3385  PRINT  CHR$ (27);"W" ;  CHR$ (0)
3386  PRINT NF$
3387  PRINT : PRINT "CHARACTER SET ": PRINT NR$: FOR I =
   33 TO 126
3388  PRINT  CHR$ (I);: NEXT I: PRINT : PRINT NF$: PRINT
3390  PRINT : PRINT "PROPORTIONAL"
3392  PRINT PN$;: FOR I = 1 TO 15: PRINT  CHR$ (AS);:
   NEXT I: PRINT PF$
3393  PRINT : PRINT : PRINT "CHARACTER SET
   ..PROPORTIONAL": PRINT PN$: FOR I = 33 TO 126: PRINT
   CHR$ (I);: NEXT I: PRINT : PRINT PF$: PRINT
3394  PRINT "USE THIS DATA STATEMENT TO DOWNLOAD THIS
   CHARACTER."
```

```
3395  PRINT "DATA 27";
3396  FOR I = 2 TO  LEN (RC$)
3397  PRINT ","; STR$ ( ASC ( MID$ (RC$,I,1)));
3398  NEXT I: PRINT : PRINT : PRINT :
3399  PR# Ø: RETURN
35ØØ  REM  ASCII CODE
351Ø  VTAB 23: HTAB 1
352Ø  INPUT "ENTER ASCII (33-126) ";AS
353Ø  IF AS < 33 OR AS > 126 THEN  PRINT BEEP$;: GOTO
   351Ø
3535  VTAB 23: FOR I = 1 TO 39: HTAB I: PRINT " ";: NEXT
   I
354Ø  GOSUB 22ØØ: RETURN
37ØØ  REM  COPY ROM
371Ø  PR# 1
3715  PRINT  CHR$ (9);"255N"
372Ø  PRINT ESC$;"*"; CHR$ (Ø);
373Ø  PR# Ø
374Ø  RETURN
4ØØØ  REM  CALCULATE A COLUMN VALUE
4Ø1Ø MM(H) = Ø: FOR J = 1 TO 7
4Ø2Ø MM(H) = MM(H) + Z(J,H) * 2 ^ (J - 1)
4Ø3Ø  NEXT J: GOSUB 41ØØ: RETURN
41ØØ  REM  PRINT A COLUMN VALUE
41Ø3  FOR I = 1 TO 3: VTAB 16 + I: HTAB 4 + H * 2: PRINT
   " ";: NEXT I
41Ø5 LV$ =  STR$ (MM(H))
41Ø6  FOR I = 1 TO LEN (LV$)
41Ø7  VTAB 16 + I: HTAB 4 + H * 2: PRINT  MID$
   (LV$,I,1);: NEXT I
412Ø  VTAB 23: HTAB 1: RETURN
```

Piechart program

```
4  HOME
5  PRINT "Please Stand By"
1Ø A = 768
2Ø  FOR I = A TO A + 12
3Ø  READ B
35  POKE I,B
4Ø  NEXT I
5Ø  DATA  32,74,255,165,25Ø,5,251
6Ø  DATA  133,252,32,63,255,96
1ØØ  REM  PIECHART
```

```
110  DIM BIT%(190,36),A$(36),PCT%(25),TXT$(48),PTXT$(25)
120 ES$ =  CHR$ (27):LF$ = CHR$ (10)
130 FF$ =  CHR$ (12):VT$ = CHR$ (11)
140 EM$ = ES$ + "E":CE$ = ES$ + "F"
145 RF$ =  CHR$ (27) +  CHR$ (12)
150  FOR I = 1 TO 148:SP$ = SP$ +  CHR$ (0): NEXT I
160  FOR I = 1 TO 79:SS$ = SS$ + " ": NEXT I
1000  REM  SET PROGRAM CONSTANTS
1010 MASK%(1) = 64:MASK%(4) = 8
1020 MASK%(2) = 32:MASK%(5) = 4
1030 MASK%(3) = 16:MASK%(6) = 2
1040 LX = 20:LY = 20
1050 XFAC = 190 / LX:YFAC = 216 / LY
1060  FOR I = 0 TO 48
1070 TXT$(I) = SS$
1080  NEXT I
1090  GOSUB 7000
1092  HOME : PRINT : PRINT : PRINT : PRINT
1093  PRINT "THIS PROGRAM TAKES ABOUT"
1094  PRINT "2 MINUTES TO RUN. PLEASE"
1095  PRINT "TURN ON YOUR PRINTER AND"
1096  PRINT "STAND BY..............."
1097  PRINT : PRINT : PRINT
1098  FOR I = 1 TO 31: PRINT "0";: NEXT I
1099  PRINT " ": PRINT " "
1100  FOR I = 1 TO NP%: PRINT "0";: NEXT I
1110  PRINT " "
1120  VTAB 12: HTAB 1
2000  REM  PLOT CURVE
2010 RAD = 9
2020 X1 = 19:Y1 = 10
2030  FOR ANG = 0 TO 360 STEP 12
2040 R1 = ANG * 6.28 / 360
2050 X2 = RAD *  COS (R1) + 10:Y2 = RAD *  SIN (R1) + 10
2060  GOSUB 4000
2070  NEXT ANG
2075  VTAB 14: HTAB 1
2080  FOR PI = 1 TO NP%
2090 X1 = 10:Y1 = 10
2100 TP% = TP% + PCT%(PI)
2110 ANG = 360 * TP% * .01
2120 R1 = ANG * 6.28 / 360
2130 X2 = RAD *  COS (R1) + 10:Y2 = RAD *  SIN (R1) + 10
```

```
2140  GOSUB 4000
2150  GOSUB 6000
2160  NEXT PI
3000  REM  SEND BIT IMAGE MAP TO PRINTER
3090  PR# 1
3100  PRINT  CHR$ (9); "0N"
3110 X = (40 -  LEN (TI$) / 2)
3120  FOR I = 1 TO X: PRINT " ";: NEXT I
3130  PRINT EM$;TI$;CE$;LF$
3140  PRINT VT$;VT$;VT$
3150  PRINT ES$;"A"; CHR$ (6)
3160  FOR I = 0 TO 48: PRINT TXT$(I): NEXT I
3165  PRINT RF$;VT$;VT$;VT$;
3166  PRINT LF$;LF$;LF$;LF$;LF$;LF$
3170  FOR ROW = 0 TO 35
3180  PRINT ES$;"K"; CHR$ (82); CHR$ (1);SP$;
3190  FOR COL = 1 TO 190: PRINT  CHR$ (BIT%(COL,ROW));:
   NEXT
3192  PRINT " "
3210  NEXT ROW
3250  PRINT ES$;"2";FF$
3255  PR# 0
3257  HOME
3260  END
4000  REM   DRAW A LINE FROM X1,Y1 TO X2,Y2
4010 XL = X2 - X1:YL = Y2 - Y1
4020 NX =  ABS (XL * XFAC):NY =  ABS (YL * YFAC)
4030  IF NX < NY THEN NX = NY
4040 NS% =  INT (NX + 1)
4050 DX = XL / NS%:DY = YL / NS%
4060  FOR I = 1 TO NS%
4070 X1 = X1 + DX:Y1 = Y1 + DY
4080  GOSUB 5000
4090  NEXT I
4095  PRINT "*";
4100  RETURN
5000  REM  PLOT A POINT AT X1,Y1
5010 XX = X1 * XFAC:YY = Y1 * YFAC
5020 COL =  INT (XX) + 1
5030 ROW =  INT (YY / 6)
5040 XIT% =  INT (YY - (6 * ROW)) + 1
5042  POKE 250,BIT%(COL,ROW)
5044  POKE 251,MASK%(XIT%)
5046  CALL 768
```

```
5050 BIT%(COL,ROW) =  PEEK (252)
5060  RETURN
6000  REM
6010 MA% = (ANG + PA%) / 2
6020 R1 = MA% * 6.28 / 360
6030 X3 =  INT (20 *  SIN (R1)):Y3 =  INT (22 *  COS
   (R1))
6040 X4 = 22 + X3:Y4 = 40 + Y3
6045  IF (MA% > 70 AND MA% < 110) THEN  GOSUB 6300: GOTO
   6070
6047  IF (MA% > 250 AND MA% < 290) THEN  GOSUB 6300:
   GOTO 6070
6050  IF MA% > 270 OR MA% < 90 THEN GOSUB 6100: GOTO
   6070
6060  GOSUB 6200
6070 PA% = ANG
6080  RETURN
6100 MM$ = TXT$(X4)
6102 LL$ =  LEFT$ (MM$,Y4)
6104 PP =  LEN (PTXT$(PI))
6106 RR$ =  RIGHT$ (MM$,80 - (Y4 + PP))
6108 TXT$(X4) = LL$ + PTXT$(PI) + RR$
6110  RETURN
6200 MM$ = TXT$(X4)
6202 PP =  LEN (PTXT$(PI))
6204 LL$ =  LEFT$ (MM$,(Y4 - PP))
6206 RR$ =  RIGHT$ (MM$,(80 - Y4))
6208 TXT$(X4) = LL$ + PTXT$(PI) + RR$
6210  RETURN
6300 MM$ = TXT$(X4)
6310 PP =  INT ( LEN (PTXT$(PI)) / 2)
6320 LL$ =  LEFT$ (MM$,(Y4 - PP))
6330 RR$ =  RIGHT$ (MM$,(80 - Y4))
6340 TXT$(X4) = LL$ + PTXT$(PI) + RR$
6350  RETURN
7000  REM
7010  HOME : PRINT : PRINT : PRINT
7020  INPUT "ENTER TITLE FOR CHART ";TI$
7025  IF  LEN (TI$) < = 40 THEN 7030
7027  PRINT  CHR$ (7);"TITLE TOO LONG - 40 CHAR. MAX ":
   GOTO 7000
7030 AS% = 0:AL% = 100
7035  FOR I = 1 TO 24
7040  HOME
```

```
7050  PRINT "TOTAL SO FAR     : ";AS%
7060  PRINT "TOTAL REMAINING : ";AL%
7070  INPUT "ENTER  %  FOR FIELD ";PCT%(I)
7080  IF PCT%(I) > AL% OR PCT%(I) = 0 THEN PCT%(I) = AL%
7090 AL% = AL% - PCT%(I)
7100 AS% = AS% + PCT%(I)
7110  INPUT "ENTER DESCRIPTION OF FIELD : ";PTXT$(I)
7120  IF  LEN (PTXT$(I)) > 15 THEN  PRINT "FIELD TOO
   LONG - 15 CHAR. MAX": GOTO 7110
7130  IF AL% = 0 THEN  GOTO 7200
7140  NEXT I
7200 NP% = I
7210  IF NP% = 1 THEN 7030
7220  HOME
7230  RETURN
```

Printer setup utility

```
10  REM  PROGRAM TO SET UP RADIX
20 BEEP$ =  CHR$ (7)
40 ESC$ =  CHR$ (27):TB = 5: DIM TBS(256)
80  HOME
90 TI$ = "MAIN MENU"
100  GOSUB 2560
110  PRINT  TAB( TB);"0. EXIT "
120  PRINT  TAB( TB);"1. SELECT  CHARACTER SET."
130  PRINT  TAB( TB);"2. SELECT PRINTING MODES"
140  PRINT  TAB( TB);"3. SELECT PITCH "
150  PRINT  TAB( TB);"4. SELECT LINE SPACING"
160  PRINT  TAB( TB);"5. SET MARGINS, TABS & FORMS"
170  GOSUB 2650
180  IF S < 0 OR S > 5 THEN  PRINT BEEP$;: GOTO 170
190  IF S = 0 THEN HOME : END
200  ON S GOSUB 220,490,360,1410,650
210  GOTO 80
220  REM  SUBROUTINE TO DISPLAY CHARACTER SET MENU
240 TI$ = "CHARACTER SET MENU"
250  GOSUB 2560
260  PRINT  TAB( TB);"0. RETURN TO MAIN MENU"
270  PRINT  TAB( TB);"1. SELECT NLQ CHARACTER SET"
280  PRINT  TAB( TB);"2. CANCEL NLQ CHARACTER SET"
290  PRINT  TAB( TB);"3. SELECT ITALIC CHARACTER SET"
300  PRINT  TAB( TB);"4. CANCEL ITALIC CHARACTER SET"
310  GOSUB 2650
```

```
320  IF S < Ø OR S > 4 THEN  PRINT BEEP$;: GOTO 31Ø
33Ø  IF S = Ø THEN  RETURN
34Ø  ON S GOSUB 131Ø,136Ø,18ØØ,184Ø
35Ø  GOTO 22Ø
36Ø  REM   DISPLAY PITCHES MENU
38Ø TI$ = "PITCHES MENU"
39Ø  GOSUB 256Ø
4ØØ  PRINT  TAB( TB);"Ø. RETURN TO MAIN MENU"
41Ø  PRINT  TAB( TB);"1. SELECT PICA PITCH"
42Ø  PRINT  TAB( TB);"2. SELECT ELITE PITCH"
43Ø  PRINT  TAB( TB);"3. SELECT CONDENSED PITCH"
44Ø  GOSUB 265Ø
45Ø  IF S < Ø OR S > 3 THEN  PRINT BEEP$;: GOTO 44Ø
46Ø  IF S = Ø THEN  RETURN
47Ø  ON S GOSUB 83Ø,88Ø,93Ø
48Ø  GOTO 36Ø
49Ø  REM DISPLAY PRINTING MODE
5ØØ TI$ = "PRINTING MODES MENU"
51Ø  GOSUB 256Ø
53Ø  PRINT  TAB( TB);"Ø. RETURN TO MAIN MENU"
54Ø  PRINT  TAB( TB);"1. SELECT EXPANDED MODE"
55Ø  PRINT  TAB( TB);"2. CANCEL EXPANDED MODE"
56Ø  PRINT  TAB( TB);"3. SELECT EMPHASIZED MODE"
57Ø  PRINT  TAB( TB);"4. CANCEL EMPHASIZED MODE"
58Ø  PRINT  TAB( TB);"5. SELECT DOUBLE STRIKE MODE"
59Ø  PRINT  TAB( TB);"6. CANCEL DOUBLE STRIKE MODE"
6ØØ  GOSUB 2650
61Ø  IF S < Ø OR S > 6 THEN  PRINT BEEP$;: GOTO 6ØØ
62Ø  IF S = Ø THEN  RETURN
63Ø  ON S GOSUB 17ØØ,175Ø,24ØØ,244Ø,248Ø,252Ø
64Ø  GOTO 49Ø
65Ø  REM
66Ø  REM  DISPLAY MARGIN, TABS AND FORMS
67Ø TI$ = "MARGINS, TABS & FORMS MENU"
68Ø  GOSUB 256Ø
69Ø  PRINT  TAB( TB);"Ø. RETURN TO MAIN MENU"
7ØØ  PRINT  TAB( TB);"1. SET HORIZONTAL TABS"
71Ø  PRINT  TAB( TB);"2. SET VERTICAL TABS"
72Ø  PRINT  TAB( TB);"3. SET LEFT MARGIN"
73Ø  PRINT  TAB( TB);"4. SET RIGHT MARGIN"
74Ø  PRINT  TAB( TB);"5. SET TOP MARGIN"
75Ø  PRINT  TAB( TB);"6. SET BOTTOM MARGIN"
76Ø  PRINT  TAB( TB);"7. CANCEL TOP & BOTTOM MARGINS"
77Ø  PRINT  TAB( TB);"8. SET PAGE LENGTH"
```

```
780  GOSUB 2650
790  IF S < 0 OR S > 8 THEN  PRINT BEEP$;: GOTO 780
800  IF S = 0 THEN  RETURN
810  ON S GOSUB 2050,2360,980,1060,1130,1210,1280,1880
820  GOTO 650
830  REM  SELECT PICA
850 S$ = ESC$ + "B" +  CHR$ (1)
860  GOSUB 2730
870  RETURN
880  REM  SELECT ELITE
890 S$ = ESC$ + "B" +  CHR$ (2)
900  GOSUB 2730
910  RETURN
930  REM  SELECT CONDENSED
940 S$ = ESC$ + "B" +  CHR$ (3)
960  GOSUB 2730
970  RETURN
980  REM  SET LEFT MARGIN
1000  GOSUB 2770
1010  INPUT "ENTER NEW LEFT MARGIN (1-255) ";X
1020  IF X < 1 OR X > 255 THEN  PRINT BEEP$;: GOTO 1000
1030 S$ = ESC$ + "M" +  CHR$ (X)
1040  GOSUB 2730
1050  RETURN
1060  REM  SET RIGHT MARGIN
1080  GOSUB 2770
1090  INPUT "ENTER NEW RIGHT MARGIN (1-255) ";X
1100  IF X < 1 OR X > 255 THEN  PRINT BEEP$;: GOTO 1080
1110 S$ = ESC$ + "Q" +  CHR$ (X)
1120  GOSUB 2730: RETURN
1130  REM  SET TOP MARGIN
1150  GOSUB 2770
1160  INPUT "ENTER NEW TOP MARGIN (1-16) ";X
1170  IF X < 1 OR X > 16 THEN  PRINT BEEP$;: GOTO 1150
1180 S$ = ESC$ + "R" +  CHR$ (X)
1190  GOSUB 2730
1200  RETURN
1210  REM  SET BOTTOM MARGIN
1230  GOSUB 2770
1240  INPUT "ENTER NEW BOTTOM MARGIN (1-127) ";X
1250  IF X < 1 OR X > 127 THEN  PRINT BEEP$;: GOTO 1230
1260 S$ = ESC$ + "N" +  CHR$ (X)
1270  GOSUB 2730: RETURN
1280  REM CANCEL TOP & BOTTOM MARGIN
```

```
1300 S$ = ESC$ + "O": GOSUB 2730: RETURN
1310  REM  SELECT NLQ
1330 S$ = ESC$ + "B" +  CHR$ (4)
1340  GOSUB 2730: RETURN
1360  REM  CANCEL NLQ
1380 S$ = ESC$ + "B" +  CHR$ (5)
1390  GOSUB 2730: RETURN
1410  REM  SELECT LINE SPACING
1430 TI$ = "LINE SPACING MENU"
1440  GOSUB 2560
1450  PRINT  TAB( TB);"0. RETURN TO MAIN MENU"
1460  PRINT  TAB( TB);"1. SELECT 1/6 INCH LINE SPACING"
1470  PRINT  TAB( TB);"2. SELECT 1/8 INCH LINE SPACING"
1480  PRINT  TAB( TB);"3. SELECT 7 DOT GRAPHICS SPACING"
1490  PRINT  TAB( TB);"4. SELECT N/144 INCH SPACING"
1500  GOSUB 2650
1510  IF S < 0 OR S > 4 THEN  PRINT BEEP$;: GOTO 1500
1520  IF S = 0 THEN  RETURN
1530  ON S GOSUB 1550,1580,1610,1640
1540  GOTO 1410
1550  REM  SELECT 1/6 INCH LINE SPACING
1570 S$ = ESC$ + "2": GOSUB 2730: RETURN
1580  REM  SELECT 1/8 INCH LINE SPACING
1600 S$ = ESC$ + "0": GOSUB 2730: RETURN
1610  REM  SELECT 7 DOT GRAPHICS SPACING
1630 S$ = ESC$ + "1": GOSUB 2730: RETURN
1640  REM  SELECT N/144 INCH LINE SPACING
1660  GOSUB 2770
1670  INPUT "ENTER LINE SPACE (0-255) ";X
1680  IF X < 0 OR X > 255 THEN  PRINT BEEP$;: GOTO 1660
1690 S$ = ESC$ + "3" + CHR$ (X): GOSUB 2730: RETURN
1700  REM  SELECT EXPANDED
1720 S$ = ESC$ + "W" + CHR$ (1)
1730  GOSUB 2730
1740  RETURN
1750  REM  CANCEL EXPANDED
1770 S$ = ESC$ + "W" +  CHR$ (0)
1780  GOSUB 2730
1790  RETURN
1800  REM  SELECT ITALIC
1820 S$ = ESC$ + "4": GOSUB 2730
1830  RETURN
1840  REM  CANCEL ITALIC
1860 S$ = ESC$ + "5": GOSUB 2730
```

```
1870  RETURN
1880  REM  SET PAGE LENGTH
1900  GOSUB 2770
1910  PRINT "PAGE LENGTH IN INCHES OR LINES (I,L)?"
1920  PRINT  TAB( TB);
1930  GET A$
1940  IF A$ = "I" THEN 1970
1950  IF A$ = "L" THEN 2010
1960  PRINT BEEP$;: GOTO 1930
1970  INPUT "LENGTH OF PAGE IN INCHES (1-32) ";X
1980  IF X < 1 OR X > 32 THEN PRINT BEEP;: GOTO 1900
1990 S$ = ESC$ + "C" +  CHR$ (0) +  CHR$ (X)
2000  GOSUB 2730: RETURN
2010  INPUT "LENGTH OF PAGE IN LINES (1-127) ";X
2020  IF X < 1 OR X > 127 THEN  PRINT BEEP$;: GOTO 1900
2030 S$ = ESC$ + "C" +  CHR$ (X)
2040  GOSUB 2730: RETURN
2050  REM  SET HORIZONTAL TAB
2070 S$ = ESC$ + "D":MAX = 255: GOSUB 2080: RETURN
2080  REM  SET TABS
2100  GOSUB 2770
2110  PRINT "WOULD YOU LIKE TO SET THE TABS IN"
2120  PRINT  TAB( TB);"REGULAR INTERVALS, OR SPECIFY"
2130  PRINT  TAB( TB);"EACH ONE INDIVIDUALLY (R,I) "
2140  GET A$
2150  IF A$ = "R" THEN 2300
2160  IF A$ = "I" THEN 2180
2170  PRINT BEEP$;: GOTO 2080
2180  PRINT :I = 2:TBS(1) =  - 1
2190  PRINT  TAB( TB);"ENTER THE LIST OF TABS, IN "
2200  PRINT  TAB( TB);"ASCENDING ORDER. NO MORE THAN
   ";MAX;"."
2210  PRINT  TAB( TB): INPUT "ENTER TAB ";TBS(I)
2220  IF TBS(I) < 0 OR TBS(I) > 255 THEN 2170
2230  IF TBS(I) = 0 THEN I = 1: GOTO 2270
2240  IF TBS(I) < = TBS(I - 1) THEN 2170
2250 I = I + 1: IF I > MAX THEN 2170
2260  GOTO 2210
2270 I = I + 1
2280 S$ = S$ +  CHR$ (TBS(I)): IF TBS(I) < > 0 THEN 2270
2285  GOSUB 2730
2290  RETURN
2300  PRINT : PRINT  TAB( TB);: INPUT "ENTER INTERVAL
   ";X
```

```
2310  IF X < 0 OR X > 255 THEN  PRINT BEEP$;: GOTO 2080
2320  FOR I = 1 TO 255 STEP X
2330 MAX = MAX - 1: IF MAX = 0 THEN 2350
2340 S$ = S$ +  CHR$ (I): NEXT I
2350 S$ = S$ +  CHR$ (0): GOSUB 2730: RETURN
2360  REM  VERTICAL TABS
2380 S$ = ESC$ + "P":MAX = 20: GOSUB 2080
2390  RETURN
2400  REM  SELECT EMPHASIZED
2420 S$ = ESC$ + "E": GOSUB 2730
2430  RETURN
2440  REM  CANCEL EMPHASIZED
2460 S$ = ESC$ + "F": GOSUB 2730
2470  RETURN
2480  REM  DOUBLE-STRIKE
2500 S$ = ESC$ + "G": GOSUB 2730
2510  RETURN
2520  REM  CANCEL DOUBLE-STRIKE
2540 S$ = ESC$ + "H": GOSUB 2730
2550  RETURN
2560  REM  PRINT A MENU TITLE
2570  HOME
2580  PRINT : PRINT : PRINT
2590  PRINT  TAB( 6);"---RADIX PRINTER SETUP ---"
2600  PRINT
2610  PRINT  TAB( (40 - LEN (TI$)) / 2);TI$
2620  PRINT : PRINT
2630  RETURN
2650  REM  SELECTION
2660  VTAB 19: HTAB 10: PRINT "HIT <P> FOR SAMPLE PRINT"
2665  VTAB 21: HTAB 10: PRINT "SELECTION ";
2670  GET C$
2675  IF C$ = "P" THEN  GOSUB 3000: GOTO 2650
2680  IF C$ < "0" OR C$ > "9" THEN  PRINT BEEP$;: GOTO
   2670
2690 S = VAL (C$)
2700  VTAB 20:
2710  FOR H = 10 TO 40: HTAB H: PRINT " ";: NEXT H
2720  RETURN
2730  REM  OUTPUT COMMAND STRING
2750  PR# 1
2755  PRINT S$;
2758  PR# 0
2760  RETURN
```

```
2770  REM  CLEAR SCREEN AND POSITION CURSOR
2790  HOME : VTAB 10: HTAB TB: RETURN
3000  REM  PRINT
3005  PR# 1
3007  PRINT  CHR$ (9);"255N"
3010  FOR I = 1 TO 4: FOR J = 33 TO 126
3020  PRINT  CHR$ (J);: NEXT J
3030  PRINT : NEXT I
3040  PR# 0
3050  RETURN
```

Appendix D

TRS-80 Computers

All that's required to connect Radix to your TRS-80 is a cable. It is available at your Radix dealer.

Setting the Switches

When connecting Radix to a TRS-80 we recommend that you set the DIP switches in Radix as shown below. Since you will be using the parallel interface, the settings of switch B have no effect.

Table D-1
Recommended DIP switch settings for TRS-80

Switch	Setting	Function
A-1	ON	11 inch page size
A-2	ON	Normal print density
A-3	ON	10 CPI pitch
A-4	ON	Normal characters
A-5	ON	1/6 inch line feed
A-6	ON	U.S.A. Character set
A-7	ON	
A-8	ON	
C-1	ON	Paper-out detector active
C-2	OFF	Parallel interface
C-3	OFF	8-bit interface
C-4	ON	Auto line feed

TRS-80 BASIC

You may have to initialize your Model II to direct LPRINT statements to the printer. Use the SYSTEM "FORMS" command to do it.

Table D-2
TRS-80 Model I parallel cable

Radix			TRS-80 Model I	
Pin No.	**Function**		**Pin No.**	**Function**
1	STROBE	———	1	STROBE
2	D1	———	3	D1
3	D2	———	5	D2
4	D3	———	7	D3
5	D4	———	9	D4
6	D5	———	11	D5
7	D6	———	13	D6
8	D7	———	15	D7
9	D8	———	17	D8
11	BUSY	———	21	READY

Table D-3
TRS-80 Model II parallel cable

Radix			TRS-80 Model II	
Pin No.	**Function**		**Pin No.**	**Function**
1	STROBE	———	1	STROBE
2	D1	———	3	D1
3	D2	———	5	D2
4	D3	———	7	D3
5	D4	———	9	D4
6	D5	———	11	D5
7	D6	———	13	D6
8	D7	———	15	D7
9	D8	———	17	D8
10	ACK	———	19	ACK
11	BUSY	———	21	BUSY

TRS-80 uses another version of Microsoft Basic. Most of the programs in this book will work just as they are, but the TRS-80 does have a few unique "problem codes." They are 0, 10, 11, and 12. None of these are passed properly to the printer.

You can bypass the TRS-80's BASIC and send these codes directly to the printer with the following short routine. The variable N must be set equal to the code that you wish to pass (in our example it's 0).

```
90 N = 0
100 IF PEEK(14312)<>63 THEN 100
110 POKE 14312,N
```

Or you can use this special printer driver that will solve all your problems. Just run this program first, and then any codes sent by a BASIC program will be sent directly to the printer. This program is for the TRS-80 Model III.

```
5 REM  DRIVER FOR TRS-80 III
10 AD=16571
20 FOR I=0 TO 14
30 READ A:POKE AD+I,A
40 NEXT I
50 POKE 16422,187
60 POKE 16423,64
70 DATA 33,232,55,203,126,32,252,33,17,
   0,57,126,211,251,201
80 END
```

And here is a version for the TRS-80 Model I.

```
5 REM  DRIVER FOR THE TRS-80 I
10 AD=16571
20 FOR I=0 TO 15
30 READ A:POKE AD+I,A
40 NEXT I
50 POKE 16422,187
60 POKE 16423,64
70 DATA 33,232,55,203,126,32,252,33,17,
   0,57,126,50,232,55,201
80 END
```

Listing programs

To list a BASIC program that is in your TRS-80's memory on Radix, type LLIST. This directs the listing to the printer instead of the screen.

Program Listings

Download character editing utility

```
10 'Program to allow editing down-load characters.
20 'for the RADIX printer.
30 '
40 'Initialization.
45 CLEAR 1000
50 DIM Z(8,12),MM(11)
60 AS=33 : ESC$ = CHR$(27)
65 PN$=ESC$+"X"+CHR$(1):PF$=ESC$+"X"+CHR$(0)
67 NN$=ESC$+"$"+CHR$(1):NF$=ESC$+"$"+CHR$(0)
80 CS$="C":SC$=CHR$(143):SS$="@"
90 GOSUB 1910
100 '
110 'Main loop.
120 A$=INKEY$:IF A$="" THEN 120
150 IF A$ = "+" THEN GOSUB 1050 : GOTO 340 'Wider.
160 IF A$ = "-" THEN GOSUB 1080 : GOTO 340 'Narrower.
170 IF A$ = "T" OR A$="t" THEN GOSUB 1110 : GOTO 340
   'Descender.
180 IF A$="Q" OR A$="q" THEN CLS : END
190 IF A$="P" OR A$="p" THEN GOSUB 1350 : GOTO 340
210 IF A$="C" OR A$="c" THEN GOSUB 1910 : GOTO 340
220 IF A$="A" OR A$="a" THEN GOSUB 1670 : GOTO 340
240 IF A$="R" OR A$="r" THEN GOSUB 2010 : GOTO 340
270 IF A$=CHR$(8) THEN GOSUB 900:GOTO 340 'Left.
280 IF A$=CHR$(9) THEN GOSUB 920:GOTO 340 'Right.
290 IF A$=CHR$(10) THEN GOSUB 940:GOTO 340 'Down.
300 IF A$=CHR$(91) THEN GOSUB 960:GOTO 340 'Up.
310 IF A$="I" OR A$="i" THEN GOSUB 980:GOTO 340 'Insert.
320 IF A$= "D" OR A$="d" THEN GOSUB 1020:GOTO 340
   'Delete.
340 GOTO 120
390 '
400 'Subroutine to paint screen.
410 CLS
420 GOSUB 1770
430 '
440 'Draw grid.
```

```
450 PRINT @2*64+5,"M1  M2  M3  M4  M5  M6  M7  M8  M9
   M10 M11"
470 PRINT @3*64+4,"!---!---!---!---!---!---!---!---!---
   !---!---!"
480 FOR I=0 TO 6:PRINT @(I+4)*64+1,2[I;
485 PRINT @(I+4)*64+4,"!";TAB(48);"!";:NEXT I
490 PRINT @11*64+4,"!---!---!---!---!---!---!---!---!---
   !---!---!";
620 '
630 'Put in dots.
640 FOR H = 1 TO 11 : FOR J = 1 TO 7 : Z(J,H) = 0
680 NEXT J : NEXT H
690 FOR H = 1 TO 11 : GOSUB 1190 : NEXT H
700 X=1:Y=1:G=1:H=1
710 GOSUB 1290
720 '
730 'Paint menu.
732 PRINT @49,"CURSOR MOVEMENT";
734 PRINT @1*64+50,"LEFT ARROW";
736 PRINT @2*64+50,"RIGHT ARROW";
738 PRINT @3*64+50,"UP ARROW";
739 PRINT @4*64+50,"DOWN ARROW";
745 PRINT @5*64+50,"P)RINT CHAR.";
750 PRINT @6*64+50,"A)SCII SET";
760 PRINT @7*64+50,"C)LEAR DOTS";
770 PRINT @8*64+50,"Q)UIT";
780 PRINT @9*64+50,"R)OM COPY";
790 PRINT @10*64+50,"T)OGGLE DESC.";
820 PRINT @11*64+50,"I)NSERT DOT";
830 PRINT @12*64+50,"D)ELETE DOT";
840 PRINT @13*64+50,"+) WIDER CHAR.";
850 PRINT @14*64+50,"-) NARROWER";
870 RETURN
880 '
890 'Edit subroutines.
900 GOSUB 1230:Y=Y-1:H=H-1:IF Y<1 THEN Y=1:H=1
910 GOSUB 1290:RETURN
920 GOSUB 1230:Y=Y+1:H=H+1:IF Y>11 THEN Y=11:H=11
930 GOSUB 1290:RETURN
940 GOSUB 1230:X=X+1:G=G+1:IF X>7 THEN X=7:G=7
950 GOSUB 1290:RETURN
960 GOSUB 1230:X=X-1:G=G-1:IF X<1 THEN X=1:G=1
970 GOSUB 1290:RETURN
980 IF Z(G,H-1)=1 OR Z(G,H+1)=1 THEN RETURN
```

```
990 Z(G,H) = 1
1000 PRINT @(X+3)*64+Y*4+2,SS$;
1010 GOSUB 1140 : RETURN
1020 Z(G,H)=0
1030 PRINT @(X+3)*64+Y*4+2,CS$;
1040 GOSUB 1140 : RETURN
1050 IF PROWID = 11 THEN  RETURN
1060 PROWID = PROWID + 1
1070 GOSUB 1770 : RETURN
1080 IF PROWID = 4 THEN  RETURN
1090 PROWID = PROWID - 1
1100 GOSUB 1770 : RETURN
1110 IF DESC = 1 THEN DESC = 0 : GOTO 1130
1120 DESC = 1
1130 GOSUB 1770 : RETURN
1140 '
1150 'Subroutine to calculate a column value and print
   it.
1160 MM(H) = 0 : FOR J=1 TO 7
1170 MM(H)=MM(H)+Z(J,H)*2[(J-1)
1180 NEXT J
1190 '
1200 'Subroutine to print a column value.
1205 PRINT @12*64+H*4+1,"   ";
1210 PRINT @12*64+H*4+1,RIGHT$(STR$(MM(H)),3);
1220 RETURN
1230 '
1240 'Subroutine to remove the cursor.
1250 PRINT @(X+3)*64+Y*4+2,"";
1260 IF Z(G,H) = 0 THEN PRINT "  ";
1270 IF Z(G,H) = 1 THEN  PRINT SC$;
1280 RETURN
1290 '
1300 'Subroutine to place the cursor.
1310 PRINT @(X+3)*64+Y*4+2,"";
1320 IF Z(G,H)=1 THEN  PRINT SS$;
1330 IF Z(G,H)=0 THEN  PRINT CS$;
1340 RETURN
1350 '
1360 'Subroutine to print current character.
1370 GOSUB 2080
1380 LPRINT "ASCII code =" AS : LPRINT
1400 LPRINT REC$ ; 'Download the character.
1410 LPRINT CHR$(27) "@" ;
```

```
1460 LPRINT CHR$(27) "B" CHR$(3) "Condensed"
1470 LPRINT NN$ STRING$(21,AS)
1480 LPRINT NF$
1490 LPRINT CHR$(27) "B" CHR$(2) "Elite"
1500 LPRINT NN$ STRING$(15,AS)
1510 LPRINT NF$
1520 LPRINT CHR$(27) "B" CHR$(1) "Pica"
1530 LPRINT NN$ STRING$(12,AS)
1540 LPRINT NF$
1550 LPRINT CHR$(27) "W" CHR$(1) "Expanded"
1560 LPRINT NN$ STRING$(6,AS)
1570 LPRING NF$ CHR$(27) "W" CHR$(0)
1573 LPRINT:LPRINT "CHARACTER SET (NORMAL SPACING) "
1574 LPRINT NN$
1575 FOR I=33 TO 126:LPRINT CHR$(I);:NEXT I:LPRINT
1576 FOR I=160 TO 254:LPRINT CHR$(I);:NEXT
   I:LPRINT:LPRINT
1577 LPRINT NF$
1580 LPRINT "Proportional"
1590 LPRINT PN$ STRING$(15,AS)
1592 LPRINT PF$
1595 LPRINT:LPRINT "CHARACTER SET (PROPORTIONAL
   SPACING)":LPRINT PN$;
1596 FOR I=33 TO 126:LPRINT CHR$(I);:NEXT I:LPRINT
1597 FOR I=160 TO 254:LPRINT CHR$(I);:NEXT I:LPRINT
1600 LPRINT PF$
1610 LPRINT : LPRINT : LPRINT
1620 LPRINT "Use this data statement to download this
   character."
1630 GOSUB 2080 : LPRINT "DATA 27" ;
1640 FOR I = 2 TO LEN(REC$)
1650 LPRINT "," STR$(ASC(MID$(REC$,I,1))) ;
1660 NEXT I : LPRINT : LPRINT : LPRINT : LPRINT : RETURN
1670 '
1680 'Subroutine to input desired character code.
1690 PRINT @14*64,"";
1700 INPUT "Enter ASCII code (33-126 OR 160-254) --> " ;
   AS
1710 GOSUB 2040
1720 IF AS > 32 AND  AS < 127 THEN  GOTO 1760
1730 IF AS > 159 AND AS < 255 THEN GOTO 1760
1740 GOTO 1690
1760 GOSUB 1770 : RETURN
1770 '
```

```
178Ø 'Subroutine to display header.
179Ø PRINT @1,"ASCII CODE = ";AS;" ";
18ØØ PRINT "(" CHR$(AS AND &H7F) ;
181Ø IF AS > 127 THEN PRINT " + 128" ;
182Ø PRINT ")          " ;
183Ø PRINT @3Ø,"DESCENDER = ";DESC;
188Ø PRINT @1*64+9,STRING$(11," ");
189Ø PRINT @1*64+1,"WIDTH : ";STRING$(PROWID,"*");
19ØØ RETURN
191Ø '
192Ø 'Subroutine to clear current character.
193Ø PROWID = 11 : DESC = Ø
194Ø FOR H = 1 TO 11 : MM(H) = Ø : NEXT H
195Ø GOSUB 39Ø : RETURN
2Ø1Ø '
2Ø2Ø 'Subroutine to perform a ROM copy.
2Ø3Ø LPRINT ESC$ "*" CHR$(Ø): RETURN
2Ø4Ø '
2Ø5Ø 'Subroutine to erase query message.
2Ø6Ø PRINT @14*64,STRING$(5Ø," ");
2Ø7Ø RETURN
2Ø8Ø '
2Ø9Ø 'Subroutine to build command string.
21ØØ REC$ = ESC$ + "*" + CHR$(1)
211Ø REC$ = REC$ + CHR$(AS) + CHR$(DESC*16 + PROWID)
212Ø FOR I = 1 to 11 : REC$ = REC$ + CHR$(MM(I)) : NEXT
   I
213Ø RETURN
```

Piechart program

```
1Ø 'Program to print a piechart on the RADIX.
15 CLEAR 4ØØØ
2Ø CLS
21 'DIRECT-TO-PRINTER DRIVER FOR TRS-8Ø MODEL III
22 AD=16571
23 FOR I=Ø TO 14
24 READ A : POKE AD + I, A
25 NEXT I
26 POKE 16422,187
27 POKE 16423,64
28 DATA 33,232,55,2Ø3,126,32,252,33,17,Ø,57,126,
   211,251,2Ø1
29 '
3Ø 'Initialize program constants.
```

```
40 ESC$ = CHR$(27)        : LF$=CHR$(10)
50 FF$ = CHR$(12)         : VTAB$ = CHR$(11)
60 REVFF$ = ESC$ + FF$
70 'Emphasized & expanded modes.
80 TITLE$ = ESC$ + "E" + ESC$ + "W" + CHR$(1)
90 NTITLE$ = ESC$ + "F" + ESC$ + "W" + CHR$(0)
110 DIM BIT%(190,36),PCT%(25)
120 DIM TEXT$(48),PIECETEXT$(25)
130 MASK%(1) = 128          : MASK%(4) = 16
140 MASK%(2) = 64         : MASK%(5) = 8
150 MASK%(3) = 32         : MASK%(6) = 4
160 LX = 20               : LY = 20
170 LXFAC = 190/LX        : LYFAC = 216/LY
180 FOR I= 0 TO 48
190 TEXT$(I) = STRING$(79," ")
200 NEXT I
210 GOSUB 1040
215 GOSUB 2000
217 PRINT @64*7,"";
220 '
230 'Plot curve
240 RAD=8
250 X1 = 19                 : Y1 = 10
270 FOR ANG% = 0 TO 360 STEP 15
280 RNG = ANG%*6.28/360
290 X2 = RAD*COS(RNG)+10  : Y2 = RAD*SIN(RNG)+10
300 GOSUB 640
310 NEXT ANG%
315 PRINT @64*9,"";:
320 FOR PIECE% = 1 TO PCNT%
330 X1 = 10 : Y1 = 10
340 TPCT%=TPCT%+PCT%(PIECE%)
350 ANG%=360*TPCT%*.01
360 RNG = ANG%*6.28/360
370 X2 = RAD*COS(RNG)+10    : Y2 = RAD*SIN(RNG)+10
380 GOSUB 640
390 GOSUB 870
400 NEXT PIECE%
410 '
420 'Send chart title to printer.
440 LPRINT ESC$ "A" CHR$(6) REVFF$ VTAB$ ;
450 LPRINT TITLE$ STRING$(16-LEN(CTITLE$)/2," ") ;
460 LPRINT CTITLE$ NTITLE$
```

```
47Ø LPRINT VTAB$ VTAB$ ;
48Ø FOR I = Ø TO 48
49Ø LPRINT TEXT$(I) : NEXT I
5ØØ '
51Ø 'Send bit image map to printer.
515 LPRINT ESC$ "A" CHR$(6) ;
52Ø LPRINT REVFF$ VTAB$ VTAB$ VTAB$ ;
53Ø LPRINT LF$ LF$ LF$ LF$ LF$ LF$
54Ø FOR ROW% = 2 TO 33
55Ø LPRINT "                       " ;
56Ø LPRINT ESC$ "K" CHR$(171) CHR$(Ø) ;
57Ø FOR COL% = 1 TO 171
58Ø LPRINT CHR$(BIT%(COL%,ROW%)) ; : NEXT
59Ø LPRINT
61Ø NEXT ROW%
62Ø LPRINT ESC$ "2" FF$
63Ø END
64Ø '
65Ø 'Subroutine to draw a line from X1,Y1 to X2,Y2.
66Ø '
67Ø XL = X2 - X1        : YL = Y2 - Y1
68Ø NX = ABS(XL*LXFAC) : NY = ABS(YL*LYFAC)
69Ø IF NX < NY THEN NX = NY
7ØØ NS% = INT(NX+1)
71Ø DX = XL/NS%        : DY = YL/NS%
72Ø FOR I% = 1 TO NS%
73Ø X1 = X1 + DX       : Y1 = Y1 + DY
74Ø GOSUB 78Ø
75Ø NEXT I%
76Ø PRINT "*";
77Ø RETURN
78Ø '
79Ø 'Subroutine to plot a point at X1,Y1.
8ØØ '
81Ø XX = X1 * LXFAC      : YY = Y1 * LYFAC
82Ø COL% = INT(XX) + 1
83Ø ROW% = INT(YY/6)
84Ø XIT% = INT(YY - ROW% * 6)+1
85Ø BIT%(COL%,ROW%) = BIT%(COL%,ROW%) OR MASK%(XIT%)
86Ø RETURN
87Ø '
88Ø 'Subroutine to arrange field descriptions.
89Ø '
9ØØ MIDANG%=(ANG%+PREVANG%)/2
```

```
910 RNG = MIDANG%*6.28/360
920 X3 = INT(24*SIN(RNG)+.5) : Y3 = INT(20*COS(RNG))
930 X4 = 24 + X3 : Y4 = 42 + Y3
940 IF (MIDANG% > 70 AND MIDANG% < 110) THEN 990
950 IF (MIDANG% > 250 AND MIDANG% < 290) THEN 990
960 IF MIDANG%>270 OR MIDANG%<90 THEN 1010
970 MID$(TEXT$(X4), Y4-LEN(PIECETEXT$(PIECE%))) =
   PIECETEXT$(PIECE%)
980 GOTO 1020
990 MID$(TEXT$(X4),Y4-INT(LEN(PIECETEXT$(PIECE%))/
   2))=PIECETEXT$(PIECE%)
1000 GOTO 1020
1010 MID$(TEXT$(X4),Y4) = PIECETEXT$(PIECE%)
1020 PREVANG%=ANG%
1030 RETURN
1040 '
1050 'Subroutine to query user for data.
1060 '
1070 CLS: PRINT : PRINT : PRINT :
1080 PRINT "ENTER TITLE FOR CHART";
1085 INPUT CTITLE$
1090 IF LEN(CTITLE$) <= 32 THEN 1110
1100 PRINT "TITLE TOO LONG - 32 CHAR. MAX" : GOTO 1080
1110 SOFAR%=0 : LFT%=100
1120 FOR I=1 TO 24
1130 CLS
1140 PRINT "          ENTER PARAMETERS FOR PIE-CHART"
1150 PRINT "               TOTAL SO FAR :   ";
1160 PRINT USING "###";SOFAR%
1170 PRINT "               TOTAL REMAINING: ";
1180 PRINT USING "###";LFT%
1190 PRINT :PRINT :PRINT :PRINT
1200 PRINT "ENTER PERCENTAGE FOR FIELD:   ";
1205 INPUT PCT%(I)
1210 IF PCT%(I)>LFT% OR PCT%(I)=0 THEN PCT%(I)=LFT%
1220 LFT%=LFT%-PCT%(I)
1230 SOFAR%=SOFAR%+PCT%(I)
1240 PRINT :PRINT
1250 PRINT "ENTER DESCRIPTION OF FIELD:  ";
1255 INPUT PIECETEXT$(I)
1260 IF LEN(PIECETEXT$(I))<16 THEN 1280
1270 PRINT "FIELD TOO LONG - 15 CHAR. MAX": GOTO 1250
1280 IF LFT%=0 GOTO 1300
1290 NEXT I
```

```
1300 PCNT%=I
1310 IF PCNT%=1 THEN 1110
1320 CLS
1330 RETURN
2000 REM
2010 CLS
2020 PRINT:PRINT:PRINT
2030 PRINT "THIS PROGRAM TAKES ABOUT TWO MINUTES TO RUN"
2040 PRINT "PLEASE TURN ON YOUR PRINTER AND STAND BY..."
2050 PRINT:PRINT
2060 PRINT ":::::::::::::::::::::::::"
2070 PRINT
2080 FOR I=1 TO PCNT%:PRINT ":"; : NEXT I
2090 RETURN
```

Printer setup utility

```
10 'Program to setup RADIX printer as directed.
20 '
30 'Initialize.
35 CLEAR 1000
40 ESC$ = CHR$(27) : TB = 15 : DIM TBS(256)
60 '
70 'Display MAIN menu.
80 CLS
90 TITLE$ = "MAIN MENU"
100 GOSUB 2560
110 PRINT TAB(TB) "0. Exit."
120 PRINT TAB(TB) "1. Select CHARACTER SET."
130 PRINT TAB(TB) "2. Select PRINTING MODES."
140 PRINT TAB(TB) "3. Select PITCH."
150 PRINT TAB(TB) "4. Select LINE SPACING."
160 PRINT TAB(TB) "5. Set MARGINS, TABS & FORMS."
170 GOSUB 2650
180 IF S<0 OR S>5 THEN 170
190 IF S = 0 THEN END
200 ON S GOSUB 220,490,360,1410,650
210 GOTO 60
220 '
230 'Subroutine to display CHARACTER SET menu.
240 TITLE$ = "CHARACTER SET MENU"
250 GOSUB 2560
260 PRINT TAB(TB) "0. Return to main menu."
270 PRINT TAB(TB) "1. Select NLQ character set."
280 PRINT TAB(TB) "2. Cancel NLQ character set."
```

```
290 PRINT TAB(TB) "3. Select ITALIC character set."
300 PRINT TAB(TB) "4. Cancel ITALIC character set."
310 GOSUB 2650
320 IF S<0 OR S>4 THEN 310
330 IF S = 0 THEN RETURN
340 ON S GOSUB 1310,1360,1800,1840
350 GOTO 220
360 '
370 'Subroutine to display PITCHES menu.
380 TITLE$ = "PITCHES MENU"
390 GOSUB 2560
400 PRINT TAB(TB) "0. Return to main menu."
410 PRINT TAB(TB) "1. Select PICA pitch."
420 PRINT TAB(TB) "2. Select ELITE pitch."
430 PRINT TAB(TB) "3. Select CONDENSED pitch."
440 GOSUB 2650
450 IF S<0 OR S>3 THEN 440
460 IF S = 0 THEN RETURN
470 ON S GOSUB 830,880,930
480 GOTO 360
490 '
500 'Subroutine to display PRINTING MODES menu.
510 TITLE$ = "PRINTING MODES MENU"
520 GOSUB 2560
530 PRINT TAB(TB) "0. Return to main menu."
540 PRINT TAB(TB) "1. Select EXPANDED mode."
550 PRINT TAB(TB) "2. Cancel EXPANDED mode."
560 PRINT TAB(TB) "3. Select EMPHASIZED mode."
570 PRINT TAB(TB) "4. Cancel EMPAHASIZED mode."
580 PRINT TAB(TB) "5. Select DOUBLE-STRIKE mode."
590 PRINT TAB(TB) "6. Cancel DOUBLE-STRIKE mode."
600 GOSUB 2650
610 IF S<0 OR S>6 THEN 600
620 IF S = 0 THEN RETURN
630 ON S GOSUB 1700,1750,2400,2440,2480,2520
640 GOTO 490
650 '
660 'Subroutine to display MARGINS, TABS & FORMS menu.
670 TITLE$ = "MARGINS, TABS & FORMS MENU"
680 GOSUB 2560
690 PRINT TAB(TB) "0. Return to main menu."
700 PRINT TAB(TB) "1. Set HORIZONTAL TABS."
710 PRINT TAB(TB) "2. Set VERTICAL TABS."
720 PRINT TAB(TB) "3. Set LEFT MARGIN."
```

```
73Ø PRINT TAB(TB) "4. Set RIGHT MARGIN."
74Ø PRINT TAB(TB) "5. Set TOP MARGIN."
75Ø PRINT TAB(TB) "6. Set BOTTOM MARGIN."
76Ø PRINT TAB(TB) "7. Cancel TOP AND BOTTOM MARGINS."
77Ø PRINT TAB(TB) "8. Set PAGE LENGTH."
78Ø GOSUB 265Ø
79Ø IF S<Ø OR S>8 THEN 78Ø
8ØØ IF S = Ø THEN RETURN
81Ø ON S GOSUB 2Ø5Ø,236Ø,98Ø,1Ø6Ø,113Ø,121Ø,128Ø,188Ø
82Ø GOTO 65Ø
83Ø '
84Ø 'Subroutine to select PICA pitch.
85Ø S$ = ESC$ + "B" + CHR$(1)
86Ø GOSUB 273Ø
87Ø RETURN
88Ø '
89Ø 'Subroutine to select ELITE pitch.
9ØØ S$ = ESC$ + "B" + CHR$(2)
91Ø GOSUB 273Ø
92Ø RETURN
93Ø '
94Ø 'Subroutine to select CONDENSED pitch.
95Ø S$ = ESC$ + "B" + CHR$(3)
96Ø GOSUB 273Ø
97Ø RETURN
98Ø '
99Ø 'Subroutine to set LEFT MARGIN.
1ØØØ GOSUB 277Ø
1Ø1Ø INPUT "Enter new left margin (1-255)" ; X
1Ø2Ø IF X < 1 OR X > 255 THEN  GOTO 1ØØØ
1Ø3Ø S$ = ESC$ + "M" + CHR$(X)
1Ø4Ø GOSUB 273Ø
1Ø5Ø RETURN
1Ø6Ø '
1Ø7Ø 'Subroutine to set RIGHT MARGIN.
1Ø8Ø GOSUB 277Ø
1Ø9Ø INPUT "Enter new right margin (1-255)" ; X
11ØØ IF X < 1 OR X > 255 THEN  GOTO 1Ø8Ø
111Ø S$ = ESC$ + "Q" + CHR$(X)
112Ø GOSUB 273Ø : RETURN
113Ø '
114Ø 'Subroutine to set TOP MARGIN.
115Ø GOSUB 277Ø
116Ø INPUT "Enter new top margin (1-16)" ; X
```

```
1170 IF X < 1 OR X > 16 THEN  GOTO 1150
1180 S$ = ESC$ + "R" + CHR$(X)
1190 GOSUB 2730
1200 RETURN
1210 '
1220 'Subroutine to set BOTTOM MARGIN.
1230 GOSUB 2770
1240 INPUT "Enter new bottom margin (1-127)" ; X
1250 IF X < 1 OR X > 127 THEN  GOTO 1230
1260 S$ = ESC$ + "N" + CHR$(X)
1270 GOSUB 2730 : RETURN
1280 '
1290 'Subroutine to cancel TOP & BOTTOM MARGINS.
1300 S$ = ESC$ + "O" : GOSUB 2730 : RETURN
1310 '
1320 'Subroutine to select NLQ character set.
1330 S$ = ESC$ + "B" + CHR$(4)
1340 GOSUB 2730 : RETURN
1360 '
1370 'Subroutine to cancel NLQ character set.
1380 S$ = ESC$ + "B" + CHR$(5)
1390 GOSUB 2730
1400 RETURN
1410 '
1420 'Subroutine to select LINE SPACING.
1430 TITLE$ = "LINE SPACING MENU"
1440 GOSUB 2560
1450 PRINT TAB(TB) "0. Return to main menu."
1460 PRINT TAB(TB) "1. Select 1/6 inch line spacing."
1470 PRINT TAB(TB) "2. Select 1/8 inch line spacing."
1480 PRINT TAB(TB) "3. Select 7 dot graphics spacing."
1490 PRINT TAB(TB) "4. Select n/144 inch spacing."
1500 GOSUB 2650
1510 IF S<0 OR S>4 THEN 1500
1520 IF S = 0 THEN RETURN
1530 ON S GOSUB 1550,1580,1610,1640
1540 GOTO 1410
1550 '
1560 'Subroutine to select 1/6 inch line spacing.
1570 S$ = ESC$ + "2" : GOSUB 2730 : RETURN
1580 '
1590 'Subroutine to select 1/8 inch line spacing.
1600 S$ = ESC$ + "0" : GOSUB 2730 : RETURN
```

```
1610 '
1620 'Subroutine to select 7 dot graphics spacing.
1630 S$ = ESC$ + "1" : GOSUB 2730 : RETURN
1640 '
1650 'Subroutine to select n/144 inch line spacing.
1660 GOSUB 2770
1670 INPUT "Enter line space in n/144 ths of an inch"; X
1680 IF X < 0 OR X > 255 THEN 1660
1690 S$ = ESC$ + "3" + CHR$(X) : GOSUB 2730 : RETURN
1700 '
1710 'Subroutine to select EXPANDED print.
1720 S$ = ESC$ + "W" + CHR$(1)
1730 GOSUB 2730
1740 RETURN
1750 '
1760 'Subroutine to cancel EXPANDED print.
1770 S$ = ESC$ + "W" + CHR$(0)
1780 GOSUB 2730
1790 RETURN
1800 '
1810 'Subroutine to select ITALIC character set.
1820 S$ = ESC$ + "4" : GOSUB 2730
1830 RETURN
1840 '
1850 'Subroutine to cancel ITALIC character set.
1860 S$ = ESC$ + "5" : GOSUB 2730
1870 RETURN
1880 '
1890 'Subroutine to set PAGE LENGTH.
1900 GOSUB 2770
1910 PRINT "Page length in Inches or Lines (I,L)?"
1920 PRINT TAB(TB) ;
1930 A$ = INKEY$ : IF A$ = "" THEN 1930
1940 IF A$ = "I" OR A$ ="i" THEN 1970
1950 IF A$ = "L" OR A$ ="l" THEN 2010
1960 GOTO 1930
1970 INPUT "Length of page in inches (1-32)" ; X
1980 IF X < 1 OR X > 32 THEN 1900
1990 S$ = ESC$ + "C" + CHR$(0) + CHR$(X)
2000 GOSUB 2730 : RETURN
2010 INPUT "Length of page in lines (1-127)" ; X
2020 IF X < 1 OR X > 127 THEN 1900
2030 S$ = ESC$ + "C" + CHR$(X)
2040 GOSUB 2730 : RETURN
```

```
2050 '
2060 'Subroutine to set HORIZONTAL TABS.
2070 S$ = ESC$ + "D" : MAX = 255 : GOSUB 2080 : RETURN
2080 '
2090 'Subroutine to set tabs, either horiz or vert.
2100 GOSUB 2770
2110 PRINT "Would you like to set the tabs in"
2120 PRINT TAB(TB) "Regular intervals, or specify"
2130 PRINT TAB(TB) "each one Individually (R,I)"
2140 A$ = INKEY$ : IF A$ = "" THEN 2140
2150 IF A$ = "R" OR A$ = "r" THEN 2300
2160 IF A$ = "I" OR A$ = "i" THEN 2180
2170 GOTO 2080
2180 PRINT : I = 2 : TBS(1) = -1
2190 PRINT TAB(TB) "Enter the list of tabs, in"
2200 PRINT TAB(TB) "ascending order. No more than" MAX
   "."
2210 PRINT TAB(TB) : INPUT "Enter a tab" ; TBS(I)
2220 IF TBS(I) < 0 OR TBS(I) > 255 THEN 2170
2230 IF TBS(I) = 0 THEN I = 1 : GOTO 2270
2240 IF TBS(I) <= TBS(I-1) THEN 2170
2250 I = I + 1 : IF I > MAX THEN 2170
2260 GOTO 2210
2270 I = I + 1
2280 S$ = S$ + CHR$(TBS(I)) : IF TBS(I) <> 0 THEN 2270
2285 S$=S$+CHR$(0):GOSUB 2730
2290 RETURN
2300 PRINT : PRINT TAB(TB) ; : INPUT "Enter interval" ;
   X
2310 IF X < 0 OR X > 255 THEN 2080
2320 FOR I = 1 TO 255 STEP X
2330 MAX = MAX -1 : IF MAX = 0 THEN 2350
2340 S$ = S$ + CHR$(I) : NEXT I
2350 S$ = S$ + CHR$(0) : GOSUB 2730 : RETURN
2360 '
2370 'Subroutine to set VERTICAL TABS.
2380 S$ = ESC$ + "P" : MAX = 20 : GOSUB 2080
2390 RETURN
2400 '
2410 'Subroutine to select EMPHASIZED printing.
2420 S$ = ESC$ + "E" : GOSUB 2730
2430 RETURN
2440 '
2450 'Subroutine to cancel EMPHASIZED printing.
```

```
2460 S$ = ESC$ + "F" : GOSUB 2730
2470 RETURN
2480 '
2490 'Subroutine to select DOUBLE-STRIKE printing.
2500 S$ = ESC$ + "G" : GOSUB 2730
2510 RETURN
2520 '
2530 'Subroutine to cancel DOUBLE-STRIKE printing.
2540 S$ = ESC$ + "H" : GOSUB 2730
2550 RETURN
2560 '
2570 'Subroutine to print a menu title.
2580 CLS
2600 PRINT TAB(18) "--- RADIX PRINTER SETUP ---"
2610 PRINT
2620 PRINT TAB((64-LEN(TITLE$))/2) TITLE$
2630 PRINT
2640 RETURN
2650 '
2660 'Subroutine to input menu selection.
2670 PRINT @960+TB,"Enter selection. or hit P for a
   print out";
2680 C$ = INKEY$ : IF C$ = "" THEN 2680
2685 IF C$="P" OR C$="p" THEN GOSUB 3000:GOTO 2660
2690 IF C$ < "0" OR C$ > "9" THEN  GOTO 2680
2700 S = VAL(C$)
2710 PRINT @960,STRING$(63," ");
2720 RETURN
2730 '
2740 'Subroutine to output command string.
2750 LPRINT S$ ;
2760 RETURN
2770 '
2780 'Subroutine to clear screen & position cursor.
2790 CLS : PRINT @320+TB, "" ; : RETURN
3000 FOR I=1 TO 4:FOR J=32 TO 126:LPRINT CHR$(J);:NEXT J
3010 FOR J=160 TO 254:LPRINT CHR$(J);: NEXT J
3015 LPRINT
3020 NEXT I:RETURN
```

Appendix E

Kaypro, Osborne and Other CP/M Computers

All that you need to connect Radix to an Osborne 1 or Kaypro computer is a cable. Your Radix dealer can provide the cable that you need.

Setting the Switches

When connecting Radix to an Osborne 1, Kaypro, or other CP/M computer, we recommend that you set the DIP switches in Radix as shown below. (Although our chart indicates switch C-2 set for a parallel interface, a serial interface will work also.)

Table E-1
Recommended DIP switch settings for Kaypro

Switch	Setting	Function
A-1	ON	11 inch page size
A-2	ON	Normal print density
A-3	ON	10 CPI pitch
A-4	ON	Normal characters
A-5	ON	1/6 inch line feed
A-6	ON	U.S.A. Character set
A-7	ON	
A-8	ON	
C-1	ON	Paper-out detector active
C-2	OFF	Parallel interface
C-3	OFF	8-bit interface
C-4	OFF	No auto line feed

When you connect your printer to your Osborne 1 you must use the SETUP program to tell the computer whether Radix is connected to the Osborne 1's serial or parallel interface (either will work).

Table E-2
Kaypro parallel cable

Radix		**Kaypro**	
Pin No.	**Function**	**Pin No.**	**Function**
1	STROBE	1	STROBE
2	DATA1	2	DATA1
3	DATA2	3	DATA2
4	DATA3	4	DATA2
5	DATA4	5	DATA2
6	DATA5	6	DATA2
7	DATA6	7	DATA2
8	DATA7	8	DATA2
9	DATA8	9	DATA8
11	BUSY	11	BUSY
16	SIG GND	16	SIG GND

Table E-3
Osborne 1 parallel cable

Radix		**Osborne 1**	
Pin No.	**Function**	**Pin No.**	**Function**
2	DATA1	1	DATA0
6	DATA5	2	DATA4
3	DATA2	3	DATA1
7	DATA6	4	DATA5
4	DATA3	5	DATA2
8	DATA7	6	DATA6
5	DATA4	7	DATA3
9	DATA8	8	DATA7
1	STROBE	11	STROBE
11	BUSY	15	BUSY
16	SIG GND	16	SIG GND

Using MBASIC

Many CP/M computers use Microsoft BASIC (called MBASIC). If you have a CP/M-80 computer that uses Microsoft BASIC the program listings given here should work with your computer also.

MBASIC is a very close relative of the IBM-Microsoft BASIC used in this book. The only difference is that MBASIC "interprets" CHR$(9) and substitutes a group of spaces to simulate a tab. You can send a horizontal tab to Radix by using CHR$(137) instead of CHR$(9).

Some versions of Microsoft BASIC will add a carriage return and line feed at the end of every 80 (or sometimes 132) characters. To print lines longer than 80 (or 132) characters (as when doing dot graphics) you must define a wider printer width. The following statement will prevent the computer from inserting unwanted codes.

```
10 WIDTH LPRINT 255
```

Listing programs

Microsoft BASIC uses the "L" prefix on several commands to direct them to the printer. To list programs on the printer, just type LLIST. To direct program output to the printer, use LPRINT in place of PRINT.

Program Listings

The following programs are in Microsoft BASIC for the Kaypro.

Download character editing utility

```
10 'Program to allow editing down-load characters.
20 'for the RADIX printer.
30 '
40 'Initialization.
50 DIM Z(8,12),MM(11)
60 WIDTH 255
70 AS=33 :
```

```
80 CS$=CHR$(91)+CHR$(93):SC$=STRING$(2,159):
   CR$=STRING$(2,127)
90 RAMNML$ = CHR$(27) + "$" + CHR$(1)
100 RAMNMLOFF$ = CHR$(27) + "$" + CHR$(0)
110 RAMPRO$ = CHR$(27) + "X" + CHR$(1)
120 RAMPROOFF$ = CHR$(27) + "X" + CHR$(0)
130 DEF FNL$(ROW,COL) = CHR$(27) + "=" + CHR$(ROW+32) +
   CHR$(COL+32)
140 LPRINT CHR$(27) "@" ; : WIDTH "LPT1:",255
150 GOSUB 1660
160 '
170 'Main loop.
180 A$=INKEY$:IF A$="" THEN 180
190 IF A$ = "+" OR A$ = "=" THEN GOSUB 820 : GOTO 340
   'Wider.
200 IF A$ = "-" OR A$ = CHR$(95) THEN GOSUB 850 : GOTO
   340 'Narrower.
210 IF A$ = "D" OR A$ = "d" THEN GOSUB 880 : GOTO 340
   'Descender.
220 IF A$="Q" OR A$="q" THEN GOSUB 350 : END
230 IF A$="P" OR A$="p" THEN GOSUB 1120 : GOTO 340
240 IF A$="C" OR A$="c" THEN GOSUB 1660 : GOTO 340
250 IF A$="A" OR A$="a" THEN GOSUB 1480 : GOTO 340
260 IF A$="R" OR A$="r" THEN GOSUB 1710 : GOTO 340
270 IF A$=CHR$(8) THEN GOSUB 670:GOTO 340 'Left.
280 IF A$=CHR$(12) THEN GOSUB 690:GOTO 340 'Right.
290 IF A$=CHR$(10) THEN GOSUB 710:GOTO 340 'Down.
300 IF A$=CHR$(11) THEN GOSUB 730:GOTO 340 'Up.
310 IF A$=CHR$(13) THEN GOSUB 750:GOTO 340 'Insert.
320 IF A$=CHR$(127) THEN GOSUB 790:GOTO 340 'Delete.
330 PRINT CHR$(7);
340 GOTO 180
350 PRINT CHR$(26);
360 RETURN
370 '
380 ' Subroutine to paint screen.
390 PRINT CHR$(26);
400 GOSUB 1560
410 '
420 'Draw grid.
430 FOR I=0 TO 6:PRINT FNL$(5+I*2,6); 2^I;:NEXT I
440 '
450 'Put in dots.
460 FOR H = 1 TO 11 : FOR J = 1 TO 7 : Z(J,H) = 0
```

```
470 NEXT J : NEXT H
480 FOR H = 1 TO 11 : GOSUB 960 : NEXT H
490 X=1:Y=1:G=1:H=1
500 GOSUB 1060
510 '
520 'Paint menu.
530  PRINT FNL$(6,47) "P -- Print the character."
540  PRINT FNL$(7,47) "A -- Set ASCII code."
550  PRINT FNL$(8,47) "C -- Clear all dots."
560  PRINT FNL$(9,47) "Q -- Quit."
570  PRINT FNL$(10,47) "R -- Perform ROM copy."
580 PRINT FNL$(11,45) "Arrow keys move cursor"
590 PRINT FNL$(12,45) "RET -- place a dot.";
600 PRINT FNL$(13,45) "DEL -- remove a dot.";
610  PRINT FNL$(14,47) "+ -- make character wider." ;
620  PRINT FNL$(15,47) "- -- make character narrower." ;
630  PRINT FNL$(16,47) "D -- Toggle descender mode." ;
640 RETURN
650 '
660 'Edit subroutines.
670 GOSUB 1000:Y=Y-3:H=H-1:IF Y<1 THEN PRINT CHR$(7);
   :Y=1:H=1
680 GOSUB 1060:RETURN
690 GOSUB 1000:Y=Y+3:H=H+1:IF Y>31 THEN PRINT CHR$(7);
   :Y=31:H=11
700 GOSUB 1060:RETURN
710 GOSUB 1000:X=X+2:G=G+1:IF X>13 THEN PRINT CHR$(7);
   :X=13:G=7
720 GOSUB 1060:RETURN
730 GOSUB 1000:X=X-2:G=G-1:IF X<1 THEN PRINT CHR$(7);
   :X=1:G=1
740 GOSUB 1060:RETURN
750 IF Z(G,H-1)=1 OR Z(G,H+1)=1 THEN PRINT CHR$(7);
   :RETURN
760 Z(G,H) = 1
770 PRINT FNL$(X+4,Y+10) CR$ ;
780 GOSUB 910 : RETURN
790 Z(G,H)=0
800  PRINT FNL$(X+4,Y+10) CS$ ;
810 GOSUB 910 : RETURN
820 IF PROWID = 11 THEN PRINT CHR$(7);  : RETURN
830 PROWID = PROWID + 1
840 GOSUB 1560 : RETURN
850 IF PROWID = 4 THEN PRINT CHR$(7);  : RETURN
```

```
860 PROWID = PROWID - 1
870 GOSUB 1560 : RETURN
880 IF DESC = 1 THEN DESC = 0 : GOTO 900
890 DESC = 1
900 GOSUB 1560 : RETURN
910 '
920 'Subroutine to calculate a column value & print it.
930 MM(H) = 0 : FOR J=1 TO 7
940 MM(H)=MM(H)+Z(J,H)*2^(J-1)
950 NEXT J : GOSUB 960 : RETURN
960 '
970 'Subroutine to print a column value.
980  PRINT FNL$(19,7+3*H); RIGHT$("  "+STR$(MM(H)),3) ;
990 RETURN
1000 '
1010 'Subroutine to remove the cursor.
1020 PRINT FNL$(X+4, Y+10);
1030 IF Z(G,H) = 0 THEN PRINT "  " ;
1040 IF Z(G,H) = 1 THEN  PRINT SC$ ;
1050 RETURN
1060 '
1070 'Subroutine to place the cursor.
1080 PRINT FNL$(X+4,Y+10);
1090 IF Z(G,H)=1 THEN  PRINT CR$ ;
1100 IF Z(G,H)=0 THEN  PRINT CS$ ;
1110 RETURN
1120 '
1130 'Subroutine to print current character.
1140 GOSUB 1780
1150 LPRINT "ASCII code =" AS : LPRINT
1160 LPRINT REC$ ; 'Download the character.
1170 LPRINT CHR$(27) "B" CHR$(3) "Condensed"
1180 LPRINT RAMNML$ STRING$(21,AS)
1190 LPRINT RAMNMLOFF$
1200 LPRINT CHR$(27) "B" CHR$(2) "Elite"
1210 LPRINT RAMNML$ STRING$(15,AS)
1220 LPRINT RAMNMLOFF$
1230 LPRINT CHR$(27) "B" CHR$(1) "Pica"
1240 LPRINT RAMNML$ STRING$(12,AS)
1250 LPRINT RAMNMLOFF$
1260 LPRINT CHR$(27) "W" CHR$(1) "Expanded"
1270 LPRINT RAMNML$ STRING$(6,AS)
1280 LPRINT RAMNMLOFF$ CHR$(27) "W" CHR$(0)
1290 LPRINT "Character set (normal width)"
```

```
1300 LPRINT RAMNML$;
1310 FOR I=33 TO 126 : LPRINT CHR$(I); : NEXT : LPRINT
1320 FOR I=160 TO 254 : LPRINT CHR$(I); : NEXT : LPRINT
1330 LPRINT RAMNMLOFF$
1340 LPRINT "Proportional"
1350 LPRINT RAMPRO$ STRING$(15,AS)
1360 LPRINT RAMPROOFF$
1370 LPRINT "Character set (proportional)"
1380 LPRINT RAMPRO$;
1390 FOR I=33 TO 126 : LPRINT CHR$(I); : NEXT : LPRINT
1400 FOR I=160 TO 254 : LPRINT CHR$(I); : NEXT : LPRINT
1410 LPRINT RAMPROOFF$
1420 LPRINT : LPRINT : LPRINT
1430 LPRINT "Use this data statement to download this
   character."
1440 GOSUB 1780 : LPRINT "DATA 27" ;
1450 FOR I = 2 TO LEN(REC$)
1460 LPRINT "," STR$(ASC(MID$(REC$,I,1))) ;
1470 NEXT I : LPRINT : LPRINT : LPRINT : LPRINT : RETURN
1480 '
1490 'Subroutine to input desired character code.
1500 PRINT FNL$(23,5);
1510 INPUT; "Enter ASCII code (33-126 OR 160-254) --> "
   ; AS
1520 GOSUB 1740
1530 IF AS < 33 OR AS > 254 THEN PRINT CHR$(7);  :
   GOTO 1500
1540 IF AS < 160 AND AS > 126 THEN PRINT CHR$(7);  :
   GOTO 1500
1550 GOSUB 1560 : RETURN
1560 '
1570 'Subroutine to display header.
1580  PRINT FNL$(1,1) "ASCII CODE =" AS ;
1590 PRINT "(" CHR$(AS AND &H7F) ;
1600 IF AS > 127 THEN PRINT " + 128" ;
1610 PRINT ")          " ;
1620  PRINT FNL$(1,30) "DESCENDER =" DESC ;
1630  PRINT FNL$(3,10) STRING$(33, " ") ;
1640  PRINT FNL$(3,2) "WIDTH : " STRING$
   (PROWID*3, "*") ;
1650 RETURN
1660 '
1670 'Subroutine to clear current character.
1680 PROWID = 11 : DESC = 0
```

```
169Ø FOR H = 1 TO 11 : MM(H) = Ø : NEXT H
17ØØ GOSUB 37Ø : RETURN
171Ø '
172Ø 'Subroutine to perform a ROM copy.
173Ø LPRINT CHR$(27) "*" CHR$(Ø) ; : RETURN
174Ø '
175Ø 'Subroutine to erase query message.
176Ø PRINT FNL$(23,5) STRING$(7Ø," ") ;
177Ø RETURN
178Ø '
179Ø 'Subroutine to build command string.
18ØØ REC$ = CHR$(27) + "*" + CHR$(1)
181Ø REC$ = REC$ + CHR$(AS) + CHR$(DESC*16 + PROWID)
182Ø FOR I = 1 TO 11 : REC$ = REC$ + CHR$(MM(I)) : NEXT
   I
183Ø RETURN
```

Piechart program

```
Ø 'Program to print a piechart on the RADIX.
2Ø '
3Ø 'Initialize program constants.
4Ø ESC$ = CHR$(27)       : LF$ =CHR$(1Ø)
5Ø FF$ = CHR$(12)        : VTAB$ = CHR$(11)
6Ø REVFF$ = ESC$ + FF$
7Ø 'Emphasized & expanded modes.
8Ø TITLE.ON$ = ESC$ + "E" + ESC$ + "W" + CHR$(1)
9Ø TITLE.OFF$ = ESC$ + "F" + ESC$ + "W" + CHR$(Ø)
1ØØ '
11Ø DIM BIT%(19Ø,36),A$(36),PCT%(25)
12Ø DIM TEXT$(48),PIECETEXT$(25)
13Ø MASK%(1) = 128       : MASK%(4) = 16
14Ø MASK%(2) = 64        : MASK%(5) = 8
15Ø MASK%(3) = 32        : MASK%(6) = 4
16Ø LX = 2Ø              : LY = 2Ø
17Ø LXFAC = 19Ø/LX       : LYFAC = 216/LY
18Ø FOR I= Ø TO 48
19Ø TEXT$(I) = SPACE$(79)
2ØØ NEXT I
21Ø GOSUB 1Ø4Ø
22Ø '
23Ø ' Plot curve
24Ø RAD = 9
25Ø X1 = 19              : Y1 = 1Ø
27Ø FOR ANG% = Ø TO 36Ø STEP 12
```

```
280 RANG = ANG%*6.28/360
290 X2 = RAD*COS(RANG)+10  : Y2 = RAD*SIN(RANG)+10
300 GOSUB 640
310 NEXT ANG%
320 FOR PIECE% = 1 TO NUMBER.PIECES%
330 X1 = 10                : Y1 = 10
340 TOTAL.PCT%=TOTAL.PCT%+PCT%(PIECE%)
350 ANG%=360*TOTAL.PCT%*.01
360 RANG = ANG%*6.28/360
370 X2 = RAD*COS(RANG)+10  : Y2 = RAD*SIN(RANG)+10
380 GOSUB 640
390 GOSUB 870
400 NEXT PIECE%
410 '
420 'Send chart title to printer.
440 LPRINT ESC$ "A" CHR$(6) REVFF$ VTAB$ ;
450 LPRINT TITLE.ON$ SPACE$(20-LEN(TITLE$)/2) ;
460 LPRINT TITLE$ TITLE.OFF$
470 LPRINT VTAB$ VTAB$ ;
480 FOR I = 0 TO 48
490 LPRINT TEXT$(I) : NEXT I
500 '
510 'Send bit image map to printer.
520 LPRINT REVFF$ VTAB$ VTAB$ VTAB$ ;
530 LPRINT LF$ LF$ LF$ LF$ LF$ LF$
540 FOR ROW% = 0 TO 35
550 LPRINT "                        " ;
560 LPRINT ESC$ "K" CHR$(190) CHR$(0) ;
570 FOR COL% = 1 TO 190
580 LPRINT CHR$(BIT%(COL%,ROW%)) ; : NEXT
590 LPRINT
600 PRINT CHR$(127) CHR$(127);
610 NEXT ROW%
620 LPRINT ESC$ "2" FF$
630 END
640 '
650 'Subroutine to draw a line from X1,Y1 to X2,Y2.
660 '
670 XL = X2 - X1         : YL = Y2 - Y1
680 NX = ABS(XL*LXFAC) : NY = ABS(YL*LYFAC)
690 IF NX < NY THEN NX = NY
700 NS% = INT(NX+1)
710 DX = XL/NS%          : DY = YL/NS%
720 FOR I% = 1 TO NS%
```

```
730 X1 = X1 + DX       : Y1 = Y1 + DY
740 GOSUB 780
750 NEXT I%
760 PRINT CHR$(8) "==>";
770 RETURN
780 '
790 'Subroutine to plot a point at X1,Y1.
800 '
810 XX = X1 * LXFAC    : YY = Y1 * LYFAC
820 COL% = INT(XX) + 1
830 ROW% = INT(YY/6)
840 XIT% = INT(YY - ROW% * 6)+1
850 BIT%(COL%,ROW%) = BIT%(COL%,ROW%) OR MASK%(XIT%)
860 RETURN
870 '
880 'Subroutine to arrange field descriptions.
890 '
900 MIDANG%=(ANG%+PREVANG%)/2
910 RANG = MIDANG%*6.28/360
920 X3 = INT(24*SIN(RANG)+.5) : Y3 = INT(20*COS(RANG))
930 X4 = 24 + X3              : Y4 = 42 + Y3
940 IF (MIDANG% > 70 AND MIDANG% < 110) THEN 990
950 IF (MIDANG% > 250 AND MIDANG% < 290) THEN 990
960 IF MIDANG%>270 OR MIDANG%<90 THEN 1010
970 MID$(TEXT$(X4),Y4-LEN(PIECETEXT$(PIECE%)))
   =PIECETEXT$(PIECE%)
980 GOTO 1020
990 MID$(TEXT$(X4),Y4-LEN(PIECETEXT$(PIECE%))\2)
   =PIECETEXT$(PIECE%)
1000 GOTO 1020
1010 MID$(TEXT$(X4),Y4) = PIECETEXT$(PIECE%)
1020 PREVANG%=ANG%
1030 RETURN
1040 '
1050 'Subroutine to query user for data.
1060 '
1070 PRINT CHR$(26) : PRINT : PRINT :
1080 INPUT "ENTER TITLE FOR CHART: ",TITLE$
1090 IF LEN(TITLE$) <= 40 THEN 1110
1100 PRINT "TITLE TOO LONG - 40 CHAR. MAX" : GOTO 1080
1110 AMT.SOFAR%=0       : AMT.LEFT%=100
1120 FOR I=1 TO 24
1130 PRINT CHR$(26);
```

```
1140 PRINT "                    ENTER PARAMETERS FOR
   PIECHART"
1150 PRINT "                         TOTAL SO FAR :   ";
1160 PRINT USING "###";AMT.SOFAR%
1170 PRINT "                         TOTAL REMAINING: ";
1180 PRINT USING "###";AMT.LEFT%
1190 PRINT :PRINT :PRINT :PRINT
1200 INPUT "ENTER PERCENTAGE FOR FIELD:   ",PCT%(I)
1210 IF PCT%(I)>AMT.LEFT% OR PCT%(I)=0 THEN
   PCT%(I)=AMT.LEFT%
1220 AMT.LEFT%=AMT.LEFT%-PCT%(I)
1230 AMT.SOFAR%=AMT.SOFAR%+PCT%(I)
1240 PRINT :PRINT
1250 INPUT "ENTER DESCRIPTION OF FIELD:
   ",PIECETEXT$(I)
1260 IF LEN(PIECETEXT$(I))<16 THEN 1280
1270 PRINT "FIELD TOO LONG - 15 CHAR. MAX": GOTO 1250
1280 IF AMT.LEFT%=0 GOTO 1300
1290 NEXT I
1300 NUMBER.PIECES%=I
1310 IF NUMBER.PIECES%=1 THEN 1110
1320 PRINT CHR$(26);
1330 RETURN
```

Printer setup utility

```
10 'Program to setup RADIX printer as directed.
20 '
30 'Initialize.
40 ESC$ = CHR$(27) : TB = 25 : DIM TBS(256)
50 '
60 'Display MAIN menu.
70 TITLE$ = "MAIN MENU"
80 GOSUB 2280
90 PRINT TAB(TB) "0. Exit."
100 PRINT TAB(TB) "1. Select CHARACTER SET."
110 PRINT TAB(TB) "2. Select PRINTING MODES."
120 PRINT TAB(TB) "3. Select PITCH."
130 PRINT TAB(TB) "4. Select LINE SPACING."
140 PRINT TAB(TB) "5. Set MARGINS, TABS & FORMS."
150 GOSUB 2370
160 IF S<0 OR S>5 THEN PRINT CHR$(7) : GOTO 150
170 IF S = 0 THEN PRINT CHR$(26); : END
180 ON S GOSUB 200,470,340,1230,630
190 GOTO 50
```

```
200 '
210 'Subroutine to display CHARACTER SET menu.
220 TITLE$ = "CHARACTER SET MENU"
230 GOSUB 2280
240 PRINT TAB(TB) "0. Return to main menu."
250 PRINT TAB(TB) "1. Select NLQ character set."
260 PRINT TAB(TB) "2. Cancel NLQ character set."
270 PRINT TAB(TB) "3. Select ITALIC character set."
280 PRINT TAB(TB) "4. Cancel ITALIC character set."
290 GOSUB 2370
300 IF S<0 OR S>4 THEN PRINT CHR$(7) : GOTO 290
310 IF S = 0 THEN RETURN
320 ON S GOSUB 1170,1200,1580,1610
330 GOTO 200
340 '
350 'Subroutine to display PITCHES menu.
360 TITLE$ = "PITCHES MENU"
370 GOSUB 2280
380 PRINT TAB(TB) "0. Return to main menu."
390 PRINT TAB(TB) "1. Select PICA pitch."
400 PRINT TAB(TB) "2. Select ELITE pitch."
410 PRINT TAB(TB) "3. Select CONDENSED pitch."
420 GOSUB 2370
430 IF S<0 OR S>3 THEN PRINT CHR$(7) : GOTO 420
440 IF S = 0 THEN RETURN
450 ON S GOSUB 810,840,870
460 GOTO 340
470 '
480 'Subroutine to display PRINTING MODES menu.
490 TITLE$ = "PRINTING MODES MENU"
500 GOSUB 2280
510 PRINT TAB(TB) "0. Return to main menu."
520 PRINT TAB(TB) "1. Select EXPANDED mode."
530 PRINT TAB(TB) "2. Cancel EXPANDED mode."
540 PRINT TAB(TB) "3. Select EMPHASIZED mode."
550 PRINT TAB(TB) "4. Cancel EMPHASIZED mode."
560 PRINT TAB(TB) "5. Select DOUBLE-STRIKE mode."
570 PRINT TAB(TB) "6. Cancel DOUBLE-STRIKE mode."
580 GOSUB 2370
590 IF S<0 OR S>6 THEN PRINT CHR$(7) : GOTO 580
600 IF S = 0 THEN RETURN
610 ON S GOSUB 1520,1550,2160,2190,2220,2250
620 GOTO 470
630 '
```

```
640 'Subroutine to display MARGINS, TABS & FORMS menu.
650 TITLE$ = "MARGINS, TABS & FORMS MENU"
660 GOSUB 2280
670 PRINT TAB(TB) "0. Return to main menu."
680 PRINT TAB(TB) "1. Set HORIZONTAL TABS."
690 PRINT TAB(TB) "2. Set VERTICAL TABS."
700 PRINT TAB(TB) "3. Set LEFT MARGIN."
710 PRINT TAB(TB) "4. Set RIGHT MARGIN."
720 PRINT TAB(TB) "5. Set TOP MARGIN."
730 PRINT TAB(TB) "6. Set BOTTOM MARGIN."
740 PRINT TAB(TB) "7. Cancel TOP & BOTTOM MARGINS."
750 PRINT TAB(TB) "8. Set PAGE LENGTH."
760 GOSUB 2370
770 IF S<0 OR S>8 THEN PRINT CHR$(7) : GOTO 760
780 IF S = 0 THEN RETURN
790 ON S GOSUB 1810,2120,900,960,1020,1080,1140,1640
800 GOTO 630
810 '
820 'Subroutine to select PICA pitch.
830 S$ = ESC$ + "B" + CHR$(1) : GOSUB 2460 : RETURN
840 '
850 'Subroutine to select ELITE pitch.
860 S$ = ESC$ + "B" + CHR$(2) : GOSUB 2460 : RETURN
870 '
880 'Subroutine to select CONDENSED pitch.
890 S$ = ESC$ + "B" + CHR$(3) : GOSUB 2460 : RETURN
900 '
910 'Subroutine to set LEFT MARGIN.
920 GOSUB 2500
930 INPUT "Enter new left margin (1-255)" ; X
940 IF X < 1 OR X > 255 THEN PRINT CHR$(7) : GOTO 920
950 S$ = ESC$ + "M" + CHR$(X) : GOSUB 2460 : RETURN
960 '
970 'Subroutine to set right MARGIN
980 GOSUB 2500
990 INPUT "Enter new right margin (1-255)" ; X
1000 IF X < 1 OR X > 255 THEN PRINT CHR$(7) : GOTO 980
1010 S$ = ESC$ + "Q" + CHR$(X) : GOSUB 2460 : RETURN
1020 '
1030 'Subroutine to set TOP MARGIN.
1040 GOSUB 2500
1050 INPUT "Enter new top margin (1-16)" ; X
1060 IF X < 1 OR X > 16 THEN PRINT CHR$(7) : GOTO 1040
1070 S$ = ESC$ + "R" + CHR$(X) : GOSUB 2460 : RETURN
```

```
1080 '
1090 'Subroutine to set BOTTOM MARGIN.
1100 GOSUB 2500
1110 INPUT "Enter new bottom margin (1-127)" ; X
1120 IF X < 1 OR X > 127 THEN PRINT CHR$(7) : GOTO 1100
1130 S$ = ESC$ + "N" + CHR$(X) : GOSUB 2460 : RETURN
1140 '
1150 'Subroutine to CANCEL TOP & BOTTOM MARGINS.
1160 S$ = ESC$ + "O" : GOSUB 2460 : RETURN
1170 '
1180 'Subroutine to select NLQ character set.
1190 S$ = ESC$ + "B" + CHR$(4) : GOSUB 2460 : RETURN
1200 '
1210 'Subroutine to cancel NLQ character set.
1220 S$ = ESC$ + "B" + CHR$(5) : GOSUB 2460 : RETURN
1230 '
1240 'Subroutine to select LINE SPACING.
1250 TITLE$ = "LINE SPACING MENU"
1260 GOSUB 2280
1270 PRINT TAB(TB) "0. Return to main menu."
1280 PRINT TAB(TB) "1. Select 1/6 inch line spacing."
1290 PRINT TAB(TB) "2. Select 1/8 inch line spacing."
1300 PRINT TAB(TB) "3. Select 7 dot graphics spacing."
1310 PRINT TAB(TB) "4. Select n/144 inch spacing."
1320 GOSUB 2370
1330 IF S<0 OR S>4 THEN PRINT CHR$(7) : GOTO 1320
1340 IF S = 0 THEN RETURN
1350 ON S GOSUB 1370,1400,1430,1460
1360 GOTO 1230
1370 '
1380 'Subroutine to select 1/6 inch line spacing.
1390 S$ = ESC$ + "2" : GOSUB 2460 : RETURN
1400 '
1410 'Subroutine to select 1/8 inch line spacing.
1420 S$ = ESC$ + "0" : GOSUB 2460 : RETURN
1430 '
1440 'Subroutine to select 7 dot graphics spacing.
1450 S$ = ESC$ + "1" : GOSUB 2460 : RETURN
1460 '
1470 'Subroutine to select n/144 inch line spacing.
1480 GOSUB 2500
1490 INPUT "Enter line space in 1/144 ths of an inch"; X
1500 IF X < 0 OR X > 255 THEN PRINT CHR$(7) : GOTO 1480
1510 S$ = ESC$ + "3" + CHR$(X) : GOSUB 2460 : RETURN
```

```
1520 '
1530 'Subroutine to select EXPANDED print.
1540 S$ = ESC$ + "W" + CHR$(1) : GOSUB 2460 : RETURN
1550 '
1560 'Subroutine to cancel EXPANDED printing.
1570 S$ = ESC$ + "W" + CHR$(0) : GOSUB 2460 : RETURN
1580 '
1590 'Subroutine to select ITALIC character set.
1600 S$ = ESC$ + "4" : GOSUB 2460 : RETURN
1610 '
1620 'Subroutine to cancel ITALIC character set.
1630 S$ = ESC$ + "5" : GOSUB 2460 : RETURN
1640 '
1650 'Subroutine to set PAGE LENGTH.
1660 GOSUB 2500
1670 PRINT "Page length in Inches or Lines (I,L)?"
1680 PRINT TAB(TB) ;
1690 A$ = INKEY$ : IF A$ = "" THEN 1690
1700 IF A$ = "I" OR A$ ="i" THEN 1730
1710 IF A$ = "L" OR A$ ="l" THEN 1770
1720 PRINT CHR$(7) : GOTO 1690
1730 INPUT "Length of page in inches (1-32)" ; X
1740 IF X < 1 OR X > 32 THEN PRINT CHR$(7) : GOTO 1660
1750 S$ = ESC$ + "C" + CHR$(0) + CHR$(X)
1760 GOSUB 2460 : RETURN
1770 INPUT "Length of page in lines (1-127)" ; X
1780 IF X < 1 OR X > 127 THEN PRINT CHR$(7) : GOTO 1660
1790 S$ = ESC$ + "C" + CHR$(X)
1800 GOSUB 2460 : RETURN
1810 '
1820 'Subroutine to set HORIZONTAL TABS.
1830 S$ = ESC$ + "D" : MAX = 255 : GOSUB 1840 : RETURN
1840 '
1850 'Subroutine to set tabs, either horiz or vert.
1860 GOSUB 2500
1870 PRINT "Would you like to set the tabs in"
1880 PRINT TAB(TB) "Regular intervals, or specify"
1890 PRINT TAB(TB) "each one Individually (R,I)"
1900 A$ = INKEY$ : IF A$ = "" THEN 1900
1910 IF A$ = "R" OR A$ = "r" THEN 2060
1920 IF A$ = "I" OR A$ = "i" THEN 1940
1930 PRINT CHR$(7) : GOTO 1840
1940 PRINT : I = 2 : TBS(1) = -1
1950 PRINT TAB(TB) "Enter the list of tabs, in"
```

```
1960 PRINT TAB(TB) "ascending order. No more than" MAX
   "."
1970 PRINT TAB(TB); : INPUT "Enter a tab" ; TBS(I)
1980 IF TBS(I) < 0 OR TBS(I) > 255 THEN 1930
1990 IF TBS(I) = 0 THEN I = 1 : GOTO 2030
2000 IF TBS(I) <= TBS(I-1) THEN 1930
2010 I = I + 1 : IF I > MAX THEN 1930
2020 GOTO 1970
2030 I = I + 1
2040 S$ = S$ + CHR$(TBS(I)) : IF TBS(I) <> 0 THEN 2030
2050 S$ = S$ + CHR$(0) : GOSUB 2460 : RETURN
2060 PRINT : PRINT TAB(TB) ; : INPUT "Enter interval" ;
   X
2070 IF X < 0 OR X > 255 THEN 1930
2080 FOR I = 1 TO 255 STEP X
2090 MAX = MAX - 1 : IF MAX = 0 THEN 2110
2100 S$ = S$ + CHR$(I) : NEXT I
2110 S$ = S$ + CHR$(0) : GOSUB 2460 : RETURN
2120 '
2130 'Subroutine to set VERTICAL TABS.
2140 S$ = ESC$ + "P" : MAX = 20 : GOSUB 1840
2150 RETURN
2160 '
2170 'Subroutine to select EMPHASIZED printing.
2180 S$ = ESC$ + "E" : GOSUB 2460 : RETURN
2190 '
2200 'Subroutine to cancel EMPHASIZED printing.
2210 S$ = ESC$ + "F" : GOSUB 2460 : RETURN
2220 '
2230 'Subroutine to select DOUBLE-STRIKE printing.
2240 S$ = ESC$ + "G" : GOSUB 2460 : RETURN
2250 '
2260 'Subroutine to cancel DOUBLE-STRIKE printing.
2270 S$ = ESC$ + "H" : GOSUB 2460 : RETURN
2280 '
2290 'Subroutine to print a menu title.
2300 PRINT CHR$(26);
2310 PRINT : PRINT : PRINT
2320 PRINT TAB(27) "--- RADIX PRINTER SETUP ---"
2330 PRINT
2340 PRINT TAB((80-LEN(TITLE$))/2) TITLE$
2350 PRINT : PRINT
2360 RETURN
2370 '
2380 'Subroutine to input menu selection.
```

```
2390 PRINT ESC$ "=" CHR$(20+32) CHR$(18+32);
2400 PRINT "Enter selection or press P for print
   sample."
2410 C$ = INKEY$ : IF C$ = "" THEN 2410
2420 IF C$ = "P" OR C$ = "p" THEN GOSUB 2530 : GOTO 2370
2430 IF C$ < "0" OR C$ > "9" THEN PRINT CHR$(7) : GOTO
   2390
2440 S = VAL(C$)
2450 RETURN
2460 '
2470 'Subroutine to output command string.
2480 LPRINT S$ ;
2490 RETURN
2500 '
2510 'Subroutine to clear screen & position cursor.
2520 PRINT CHR$(26) ESC$ "=" CHR$(10+32) CHR$(TB+32); :
   RETURN
2530 '
2540 ' Subroutine to print sample
2550 FOR I = 1 TO 4 : FOR J = 33 TO 127
2560 LPRINT CHR$(J) ;
2570 NEXT : LPRINT : NEXT
2580 RETURN
```

Appendix F
The Parallel Interface

Radix has both a parallel interface and a serial interface to communicate with the computer that it is connected to. The operating specifications of the parallel interface are as follows:

Data transfer rate:	1,000 to 6,000 characters per second
Synchronization:	Via externally supplied $\overline{\text{STROBE}}$ pulses
Handshaking:	$\overline{\text{ACK}}$ and BUSY signals
Logic level:	Compatible with TTL level

Radix's parallel interface connects to the computer by a 36 pin connector on the back of the printer. This connector mates with an Amphenol 57-30360 connector. The functions of the various pins are summarized in Table F-1.

Functions of the Connector Signals

Communications between the computer and the Radix use many of the pins of the connector. To understand how the system of communications works we need to look at the functions of the various signals carried by the pins of the interface connector.

Pin 1 carries the $\overline{\text{STROBE}}$ pulse signal from the computer to the printer. This signal is normally held high by the computer. When the computer has data ready for the printer it sets this signal to a low value for at least 0.5 microseconds. When the printer sees this pulse on the strobe pin, it reads the data that the computer supplies on pins 2 through 9. Each of these lines carries one bit of information. A logical "1" is represented by a high signal level, and a logical "0" is represented by a low signal level. The computer must maintain these signals for a period beginning at least 0.5 microseconds before the strobe pulse starts and continuing for at least 0.5 microseconds after the strobe pulse ends.

When the Radix has successfully received the byte of data from the computer it sets pin 10 low for approximately 9 microseconds. This signal acknowledges the receipt of the data and so is called the $\overline{\text{ACK}}$ (for "acknowledge") signal.

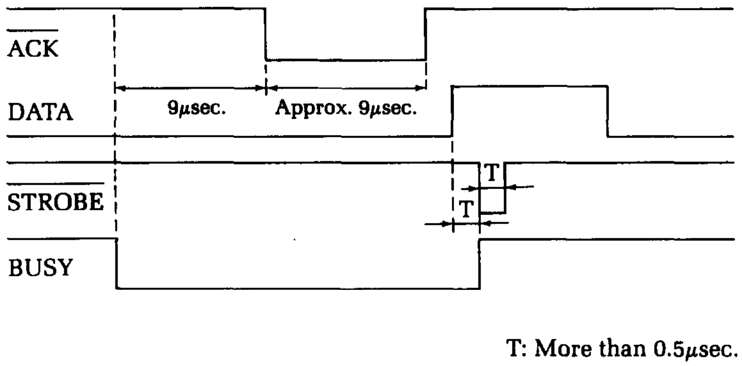


Figure F-1. *Radix interface timing diagram.*

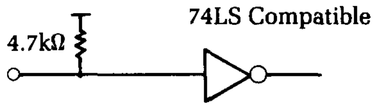
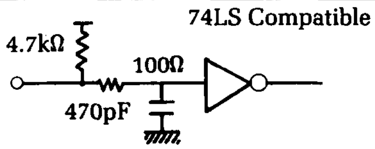
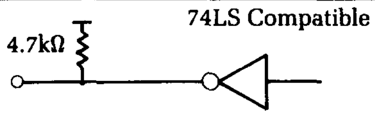
Signal Name	Circuit Example
DATA 1 - DATA 8 (To Printer)	
$\overline{\text{STROBE}}$ (To Printer)	
BUSY, $\overline{\text{ACK}}$ (From Printer)	

Figure F-2. *Typical interface circuit.*

Table F-1
Parallel interface pin functions

Pin No.	Signal Name	Direction	Function
1	$\overline{\text{STROBE}}$	IN	Signals when data is ready to be read. Signal goes from HIGH to LOW (for at least 0.5 microseconds) when data is available.
2	DATA1	IN	These signals provide the information of the first to eighth bits of parallel data. Each signal is at a HIGH level for a logical 1 and at a LOW level for a logical 0.
3	DATA2	IN	
4	DATA3	IN	
5	DATA4	IN	
6	DATA5	IN	
7	DATA6	IN	
8	DATA7	IN	
9	DATA8	IN	
10	$\overline{\text{ACK}}$	OUT	A 9 microsecond LOW pulse acknowledges receipt of data.
11	BUSY	OUT	When this signal goes LOW the printer is ready to accept data.
12	PAPER OUT	OUT	This signal is normally LOW. It will go HIGH if Radix runs out of paper. This signal can be held LOW permanently by turning DIP switch C-1 off.
13	SELECTED	OUT	This signal is HIGH when the printer is on-line.
14-15	N/C		Unused.
16	SIGNAL GND		Signal ground.
17	CHASSIS GND		Printer's chassis ground, isolated from logic ground.
18	+5VDC	OUT	External supply of +5VDC.
19-30	GND		Twisted pair return signal ground level.
31	$\overline{\text{RESET}}$	IN	When this signal goes LOW the printer is reset to its power-on condition.
32	$\overline{\text{ERROR}}$	OUT	This signal is normally HIGH. This signal goes LOW to signal that the printer cannot print due to an error condition.
33	EXT GND		External ground.
34-36	N/C		Unused.

Pin 11 reports when the Radix is not able to receive data. The signal is called BUSY. When this signal is high, Radix cannot receive data. This signal will be high during data transfer, when the printer is off-line and when an error condition exists.

Radix will report that it has run out of paper by making the PAPER OUT signal on pin 12 high. This pin can be held low by turning DIP switch C-1 off. When the printer is in the on-line state pin 13 is held high. This signal (SELECTED) tells the computer that the printer is ready to receive data.

Pins 14, 15, and 34-36 are not used, while pins 16, 17, 19-30 and 33 are grounded. Pin 18 is connected to the +5VDC supply in the printer.

Pin 31 can be used to reset the printer. If this signal ($\overline{\text{RESET}}$) goes low the printer will reinitialize. Pin 32 is used to report error conditions in the printer. This signal ($\overline{\text{ERROR}}$) is high during normal operation and goes low to report that the printer cannot print due to an error condition.

Appendix G

Serial Interface Specifications

Radix provides a very flexible RS232C serial interface. It can communicate at rates from 150 to 19,200 baud (bits per second) and supports four different kinds of *handshaking*. This interface can also function as a 20mA current loop interface. The operating specifications of the interface are as follows:

Data transfer rate:	150-19200
Word length:	1 start bit
	7 or 8 data bits
	Odd, even or no parity
	1 or 2 stop bits
Signal levels:	Mark or logical 1, − 3 to − 15 volts or current ON
	Space or logical 0, + 3 to + 15 volts or current OFF
Handshaking:	Serial busy, 1 byte mode
	Serial busy, 1 block mode
	ACK mode
	XON/XOFF mode

Note: 19200 baud can be used only with a RS232C interface; it cannot be used with a 20mA current loop interface.

Radix has a DB-25 female connector on the back to connect to a computer. The functions of the pins are summarized in Table G-1.

Configuring the Serial Interface

DIP switch B controls the configuration of the serial interface. Switch B is located under Radix's front cover. Table G-2 describes the functions of the individual switches in DIP switch B.

Table G-1
Serial interface pin functions

Pin No.	Signal Name	Direction	Function
1	GND	—	Printer's chassis ground.
2	TXD	OUT	This pin carries data from the printer.
3	RXD	IN	This pin carries data to the printer.
4	RTS	OUT	This is ON when the printer is ready to receive data.
5	CTS	IN	This pin is ON when the computer is ready to send data.
6	DSR	IN	This pin is ON when the computer is ready to send data. Radix does not check this pin.
7	GND	—	Signal ground.
8	DCD	IN	This pin is ON when the computer is ready to send data.
9	TTY TXDR	—	This pin is the return path for data transmitted from the printer on the 20mA current loop.
10	TTY TXD	OUT	This pin carries data from the printer on the 20mA current loop.
11	RCH	OUT	This is the signal line for the serial busy protocols. This pin goes OFF when Radix's buffer fills, and ON when Radix is ready to receive data. In the busy protocols this line carries the same signal as pin 20.
12	N/C		Unused
13	GND	—	Signal ground
14-16	N/C		Unused
17	TTY TXDR	—	This pin is the return path for data transmitted from the printer on the 20mA current loop.
18	TTY RXDR	—	This pin is the return path for data transmitted to the printer on the 20mA current loop.
19	TTY RXD	IN	This pin carries data to the printer on the 20mA current loop.
20	DTR	OUT	Radix turns this pin ON when it is ready to receive data.
21-22	N/C		Unused.
23	TTY RXDR	—	This pin is the return path for data transmitted to the printer on the 20mA current loop.
24	TTY TXD	OUT	This pin carries data from the printer on the 20mA current loop.
25	TTY RXD	IN	This pin carries data to the printer on the 20mA current loop.

Table G-2
DIP switch B

Switch	ON	OFF
B-1	2 stop bits	1 stop bit
B-2	7 data bits	8 data bits
B-3	Parity checked	No parity
B-4	Handshaking protocols—see Table G-3	
B-5		
B-6	Odd parity	Even parity
B-7	Data transfer rate—see Table G-4	
B-8		
B-9		
B-10	Not used	

Table G-3
Handshaking protocols

Protocol	Switch B-4	Switch B-5
Serial busy, 1 byte mode	OFF	OFF
Serial busy, 1 block mode	ON	OFF
ACK mode	OFF	ON
XON/XOFF mode	ON	ON

Table G-4
Data transfer rates

Baud rate	Switch B-7	Switch B-8	Switch B-9
150	OFF	OFF	OFF
300	OFF	OFF	ON
600	OFF	ON	OFF
1200	OFF	ON	ON
2400	ON	OFF	OFF
4800	ON	OFF	ON
9600	ON	ON	OFF
19200	ON	ON	ON

Radix's Serial Protocols

Radix has four serial protocols selected by DIP switches B-4 and B-5. Figure G-2 shows a typical byte of serial data and Figure G-3 shows timing charts for the 4 protocols.

Serial busy protocols

In the serial busy protocols, Radix uses DTR (pin 20) and RCH (pin 11) to signal to the computer when it is able to accept data. These two pins go ON when Radix is ready to accept data. In the 1 byte mode they go OFF after each character is received. In the 1 block mode they only go OFF when Radix's buffer approaches capacity. In both cases they will stay OFF if the buffer is too full to accept more data.

XON/XOFF protocol

The XON/XOFF protocol uses the ASCII characters ⟨DC1⟩ and ⟨DC3⟩ (sometimes called XON and XOFF, respectively) to communicate with the computer. When Radix's buffer approaches capacity Radix will send a DC3 (ASCII 19) on TXD (pin 2) to tell the computer that it must stop sending data. When Radix is able to receive more data it sends a DC1 (ASCII 17) on TXD. The computer can then send more data until Radix sends another DC3.

ACK protocol

In the ACK protocol, Radix sends an ACK (ASCII 6) on TXD (pin 2) each time that it is prepared to receive a byte of data.

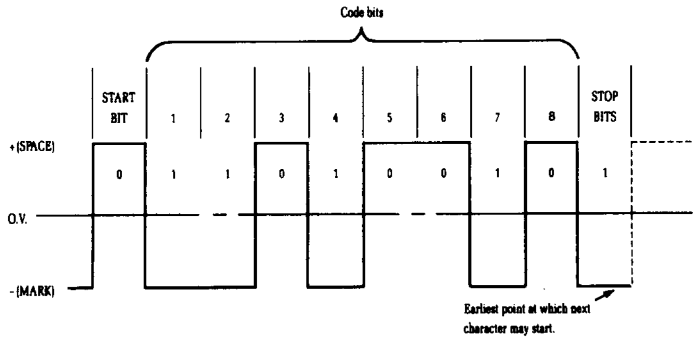


Figure G-1. *Typical data byte on the serial interface.*

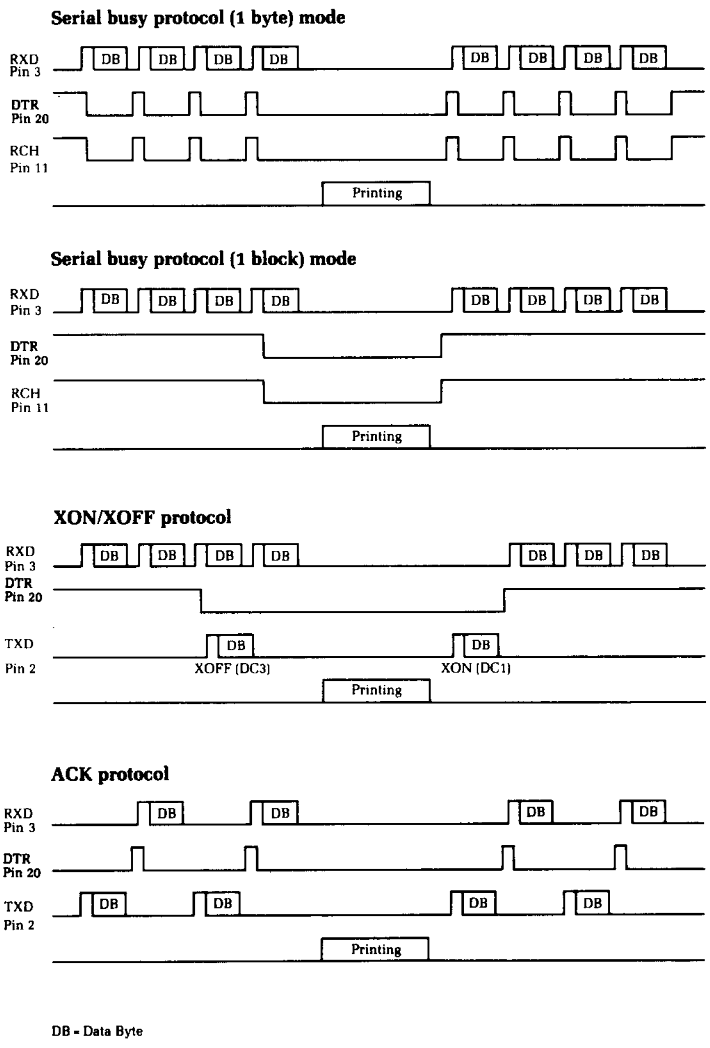


Figure G-2. *Serial protocol timing charts.*

Appendix H

DIP Switch Settings

The DIP (dual in-line package) switches control some of the functions of Radix. A DIP switch actually contains several individual switches. Radix has one DIP switch with 8 individual switches in it, one with 10 individual switches, and one DIP switch with 4 individual switches. Figure H-1 is a drawing of a typical DIP switch.

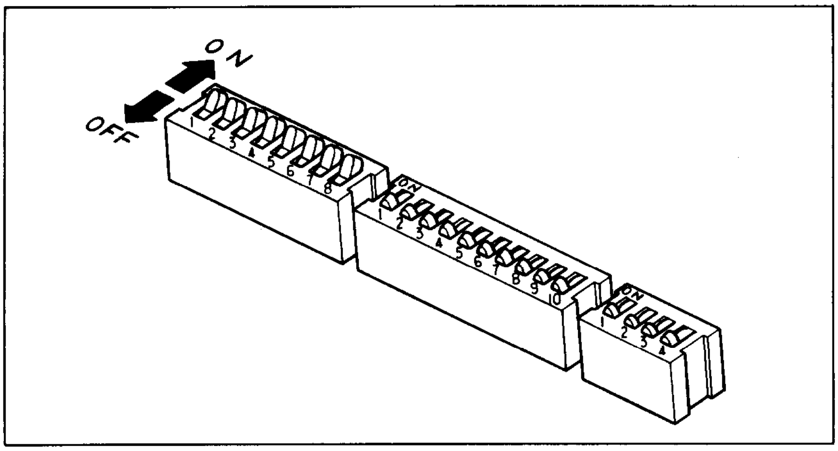


Figure H-1. *A DIP switch is actually a series of several small switches.*

All three DIP switches are readily accessible from the top. They are located in the compartment with the print head, and can be seen by opening the front cover. To change the setting of a switch, use a ballpoint pen or a similar object. The "on" position for a switch is towards the back of the printer; "off" is towards the front.

Never change the settings of any of the DIP switches when the power is on. Turn off both the printer and your computer.

Table H-1 summarizes the functions of DIP switches A and C.

DIP switch B controls the serial interface and is covered in Appendix G. The individual switches on DIP switch A are called A-1 through A-8; those on switch C are C-1 through C-4.

Table H-1
DIP switch settings

Switch	ON	OFF
Switch A		
A-1	11″ page length	12″ page length
A-2	Normal print	Emphasized print
A-3	10 CPI (pica pitch)	17 CPI (condensed pitch)
A-4	Normal	NLQ
A-5	1/6″ line feed	1/8″ line feed
A-6	International character set selection — see Table H-2	
A-7		
A-8		
Switch C		
C-1	Paper-out detector on	Ignore paper-out
C-2	Serial interface	Parallel interface
C-3	7-bit interface	8-bit interface
C-4	Auto LF with CR	LF must be from host

DIP switch A controls the default settings for printing functions. DIP switch C controls the interface.

Switch Functions

Switch	**Function**
A-1	Switch A-1 sets the default page length for Radix. If switch A-1 is ON, the page length is set to 11″. When switch A-1 is OFF the page length is set to 12″. This switch is set ON at the factory.
A-2	This switch selects either normal or emphasized print for the default. If this switch is ON then Radix will print normal type when the power is turned on. If this switch is OFF then Radix will print emphasized type when the power is turned on. This switch is set ON at the factory. This switch has no effect if switch A-4 is off.

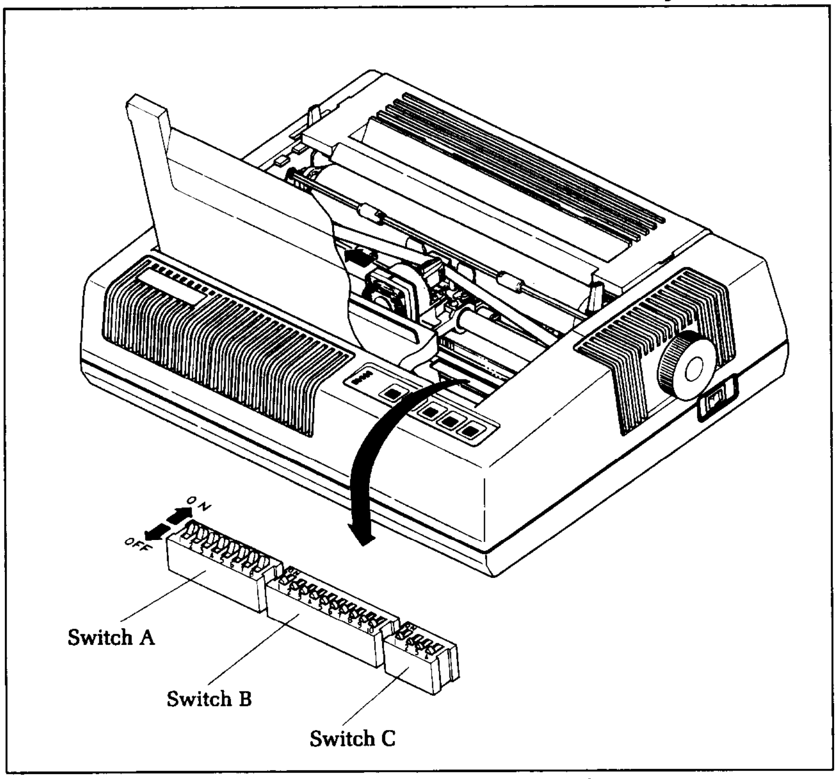


Figure H-2. *Radix's DIP switches are located under the front cover.*

A-3	This switch selects the default character pitch. If this switch is ON the default pitch is 10 CPI. If this switch is OFF the default pitch is 17 CPI. This switch is set ON at the factory. This switch has no effect if switch A-4 is off.
A-4	Switch A-4 selects the default character style. If this switch is ON then the default character style is normal characters. If this switch is OFF then the default character style is near letter quality. If this switch is OFF then switches A-2 and A-3 have no effect. This switch is set ON at the factory.
A-5	This switch sets the default line spacing. When this switch is ON the default line spacing is set to 1/6 inch. This means that Radix will advance the paper 1/6 inch each time it receives a line feed. When this switch is OFF the default line spacing is 1/8 inch. This switch is set ON at the factory.

A-6 – A-8 These three switches determine the default international character set as shown in Table H-2. These switches are all set ON at the factory.

Table H-2
International character sets

Switch	USA	England	Germany	Denmark	France	Sweden	Italy	Spain
A-6	ON	OFF	ON	OFF	ON	OFF	ON	OFF
A-7	ON	ON	OFF	OFF	ON	ON	OFF	OFF
A-8	ON	ON	ON	ON	OFF	OFF	OFF	OFF

C-1 This switch disables the paper-out sensor. If this switch is ON the printer will signal the computer when it runs out of continuous paper and will stop printing. If this switch is OFF the printer will ignore the paper-out sensor and will continue printing. This switch is set ON at the factory.

C-2 This switch selects the active interface. Turn this switch ON to use the serial interface. Turn this switch OFF to use the parallel interface. This switch is set OFF at the factory.

C-3 This switch controls the eighth bit of the parallel interface. If this switch is ON the printer will only read the first seven bits on the parallel interface and ignores the eighth bit. If this switch is OFF all eight bits will be read. This switch is set OFF at the factory.

C-4 When this switch is ON, Radix will automatically advance the paper one line every time it receives a carriage return. When this switch is OFF, the computer must send a line feed command every time the paper is to advance. (Most BASICs send a line feed with every carriage return, therefore, this switch should usually be off.) This switch is set OFF at the factory.

Appendix I

ASCII Codes

Standard and Italic Characters

Decimal	Character		Function	Decimal	Character		
0	NUL		End tab settings	47	/	/	
7	BEL		Bell	48	0	0	
8	BS		Backspace	49	1	1	
9	HT		Horizontal tab	50	2	2	
10	LF		Line feed	51	3	3	
11	VT		Vertical tab	52	4	4	
12	FF		Form feed	53	5	5	
13	CR		Carriage return	54	6	6	
14	SO		Expanded print on	55	7	7	
15	SI		Condensed print on	56	8	8	
17	DC1		On line	57	9	9	
18	DC2		Pica pitch	58	:	:	
19	DC3		Off line	59	;	;	
20	DC4		Expanded print off	60	<	<	
27	ESC		Escape	61	=	=	
30	RS		End macro	62	>	>	
32			Space	63	?	?	
33	!	!		64	@	@	*
34	"	"		65	A	A	
35	#	#	*	66	B	B	
36	$	$		67	C	C	
37	%	%		68	D	D	
38	&	&		69	E	E	
39	'	'	Apostrophe	70	F	F	
40	(	(		71	G	G	
41	)	)		72	H	H	
42	*	*		73	I	I	
43	+	+		74	J	J	
44	,	,	Comma	75	K	K	
45	-	-	Hyphen	76	L	L	
46	.	.	Period	77	M	M	

*These characters may be different if you are using an international character set other than the USA set. The characters for each set are shown on the next page.

Decimal	Character		
78	N	N	
79	O	O	
80	P	P	
81	Q	Q	
82	R	R	
83	S	S	
84	T	T	
85	U	U	
86	V	V	
87	W	W	
88	X	X	
89	Y	Y	
90	Z	Z	
91	[	[	*
92	\	\	*
93	]	]	*
94	^	^	*
95	_	_	
96	`	`	*
97	a	a	
98	b	b	
99	c	c	
100	d	d	
101	e	e	
102	f	f	

Decimal	Character		
103	g	g	
104	h	h	
105	i	i	
106	j	j	
107	k	k	
108	l	l	
109	m	m	
110	n	n	
111	o	o	
112	p	p	
113	q	q	
114	r	r	
115	s	s	
116	t	t	
117	u	u	
118	v	v	
119	w	w	
120	x	x	
121	y	y	
122	z	z	
123	¨	¨	*
124	ñ	ñ	*
125	}	}	*
126	~	~	*
127	DEL		Delete

*These characters may be different if you are using an international character set other than the USA set. The characters for each set are shown below.

International Character Sets

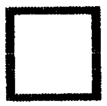
Decimal	USA	England	Germany	Denmark	France	Sweden	Italy	Spain
35	#	£	#	#	£	#	#	#
64	@	@	§	@	à	É	§	@
91	[	[	Ä	Æ	°	Ä	°	¡
92	\	\	Ö	Ø	ç	Ö	ç	Ñ
93	]	]	Ü	Å	§	Å	é	¿
94	^	^	^	^	^	Ü	^	^
96	`	`	`	`	`	é	ù	`
123	{	{	ä	æ	é	ä	à	¨
124	|	|	ö	ø	ù	ö	ò	ñ
125	}	}	ü	å	è	å	è	}
126	~	~	ß	~	¨	ü	ì	~

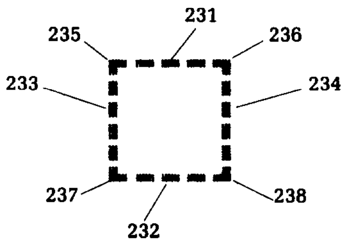
Special Characters

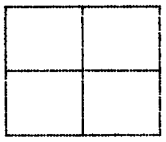
Decimal	Character	Function	Decimal	Character
128			184	Ι
135	BEL	Bell	185	σ
136	BS	Backspace	186	∞
137	HT	Horizontal tab	187	π
138	LF	Line feed	188	±
139	VT	Vertical tab	189	⊐
140	FF	Form feed	190	×
141	CR	Carriage return	191	÷
142	SO	Expanded print on	192	Ā
143	SI	Condensed print on	193	à
145	DC1	On line	194	ç
146	DC2	Pica pitch	195	£
147	DC3	Off line	196	ā
148	DC4	Expanded print off	197	µ
155	ESC	Escape	198	▫
158	RS	End macro	199	ᵓ
160	⌟		200	†
161	⌝		201	§
162	⌞		202	Ξ
163	⌜		203	⊗
164	↑		204	↳
165	↓		205	⌧
166	←		206	⊭
167	→		207	‖
168	○		208	¥
169	▲		209	Ä
170	▼		210	Ö
171	►		211	Ü
172	◄		212	¢
173	◇		213	Ñ
174	◆		214	ä
175	□		215	ö
176	Tr		216	ü
177	Å		217	ß
178	☺		218	ē
179	θ		219	é
180	⊏		220	ù
181	⊧		221	è
182	Ω		222	ñ
183	℧		223	ƒ

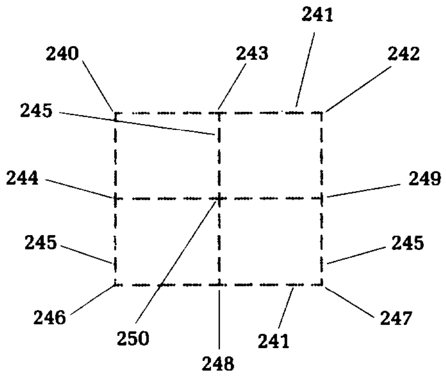
Block Graphics Characters

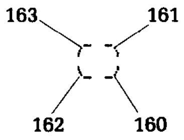
Decimal	Character	Decimal	Character
224	Space	240	┌
225	▘	241	─
226	▖	242	┐
227	▝	243	┬
228	▗	244	├
229	▚	245	│
230	▞	246	└
231	▀	247	┘
232	▄	248	┴
233	▌	249	┤
234	▐	250	┼
235	▛	251	◤
236	▜	252	◢
237	▙	253	◥
238	▟	254	◣
239	█	255	




231
235
236
233
234
237
238
232



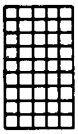
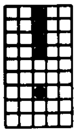
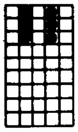
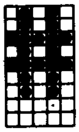
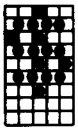
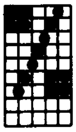
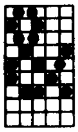
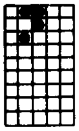
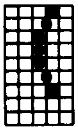
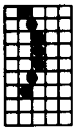
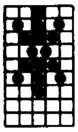
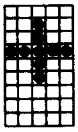
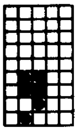
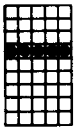
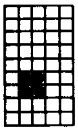
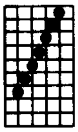
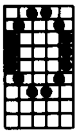
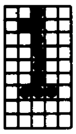
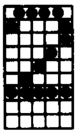
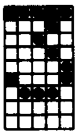
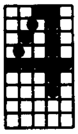
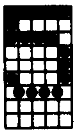
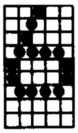
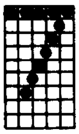
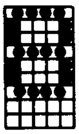
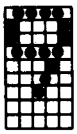
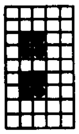
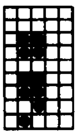

241
240
243
242
245
244
249
245
245
246
250
248
241
247

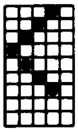

163
161
162
160

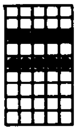
Appendix J

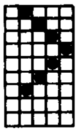
Character Style Charts

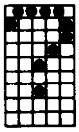
Standard Characters

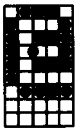
32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59


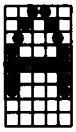
60


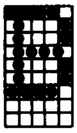
61


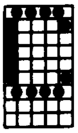
62


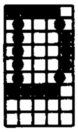
63


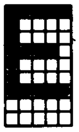
64


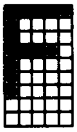
65


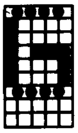
66


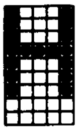
67


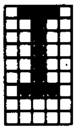
68


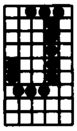
69


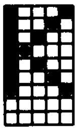
70


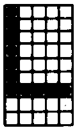
71


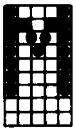
72


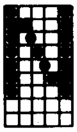
73


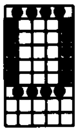
74


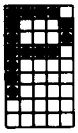
75


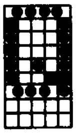
76


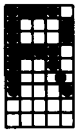
77


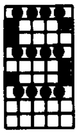
78


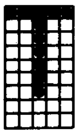
79


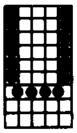
80


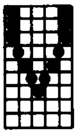
81


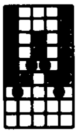
82


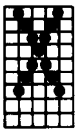
83


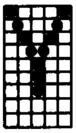
84


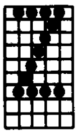
85


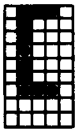
86


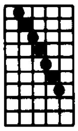
87


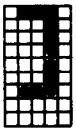
88


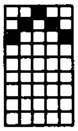
89


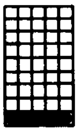
90


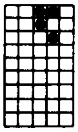
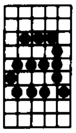
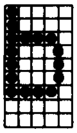
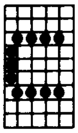
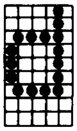
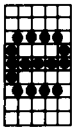
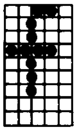
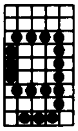
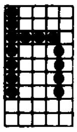
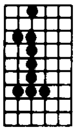
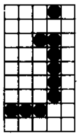
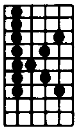
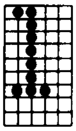
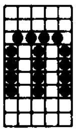
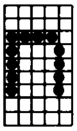
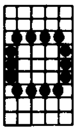
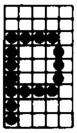
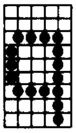
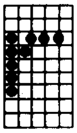
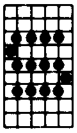
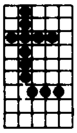
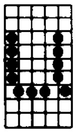
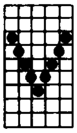
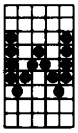
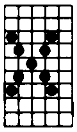
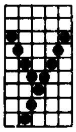
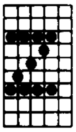
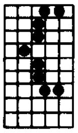
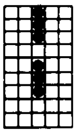
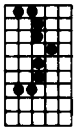
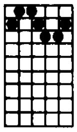
91


92


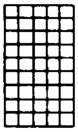
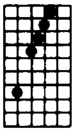
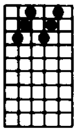
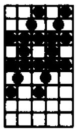
93


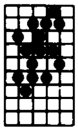
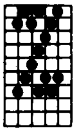
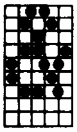
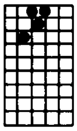
94


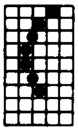
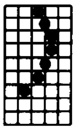
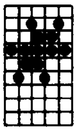
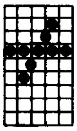
95


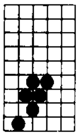
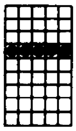
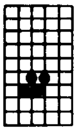
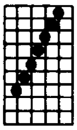
96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126


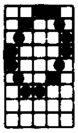
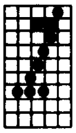
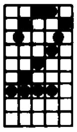
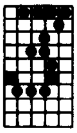
Italic Characters

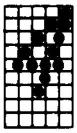
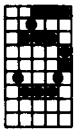
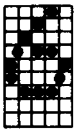
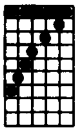
32 
33 
34 
35 

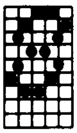
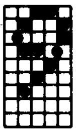
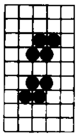
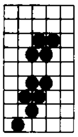
36 
37 
38 
39 

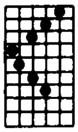
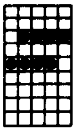
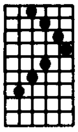
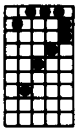
40 
41 
42 
43 

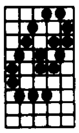
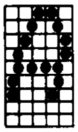
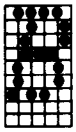
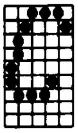
44 
45 
46 
47 

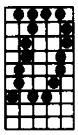
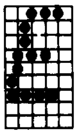
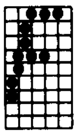
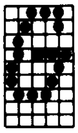
48 
49 
50 
51 

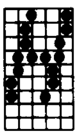
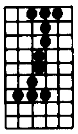
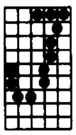
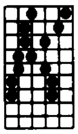
52 
53 
54 
55 

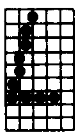
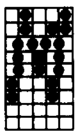
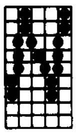
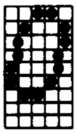
56 
57 
58 
59 

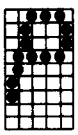
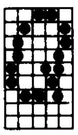
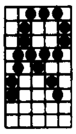
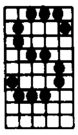
60 
61 
62 
63 

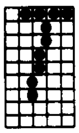
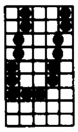
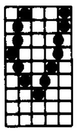
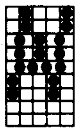
64 
65 
66 
67 

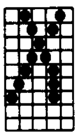
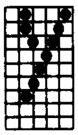
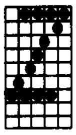
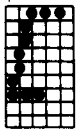
68 
69 
70 
71 

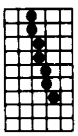
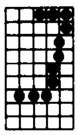
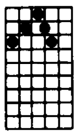
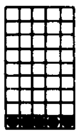
72 
73 
74 
75 

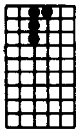
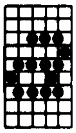
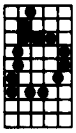
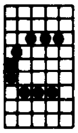
76 
77 
78 
79 

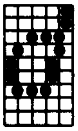
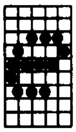
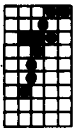
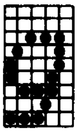
80 
81 
82 
83 

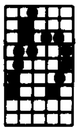
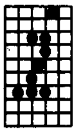
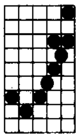
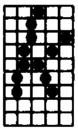
84 
85 
86 
87 

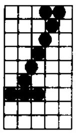
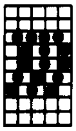
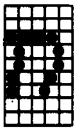
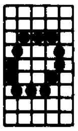
88 
89 
90 
91 

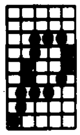
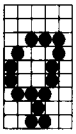
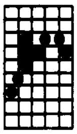
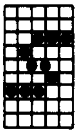
92 
93 
94 
95 

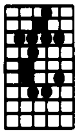
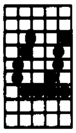
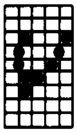
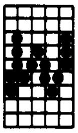
96 
97 
98 
99 

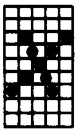
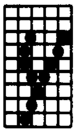
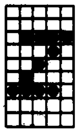
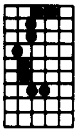
100 
101 
102 
103 

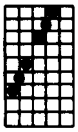
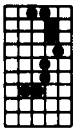
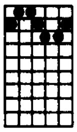
104 
105 
106 
107 

108 
109 
110 
111 

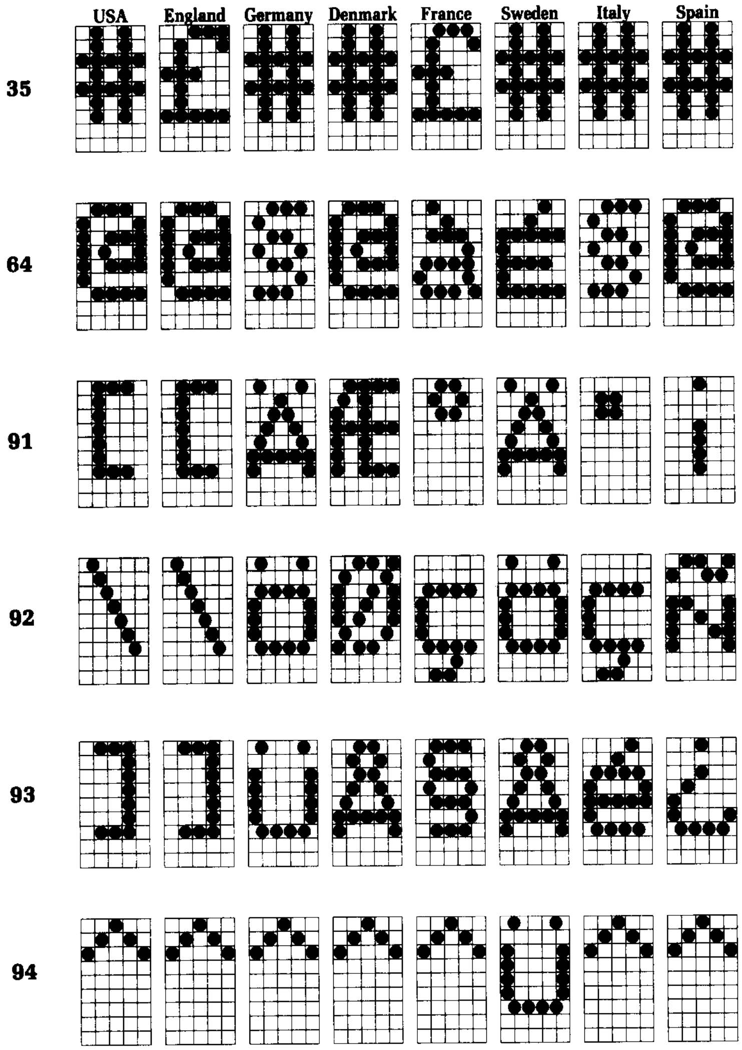
112 
113 
114 
115 

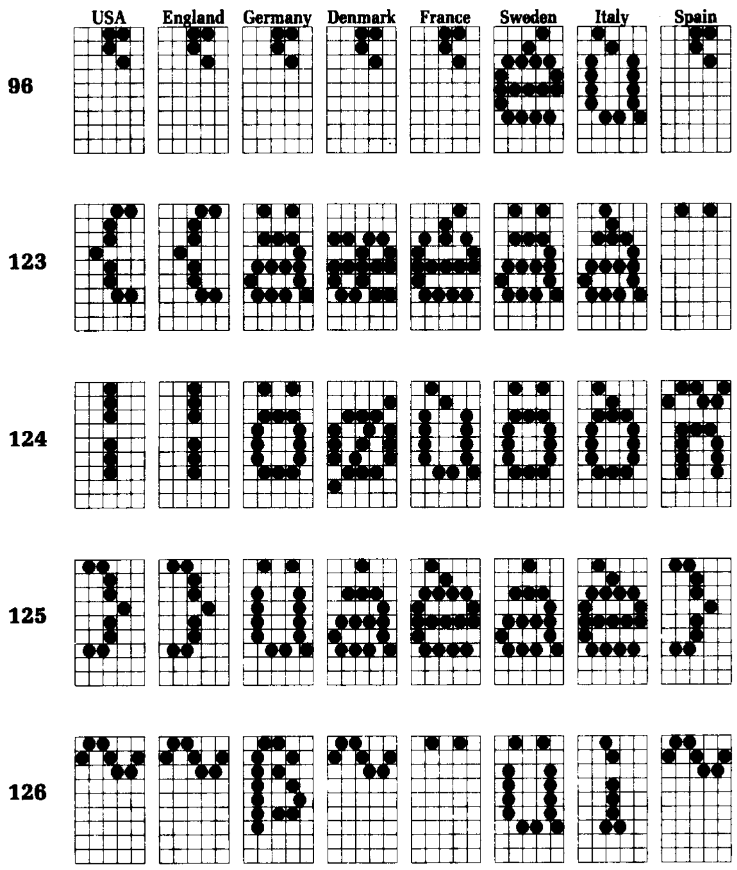
116 
117 
118 
119 

120 
121 
122 
123 

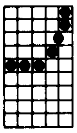
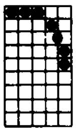
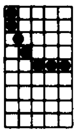
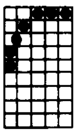
124 
125 
126 

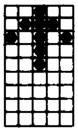
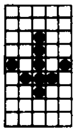
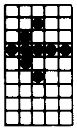
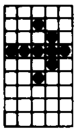
International Characters

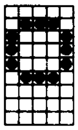
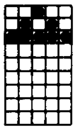
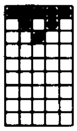
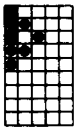


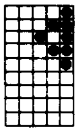
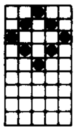
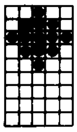
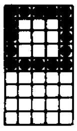

USA
England
Germany
Denmark
France
Sweden
Italy
Spain
96
123
124
125
126

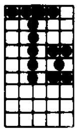
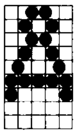
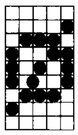
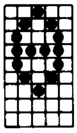
Special Characters

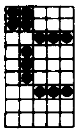
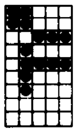
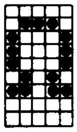
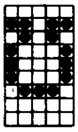
160 
161 
162 
163 

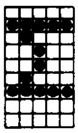
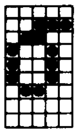
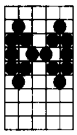
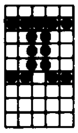
164 
165 
166 
167 

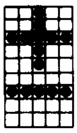
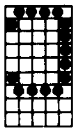
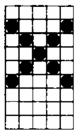
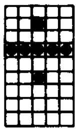
168 
169 
170 
171 

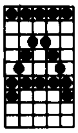
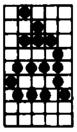
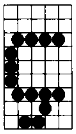
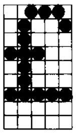
172 
173 
174 
175 

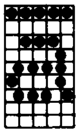
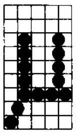
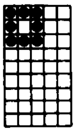
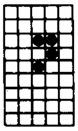
176 
177 
178 
179 

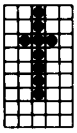
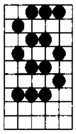
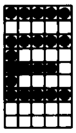
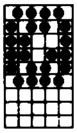
180 
181 
182 
183 

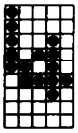
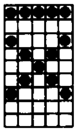
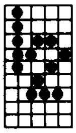
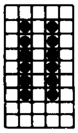
184 
185 
186 
187 

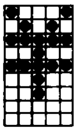
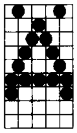
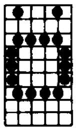
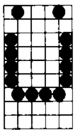
188 
189 
190 
191 

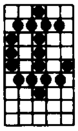
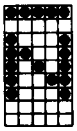
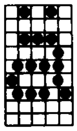
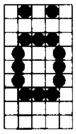
192 
193 
194 
195 

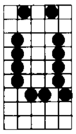
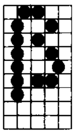
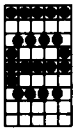
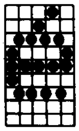
196 
197 
198 
199 

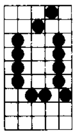
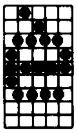
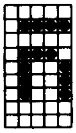
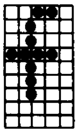
200 
201 
202 
203 

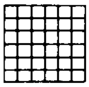
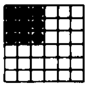
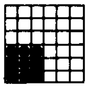
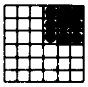
204 
205 
206 
207 

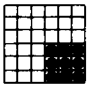
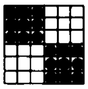
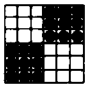
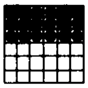
208 
209 
210 
211 

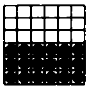
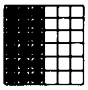
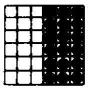
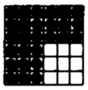
212 
213 
214 
215 

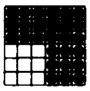
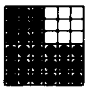
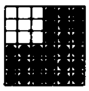
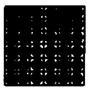
216 
217 
218 
219 

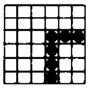
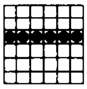
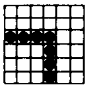
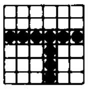
220 
221 
222 
223 

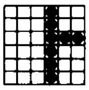
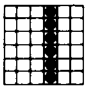
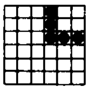
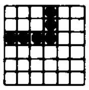
224 
225 
226 
227 

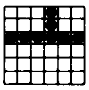
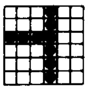
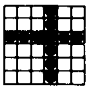
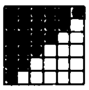
228 
229 
230 
231 

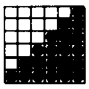
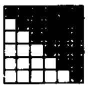
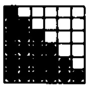
232 
233 
234 
235 

236 
237 
238 
239 

240 
241 
242 
243 

244 
245 
246 
247 

248 
249 
250 
251 

252 
253 
254 

Appendix K

Function Code Reference

The purpose of this Appendix is to provide a quick reference for the various functions available on the Radix-10 and Radix-15. The descriptions of the codes appear in the following format:

PURPOSE:	**Tells what the function code does.**
CODE:	Control code mnemonic
(decimal ASCII)	ASCII decimal equivalent
(hex ASCII)	Hexadecimal equivalent
REMARKS:	Details how the command is used.
REFERENCE:	Tells which chapter of the manual describes the command in greater detail

There are several commands that require that you specify a value (or values) to Radix. In these cases, we have used an italic "*n*" or "*m*" to indicate a variable. You should insert the ASCII code for proper value here.

Commands to Control Print Style

These commands are used to control the font style, the print pitch, and special effects.

Font style controls

PURPOSE:	**Select the standard character set.**	
CODE:	⟨ESC⟩	"5"
(decimal ASCII)	27	53
(hex ASCII)	1B	35

REMARKS: This command causes the printer to cancel the italic character set and select instead the standard character set. You can select the standard character set as the power-on default by turning DIP switch A-4 on.

REFERENCE: Chapter 7

PURPOSE:	**Select the italic character set.**	
CODE:	⟨ESC⟩	"4"
(decimal ASCII)	27	52
(hex ASCII)	1B	34

REMARKS: This command selects the italic character set.

REFERENCE: Chapter 7

PURPOSE: **Select an international character set.**

CODE:	⟨ESC⟩	"7"	n
(decimal ASCII)	27	55	n
(hex ASCII)	1B	37	n

REMARKS: This command causes the printer to select an international character set determined by the value of *n* as shown in the table below:

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

You can select a particular international character set as a power-on default by adjusting the settings of DIP switches A-6, A-7, and A-8.

REFERENCE: Chapter 10

PURPOSE: **Select the NLQ (Near Letter Quality) character set.**

CODE:	⟨ESC⟩	"B"	4
(decimal ASCII)	27	66	4
(hex ASCII)	1B	42	4

REMARKS: This command causes all subsequent printing to be done with the NLQ (Near Letter Quality) character set. This character set cannot be used in conjunction with other font styles or special print modes except for underlining. You can set NLQ characters as the power-on default by turning DIP switch A-4 off.

REFERENCE: Chapter 7

PURPOSE:	**Cancel the NLQ character set.**		
CODE:	⟨ESC⟩	"B"	5
(decimal ASCII)	27	66	5
(hex ASCII)	1B	42	05

REMARKS: This command causes the printer to cancel the NLQ character set and return to the standard (also known as "draft") character set.

REFERENCE: Chapter 7

Font pitch controls

PURPOSE:	**Set the print pitch to pica (10 characters/inch).**		
CODE:	⟨ESC⟩	"B"	1
(decimal ASCII)	27	66	1
(hex ASCII)	1B	42	01

REMARKS: This command causes all subsequent printing to be done in pica type. This command also sets the maximum number of print columns to 80 on the Radix-10 and 136 on the Radix-15. You can select pica type as the power-on default by turning DIP switch A-3 on.

REFERENCE: Chapter 7

PURPOSE:	**Set the print pitch to elite (12 characters/inch).**		
CODE:	⟨ESC⟩	"B"	2
(decimal ASCII)	27	66	2
(hex ASCII)	1B	42	02

REMARKS: This command causes all subsequent printing except NLQ characters to be done in elite type. This command also sets the maximum number of print columns to 96 on the Radix-10 and 163 on the Radix-15.

REFERENCE: Chapter 7

PURPOSE:	**Set the print pitch to condensed (17 characters/inch).**		
CODE:	⟨ESC⟩	"B"	3
(decimal ASCII)	27	66	3
(hex ASCII)	1B	42	03

REMARKS: This command causes all subsequent printing except NLQ characters to be done in condensed type of 17 characters per inch. This command also sets the maximum number of print columns to 136 on the Radix-10 and 233 on the Radix-15. You can select condensed type as the power-on default by turning DIP switch A-3 off.

REFERENCE: Chapter 7

PURPOSE:	**Set the print pitch to pica (10 characters/inch).**
CODE:	⟨DC2⟩
(decimal ASCII)	18
(hex ASCII)	12

REMARKS: This command is the same as ⟨ESC⟩ "B" 1, but can be used in applications where a single-character command is required.

REFERENCE: Chapter 7

PURPOSE:	**Set the print pitch to condensed (17 characters/inch).**
CODE:	⟨SI⟩
(decimal ASCII)	15
(hex ASCII)	0F

REMARKS: This command is the same as ⟨ESC⟩ "B" 3, but can be used in applications where a single-character command is required.

REFERENCE: Chapter 7

PURPOSE:	**Set the print pitch to condensed (17 characters/inch).**		
CODE:	〈ESC〉	〈SI〉	
(decimal ASCII)	27	15	
(hex ASCII)	1B	0F	
REMARKS:	Same as 〈SI〉, above.		
REFERENCE:	Chapter 7		

PURPOSE:	**Set the printer to expanded print.**		
CODE:	〈ESC〉	"W"	1
(decimal ASCII)	27	87	1
(hex ASCII)	1B	57	01

REMARKS: This command causes all subsequent printing except NLQ characters to be in expanded type. The size of the type is determined by the normal type size at the time the command is sent:

	Normal	Expanded
Pica	10 CPI	5 CPI
Elite	12 CPI	6 CPI
Condensed	17 CPI	8.5 CPI

REFERENCE: Chapter 7

PURPOSE:	**Set the printer to expanded print for the remainder of the current line.**
CODE:	〈SO〉
(decimal ASCII)	14
(hex ASCII)	0E

REMARKS: This command causes the printer to print expanded characters until a carriage return is sent. It can also be cancelled with 〈DC4〉. The character widths are shown above in the description of the 〈ESC〉 "W" 1 command.

REFERENCE: Chapter 7

PURPOSE:	**Set the printer to expanded print for the remainder of the current line.**		
CODE:	⟨ESC⟩	⟨SO⟩	
(decimal ASCII)	27	14	
(hex ASCII)	1B	0E	
REMARKS:	Same as ⟨SO⟩, above.		
REFERENCE:	Chapter 7		

PURPOSE:	**Cancels expanded print.**		
CODE:	⟨ESC⟩	"W"	0
(decimal ASCII)	27	87	0
(hex ASCII)	1B	57	00
REMARKS:	This command resets the print size to whatever it was before being set to expanded print.		
REFERENCE:	Chapter 7		

PURPOSE:	**Cancels expanded print.**
CODE:	⟨DC4⟩
(decimal ASCII)	20
(hex ASCII)	14
REMARKS:	This command cancels one line expanded printing set with ⟨SO⟩.
REFERENCE:	Chapter 7

Special print modes

PURPOSE:	**Select double-strike printing.**	
CODE:	⟨ESC⟩	“G”
(decimal ASCII)	27	71
(hex ASCII)	1B	47
REMARKS:	This command causes all subsequent characters except NLQ characters to be printed in double-strike. Double-strike causes all characters to be printed once, the paper moved up 1/144 inch, the characters reprinted, and the paper moved back down 1/144 inch.	
REFERENCE:	Chapter 7	

PURPOSE:	**Cancel double-strike printing.**	
CODE:	⟨ESC⟩	“H”
(decimal ASCII)	27	72
(hex ASCII)	1B	48
REMARKS:	This command cancels double-strike printing and returns the printer to its previous print style.	
REFERENCE:	Chapter 7	

PURPOSE:	**Select emphasized printing.**	
CODE:	⟨ESC⟩	“E”
(decimal ASCII)	27	69
(hex ASCII)	1B	45
REMARKS:	This command causes all subsequent characters except NLQ characters to be printed in emphasized print. Emphasized print can only be used with pica-sized characters, or enlarged pica-sized characters (10 CPI and 5 CPI), and cannot be used with superscripts or subscripts. You can select emphasized printing as the power-on default by turning DIP switch A-2 off.	
REFERENCE:	Chapter 7	

PURPOSE:	**Cancel emphasized printing.**		
CODE:	⟨ESC⟩	"F"	
(decimal ASCII)	27	70	
(hex ASCII)	1B	46	

REMARKS: This command cancels emphasized printing and returns the printer to normal printing. You can select normal printing as the power-on default by turning DIP switch A-2 on.

REFERENCE: Chapter 7

PURPOSE:	**Select underlining.**		
CODE:	⟨ESC⟩	"-"	1
(decimal ASCII)	27	45	1
(hex ASCII)	1B	2D	01

REMARKS: This command causes all subsequent characters printed to be automatically underlined. Spaces are also underlined.

REFERENCE: Chapter 7

PURPOSE:	**Cancel underlining.**		
CODE:	⟨ESC⟩	"-"	0
(decimal ASCII)	27	45	0
(hex ASCII)	1B	2D	00

REMARKS: This command cancels underlining and returns the printer to its previous print style.

REFERENCE: Chapter 7

PURPOSE:	**Select superscripts.**			
CODE:		⟨ESC⟩	"S"	0
(decimal ASCII)		27	83	0
(hex ASCII)		1B	53	00

REMARKS: This command causes all subsequent characters to be printed as superscripts. While in superscript mode, the normal bi-directional printing is cancelled and replaced with uni-directional printing. Printing is also set to double-strike mode. Superscripts may be used in conjunction with the italic font, and in pica, elite, and condensed pitches. It may not, however, be used in conjunction with emphasized print, enlarged print, or NLQ characters.

REFERENCE: Chapter 7

PURPOSE:	**Select subscripts.**			
CODE:		⟨ESC⟩	"S"	1
(decimal ASCII)		27	83	1
(hex ASCII)		1B	53	01

REMARKS: This command causes all subsequent characters to be printed as subscripts. The same conditions and restrictions apply for subscripts as do for superscripts.

REFERENCE: Chapter 7

PURPOSE:	**Cancel superscripts and subscripts.**		
CODE:		⟨ESC⟩	"T"
(decimal ASCII)		27	84
(hex ASCII)		1B	54

REMARKS: This command cancels either superscript or subscript mode. It also cancels the uni-directional printing and double-strike which the mode had set.

REFERENCE: Chapter 7

Commands to Control Vertical Position of Print Head

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

Line feed controls

PURPOSE:	**Advance the paper one line (Line Feed).**	
CODE:	〈LF〉	
(decimal ASCII)	10	
(hex ASCII)	0A	
REMARKS:	The actual distance advanced by the line feed is set either through the setting of DIP switch A-5 or through various codes which can be sent (see below). When DIP switch C-4 is "on" a line feed is automatically generated whenever the printer receives a carriage return.	
REFERENCE:	Chapter 8	

PURPOSE:	**Reverse the paper one line.**	
CODE:	〈ESC〉	〈LF〉
(decimal ASCII)	27	10
(hex ASCII)	1B	0A
REMARKS:	This command causes the printer to reverse the paper (in effect moving the print head up on the sheet) one line. The actual distance traveled is set either through the setting of DIP switch A-5 or through various codes which can be sent (see below).	
REFERENCE:	Chapter 8	

PURPOSE:	**Change the line spacing to 1/8 inch.**
CODE:	⟨ESC⟩ "0"
(decimal ASCII)	27 48
(hex ASCII)	1B 30
REMARKS:	This command sets the distance the paper advances or reverses during all subsequent line feeds to 1/8 inch. You can select 1/8 inch line spacing as the power-on default by turning DIP switch A-5 off.
REFERENCE:	Chapter 8

PURPOSE:	**Change the line spacing to 7/72 inch.**
CODE:	⟨ESC⟩ "1"
(decimal ASCII)	27 49
(hex ASCII)	1B 31
REMARKS:	This command sets the actual distance the paper advances or reverses during all subsequent line feeds to 7/72 inch.
REFERENCE:	Chapter 8

PURPOSE:	**Change the line spacing to 1/6 inch.**
CODE:	⟨ESC⟩ "2"
(decimal ASCII)	27 50
(hex ASCII)	1B 32
REMARKS:	This command sets the actual distance the paper advances or reverses during all subsequent line feeds to 1/6 inch. You can select 1/6 inch line spacing as the power-on default by turning DIP switch A-5 on.
REFERENCE:	Chapter 8

PURPOSE: **Change the line spacing to *n*/72 inch.**

CODE:	⟨ESC⟩	"A"	n
(decimal ASCII)	27	65	n
(hex ASCII)	1B	41	n

REMARKS: This command sets the distance the paper advances or reverses during all subsequent line feeds to *n*/72 inch. The value of *n* must be between 0 and 255.

REFERENCE: Chapter 8

PURPOSE: **Change the line spacing to *n*/144 inch.**

CODE:	⟨ESC⟩	"3"	n
(decimal ASCII)	27	51	n
(hex ASCII)	1B	33	n

REMARKS: This command sets the actual distance the paper advances or reverses during all subsequent line feeds to *n*/144 inch. The value of *n* must be between 0 and 255.

REFERENCE: Chapter 8

PURPOSE: **Send a one-time line feed of *n*/144 inch.**

CODE:	⟨ESC⟩	"J"	n
(decimal ASCII)	27	74	n
(hex ASCII)	1B	4A	n

REMARKS: This command causes the printer to advance the paper *n*/144 inch and return the print head to the left margin. It does not change the current value of the line spacing. The value of *n* must be between 0 and 255.

REFERENCE: Chapter 8

PURPOSE:	**Send a one-time reverse line feed of n/144 inch.**

CODE:	⟨ESC⟩	"j"	n
(decimal ASCII)	27	106	n
(hex ASCII)	1B	6A	n

REMARKS: This command causes the printer to reverse the paper *n*/144 inch and return the print head to the left margin. It does not change the current value of the line spacing. The value of n must be between 0 and 255.

REFERENCE: Chapter 8

Form feed controls

PURPOSE: **Advance paper to top of next page (Form Feed).**

CODE:	⟨FF⟩
(decimal ASCII)	12
(hex ASCII)	0C

REMARKS: The actual length of a page ejected by a form feed is set either by the setting of DIP switch A-1 or through various codes which can be sent (see below).

REFERENCE: Chapter 8

PURPOSE: **Reverse the paper to the top of the current page.**

CODE:	⟨ESC⟩	⟨FF⟩
(decimal ASCII)	27	12
(hex ASCII)	1B	0C

REMARKS: This command causes the printer to reverse the paper to the top of the current printing page (or form).

REFERENCE: Chapter 8

PURPOSE:	**Set page length to *n* lines.**

CODE:	⟨ESC⟩	"C"	*n*
(decimal ASCII)	27	67	n
(hex ASCII)	1B	43	n

REMARKS: This command sets the length of all subsequent pages to *n* lines. The value of *n* must be between 1 and 127.

REFERENCE: Chapter 8

PURPOSE:	**Set page length to *n* inches.**

CODE:	⟨ESC⟩	"C"	0	*n*
(decimal ASCII)	27	67	0	*n*
(hex ASCII)	1B	43	00	*n*

REMARKS: This command sets the length of all subsequent pages to *n* inches. The value of *n* must be between 1 and 32. You can select a power-on default form length of 11 inches or 12 inches by setting DIP switch A-1.

REFERENCE: Chapter 8

PURPOSE:	**Set the top margin.**

CODE:	⟨ESC⟩	"R"	*n*
(decimal ASCII)	27	82	n
(hex ASCII)	1B	52	n

REMARKS: This command sets the margin at the top of the page to *n*-1 lines. Printing will start on line n. The default value for n upon power-on is 1. The value of n must be between 1 and 16.

REFERENCE: Chapter 8

PURPOSE:	**Set the bottom margin.**

CODE:	〈ESC〉	"N"	*n*
(decimal ASCII)	27	78	*n*
(hex ASCII)	1B	4E	*n*

REMARKS: This command sets the margin at the bottom of the page to *n* lines. The printer will automatically execute a form feed when the number of lines left on a page is equal to *n*. The value of *n* must be between 1 and 127. This command is sometimes referred to as "skip-over-perforation."

REFERENCE: Chapter 8

PURPOSE:	**Cancel top and bottom margins.**

CODE:	〈ESC〉	"O"
(decimal ASCII)	27	79
(hex ASCII)	1B	4F

REMARKS: This command cancels both the top margin set by 〈ESC〉 "R" *n* and the bottom margin set by 〈ESC〉 "N" *n*.

REFERENCE: Chapter 8

Vertical tabs

PURPOSE:	**Advance paper to the next vertical tab position.**

CODE:	〈VT〉
(decimal ASCII)	11
(hex ASCII)	0B

REMARKS: This command causes the paper to be advanced to the next vertical tab position, or the top of the next page, whichever it finds first. The vertical tab positions are set upon power on at lines 6, 12, 18, 24, 30, 36, 42, 48, 54, and 60.

REFERENCE: Chapter 9

PURPOSE:	**Set vertical tab positions.**			
CODE:	⟨ESC⟩	"P"	*n1 n2 n3...*	0
(decimal ASCII)	27	80	*n1 n2 n3...*	0
(hex ASCII)	1B	50	*n1 n2 n3...*	00

REMARKS: This command cancels all current vertical tab positions and sets those defined at lines *n1, n2, n3*, etc. The maximum number of vertical tab positions allowed is 20. The ASCII 0 character is used as a command terminator. Each vertical tab position must be between 1 and 255, and they must be specified in ascending order.

REFERENCE: Chapter 9

PURPOSE:	**Advance the paper *n* lines.**		
CODE:	⟨ESC⟩	"a"	n
(decimal ASCII)	27	97	n
(hex ASCII)	1B	61	*n*

REMARKS: This command causes the printer to advance the paper *n* lines. It does not, however, change the current value of the vertical tab positions. The value of *n* must be between 1 and 255.

REFERENCE: Chapter 8, Chapter 9

Commands to Control Horizontal Position of Print Head

PURPOSE:	**Return print head to home position (Carriage Return).**		
CODE:	⟨CR⟩		
(decimal ASCII)	13		
(hex ASCII)	0D		

REMARKS: This command returns the print head to the home position (the left margin). If DIP switch C-4 has been set on, then this command will also cause a line feed character to be generated after the carriage return, thereby advancing to the beginning of the next print line automatically.

REFERENCE: Chapter 8

PURPOSE:	**Set the left print margin.**		
CODE:	⟨ESC⟩	"M"	n
(decimal ASCII)	27	77	n
(hex ASCII)	1B	4D	n

REMARKS: This command sets the home position returned to during the execution of all subsequent carriage returns to be print position *n*. The power on default for *n* is 1. The value of *n* must be between 1 and 255. For Radix-10 the maximum print position for pica pitch is 80, for elite is 96, and for condensed pitch is 136. For Radix-15 the maximum print position for pica pitch is 136, for elite is 163, and for condensed pitch is 233.

REFERENCE: Chapter 9

PURPOSE:	**Set the right print margin.**

CODE:	⟨ESC⟩	"Q"	*n*
(decimal ASCII)	27	81	*n*
(hex ASCII)	1B	51	*n*

REMARKS:	This command sets the right hand print margin to print position *n*. After execution of this command, any attempt to print beyond print position *n* will cause the printer to automatically generate a carriage return and a line feed before printing the remainder of the line. The value for *n* must be between 1 and 255.
REFERENCE:	Chapter 9

PURPOSE:	**Move the print head to the next horizontal tab position.**

CODE:	⟨HT⟩
(decimal ASCII)	9
(hex ASCII)	09

REMARKS:	This command causes the print head to advance to the next horizontal tab position. The horizontal tab positions are set at power-on to print positions 10, 20, 30, etc. (to the maximum print position).
REFERENCE:	Chapter 9

PURPOSE:	**Set horizontal tab positions.**

CODE:	⟨ESC⟩	"D"	*n1 n2 n3*...	0
(decimal ASCII)	27	68	*n1 n2 n3*...	0
(hex ASCII)	1B	44	*n1 n2 n3*...	00

REMARKS:	This command cancels all current horizontal tab positions and sets those defined at print positions *n1*, *n2*, *n3*, etc. The maximum number of horizontal tab positions allowed is 255. The ASCII 0 character is used as a command terminator. Each horizontal tab position must be between 1 and 255, and they must be specified in ascending order.
REFERENCE:	Chapter 9

PURPOSE:	**Skip n print positions.**		
CODE:	⟨ESC⟩	"b"	n
(decimal ASCII)	27	98	n
(hex ASCII)	1B	62	n
REMARKS:	This command causes the print head to advance n print positions to the right. It does not, however, change the current value of the horizontal tab positions. The value of n must be between 1 and 255.		
REFERENCE:	Chapter 9		

PURPOSE:	**Move the print head back one print position (backspace).**
CODE:	⟨BS⟩
(decimal ASCII)	8
(hex ASCII)	08
REMARKS:	This command shifts the print head one column to the left. If the print head is at the home position, the command is ignored. This command can be used to overstrike characters.
REFERENCE:	Chapter 10

Download Character Commands

PURPOSE:	**Define download characters into RAM.**
CODE:	〈ESC〉 "*" 1 *n1 n2 m1 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11*
	27 42 1 *n1 n2 m1 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11*
	1B 2A 01 *n1 n2 m1 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11*

REMARKS: This command is used to set up a user-defined character and store it into RAM for later use. RAM is cleared during power down. The value of *n1* is the position in RAM that this character is to occupy. It must be between 33 and 126 or between 160 and 254. That is, it must fall within the range of printable characters. The value of *n2* determines the attributes and width of the character. *m1* thru *m11* determine which dots form the character.

REFERENCE: Chapter 11

PURPOSE: **Copy standard character ROM fonts into RAM.**

CODE:	〈ESC〉	"*"	0
(decimal ASCII)	27	42	0
(hex ASCII)	1B	2A	00

REMARKS: This command takes all of the characters in the standard ASCII character set (those with ASCII values between 33 and 126; characters with ASCII values above 160 are *not* copied to RAM) and copies them into RAM. This is helpful prior to defining characters in RAM because it allows standard ROM characters to be printed on the same line as download characters.

REFERENCE: Chapter 11

PURPOSE:	**Select download character set with proportional spacing.**		
CODE:	⟨ESC⟩	"X"	1
(decimal ASCII)	27	88	1
(hex ASCII)	1B	58	01

REMARKS: This command selects the download character set using the proportional spacing defined in the character attribute data.
Note: Download characters *cannot* be mixed with other characters on the same line.

REFERENCE: Chapter 11

PURPOSE:	**Cancel download character set with proportional spacing.**		
CODE:	⟨ESC⟩	"X"	0
(decimal ASCII)	27	88	0
(hex ASCII)	1B	58	00

REMARKS: This command cancels the download character set and selects the standard ASCII character set.

REFERENCE: Chapter 11

PURPOSE:	**Select download character set with normal spacing.**		
CODE:	⟨ESC⟩	"$"	1
(decimal ASCII)	27	36	1
(hex ASCII)	1B	24	01

REMARKS: This command causes the printer to select the download character set using normal spacing and ignoring the proportional width data.
Note: Download characters *cannot* be mixed with other characters on the same line.

REFERENCE: Chapter 11

PURPOSE:	**Cancel download character set with normal spacing.**		
CODE:	⟨ESC⟩	"$"	0
(decimal ASCII)	27	36	0
(hex ASCII)	1B	24	00
REMARKS:	This command cancels the download character set and selects the standard ASCII character set.		
REFERENCE:	Chapter 11		

Commands to Control Graphics

PURPOSE:	**Print normal-density graphics.**		
CODE:	⟨ESC⟩	"K"	*n1 n2 m1 m2 m3...*
(decimal ASCII)	27	75	*n1 n2 m1 m2 m3...*
(hex ASCII)	1B	4B	*n1 n2 m1 m2 m3...*
REMARKS:	This command selects 60 dots-per-inch, column-scan, bit-image graphics mode. The values of *n1* and *n2* represent the number of graphics characters to be printed, where the total number of characters = *n2* times 256 + *n1*. The correct number of graphic data bytes (*m1*, *m2*, etc.) must follow *n2*. The ASCII value of these characters determine which pins are fired for each character.		
REFERENCE:	Chapter 12		

PURPOSE:	**Print double-density graphics.**
CODE:	⟨ESC⟩ "L" *n1 n2 m1 m2 m3...*
(decimal ASCII)	27 76 *n1 n2 m1 m2 m3...*
(hex ASCII)	1B 4C *n1 n2 m1 m2 m3...*
REMARKS:	This command selects 120 dots-per-inch, column-scan, bit-image graphics mode. The values of *n1* and *n2* are the same as in normal density graphics. The correct number of graphic data bytes (*m1, m2,* etc.) must follow *n2*. The ASCII value of these characters determine which pins are fired for each character.
REFERENCE:	Chapter 12

PURPOSE:	**Print double-density graphics with double-speed.**
CODE:	⟨ESC⟩ "y" *n1 n2 m1 m2 m3...*
(decimal ASCII)	27 121 *n1 n2 m1 m2 m3...*
(hex ASCII)	1B 79 *n1 n2 m1 m2 m3...*
REMARKS:	This command selects 120 dots-per-inch, column-scan, bit-image graphics mode with double-speed. The values of *n1* and *n2* are the same as in normal density graphics. The correct number of graphic data bytes (*m1, m2,* etc.) must follow *n2*. The ASCII value of these characters determine which pins are fired for each character.
REFERENCE:	Chapter 12

PURPOSE:	**Print quadruple-density graphics.**		
CODE:	⟨ESC⟩	"z"	*n1 n2 m1 m2 m3...*
(decimal ASCII)	27	122	*n1 n2 m1 m2 m3...*
(hex ASCII)	1B	7A	*n1 n2 m1 m2 m3...*
REMARKS:	This command selects 240 dots-per-inch, column-scan, bit-image graphics mode. The values of *n1* and *n2* are the same as in normal density graphics. The correct number of graphic data bytes (*m1*, *m2*, etc.) must follow *n2*. The ASCII value of these characters determine which pins are fired for each character.		
REFERENCE:	Chapter 12		

Macro Instruction Commands

PURPOSE:	**Define macro instruction.**			
CODE:	⟨ESC⟩	"+"	...	⟨RS⟩
(decimal ASCII)	27	43	...	30
(hex ASCII)	1B	2B	...	1E
REMARKS:	This command cancels any existing macro instruction, and replaces it with the instruction defined. The maximum number of characters allowed in the macro instruction is 16. The ⟨RS⟩ character marks the end of the macro definition.			
REFERENCE:	Chapter 10			

PURPOSE:	**Execute macro instruction.**	
CODE:	⟨ESC⟩	"!"
(decimal ASCII)	27	33
(hex ASCII)	1B	21
REMARKS:	This command executes a macro instruction that was previously defined.	
REFERENCE:	Chapter 10	

Other Commands

PURPOSE:	**Set the value of the eighth data bit to logical 1.**	
CODE:	⟨ESC⟩	"⟩"
(decimal ASCII)	27	62
(hex ASCII)	1B	3E

REMARKS: This command forces the eighth data bit of each subsequent character sent to the printer to logical 1. This code allows users with a 7-bit interface to access those characters whose ASCII code is greater than 127. This code should not be used to transmit printer control codes.

REFERENCE: Chapter 10

PURPOSE:	**Set the value of the eighth data bit to logical 0.**	
CODE:	⟨ESC⟩	"="
(decimal ASCII)	27	61
(hex ASCII)	1B	3D

REMARKS: This command forces the eighth data bit of each subsequent character sent to the printer to logical 0. This code should not be used to transmit printer control codes.

REFERENCE: Chapter 10

PURPOSE:	**Accept the value of the eighth data bit as is.**	
CODE:	⟨ESC⟩	"#"
(decimal ASCII)	27	35
(hex ASCII)	1B	23
REMARKS:	This command cancels either setting of the eighth data bit. The printer will use the value of the eighth data bit that is sent from the computer. This code allows users with a 7-bit interface to resume normal functions after accessing those characters whose ASCII code is greater than 127.	
REFERENCE:	Chapter 10	

PURPOSE:	**Delete the last character sent.**
CODE:	⟨DEL⟩
(decimal ASCII)	127
(hex ASCII)	7F
REMARKS:	This command deletes the last character received. This command is ignored if the last character received has already been printed, or if the last character received was all or part of a function code.
REFERENCE:	Chapter 10

PURPOSE:	**Set printer off line.**
CODE:	⟨DC3⟩
(decimal ASCII)	19
(hex ASCII)	13
REMARKS:	This command causes the printer to set itself off line, disregarding all subsequent characters and function codes, with the exception of ⟨DC1⟩, which will return the printer to an on line state. This is not the same as pushing the ON-LINE button. When the ON-LINE light is out the printer will not respond to ⟨DC1⟩.
REFERENCE:	Chapter 10

PURPOSE:	**Set printer on line.**		
CODE:	⟨DC1⟩		
(decimal ASCII)	17		
(hex ASCII)	11		
REMARKS:	This code resets the printer to an on line state, thus allowing it receive and process all subsequent characters and function codes. This is not the same as pushing the ON-LINE button. When the ON-LINE light is out the printer will not respond to ⟨DC1⟩.		
REFERENCE:	Chapter 10		

PURPOSE:	**Sound printer bell.**		
CODE:	⟨BEL⟩		
(decimal ASCII)	7		
(hex ASCII)	07		
REMARKS:	This command causes the printer tone to sound for approximately one-fourth second.		
REFERENCE:	Chapter 10		

PURPOSE:	**Disable the printer bell.**		
CODE:	⟨ESC⟩	"Y"	0
(decimal ASCII)	27	89	0
(hex ASCII)	1B	59	00
REMARKS:	This command causes the printer to ignore the ⟨BEL⟩ character.		
REFERENCE:	Chapter 10		

PURPOSE:	**Enable the printer bell.**		
CODE:	⟨ESC⟩	"Y"	1
(decimal ASCII)	27	89	1
(hex ASCII)	1B	59	01
REMARKS:	This command causes the printer to respond to the ⟨BEL⟩ character normally by sounding the printer bell.		
REFERENCE:	Chapter 10		

PURPOSE:	**Disable paper-out detector.**		
CODE:	⟨ESC⟩	"8"	
(decimal ASCII)	27	56	
(hex ASCII)	1B	38	

REMARKS: This command causes the printer to disregard the signal sent by the paper-out detector. The paper-out signal normally sounds the printer bell and stops printing until paper is inserted and the printer is reset. DIP switch C-1 can also be set to disable the paper-out detector.

REFERENCE: Chapter 10

PURPOSE:	**Enable paper-out detector.**		
CODE:	⟨ESC⟩	"9"	
(decimal ASCII)	27	57	
(hex ASCII)	1B	39	

REMARKS: This command restores the function of the paper-out detector.

REFERENCE: Chapter 10

PURPOSE:	**Select uni-directional printing.**		
CODE:	⟨ESC⟩	"U"	1
(decimal ASCII)	27	85	1
(hex ASCII)	1B	55	01

REMARKS: This command causes all subsequent lines to be printed in uni-directional printing. Uni-directional printing is useful in printing tables or charts, since it ensures that vertical columns of characters will be in alignment.

REFERENCE: Chapter 10

PURPOSE:	**Cancel uni-directional printing.**		
CODE:	⟨ESC⟩	"U"	0
(decimal ASCII)	27	85	0
(hex ASCII)	1B	55	00

REMARKS: This command cancels uni-directional printing, and returns to the standard bi-directional printing, which is considerably faster.

REFERENCE: Chapter 10

PURPOSE:	**Initialize printer.**	
CODE:	⟨ESC⟩	"@"
(decimal ASCII)	27	64
(hex ASCII)	1B	40

REMARKS: This command reinitializes the printer. The print buffer is cleared, and the form length, character pitch, character set, line feed pitch, and international character set are all reset to the values defined by their respective DIP switches.
The main difference between the ⟨ESC⟩ "@" command and turning the printer off and back on is that download character RAM and the macro instruction are preserved with this command.

REFERENCE: Chapter 10

Appendix L

Command Summary in Numeric Order

Control code	Function
CHR$(0)	Ends tab settings
CHR$(7)	Sounds bell
CHR$(8)	Backspace
CHR$(9)	Horizontal tab
CHR$(10)	Line feed
CHR$(11)	Vertical tab
CHR$(12)	Form feed
CHR$(13)	Carriage return
CHR$(14)	One line expanded print
CHR$(15)	Condensed print
CHR$(17)	On line
CHR$(18)	Pica type
CHR$(19)	Off line
CHR$(20)	Cancels one line expanded print
CHR$(27)	Escape **(indicated as 〈ESC〉 below)**
CHR$(30)	Ends macro instruction definition
CHR$(127)	Delete last character
〈ESC〉 CHR$(10)	Reverse line feed
〈ESC〉 CHR$(12)	Reverse feed to top of page
〈ESC〉 CHR$(14)	One line expanded print
〈ESC〉 CHR$(15)	Condensed print
〈ESC〉 "!"	Use macro
〈ESC〉 "#"	Accept eighth bit as is
〈ESC〉 "$" CHR$(0)	Cancel normal download characters
〈ESC〉 "$" CHR$(1)	Use normal download characters
〈ESC〉 "*" CHR$(0)	Copy ROM characters to download RAM
〈ESC〉 "*" CHR$(1) *n1 n2 m1 m2 . . . m11*	Define download character
〈ESC〉 "+" . . . CHR$(30)	Define macro

〈ESC〉 "-" CHR$(0)	Stop underlining
〈ESC〉 "-" CHR$(1)	Start underlining
〈ESC〉 "0"	1/8 inch line feed
〈ESC〉 "1"	7/72 inch line feed
〈ESC〉 "2"	1/6 inch line feed
〈ESC〉 "3" *n*	*n*/144 inch line feed
〈ESC〉 "4"	Italic print
〈ESC〉 "5"	Cancel italic print
〈ESC〉 "7" *n*	Select international character set
〈ESC〉 "8"	Ignore paper-out signal
〈ESC〉 "9"	Enable paper-out signal
〈ESC〉 "="	Set eighth bit to 0
〈ESC〉 "〉"	Set eighth bit to 1
〈ESC〉 "@"	Reset the printer
〈ESC〉 "A" *n*	*n*/72 inch line feed
〈ESC〉 "B" CHR$(1)	Pica print
〈ESC〉 "B" CHR$(2)	Elite print
〈ESC〉 "B" CHR$(3)	Condensed print
〈ESC〉 "B" CHR$(4)	Select NLQ (Near Letter Quality) characters
〈ESC〉 "B" CHR$(5)	Cancel NLQ characters
〈ESC〉 "C" *n*	Set page length to *n* lines
〈ESC〉 "C" CHR$(0) *n*	Set page length to *n* inches
〈ESC〉 "D" . . . CHR$(0)	Set horizontal tabs
〈ESC〉 "E"	Emphasized print
〈ESC〉 "F"	Cancel emphasized print
〈ESC〉 "G"	Double-strike print
〈ESC〉 "H"	Cancel double-strike print
〈ESC〉 "J" *n*	Single line feed of *n*/144 inches
〈ESC〉 "K" *n1 n2*	Single density graphics
〈ESC〉 "L" *n1 n2*	Double density graphics
〈ESC〉 "M" *n*	Set left margin at column *n*
〈ESC〉 "N" *n*	Set bottom margin at *n* lines
〈ESC〉 "O"	Cancel top and bottom margins
〈ESC〉 "P" . . . CHR$(0)	Set vertical tabs
〈ESC〉 "Q" *n*	Set right margin at column *n*
〈ESC〉 "R" *n*	Set top margin at line *n*
〈ESC〉 "S" CHR$(0)	Superscript on
〈ESC〉 "S" CHR$(1)	Subscript on
〈ESC〉 "T"	Cancel super and subscripts
〈ESC〉 "U" CHR$(0)	Bidirectional print
〈ESC〉 "U" CHR$(1)	Unidirectional print
〈ESC〉 "W" CHR$(0)	Cancel enlarged print
〈ESC〉 "W" CHR$(1)	Enlarged print

〈ESC〉 "X" CHR$(0)	Cancel proportional download characters
〈ESC〉 "X" CHR$(1)	Use proportional download characters
〈ESC〉 "Y" CHR$(0)	Disable bell
〈ESC〉 "Y" CHR$(1)	Enable bell
〈ESC〉 "a" *n*	Advance *n* line feeds
〈ESC〉 "b" *n*	Tab over *n* columns
〈ESC〉 "j" *n*	Reverse line feed of *n*/144 inches
〈ESC〉 "y" n1 n2	Double speed, double density graphics
〈ESC〉 "z" n1 n2	Quadruple density graphics

Appendix M

ASCII Code Conversion Chart

Standard ASCII Codes

Decimal	Hexadecimal	Binary	Control character	Character
0	00	0000 0000	Ctrl-@	NUL
1	01	0000 0001	Ctrl-A	
2	02	0000 0010	Ctrl-B	
3	03	0000 0011	Ctrl-C	
4	04	0000 0100	Ctrl-D	
5	05	0000 0101	Ctrl-E	
6	06	0000 0110	Ctrl-F	
7	07	0000 0111	Ctrl-G	BEL
8	08	0000 1000	Ctrl-H	BS
9	09	0000 1001	Ctrl-I	HT
10	0A	0000 1010	Ctrl-J	LF
11	0B	0000 1011	Ctrl-K	VT
12	0C	0000 1100	Ctrl-L	FF
13	0D	0000 1101	Ctrl-M	CR
14	0E	0000 1110	Ctrl-N	SO
15	0F	0000 1111	Ctrl-O	SI
16	10	0001 0000	Ctrl-P	
17	11	0001 0001	Ctrl-Q	DC1
18	12	0001 0010	Ctrl-R	DC2
19	13	0001 0011	Ctrl-S	DC3
20	14	0001 0100	Ctrl-T	DC4
21	15	0001 0101	Ctrl-U	
22	16	0001 0110	Ctrl-V	
23	17	0001 0111	Ctrl-W	
24	18	0001 1000	Ctrl-X	
25	19	0001 1001	Ctrl-Y	
26	1A	0001 1010	Ctrl-Z	
27	1B	0001 1011		ESC
28	1C	0001 1100		
29	1D	0001 1101		
30	1E	0001 1110		RS
31	1F	0001 1111		

Standard ASCII Codes

Decimal	Hexadecimal	Binary	Character
32	20	0010 0000	SP
33	21	0010 0001	!
34	22	0010 0010	"
35	23	0010 0011	#
36	24	0010 0100	$
37	25	0010 0101	%
38	26	0010 0110	&
39	27	0010 0111	'
40	28	0010 1000	(
41	29	0010 1001	)
42	2A	0010 1010	*
43	2B	0010 1011	+
44	2C	0010 1100	,
45	2D	0010 1101	-
46	2E	0010 1110	.
47	2F	0010 1111	/
48	30	0011 0000	0
49	31	0011 0001	1
50	32	0011 0010	2
51	33	0011 0011	3
52	34	0011 0100	4
53	35	0011 0101	5
54	36	0011 0110	6
55	37	0011 0111	7
56	38	0011 1000	8
57	39	0011 1001	9
58	3A	0011 1010	:
59	3B	0011 1011	;
60	3C	0011 1100	<
61	3D	0011 1101	=
62	3E	0011 1110	>
63	3F	0011 1111	?
64	40	0100 0000	@
65	41	0100 0001	A
66	42	0100 0010	B
67	43	0100 0011	C
68	44	0100 0100	D
69	45	0100 0101	E
70	46	0100 0110	F
71	47	0100 0111	G
72	48	0100 1000	H
73	49	0100 1001	I

	Standard ASCII Codes		
Decimal	**Hexadecimal**	**Binary**	**Character**
74	4A	0100 1010	J
75	4B	0100 1011	K
76	4C	0100 1100	L
77	4D	0100 1101	M
78	4E	0100 1110	N
79	4F	0100 1111	O
80	50	0101 0000	P
81	51	0101 0001	Q
82	52	0101 0010	R
83	53	0101 0011	S
84	54	0101 0100	T
85	55	0101 0101	U
86	56	0101 0110	V
87	57	0101 0111	W
88	58	0101 1000	X
89	59	0101 1001	Y
90	5A	0101 1010	Z
91	5B	0101 1011	[
92	5C	0101 1100	\
93	5D	0101 1101	]
94	5E	0101 1110	^
95	5F	0101 1111	_
96	60	0110 0000	`
97	61	0110 0001	a
98	62	0110 0010	b
99	63	0110 0011	c
100	64	0110 0100	d
101	65	0110 0101	e
102	66	0110 0110	f
103	67	0110 0111	g
104	68	0110 1000	h
105	69	0110 1001	i
106	6A	0110 1010	j
107	6B	0110 1011	k
108	6C	0110 1100	l
109	6D	0110 1101	m
110	6E	0110 1110	n
111	6F	0110 1111	o
112	70	0111 0000	p
113	71	0111 0001	q
114	72	0111 0010	r
115	73	0111 0011	s

Standard ASCII Codes

Decimal	**Hexadecimal**	**Binary**	**Character**
116	74	0111 0100	t
117	75	0111 0101	u
118	76	0111 0110	v
119	77	0111 0111	w
120	78	0111 1000	x
121	79	0111 1001	y
122	7A	0111 1010	z
123	7B	0111 1011	{
124	7C	0111 1100	¦
125	7D	0111 1101	}
126	7E	0111 1110	~
127	7F	0111 1111	DEL
128	80	1000 0000	
129	81	1000 0001	
130	82	1000 0010	
131	83	1000 0011	
132	84	1000 0100	
133	85	1000 0101	
134	86	1000 0110	
135	87	1000 0111	BEL
136	88	1000 1000	BS
137	89	1000 1001	HT
138	8A	1000 1010	LF
139	8B	1000 1011	VT
140	8C	1000 1100	FF
141	8D	1000 1101	CR
142	8E	1000 1110	SO
143	8F	1000 1111	SI
144	90	1001 0000	
145	91	1001 0001	DC1
146	92	1001 0010	DC2
147	93	1001 0011	DC3
148	94	1001 0100	DC4
149	95	1001 0101	
150	96	1001 0110	
151	97	1001 0111	
152	98	1001 1000	
153	99	1001 1001	
154	9A	1001 1010	
155	9B	1001 1011	ESC
156	9C	1001 1100	
157	9D	1001 1101	

	Standard ASCII Codes		
Decimal	**Hexadecimal**	**Binary**	**Character**
158	9E	1001 1110	RS
159	9F	1001 1111	
160	A0	1010 0000	╯
161	A1	1010 0001	╮
162	A2	1010 0010	╰
163	A3	1010 0011	╭
164	A4	1010 0100	↑
165	A5	1010 0101	↓
166	A6	1010 0110	←
167	A7	1010 0111	→
168	A8	1010 1000	○
169	A9	1010 1001	▲
170	AA	1010 1010	▼
171	AB	1010 1011	▶
172	AC	1010 1100	◀
173	AD	1010 1101	◇
174	AE	1010 1110	◆
175	AF	1010 1111	□
176	B0	1011 0000	Tx
177	B1	1011 0001	Å
178	B2	1011 0010	ø
179	B3	1011 0011	θ
180	B4	1011 0100	⊏
181	B5	1011 0101	⊧
182	B6	1011 0110	Ω
183	B7	1011 0111	℧
184	B8	1011 1000	Σ
185	B9	1011 1001	σ
186	BA	1011 1010	⋈
187	BB	1011 1011	⌶
188	BC	1011 1100	±
189	BD	1011 1101	⊐
190	BE	1011 1110	⋇
191	BF	1011 1111	÷
192	C0	1100 0000	Ā
193	C1	1100 0001	à
194	C2	1100 0010	ç
195	C3	1100 0011	£
196	C4	1100 0100	ā
197	C5	1100 0101	µ
198	C6	1100 0110	▪
199	C7	1100 0111	’

Standard ASCII Codes

Decimal	Hexadecimal	Binary	Character
200	C8	1100 1000	†
201	C9	1100 1001	§
202	CA	1100 1010	Ē
203	CB	1100 1011	⊗
204	CC	1100 1100	¼
205	CD	1100 1101	¤
206	CE	1100 1110	½
207	CF	1100 1111	‖
208	D0	1101 0000	¥
209	D1	1101 0001	Ä
210	D2	1101 0010	Ö
211	D3	1101 0011	Ü
212	D4	1101 0100	¢
213	D5	1101 0101	₧
214	D6	1101 0110	ä
215	D7	1101 0111	ö
216	D8	1101 1000	ü
217	D9	1101 1001	ß
218	DA	1101 1010	ê
219	DB	1101 1011	é
220	DC	1101 1100	ú
221	DD	1101 1101	è
222	DE	1101 1110	ñ
223	DF	1101 1111	ƒ
224	E0	1110 0000	
225	E1	1110 0001	▘
226	E2	1110 0010	▖
227	E3	1110 0011	▝
228	E4	1110 0100	▗
229	E5	1110 0101	▚
230	E6	1110 0110	▞
231	E7	1110 0111	▀
232	E8	1110 1000	▄
233	E9	1110 1001	▌
234	EA	1110 1010	▐
235	EB	1110 1011	▛
236	EC	1110 1100	▜
237	ED	1110 1101	▙
238	EE	1110 1110	▟
239	EF	1110 1111	█
240	F0	1111 0000	┌
241	F1	1111 0001	─

	Standard ASCII Codes		
Decimal	**Hexadecimal**	**Binary**	**Character**
242	F2	1111 0010	┐
243	F3	1111 0011	┬
244	F4	1111 0100	├
245	F5	1111 0101	│
246	F6	1111 0110	└
247	F7	1111 0111	┘
248	F8	1111 1000	┴
249	F9	1111 1001	┤
250	FA	1111 1010	┼
251	FB	1111 1011	◤
252	FC	1111 1100	◢
253	FD	1111 1101	◥
254	FE	1111 1110	◣
255	FF	1111 1111	

Appendix N

Technical Specifications

Printing

Printing method	Serial impact dot matrix
Printing speed	200 characters per second in 10 CPI
Print buffer	16 K bytes
Paper feed	12 lines/second (at 1/6 inch line spacing) Sprocket or friction feed
Printing direction	Bidirectional, logic seeking Unidirectional in bit image and NLQ modes
Character set	96 standard ASCII characters 96 italic characters 96 near letter quality (NLQ) characters 88 international characters 64 special symbols 32 block graphics characters 189 user-defined characters
Character size	2.4 mm x 2.0 mm standard 10 CPI characters
Character matrix	Standard characters: 9 dot x 9 dot Block graphics: 6 dot x 6 dot User defined: 7 dot x 4 to 11 dot Near letter quality: 17 dot x 9 dot Bit image modes: 7 or 8 dot x 60 dots/in. 7 or 8 dot x 120 dots/in. 7 or 8 dot x 240 dots/in.
Line spacing	1/6, 1/8 inch or 7/72 inch standard n/72 inch or n/144 inch programmable

Column width		Radix-10	Radix-15
	Pica	80	136
	Elite	96	163
	Condensed	136	233
	Pica expanded	40	68
	Elite expanded	48	81
	Condensed expanded	68	116

Special features
240 CPS white spacing
Automatic single sheet insertion
Near letter quality printing
Pause and feed buttons
Reverse paper feed
Short form tear-off
Easy access format switches
Self-test
Downloadable characters (proportional and standard)
Dual interface
Macro instruction
Continuous underlining
7 or 8 bit selectable interface
Ultra hi resolution bit image graphics
Vertical and horizontal tabs
Skip over perforation
15.5″ carriage (Radix-15 only)

Paper		Radix-10	Radix-15
Paper type	Single sheets	5.5-8.5 in. wide	5.5-14.5 in. wide
	Continuous paper	4-10 in. wide	4-15.5 in. wide
Thickness	One-part forms	0.07-0.10 mm	0.07-0.10 mm
	Max. 3-part forms	0.28 mm max.	0.28 mm max.

Printer		Radix-10	Radix-15
Dimensions	Height	117 mm (4.6 in.)	117 mm (4.6 in.)
	Width	414 mm (16.3 in.)	556 mm (21.9 in.)
	Depth	345 mm (13.6 in.)	345 mm (13.6 in.)
Weight		9.1 kg (20.1 lb.)	11.1 kg (24.5 lb.)

Power
120 VAC ±10% 60Hz, approx. 160W

Ribbon
Star Micronics ribbon cartridge
Radix-10: #80980070; Radix-15: #80980080
Sub-cassette: Radix-10: #80900220;
Radix-15: #80900230

Parallel interface

Interface	Centronics-compatible, 7 or 8 bit
Synchronization	By externally supplied strobe pulses
Handshaking	By ACK or BUSY signals
Logic level	TTL
Connector	57-30360 Amphenol

Serial interface

Interface	Asynchronous RS-232C/20 mA current loop
Bit rate	300, 600, 1200, 2400, 4800, 9600, 19200 baud
Word length	1 start bit 7 or 8 data bits Odd, even or no parity 1 or 2 stop bits
Handshaking	Serial busy, 1 byte mode Serial busy, 1 block mode ACK mode XON/XOFF mode

Index

DIP Switch Settings

Switch	ON	OFF	SETTING
DIP Switch A			
A-1	11″ page length	12″ page length	
A-2	Normal print	Emphasized print	
A-3	10 CPI (pica pitch)	17 CPI (condensed pitch)	
A-4	Normal	NLQ	
A-5	1/6″ line feed	1/8″ line feed	
A-6	International character set selection-see below		
A-7			
A-8			
DIP Switch B			
B-1	2 stop bits	1 stop bit	
B-2	7 data bits	8 data bits	
B-3	Parity checked	No parity	
B-4	Handshaking protocols-see below		
B-5			
B-6	Odd parity	Even parity	
B-7	Data transfer rate-see below		
B-8			
B-9			
B-10	Not used		
DIP Switch C			
C-1	Paper-out detector on	Ignore paper-out	
C-2	Serial interface	Parallel interface	
C-3	7-bit interface	8-bit interface	
C-4	Auto LF with CR	LF must be from host	

International character sets

Switch	USA	England	Germany	Denmark	France	Sweden	Italy	Spain
A-6	ON	OFF	ON	OFF	ON	OFF	ON	OFF
A-7	ON	ON	OFF	OFF	ON	ON	OFF	OFF
A-8	ON	ON	ON	ON	OFF	OFF	OFF	OFF

Handshaking protocols

Protocol	Switch B-4	Switch B-5
Serial busy, 1 byte mode	OFF	OFF
Serial busy, 1 block mode	ON	OFF
ACK mode	OFF	ON
XON/XOFF mode	ON	ON

Data transfer rates

Baud rate	Switch B-7	Switch B-8	Switch B-9
150	OFF	OFF	OFF
300	OFF	OFF	ON
600	OFF	ON	OFF
1200	OFF	ON	ON
2400	ON	OFF	OFF
4800	ON	OFF	ON
9600	ON	ON	OFF
19200	ON	ON	ON

Use the "setting" column to record the way the switches are set in your printer.