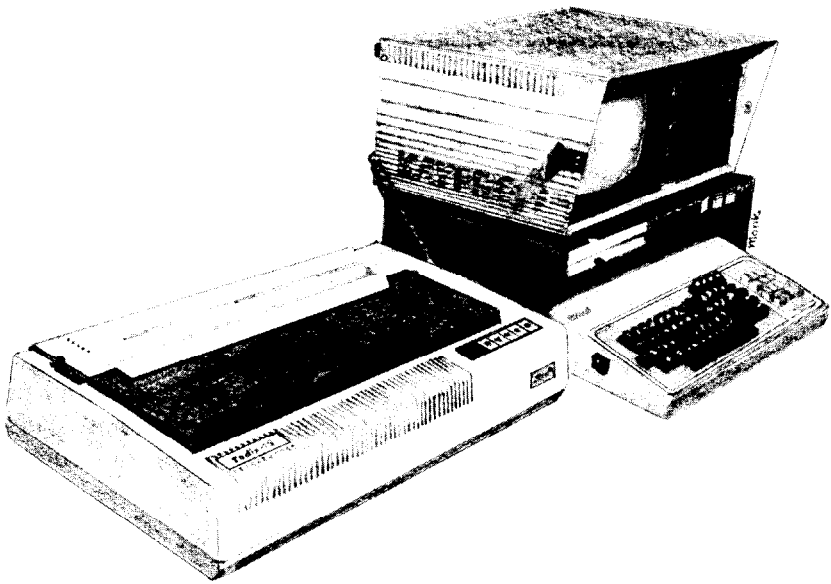


Chapter 10

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.

```
10 'Demo backspace and delete codes.
20 LPRINT "Backspace does not" ;
30 LPRINT CHR$(8) CHR$(8) CHR$(8) ; 'Three backspaces.
40 LPRINT "=== work."
50 LPRINT "Delete does not" ;
60 LPRINT CHR$(127) CHR$(127) CHR$(127) ; 'Three
   deletes.
70 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.

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

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

```
10 'Prints all block graphic characters.
20 WIDTH "LPT1:",255
30 FOR J = 160 TO 255 STEP 8
40 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

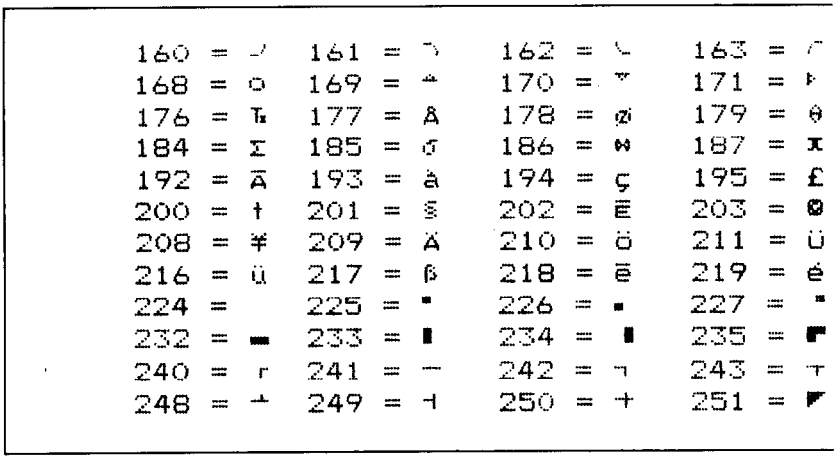


Figure 10-1.

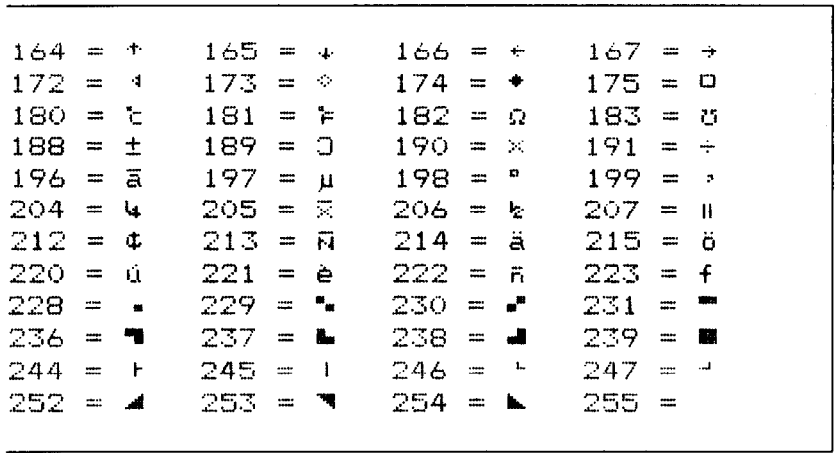
figure, and line 70 resets the line spacing to 1/6 inch. Here is what this program prints:

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)



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	ß
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-instruc-

tions (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 fea-

tures. Then it "calls" the macro in line 60. When line 70 prints the style is "plain vanilla" because the macro has reset it.

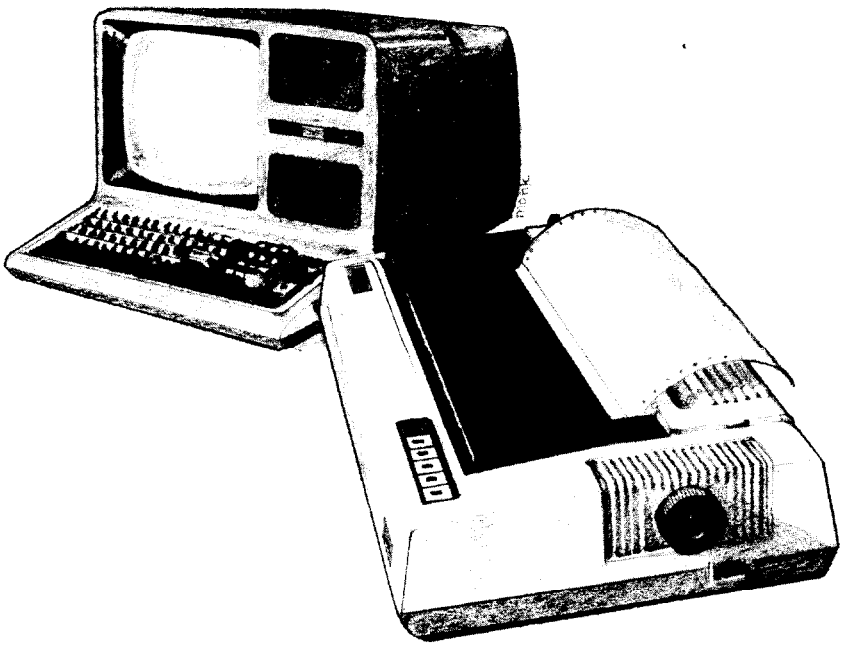
```
10 'Uses macro to reset RADIX to normal.
20 LPRINT CHR$(27) "4" ; 'Italic.
30 LPRINT CHR$(27) "G" ; 'Double-strike.
40 LPRINT CHR$(27) "W" CHR$(1) ; 'Expanded.
50 LPRINT "This line is special."
60 LPRINT CHR$(27) "!" ; 'Use the macro.
70 LPRINT "This line is normal printing."
```

```
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 chap-

ter, *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.

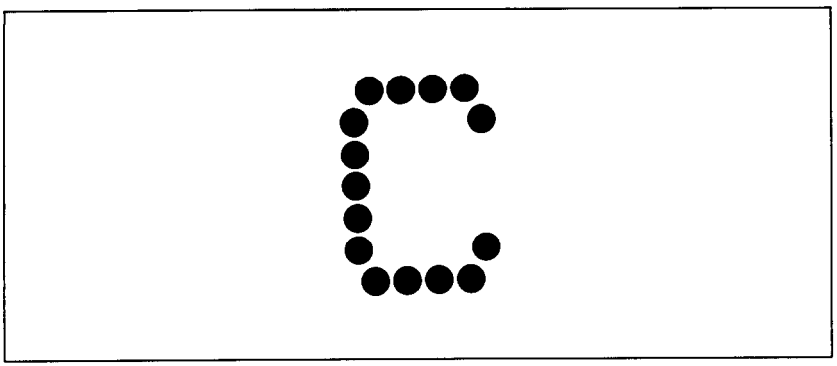


Figure 11-1. *The letter "C" is created by printing 15 dots.*

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

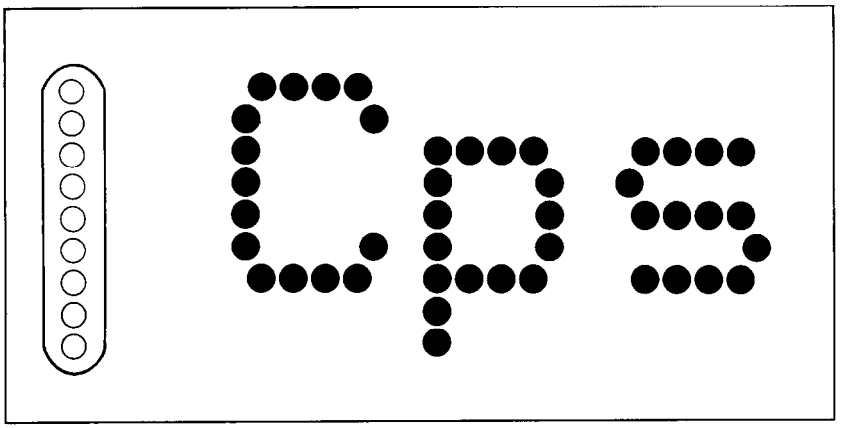


Figure 11-2. *As the printhead moves across the page, each of the wires prints one row of dots.*

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.

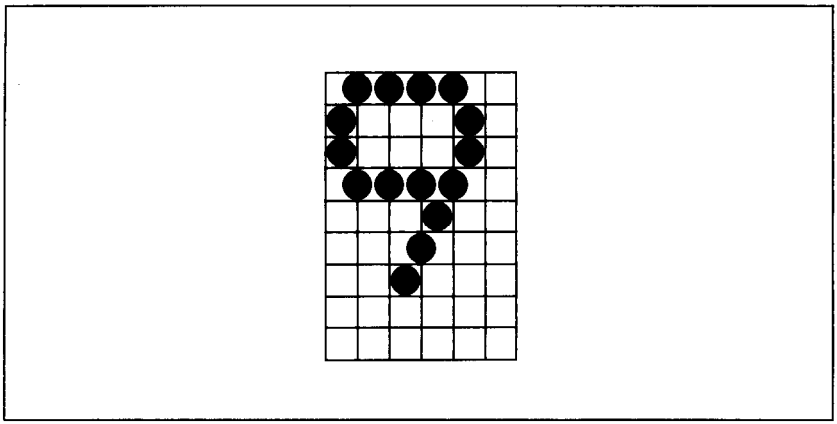


Figure 11-3. *Dots can be inside boxes or straddle the vertical lines of the grid.*

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

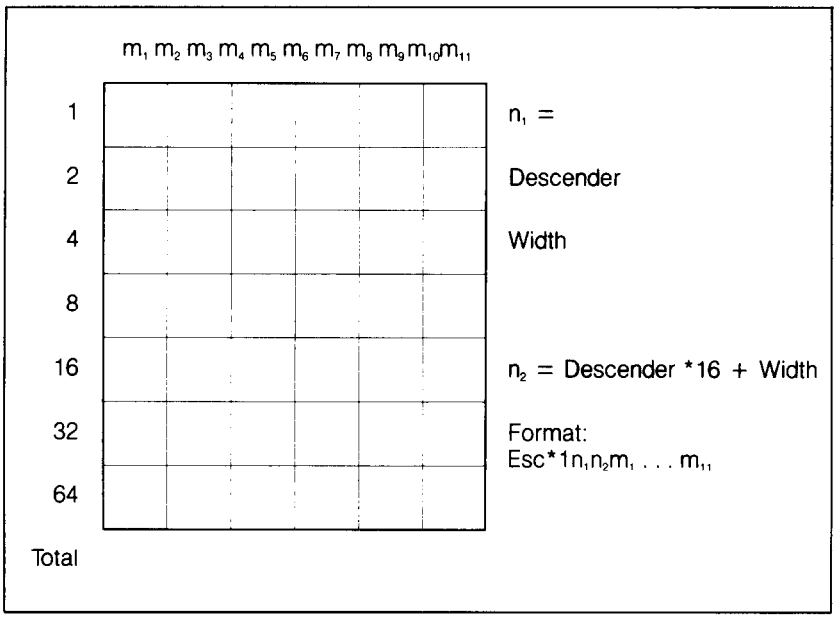


Figure 11-4. *Use this grid (or one similar to it) to define your own characters.*

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

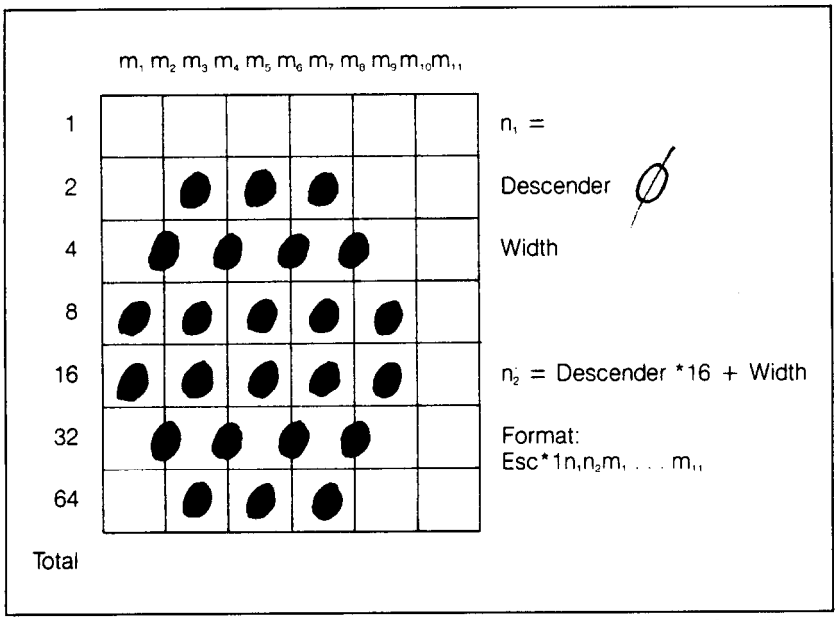


Figure 11-5. *We've designed a character and decided that it would not be a descender, hence the "0" written in.*

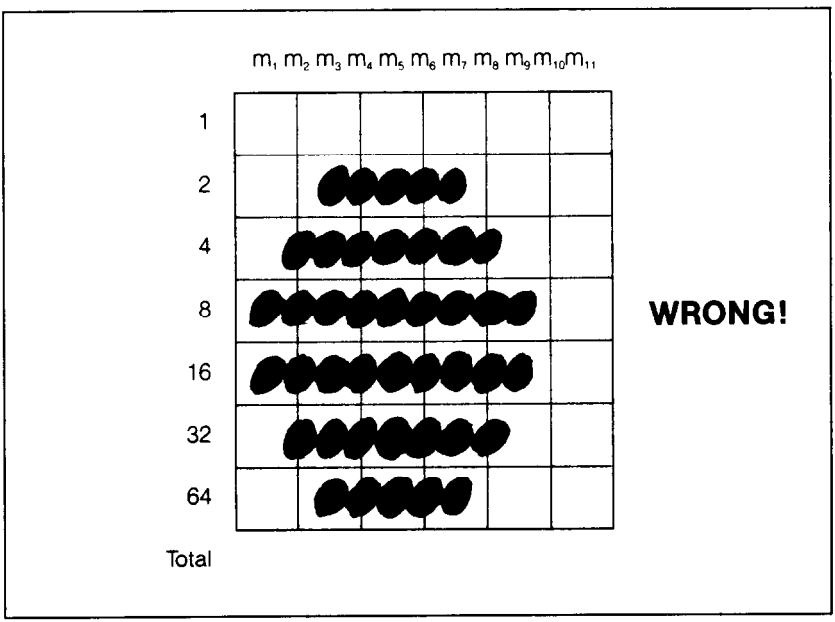


Figure 11-6. *Dots cannot overlap; those in immediately adjacent "half columns" will be ignored when the character is printed.*

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

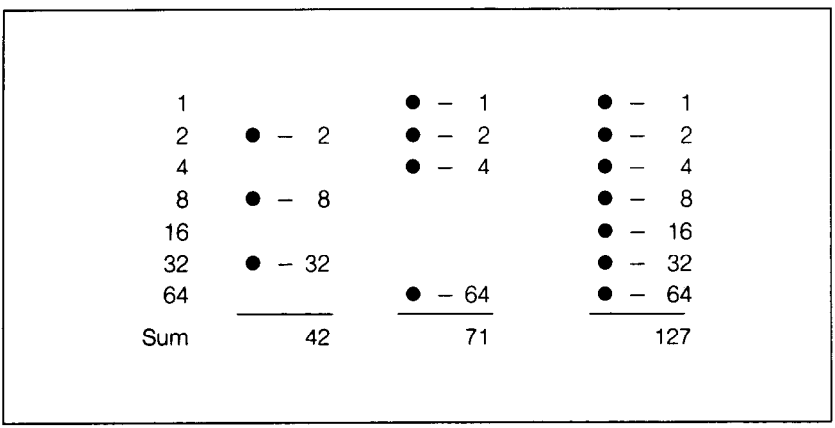


Figure 11-7. *By adding the values of each dot in a column, you'll get a unique description for any combination of dots.*

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

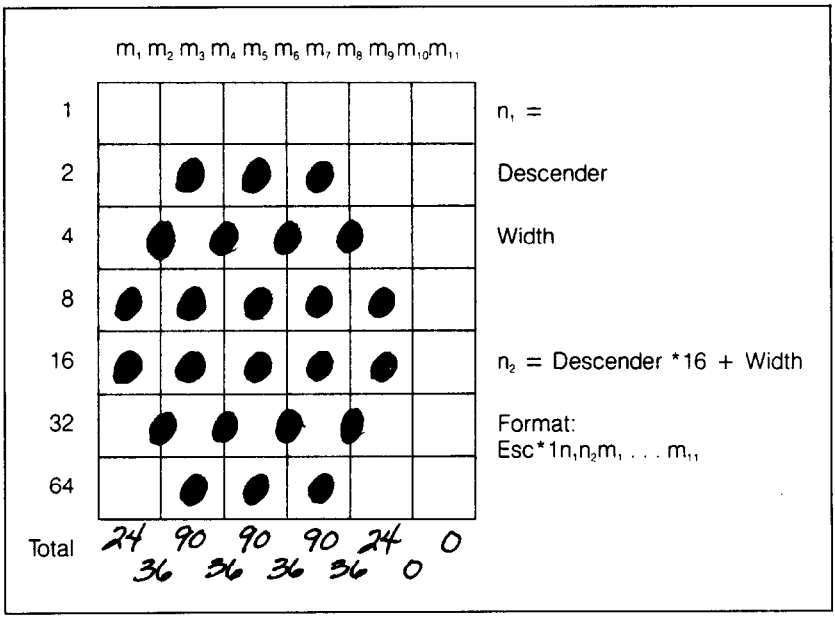


Figure 11-8. *Add the values of the dots in each column and write the sum of each column at the bottom.*

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
```

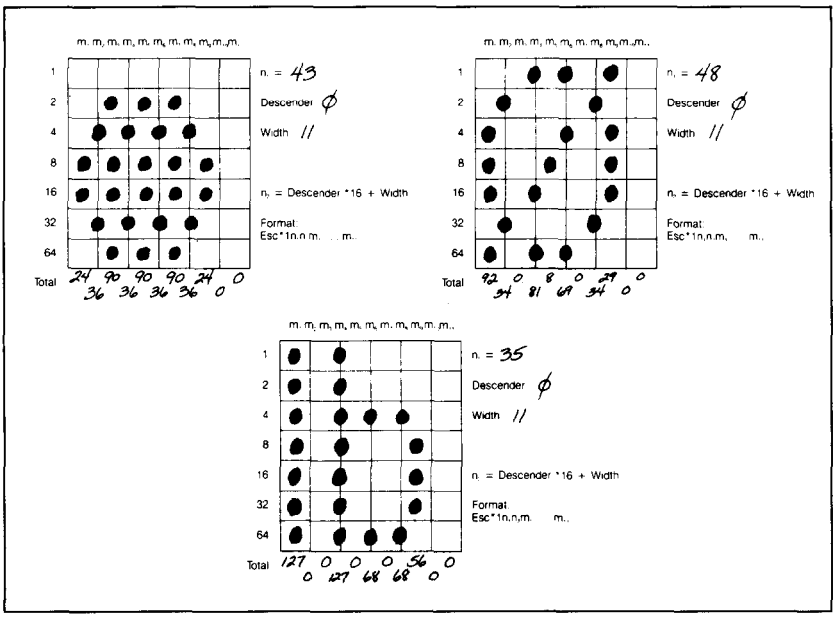


Figure 11-9. *Character designs for our three characters.*

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.)

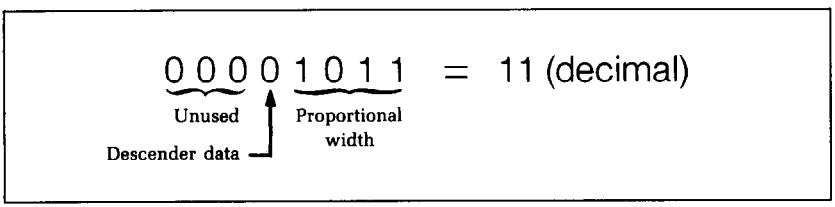


Figure 11-10. *The attribute byte* (n2) *for our bullet character.*

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.

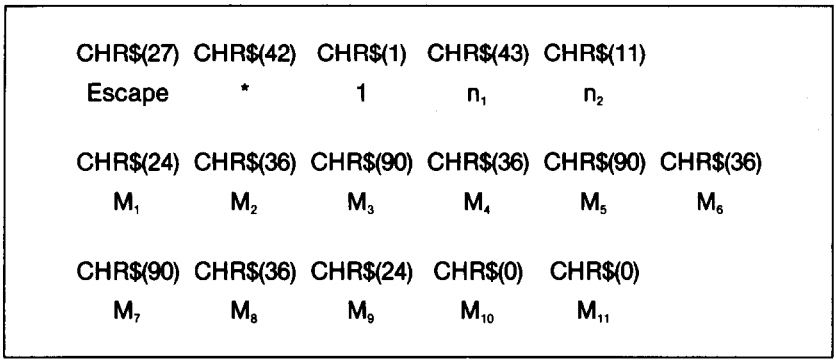


Figure 11-11. *This is the complete command to send our bullet character to the Radix printer.*

```
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:

```
•Ø℔
```

(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:

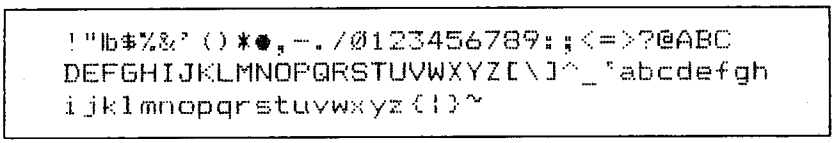


Figure 11-12. *Printout of the download character set, into which all the standard characters have been copied, and the #, +, 0 have been changed.*

```
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
```

```
170 LPRINT
180 'Select RAM set, normal spacing.
190 LPRINT CHR$(27) "$" CHR$(1) ;
200 LPRINT "               Mississippi"
210 'Cancel RAM set, normal spacing.
220 LPRINT CHR$(27) "$" CHR$(0)
230 LPRINT "RAM char set, normal spacing."
240 LPRINT
250 LPRINT
260 'Select RAM set, proportional spacing.
270 LPRINT CHR$(27) "X" CHR$(1) ;
280 LPRINT "               Mississippi"
290 'Cancel RAM set, proportional spacing.
300 LPRINT CHR$(27) "X" CHR$(0)
310 LPRINT "RAM char set, proportional spacing."
320 END
330 DATA "M",11,1,126,1,2,4,8,4,2,1,126,1
340 DATA "i",4,64,61,64,0,0,0,0,0,0,0,0
350 DATA "p",23,127,0,17,0,17,14,0,0,0,0,0
360 DATA "s",6,8,84,0,84,32,0,0,0,0,0,0
```

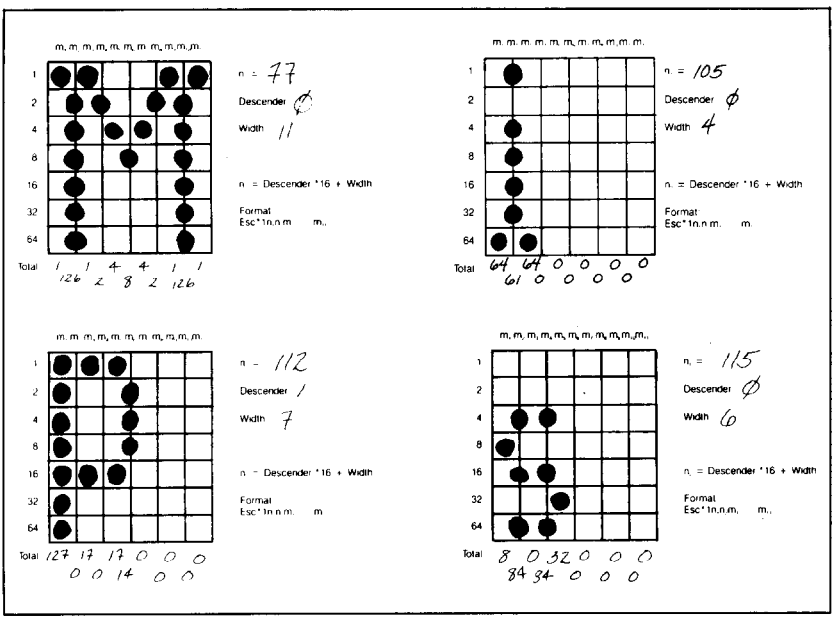


Figure 11-13. *These download characters are defined as proportional characters.*

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.

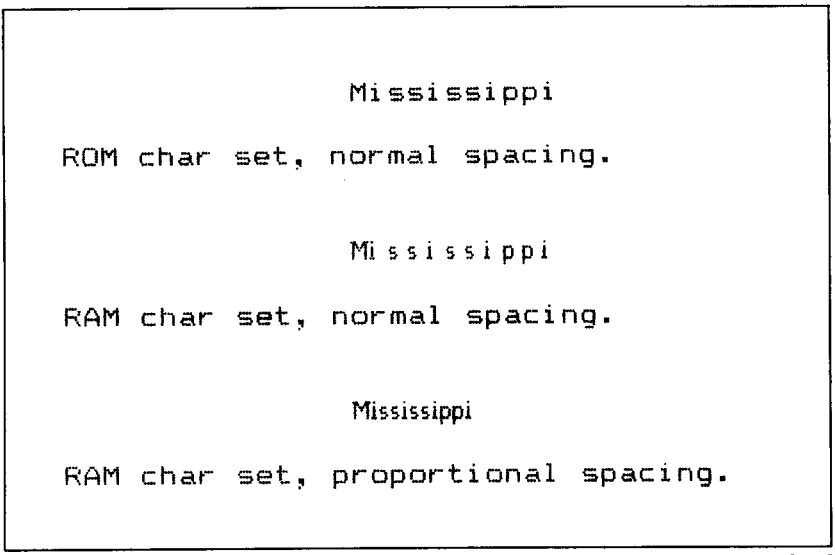


Figure 11-14. *This printout shows the same text, printed with the same download characters, in both normal and proportional widths.*

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.

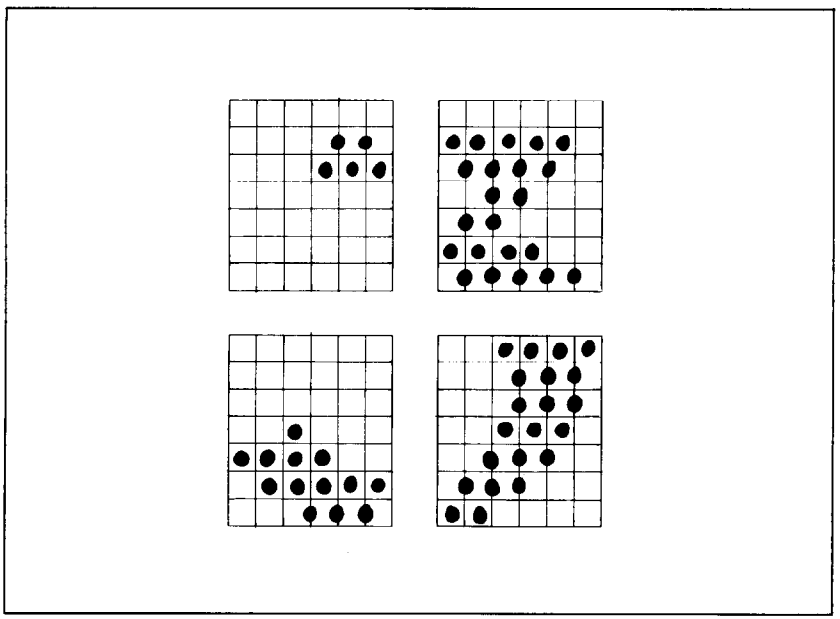


Figure 11-15. *Each digit is made up of four individual characters.*

```
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 11,0,0,0,0,0,0,4,0,4,0,4,126
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 11,0,0,0,0,0,0,12,16,14,0,6,0
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 11,0,0,0,0,0,0,0,4,2,4,2,4
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 11,0,0,0,0,0,0,64,36,88,32,16
580 DATA 11,0,0,64,32,64,56,64,60,2,12,0
590 DATA 11,0,8,4,10,5,10,5,8,4,72,4
600 DATA 11,88,38,89,38,89,6,73,4,8,6,0
610 ' FIVE
620 DATA 11,0,0,0,0,64,32,84,50,76,34,68
630 DATA 10,34,68,34,68,34,68,2,68,2,0,0
640 DATA 10,0,32,24,101,24,97,0,64,0,64,0
650 DATA 11,64,0,96,1,48,15,48,15,16,15,0
660 ' SIX
670 DATA 11,0,96,0,112,0,120,0,92,0,102,0
680 DATA 11,98,0,98,0,98,0,70,0,14,0,6
690 DATA 11,7,8,23,8,55,8,99,0,65,0,64
700 DATA 11,0,96,0,112,1,62,1,30,1,14,0
710 ' SEVEN
720 DATA 11,0,16,8,6,8,6,8,6,8,6,8
730 DATA 9,70,8,102,8,54,8,6,0,2,0,0
740 DATA 11,0,64,0,96,0,120,0,124,0,30,1
750 DATA 9,6,1,0,0,0,0,0,0,0,0,0
760 ' EIGHT
770 DATA 11,0,0,0,0,24,36,24,102,24,102,0
780 DATA 11,67,0,67,0,99,28,34,28,34,28,0
790 DATA 11,12,18,44,19,108,19,96,1,64,0,64
800 DATA 11,0,96,1,112,15,48,15,16,14,0,0
810 ' NINE
820 DATA 11,0,0,120,4,120,6,120,6,0,3,0
830 DATA 11,3,0,3,0,67,4,123,4,122,4,120
840 DATA 11,48,0,56,0,113,0,99,0,99,0,99
850 DATA 11,0,115,0,57,0,31,0,15,0,7,0
860 ' SPACE
870 DATA 11,0,0,0,0,0,0,0,0,0,0,0
```

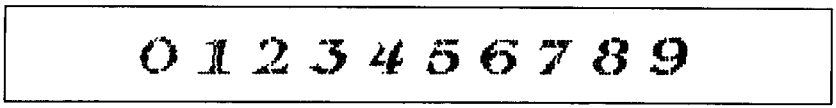


Figure 11-16. *The output for characters like this must be carefully planned.*

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.

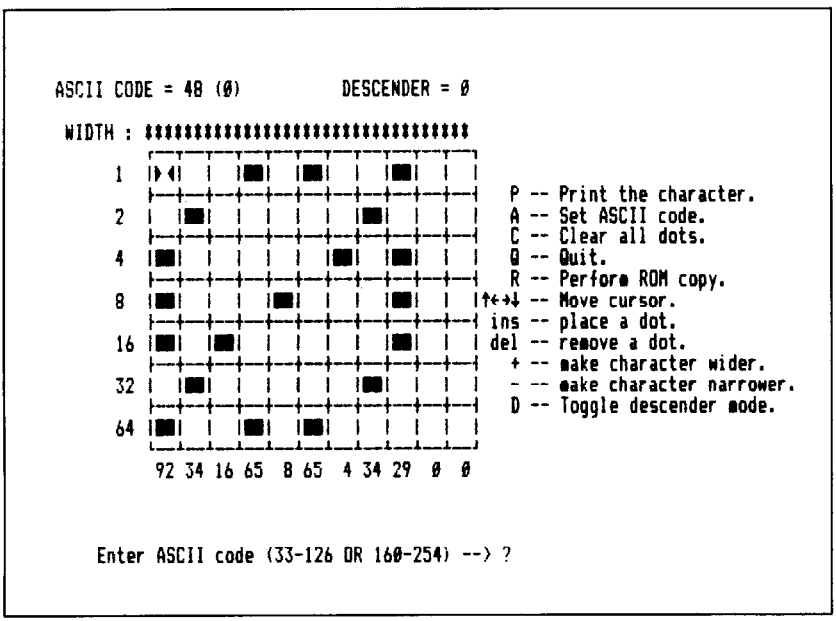


Figure 11-17. *DLEDIT screen display shows ASCII code and character layout.*

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)
```

```
630 FOR I=0 TO 6:LOCATE  5+I*2,6:PRINT 2^I;:NEXT I
640 '
650 'Put in dots.
660 FOR H = 1 TO 11 : FOR J = 1 TO 7 : Z(J,H) = 0
700 NEXT J : NEXT H
710 FOR H = 1 TO 11 : GOSUB 1200 : NEXT H
720 X=1:Y=1:G=1:H=1
730 GOSUB 1300
740 '
750 'Paint menu.
760 LOCATE 6,47 : PRINT "P -- Print the character."
770 LOCATE 7,47 : PRINT "A -- Set ASCII code."
780 LOCATE 8,47 : PRINT "C -- Clear all dots."
790 LOCATE 9,47 : PRINT "Q -- Quit."
800 LOCATE 10,47 : PRINT "R -- Perform ROM copy."
810 LOCATE 11,44 : PRINT CHR$(24) CHR$(27) CHR$(26)
   CHR$(25) ;
820 PRINT " -- Move cursor."
830 LOCATE 12,45:PRINT "ins -- place a dot.";
840 LOCATE 13,45:PRINT "del -- remove a dot.";
850 LOCATE 14,47 : PRINT "+ -- make character wider." ;
860 LOCATE 15,47 : PRINT "- -- make character narrower."
   ;
870 LOCATE 16,47 : PRINT "D -- Toggle descender mode." ;
880 RETURN
890 '
900 'Edit subroutines.
910 GOSUB 1240:Y=Y-3:H=H-1:IF Y<1 THEN BEEP:Y=1:H=1
920 GOSUB 1300:RETURN
930 GOSUB 1240:Y=Y+3:H=H+1:IF Y>31 THEN BEEP:Y=31:H=11
940 GOSUB 1300:RETURN
950 GOSUB 1240:X=X+2:G=G+1:IF X>13 THEN BEEP:X=13:G=7
960 GOSUB 1300:RETURN
970 GOSUB 1240:X=X-2:G=G-1:IF X<1 THEN BEEP:X=1:G=1
980 GOSUB 1300:RETURN
990 IF Z(G,H-1)=1 OR Z(G,H+1)=1 THEN BEEP:RETURN
1000 Z(G,H) = 1 : COLOR 31,1
1010 LOCATE X+4,Y+10 : PRINT SC$ ; : COLOR 7,0
1020 GOSUB 1150 : RETURN
1030 Z(G,H)=0 : COLOR 7,0
1040 LOCATE X+4,Y+10 : PRINT CS$ ; : COLOR 7,0
```

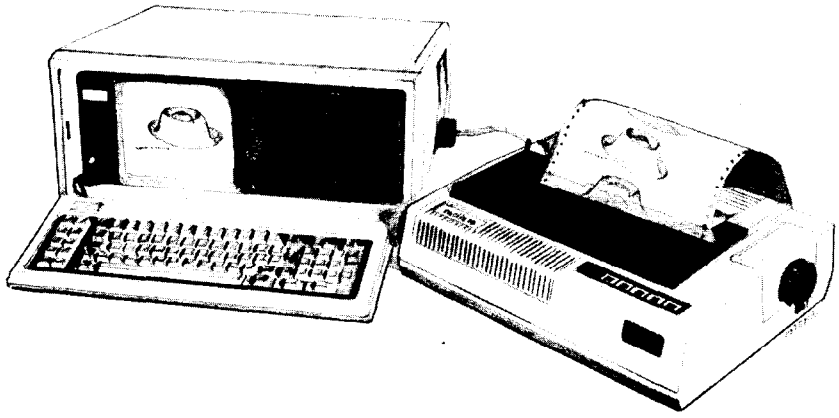
```
105Ø GOSUB 115Ø : RETURN
106Ø IF PROWID = 11 THEN BEEP : RETURN
107Ø PROWID = PROWID + 1
108Ø GOSUB 182Ø : RETURN
109Ø IF PROWID = 4 THEN BEEP : RETURN
11ØØ PROWID = PROWID - 1
111Ø GOSUB 182Ø : RETURN
112Ø IF DESC = 1 THEN DESC = Ø : GOTO 114Ø
113Ø DESC = 1
114Ø GOSUB 182Ø : RETURN
115Ø '
116Ø 'Subroutine to calculate a column value & print it.
117Ø MM(H) = Ø : FOR J=1 TO 7
118Ø MM(H)=MM(H)+Z(J,H)*2^(J-1)
119Ø NEXT J : GOSUB 12ØØ : RETURN
12ØØ '
121Ø 'Subroutine to print a column value.
122Ø LOCATE 19,7+H*3 : PRINT RIGHT$("  "+STR$(MM(H)),3)
   ;
123Ø RETURN
124Ø '
125Ø 'Subroutine to remove the cursor.
126Ø LOCATE X+4,Y+1Ø
127Ø IF Z(G,H) = Ø THEN PRINT "  " ;
128Ø IF Z(G,H) = 1 THEN COLOR 7,Ø : PRINT SC$ ;
129Ø RETURN
13ØØ '
131Ø 'Subroutine to place the cursor.
132Ø LOCATE X+4,Y+1Ø
133Ø IF Z(G,H)=1 THEN COLOR 31,1 : PRINT SC$ ; : COLOR
   7,Ø
134Ø IF Z(G,H)=Ø THEN COLOR 7,Ø : PRINT CS$ ;
135Ø RETURN
136Ø '
137Ø 'Subroutine to print current character.
138Ø GOSUB 2Ø5Ø
139Ø LPRINT "ASCII code =" AS : LPRINT
14ØØ PRINT #2,REC$ ; 'Download the character.
141Ø LPRINT CHR$(27) "B" CHR$(3) "Condensed"
142Ø LPRINT RAMNML$ STRING$(21,AS)
143Ø LPRINT RAMNMLOFF$
144Ø LPRINT CHR$(27) "B" CHR$(2) "Elite"
145Ø LPRINT RAMNML$ STRING$(15,AS)
146Ø LPRINT RAMNMLOFF$
```

```
147Ø LPRINT CHR$(27) "B" CHR$(1) "Pica"
148Ø LPRINT RAMNML$ STRING$(12,AS)
149Ø LPRINT RAMNMLOFF$
15ØØ LPRINT CHR$(27) "W" CHR$(1) "Expanded"
151Ø LPRINT RAMNML$ STRING$(6,AS)
152Ø LPRINT RAMNMLOFF$ CHR$(27) "W" CHR$(Ø)
153Ø LPRINT "Character set (normal width)"
154Ø LPRINT RAMNML$;
155Ø FOR I=33 TO 126 : LPRINT CHR$(I); : NEXT : LPRINT
156Ø FOR I=16Ø TO 254 : LPRINT CHR$(I); : NEXT : LPRINT
157Ø LPRINT RAMNMLOFF$
158Ø LPRINT "Proportional"
159Ø LPRINT RAMPRO$ STRING$(15,AS)
16ØØ LPRINT RAMPROOFF$
161Ø LPRINT "Character set (proportional)"
162Ø LPRINT RAMPRO$;
163Ø FOR I=33 TO 126 : LPRINT CHR$(I); : NEXT : LPRINT
164Ø FOR I=16Ø TO 254 : LPRINT CHR$(I); : NEXT : LPRINT
165Ø LPRINT RAMPROOFF$
166Ø LPRINT : LPRINT : LPRINT
167Ø LPRINT "Use this data statement to download this
   character."
168Ø GOSUB 2Ø5Ø : LPRINT "DATA 27" ;
169Ø FOR I = 2 TO LEN(REC$)
17ØØ LPRINT "," STR$(ASC(MID$(REC$,I,1))) ;
171Ø NEXT I : LPRINT : LPRINT : LPRINT : LPRINT : RETURN
172Ø '
173Ø 'Subroutine to input desired character code.
174Ø LOCATE 23,5
175Ø INPUT "Enter ASCII code (33-126 OR 16Ø-254) --> " ;
   AS
176Ø GOSUB 2Ø1Ø
177Ø IF AS < 33 OR AS > 254 THEN BEEP : GOTO 174Ø
178Ø IF AS < 16Ø AND AS > 126 THEN BEEP : GOTO 174Ø
181Ø GOSUB 182Ø : RETURN
182Ø '
183Ø 'Subroutine to display header.
184Ø LOCATE 1,1 : PRINT "ASCII CODE =" AS ;
185Ø PRINT "(" CHR$(AS AND &H7F) ;
186Ø IF AS > 127 THEN PRINT " + 128" ;
187Ø PRINT ")          " ;
188Ø LOCATE 1,3Ø : 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.

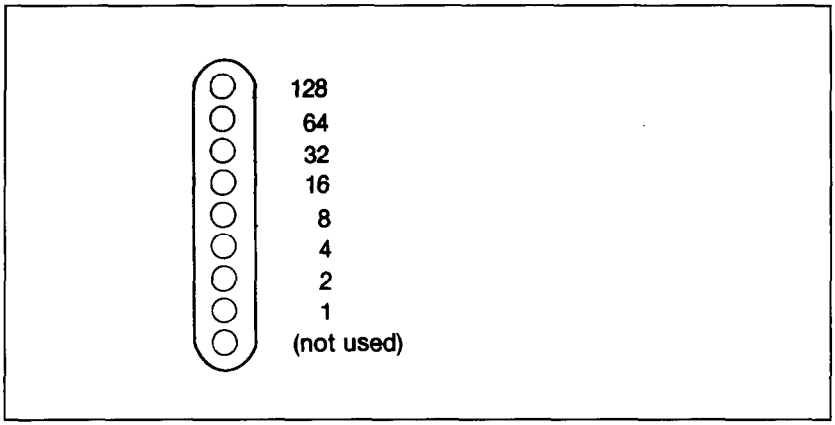


Figure 12-1. *Starting with the most significant bit at the top, each pin of the print head is assigned a value which is a power of two. Note that for 7-bit computers, the top pin has a value of 64, and the bottom two pins are unused.*

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

```
10 'Demo bit graphics.
20 PI = 3.14159
30 WID = 100
```

```
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:

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.

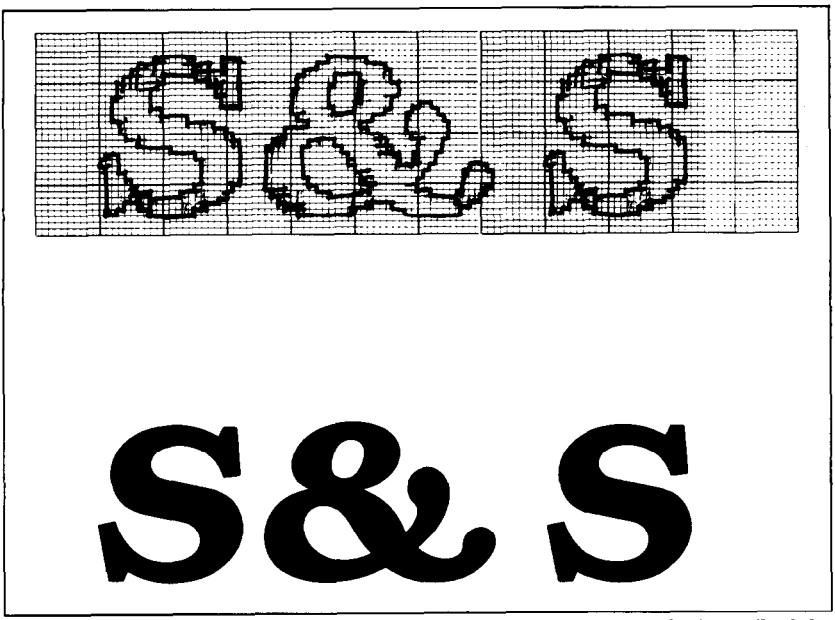


Figure 12-2. *By laying out the logo on graph paper, you can calculate all of the graphics data.*

```
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
```

S&S

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.

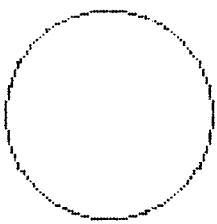
```
1Ø ' General purpose RADIX plotting program.
2Ø '
3Ø 'Set program constants.
4Ø MAXCOL% = 75        : MAXROW% = 14
```

```
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.

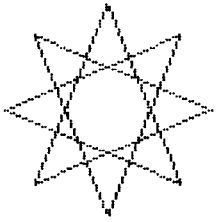
What the program section starting at line 600 actually does is to calculate starting and ending points for a line (in our circle the "lines" are very short—sometimes the starting and ending points are the same). The coordinates of the starting point of the line are assigned to variables X1 and Y1. The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

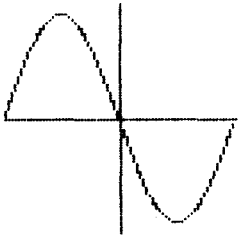
When all the points have been plotted in memory, printing begins at line 150. We first set the line spacing to 6/72 inch using the ⟨ESC⟩ "A" command. This is so that there are no gaps between rows of dots. Then the loop from line 170 to line 240 prints the dot graphics image one line (which is six dots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

```
600 '
610 'Subroutine to plot a star.
620 '
630 RAD = 9
640 FOR ANG% = 0 TO 360 STEP 45
650 RANG = ANG% * 3.14159 / 180
660 RANG2 = (ANG% + 135) * 3.14159 / 180
670 X1 = RAD * COS(RANG) + 10
680 Y1 = RAD * SIN(RANG) + 10
690 X2 = RAD * COS(RANG2) + 10
700 Y2 = RAD * SIN(RANG2) + 10
710 GOSUB 270
720 NEXT ANG%
730 RETURN
```



```
600 '
610 'Subroutine to plot a sine wave.
620 '
630 X1 = 0 : Y1 = 10 : X2 = 20 : Y2 = 10
640 GOSUB 270
650 X1 = 10 : Y1 = 0 : X2 = 10 : Y2 = 20
660 GOSUB 270
670 X1 = 0 : Y1 = 10
680 FOR X2 = 0 TO 20 STEP .2
690 Y2 = 10 - 9 * SIN(3.14159 * X2 / 10) : GOSUB 270
700 NEXT X2
710 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:

```
10 '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$(Ø)
1ØØ OPEN "LPT1:" AS #1 : WIDTH #1,255
11Ø DIM BIT%(19Ø,36),A$(36),PCT%(25)
12Ø DIM TEXT$(48),PIECETEXT$(25)
13Ø MASK%(1) = 64       : MASK%(4) = 8
14Ø MASK%(2) = 32       : MASK%(5) = 4
15Ø MASK%(3) = 16       : MASK%(6) = 2
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
28Ø RANG = ANG%*6.28/36Ø
29Ø X2 = RAD*COS(RANG)+1Ø  : Y2 = RAD*SIN(RANG)+1Ø
3ØØ GOSUB 64Ø
31Ø NEXT ANG%
32Ø FOR PIECE% = 1 TO NUMBER.PIECES%
33Ø X1 = 1Ø             : Y1 = 1Ø
34Ø TOTAL.PCT%=TOTAL.PCT%+PCT%(PIECE%)
35Ø ANG%=36Ø*TOTAL.PCT%*.Ø1
36Ø RANG = ANG%*6.28/36Ø
37Ø X2 = RAD*COS(RANG)+1Ø  : Y2 = RAD*SIN(RANG)+1Ø
38Ø GOSUB 64Ø
39Ø GOSUB 87Ø
4ØØ NEXT PIECE%
41Ø '
42Ø 'Send chart title to printer.
44Ø LPRINT ESC$ "A" CHR$(6) REVFF$ VTAB$ ;
45Ø LPRINT TITLE.ON$ SPACE$(2Ø-LEN(TITLE$)/2) ;
```

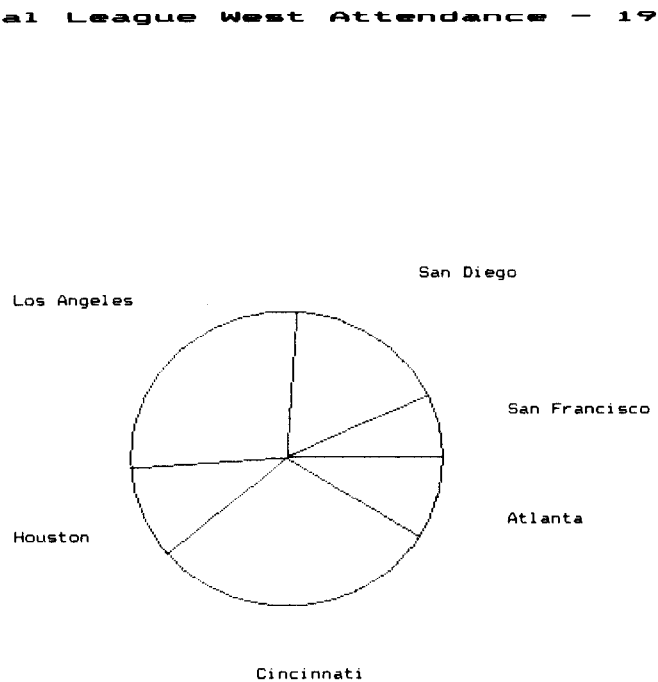
```
46Ø LPRINT TITLE$ TITLE.OFF$
47Ø LPRINT VTAB$ VTAB$ ;
48Ø FOR I = Ø TO 48
49Ø LPRINT TEXT$(I) : NEXT I
5ØØ '
51Ø 'Send bit image map to printer.
52Ø LPRINT REVFF$ VTAB$ VTAB$ VTAB$ ;
53Ø LPRINT LF$ LF$ LF$ LF$ LF$ LF$
54Ø FOR ROW% = Ø TO 35
55Ø LPRINT "                              " ;
56Ø LPRINT ESC$ "K" CHR$(19Ø) CHR$(Ø) ;
57Ø FOR COL% = 1 TO 19Ø
58Ø PRINT#1, CHR$(BIT%(COL%,ROW%)) ; : NEXT
59Ø PRINT#1, LF$
6ØØ PRINT CHR$(176) CHR$(176);
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 CHR$(29) CHR$(2Ø5) CHR$(2Ø5) CHR$(175);
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 '
```

```
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.

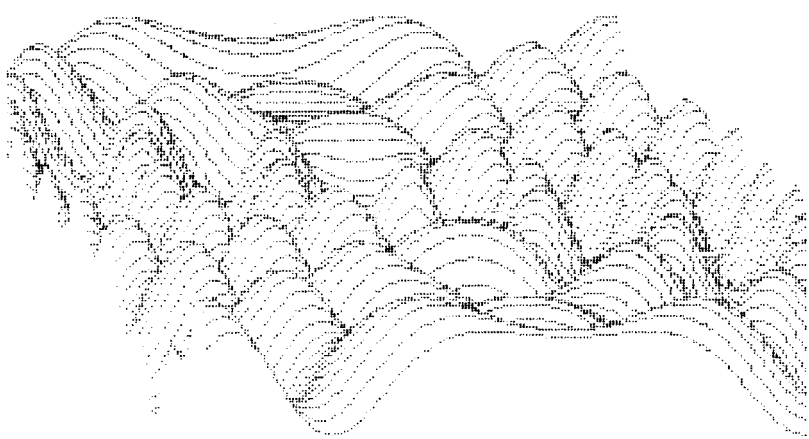
Table 12-2
Dot graphics commands

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 . . .*

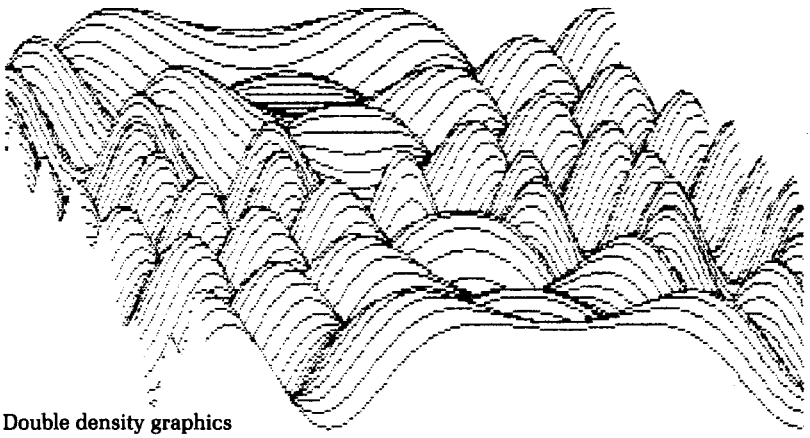
Note: If your computer does not support lowercase characters, use CHR$(121) and CHR$(122) for "y" and "z", respectively.

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 $n1 + 256*n2$.

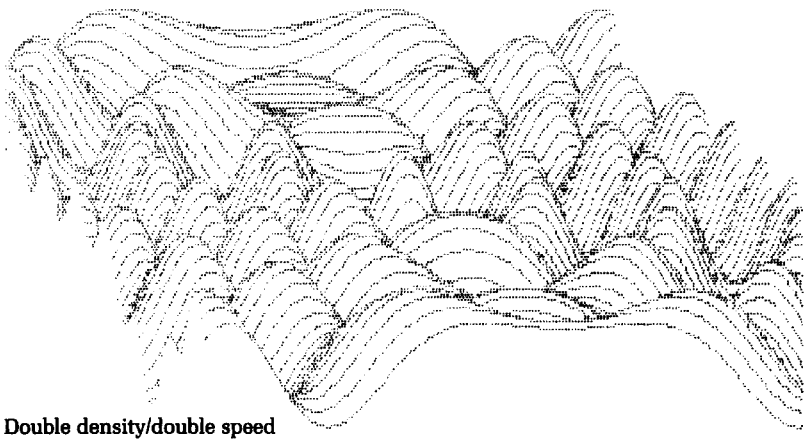
So what do these different modes do? On the following pages are actual size reproductions of printouts of the same image printed in each of the four different graphics modes. They were all printed using the plotting program in this chapter (with a rather complex set of formulas starting at line 600!).



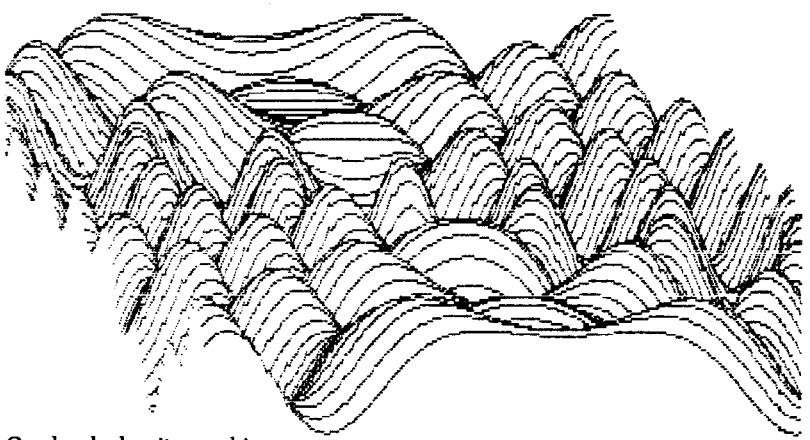
Normal density graphics



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

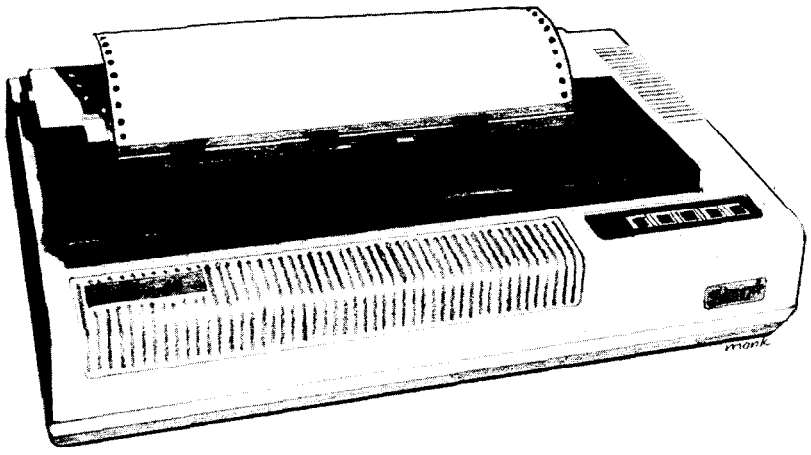
You may write some graphics programs that look just right in the listing, but the printouts aren't quite what you expected. A

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



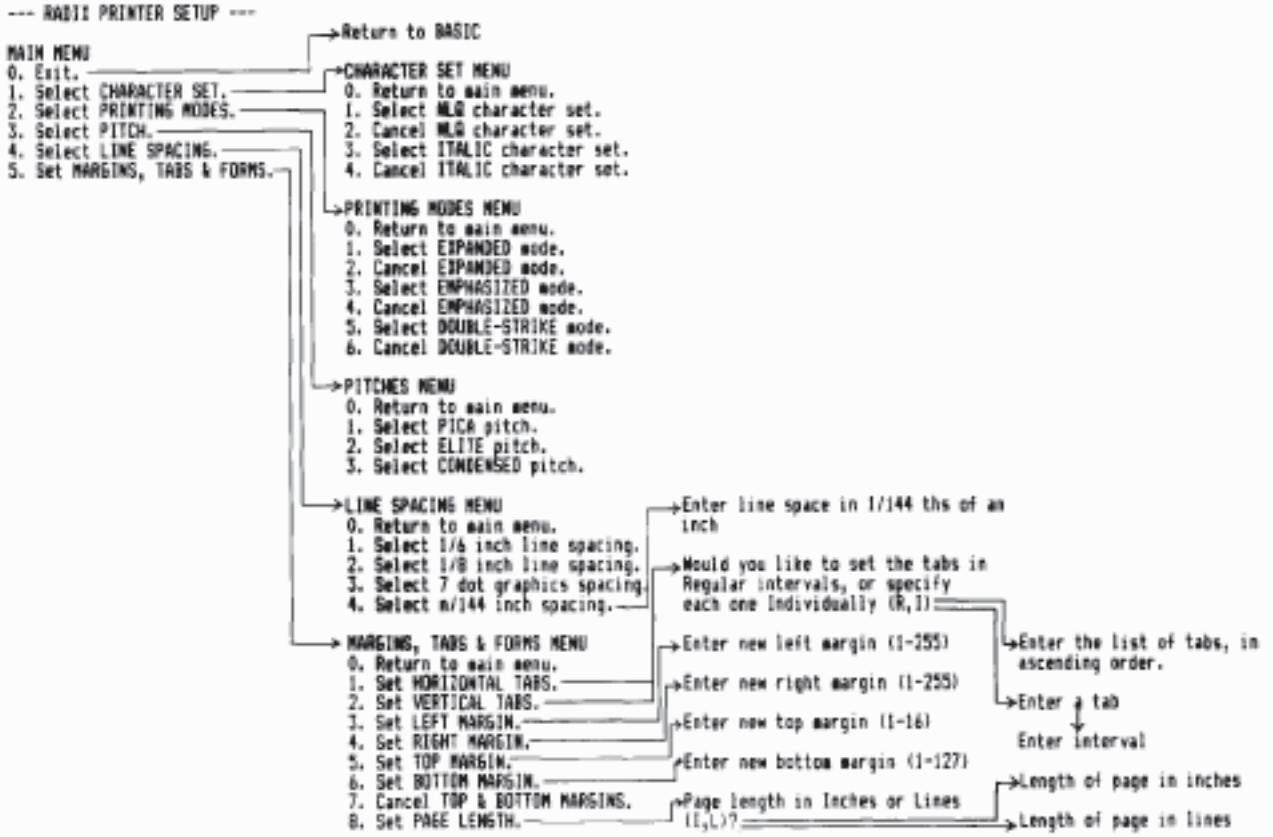
Chapter 13

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



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) "Ø. 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<Ø OR S>3 THEN BEEP : GOTO 430
450 IF S = Ø 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) "Ø. 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<Ø OR S>6 THEN BEEP : GOTO 590
610 IF S = Ø 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) "Ø. 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<Ø OR S>8 THEN BEEP : GOTO 770
790 IF S = Ø THEN RETURN
800 ON S GOSUB 1820,2130,910,970,1030,1090,1150,1650
810 GOTO 640
```

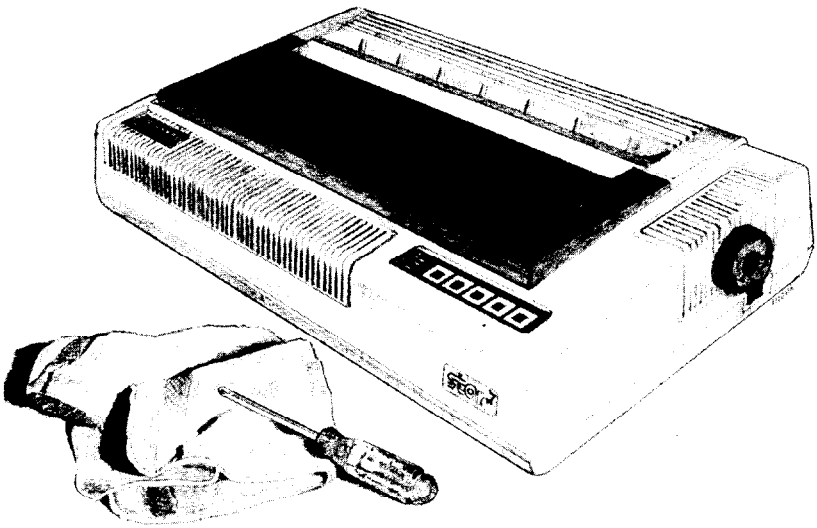
```
820 '
830 'Subroutine to select PICA pitch.
840 S$ = ESC$ + "B" + CHR$(1) : GOSUB 2460 : RETURN
850 '
860 'Subroutine to select ELITE pitch.
870 S$ = ESC$ + "B" + CHR$(2) : GOSUB 2460 : RETURN
880 '
890 'Subroutine to select CONDENSED pitch.
900 S$ = ESC$ + "B" + CHR$(3) : GOSUB 2460 : RETURN
910 '
920 'Subroutine to set LEFT MARGIN.
930 GOSUB 2500
940 INPUT "Enter new left margin (1-255)" ; X
950 IF X < 1 OR X > 255 THEN BEEP : GOTO 930
960 S$ = 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 '
```

```
125Ø 'Subroutine to select LINE SPACING.
126Ø TITLE$ = "LINE SPACING MENU"
127Ø GOSUB 229Ø
128Ø PRINT TAB(TB) "Ø. Return to main menu."
129Ø PRINT TAB(TB) "1. Select 1/6 inch line spacing."
13ØØ PRINT TAB(TB) "2. Select 1/8 inch line spacing."
131Ø PRINT TAB(TB) "3. Select 7 dot graphics spacing."
132Ø PRINT TAB(TB) "4. Select n/144 inch spacing."
133Ø GOSUB 238Ø
134Ø IF S<Ø OR S>4 THEN BEEP : GOTO 133Ø
135Ø IF S = Ø THEN RETURN
136Ø ON S GOSUB 138Ø,141Ø,144Ø,147Ø
137Ø GOTO 124Ø
138Ø '
139Ø 'Subroutine to select 1/6 inch line spacing.
14ØØ S$ = ESC$ + "2" : GOSUB 246Ø : RETURN
141Ø '
142Ø 'Subroutine to select 1/8 inch line spacing.
143Ø S$ = ESC$ + "Ø" : GOSUB 246Ø : RETURN
144Ø '
145Ø 'Subroutine to select 7 dot graphics spacing.
146Ø S$ = ESC$ + "1" : GOSUB 246Ø : RETURN
147Ø '
148Ø 'Subroutine to select n/144 inch line spacing.
149Ø GOSUB 25ØØ
15ØØ INPUT "Enter line space in 1/144 ths of an inch"; X
151Ø IF X < Ø OR X > 255 THEN BEEP : GOTO 149Ø
152Ø S$ = ESC$ + "3" + CHR$(X) : GOSUB 246Ø : RETURN
153Ø '
154Ø 'Subroutine to select EXPANDED print.
155Ø S$ = ESC$ + "W" + CHR$(1) : GOSUB 246Ø : RETURN
156Ø '
157Ø 'Subroutine to cancel EXPANDED printing.
158Ø S$ = ESC$ + "W" + CHR$(Ø) : GOSUB 246Ø : RETURN
159Ø '
16ØØ 'Subroutine to select ITALIC character set.
161Ø S$ = ESC$ + "4" : GOSUB 246Ø : RETURN
162Ø '
163Ø 'Subroutine to cancel ITALIC character set.
164Ø S$ = ESC$ + "5" : GOSUB 246Ø : RETURN
165Ø '
166Ø 'Subroutine to set PAGE LENGTH.
167Ø GOSUB 25ØØ
```

```
168Ø PRINT "Page length in Inches or Lines (I,L)?"
169Ø PRINT TAB(TB) ;
17ØØ A$ = INKEY$ : IF A$ = "" THEN 17ØØ
171Ø IF A$ = "I" OR A$ ="i" THEN 174Ø
172Ø IF A$ = "L" OR A$ ="l" THEN 178Ø
173Ø BEEP : GOTO 17ØØ
174Ø INPUT "Length of page in inches (1-32)" ; X
175Ø IF X < 1 OR X > 32 THEN BEEP : GOTO 167Ø
176Ø S$ = ESC$ + "C" + CHR$(Ø) + CHR$(X)
177Ø GOSUB 246Ø : RETURN
178Ø INPUT "Length of page in lines (1-127)" ; X
179Ø IF X < 1 OR X > 127 THEN BEEP : GOTO 167Ø
18ØØ S$ = ESC$ + "C" + CHR$(X)
181Ø GOSUB 246Ø : RETURN
182Ø '
183Ø 'Subroutine to set HORIZONTAL TABS.
184Ø S$ = ESC$ + "D" : MAX = 255 : GOSUB 185Ø : RETURN
185Ø '
186Ø 'Subroutine to set tabs, either horiz or vert.
187Ø GOSUB 25ØØ
188Ø PRINT "Would you like to set the tabs in"
189Ø PRINT TAB(TB) "Regular intervals, or specify"
19ØØ PRINT TAB(TB) "each one Individually (R,I)"
191Ø A$ = INKEY$ : IF A$ = "" THEN 191Ø
192Ø IF A$ = "R" OR A$ = "r" THEN 2Ø7Ø
193Ø IF A$ = "I" OR A$ = "i" THEN 195Ø
194Ø BEEP : GOTO 185Ø
195Ø PRINT : I = 2 : TBS(1) = -1
196Ø PRINT TAB(TB) "Enter the list of tabs, in"
197Ø PRINT TAB(TB) "ascending order. No more than" MAX
   "."
198Ø PRINT TAB(TB) : INPUT "Enter a tab" ; TBS(I)
199Ø IF TBS(I) < Ø OR TBS(I) > 255 THEN 194Ø
2ØØØ IF TBS(I) = Ø THEN I = 1 : GOTO 2Ø4Ø
2Ø1Ø IF TBS(I) <= TBS(I-1) THEN 194Ø
2Ø2Ø I = I + 1 : IF I > MAX THEN 194Ø
2Ø3Ø GOTO 198Ø
2Ø4Ø I = I + 1
2Ø5Ø S$ = S$ + CHR$(TBS(I)) : IF TBS(I) <> Ø THEN 2Ø4Ø
2Ø6Ø S$ = S$ + CHR$(Ø) : GOSUB 246Ø : RETURN
2Ø7Ø PRINT : PRINT TAB(TB) ; : INPUT "Enter interval" ;
   X
2Ø8Ø IF X < Ø OR X > 255 THEN 194Ø
2Ø9Ø 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.

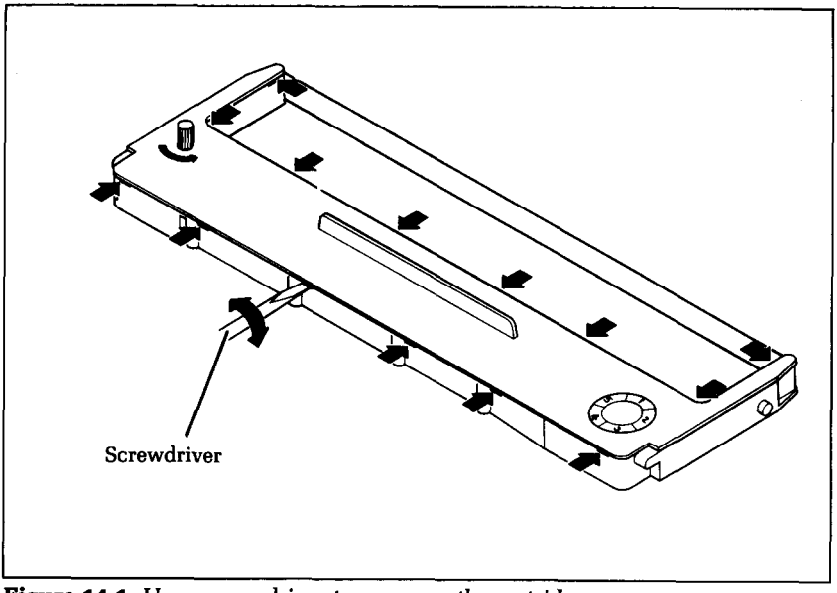


Figure 14-1. *Use a screwdriver to pry open the cartridge.*

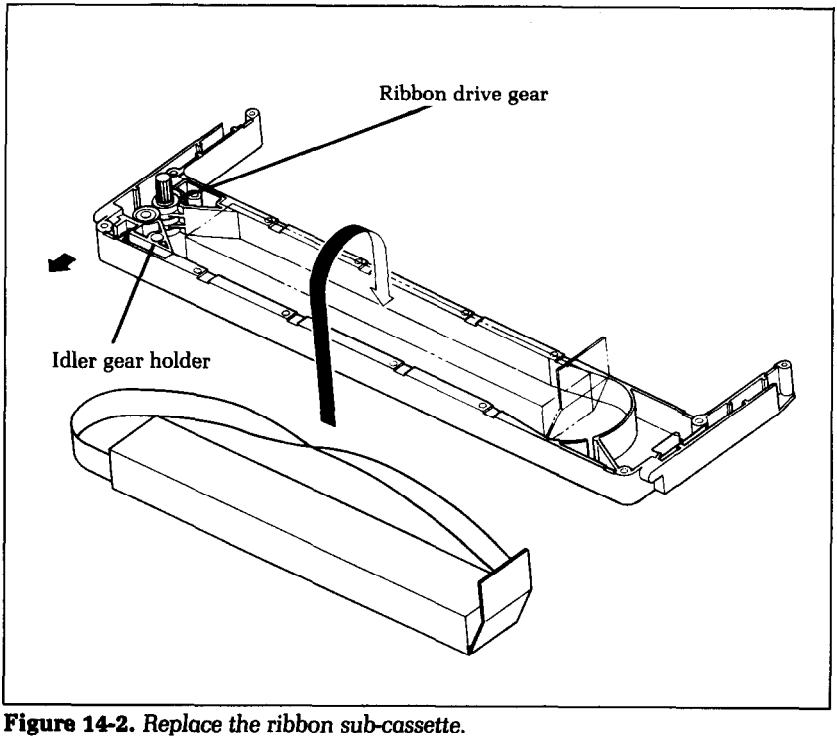


Figure 14-2. *Replace the ribbon sub-cassette.*

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!

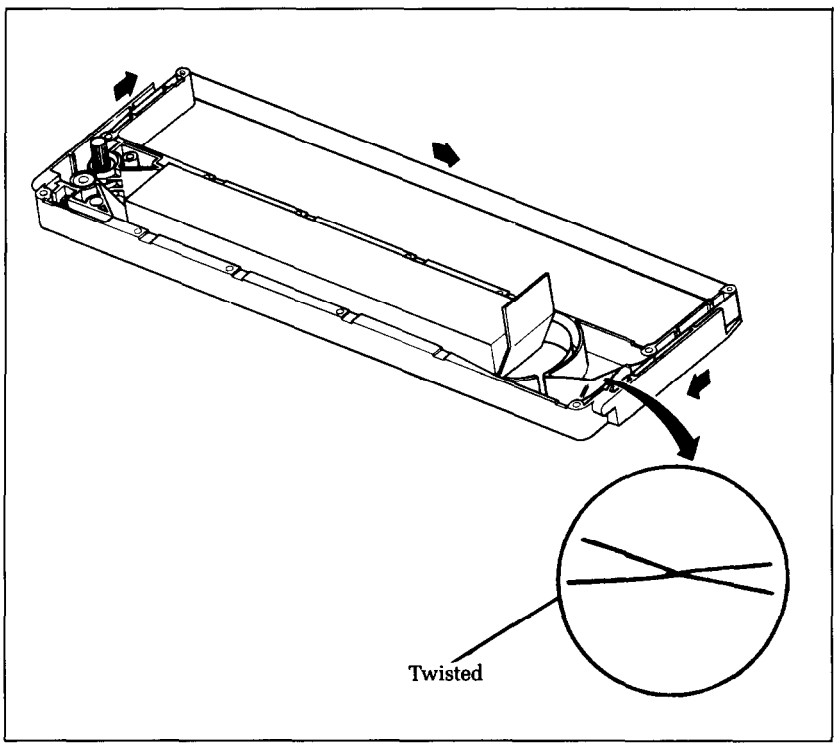


Figure 14-3. *Make sure that the ribbon is not twisted when you thread it through its path.*

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.

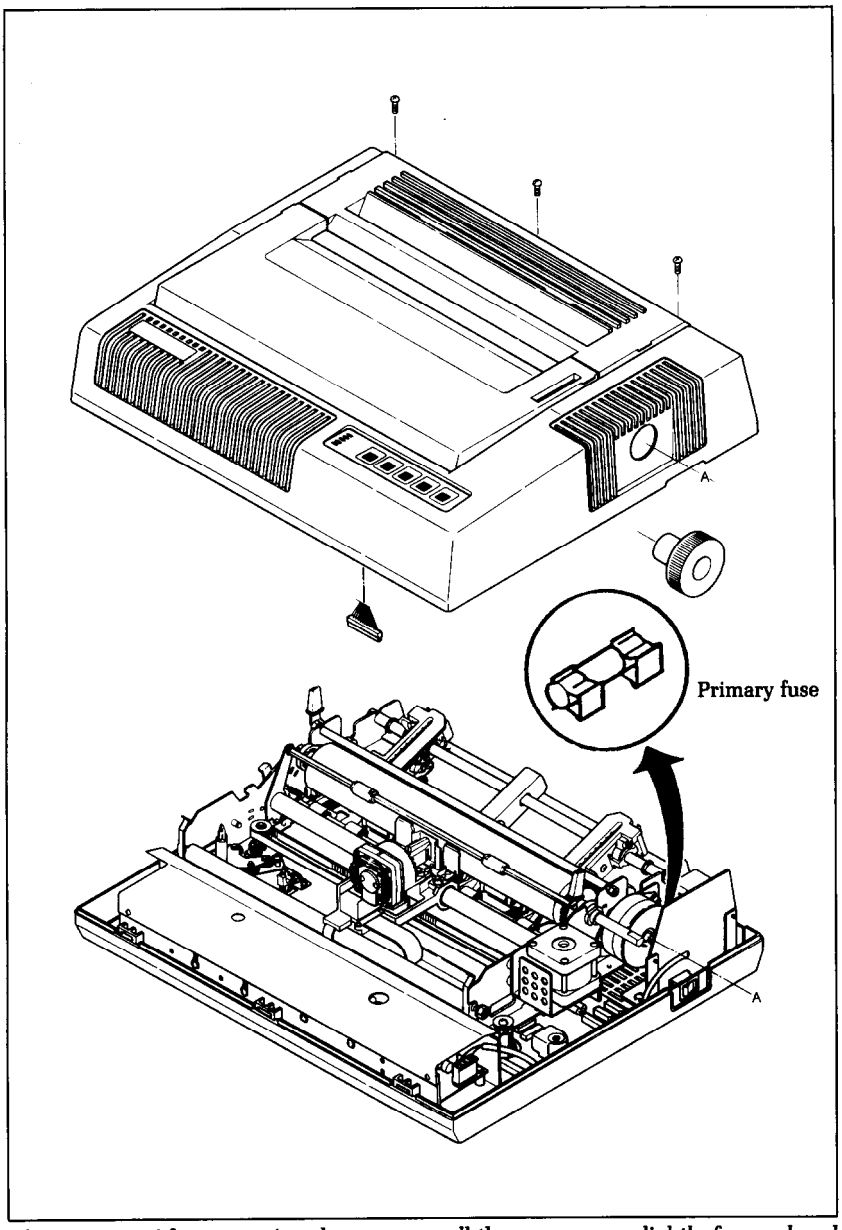


Figure 14-4. *After removing the screws, pull the upper case slightly forward and lift it off the printer. The primary fuse is located near the power switch.*

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.

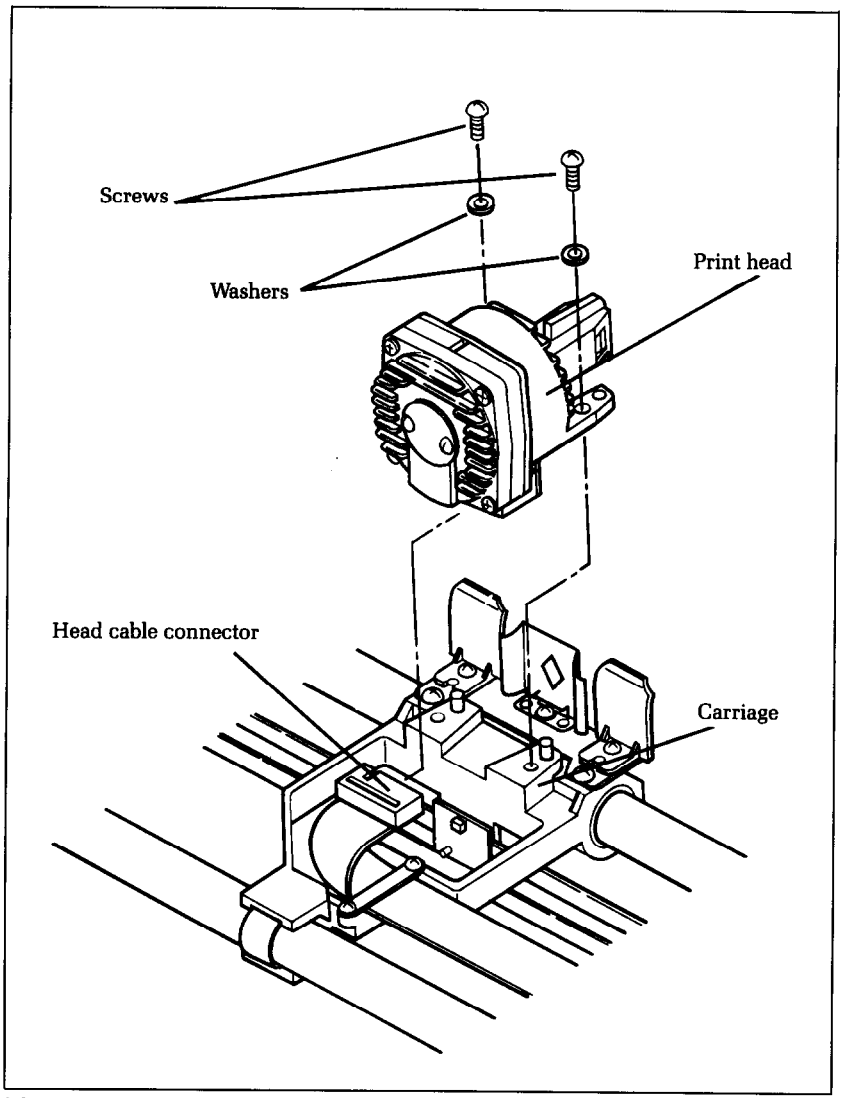


Figure 14-5. *Replacement of Radix's print head is simple.*

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.

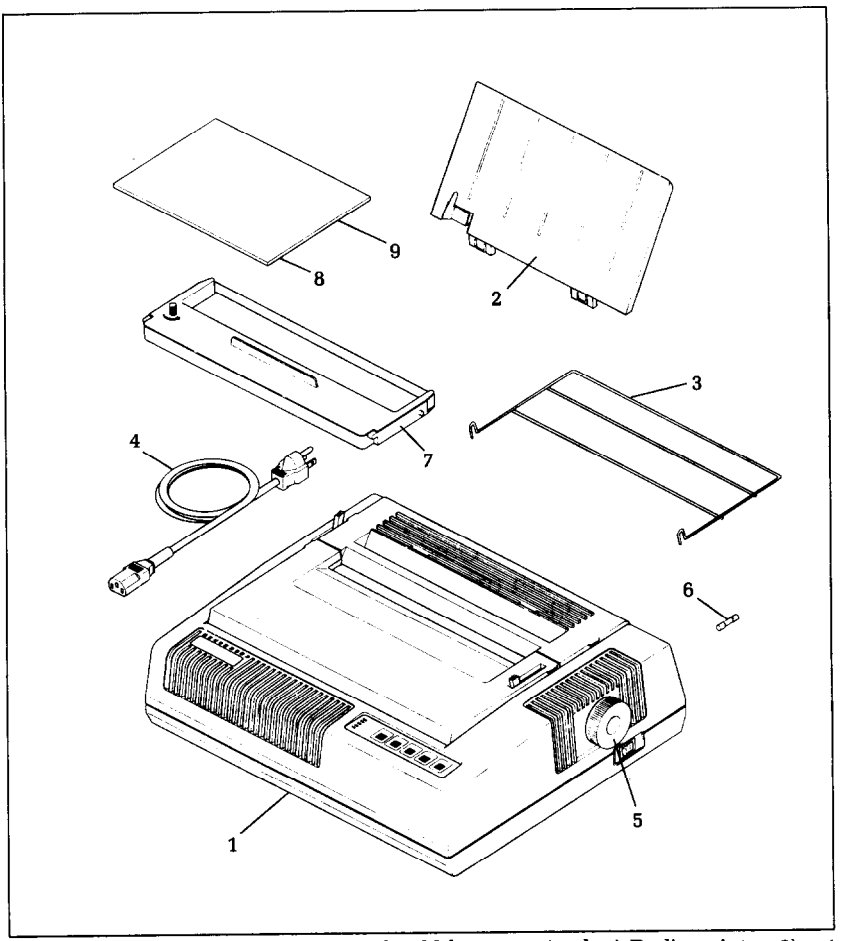


Figure A-1. *Inside the carton you should have received: 1) Radix printer, 2) cut sheet guide, 3) continuous paper guide, 4) power cord, 5) platen knob, 6) spare fuse, 7) ribbon cartridge, 8) this user's manual, and 9) warranty registration card.*

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.

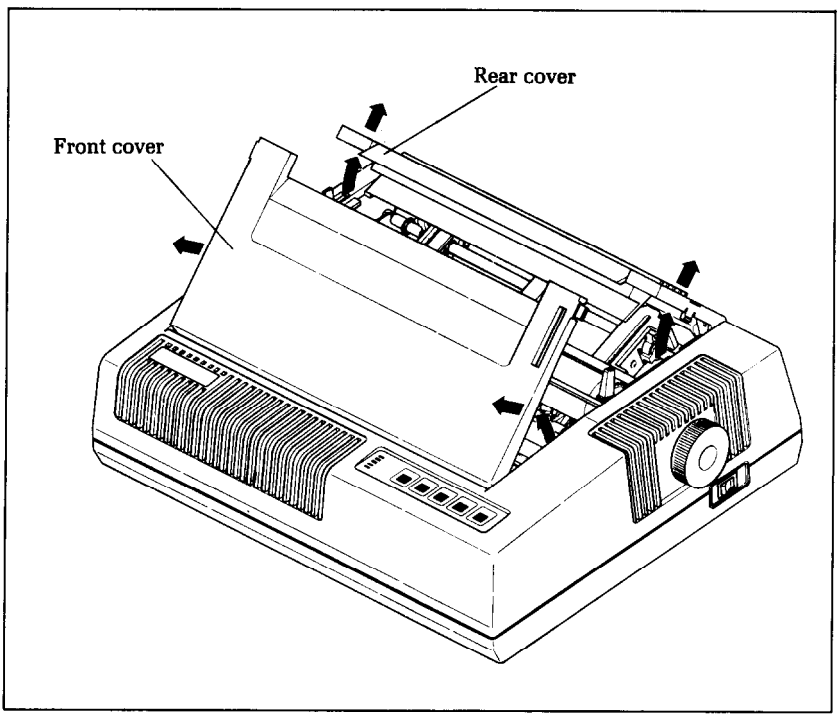


Figure A-2. *Remove the printer covers by tilting them up to about 45°, then lifting straight up.*

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.

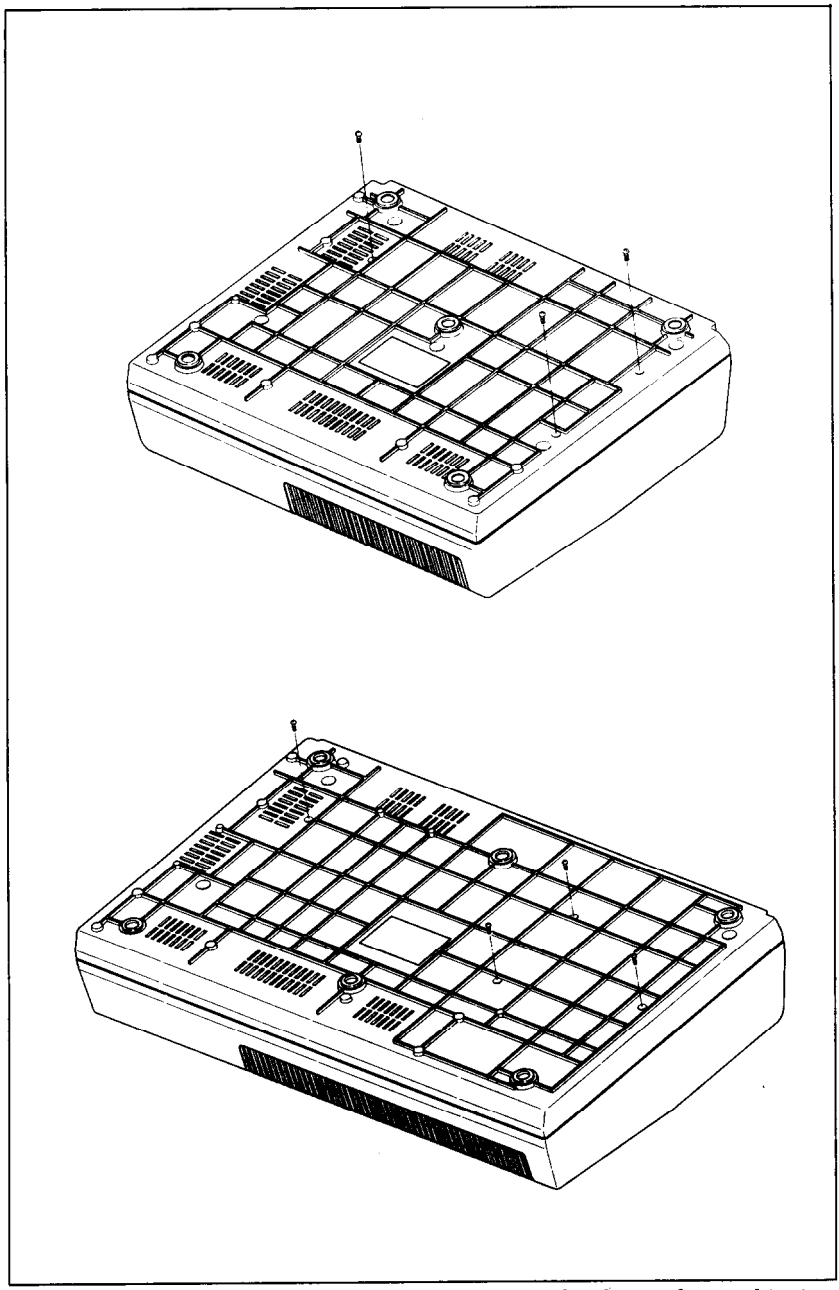


Figure A-3. *Radix-10 has three screws which secure the chassis during shipping; Radix-15 has four. They should be removed before use.*

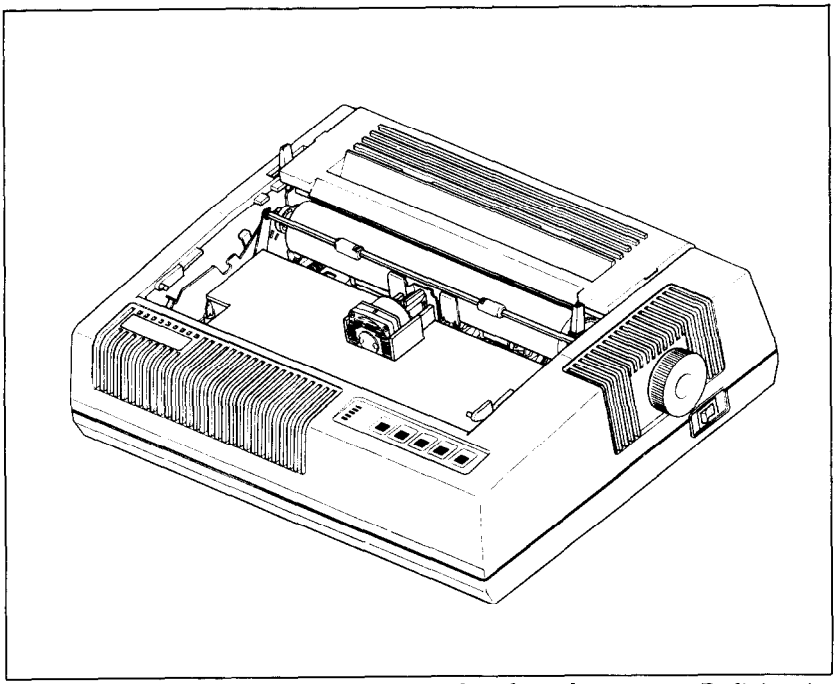


Figure A-4. *Remove the piece of cardboard packing that protects Radix's print head.*

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.

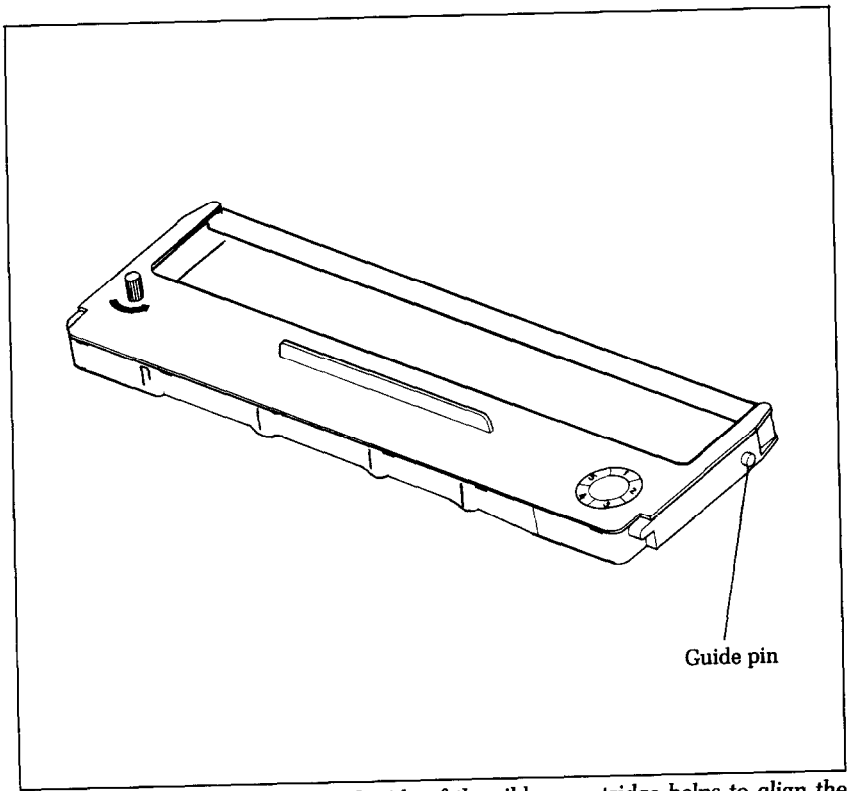


Figure A-5. *A guide pin on each side of the ribbon cartridge helps to align the cartridge during installation.*

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

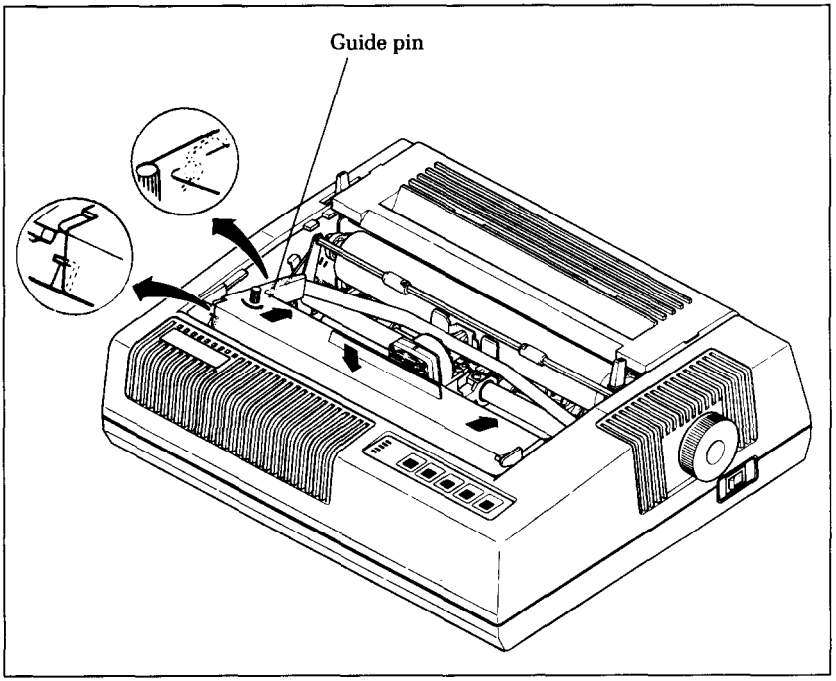


Figure A-6. *Tilt the ribbon cartridge in until the guide pins meet the hooks in the printer frame, then lower the front edge until the holder springs hold it in place.*

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!

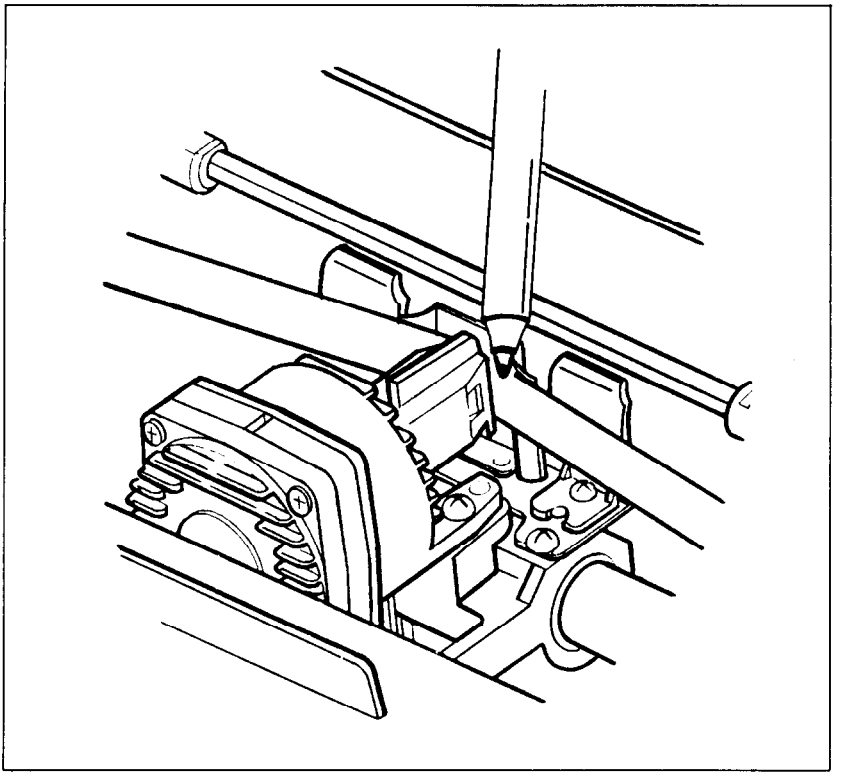


Figure A-7. *Use a ball point pen to place the ribbon between the print head and the ribbon guide. It's important that the ribbon is centered vertically between the print head and the ribbon guide.*

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
4103  FOR I = 1 TO 3: VTAB 16 + I: HTAB 4 + H * 2: PRINT
   " ";: NEXT I
4105 LV$ =  STR$ (MM(H))
4106  FOR I = 1 TO LEN (LV$)
4107  VTAB 16 + I: HTAB 4 + H * 2: PRINT  MID$
   (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
```

```
11Ø  DIM BIT%(19Ø,36),A$(36),PCT%(25),TXT$(48),PTXT$(25)
12Ø ES$ =  CHR$ (27):LF$ = CHR$ (1Ø)
13Ø FF$ =  CHR$ (12):VT$ = CHR$ (11)
14Ø EM$ = ES$ + "E":CE$ = ES$ + "F"
145 RF$ =  CHR$ (27) +  CHR$ (12)
15Ø  FOR I = 1 TO 148:SP$ = SP$ +  CHR$ (Ø): NEXT I
16Ø  FOR I = 1 TO 79:SS$ = SS$ + " ": NEXT I
1ØØØ  REM  SET PROGRAM CONSTANTS
1Ø1Ø MASK%(1) = 64:MASK%(4) = 8
1Ø2Ø MASK%(2) = 32:MASK%(5) = 4
1Ø3Ø MASK%(3) = 16:MASK%(6) = 2
1Ø4Ø LX = 2Ø:LY = 2Ø
1Ø5Ø XFAC = 19Ø / LX:YFAC = 216 / LY
1Ø6Ø  FOR I = Ø TO 48
1Ø7Ø TXT$(I) = SS$
1Ø8Ø  NEXT I
1Ø9Ø  GOSUB 7ØØØ
1Ø92  HOME : PRINT : PRINT : PRINT : PRINT
1Ø93  PRINT "THIS PROGRAM TAKES ABOUT"
1Ø94  PRINT "2 MINUTES TO RUN. PLEASE"
1Ø95  PRINT "TURN ON YOUR PRINTER AND"
1Ø96  PRINT "STAND BY............."
1Ø97  PRINT : PRINT : PRINT
1Ø98  FOR I = 1 TO 31: PRINT "Ø";: NEXT I
1Ø99  PRINT " ": PRINT " "
11ØØ  FOR I = 1 TO NP%: PRINT "Ø";: NEXT I
111Ø  PRINT " "
112Ø  VTAB 12: HTAB 1
2ØØØ  REM  PLOT CURVE
2Ø1Ø RAD = 9
2Ø2Ø X1 = 19:Y1 = 1Ø
2Ø3Ø  FOR ANG = Ø TO 36Ø STEP 12
2Ø4Ø R1 = ANG * 6.28 / 36Ø
2Ø5Ø X2 = RAD *  COS (R1) + 1Ø:Y2 = RAD *  SIN (R1) + 1Ø
2Ø6Ø  GOSUB 4ØØØ
2Ø7Ø  NEXT ANG
2Ø75  VTAB 14: HTAB 1
2Ø8Ø  FOR PI = 1 TO NP%
2Ø9Ø X1 = 1Ø:Y1 = 1Ø
21ØØ TP% = TP% + PCT%(PI)
211Ø ANG = 36Ø * TP% * .Ø1
212Ø R1 = ANG * 6.28 / 36Ø
213Ø X2 = RAD *  COS (R1) + 1Ø:Y2 = RAD *  SIN (R1) + 1Ø
```

```
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$;: GOTO 310
330  IF S = Ø THEN  RETURN
340  ON S GOSUB 1310,1360,1800,1840
350  GOTO 220
360  REM   DISPLAY PITCHES MENU
380 TI$ = "PITCHES MENU"
390  GOSUB 2560
400  PRINT  TAB( TB);"Ø. 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 < Ø OR S > 3 THEN  PRINT BEEP$;: GOTO 440
460  IF S = Ø THEN  RETURN
470  ON S GOSUB 830,880,930
480  GOTO 360
490  REM DISPLAY PRINTING MODE
500 TI$ = "PRINTING MODES MENU"
510  GOSUB 2560
530  PRINT  TAB( TB);"Ø. 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 EMPHASIZED 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 < Ø OR S > 6 THEN  PRINT BEEP$;: GOTO 600
620  IF S = Ø THEN  RETURN
630  ON S GOSUB 1700,1750,2400,2440,2480,2520
640  GOTO 490
650  REM
660  REM  DISPLAY MARGIN, TABS AND FORMS
670 TI$ = "MARGINS, TABS & FORMS MENU"
680  GOSUB 2560
690  PRINT  TAB( TB);"Ø. 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"
730  PRINT  TAB( TB);"4. SET RIGHT MARGIN"
740  PRINT  TAB( TB);"5. SET TOP MARGIN"
750  PRINT  TAB( TB);"6. SET BOTTOM MARGIN"
760  PRINT  TAB( TB);"7. CANCEL TOP & BOTTOM MARGINS"
770  PRINT  TAB( TB);"8. SET PAGE LENGTH"
```

```
78Ø  GOSUB 265Ø
79Ø  IF S < Ø OR S > 8 THEN  PRINT BEEP$;: GOTO 78Ø
8ØØ  IF S = Ø THEN  RETURN
81Ø  ON S GOSUB 2Ø5Ø,236Ø,98Ø,1Ø6Ø,113Ø,121Ø,128Ø,188Ø
82Ø  GOTO 65Ø
83Ø  REM  SELECT PICA
85Ø S$ = ESC$ + "B" +  CHR$ (1)
86Ø  GOSUB 273Ø
87Ø  RETURN
88Ø  REM  SELECT ELITE
89Ø S$ = ESC$ + "B" +  CHR$ (2)
9ØØ  GOSUB 273Ø
91Ø  RETURN
93Ø  REM  SELECT CONDENSED
94Ø S$ = ESC$ + "B" +  CHR$ (3)
96Ø  GOSUB 273Ø
97Ø  RETURN
98Ø  REM  SET LEFT MARGIN
1ØØØ  GOSUB 277Ø
1Ø1Ø  INPUT "ENTER NEW LEFT MARGIN (1-255) ";X
1Ø2Ø  IF X < 1 OR X > 255 THEN  PRINT BEEP$;: GOTO 1ØØØ
1Ø3Ø S$ = ESC$ + "M" +  CHR$ (X)
1Ø4Ø  GOSUB 273Ø
1Ø5Ø  RETURN
1Ø6Ø  REM  SET RIGHT MARGIN
1Ø8Ø  GOSUB 277Ø
1Ø9Ø  INPUT "ENTER NEW RIGHT MARGIN (1-255) ";X
11ØØ  IF X < 1 OR X > 255 THEN  PRINT BEEP$;: GOTO 1Ø8Ø
111Ø S$ = ESC$ + "Q" +  CHR$ (X)
112Ø  GOSUB 273Ø: RETURN
113Ø  REM  SET TOP MARGIN
115Ø  GOSUB 277Ø
116Ø  INPUT "ENTER NEW TOP MARGIN (1-16) ";X
117Ø  IF X < 1 OR X > 16 THEN  PRINT BEEP$;: GOTO 115Ø
118Ø S$ = ESC$ + "R" +  CHR$ (X)
119Ø  GOSUB 273Ø
12ØØ  RETURN
121Ø  REM  SET BOTTOM MARGIN
123Ø  GOSUB 277Ø
124Ø  INPUT "ENTER NEW BOTTOM MARGIN (1-127) ";X
125Ø  IF X < 1 OR X > 127 THEN  PRINT BEEP$;: GOTO 123Ø
126Ø S$ = ESC$ + "N" +  CHR$ (X)
127Ø  GOSUB 273Ø: RETURN
128Ø  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
```

```
187Ø  RETURN
188Ø  REM  SET PAGE LENGTH
19ØØ  GOSUB 277Ø
191Ø  PRINT "PAGE LENGTH IN INCHES OR LINES (I,L)?"
192Ø  PRINT  TAB( TB);
193Ø  GET A$
194Ø  IF A$ = "I" THEN 197Ø
195Ø  IF A$ = "L" THEN 2Ø1Ø
196Ø  PRINT BEEP$;: GOTO 193Ø
197Ø  INPUT "LENGTH OF PAGE IN INCHES (1-32) ";X
198Ø  IF X < 1 OR X > 32 THEN PRINT BEEP;: GOTO 19ØØ
199Ø S$ = ESC$ + "C" +  CHR$ (Ø) +  CHR$ (X)
2ØØØ  GOSUB 273Ø: RETURN
2Ø1Ø  INPUT "LENGTH OF PAGE IN LINES (1-127) ";X
2Ø2Ø  IF X < 1 OR X > 127 THEN  PRINT BEEP$;: GOTO 19ØØ
2Ø3Ø S$ = ESC$ + "C" +  CHR$ (X)
2Ø4Ø  GOSUB 273Ø: RETURN
2Ø5Ø  REM  SET HORIZONTAL TAB
2Ø7Ø S$ = ESC$ + "D":MAX = 255: GOSUB 2Ø8Ø: RETURN
2Ø8Ø  REM  SET TABS
21ØØ  GOSUB 277Ø
211Ø  PRINT "WOULD YOU LIKE TO SET THE TABS IN"
212Ø  PRINT  TAB( TB);"REGULAR INTERVALS, OR SPECIFY"
213Ø  PRINT  TAB( TB);"EACH ONE INDIVIDUALLY (R,I) "
214Ø  GET A$
215Ø  IF A$ = "R" THEN 23ØØ
216Ø  IF A$ = "I" THEN 218Ø
217Ø  PRINT BEEP$;: GOTO 2Ø8Ø
218Ø  PRINT :I = 2:TBS(1) =  - 1
219Ø  PRINT  TAB( TB);"ENTER THE LIST OF TABS, IN "
22ØØ  PRINT  TAB( TB);"ASCENDING ORDER. NO MORE THAN
   ";MAX;"."
221Ø  PRINT  TAB( TB): INPUT "ENTER TAB ";TBS(I)
222Ø  IF TBS(I) < Ø OR TBS(I) > 255 THEN 217Ø
223Ø  IF TBS(I) = Ø THEN I = 1: GOTO 227Ø
224Ø  IF TBS(I) < = TBS(I - 1) THEN 217Ø
225Ø I = I + 1: IF I > MAX THEN 217Ø
226Ø  GOTO 221Ø
227Ø I = I + 1
228Ø S$ = S$ +  CHR$ (TBS(I)): IF TBS(I) < > Ø THEN 227Ø
2285  GOSUB 273Ø
229Ø  RETURN
23ØØ  PRINT : PRINT  TAB( TB);: INPUT "ENTER INTERVAL
   ";X
```

```
2310  IF X < Ø OR X > 255 THEN  PRINT BEEP$;: GOTO 2Ø8Ø
2320  FOR I = 1 TO 255 STEP X
2330 MAX = MAX - 1: IF MAX = Ø THEN 2350
2340 S$ = S$ +  CHR$ (I): NEXT I
2350 S$ = S$ +  CHR$ (Ø): GOSUB 2730: RETURN
2360  REM  VERTICAL TABS
2380 S$ = ESC$ + "P":MAX = 2Ø: GOSUB 2Ø8Ø
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( (4Ø - LEN (TI$)) / 2);TI$
2620  PRINT : PRINT
2630  RETURN
2650  REM  SELECTION
2660  VTAB 19: HTAB 1Ø: PRINT "HIT <P> FOR SAMPLE PRINT"
2665  VTAB 21: HTAB 1Ø: PRINT "SELECTION ";
2670  GET C$
2675  IF C$ = "P" THEN  GOSUB 3ØØØ: GOTO 265Ø
2680  IF C$ < "Ø" OR C$ > "9" THEN  PRINT BEEP$;: GOTO
   267Ø
2690 S = VAL (C$)
2700  VTAB 2Ø:
2710  FOR H = 1Ø TO 4Ø: HTAB H: PRINT " ";: NEXT H
2720  RETURN
2730  REM  OUTPUT COMMAND STRING
2750  PR# 1
2755  PRINT S$;
2758  PR# Ø
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
```