ภาคผนวกที่ 3
รหัสของเครื่องพิมพ์ EPSON

A CLOSER LOOK AT YOUR RX

This chapter is devoted to relieving your curiosity about the "secrets" of your KX printer's operation. Feel free to pass over this if you're an old hand at the printer controls or not particularly inquisitive. Of course, the more you know about your printer, the more comfortable you'll be and the more you'll ultimately be able to do with it. We plan to stick to the basics so don't be worried about being subjected her-e to a sensory overload of arcane computerese.

With that said, let's take a brief, closer look at this remarkable new machine of yours, focusing first on the print head.

One Head's Better Than None

The print head might be thought of as a tiny Gatling gun that uses electrical impulses to "fire" pins instead of bullets. The pins are arranged vertically in a nine-pin column. Rather than just one pin at a time, it can fire any number in any combination. The print head is shown in Fig. 3-1.

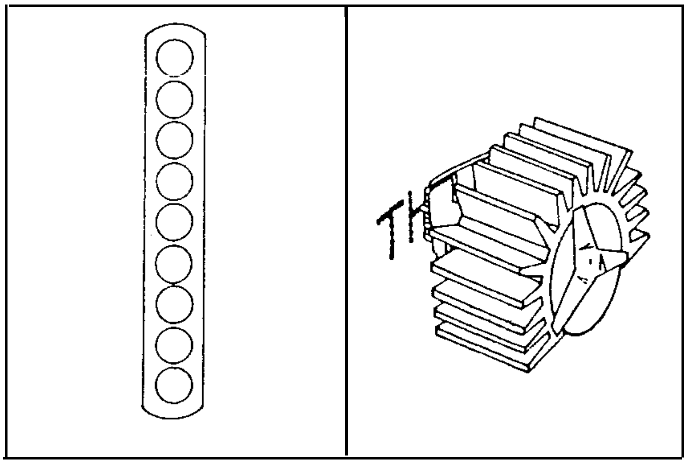


Fig. 3-1 The RX Print Head

Tiiese pins (actually wires) strike the ribbon to make the dots of the dot matrix character. Because there is only one column of pins, the head moves sideways to make a complete character. Each letter, number, or symbol is formed within a six-column, nine-row matrix at up to 100 characters a second.

(Hint: You're right, that is fast. So fast that the head becomes hot and can hurt if you touch it before it cools. So keep that in mind when changing the ribbon or otherwise doing something inside the printing area.)

Printing of the letter T is illustrated in Fig. 3-2. **To** print an uppercase T, the print head fires pin 1 in columns 1 and 2; pins **1** through 7 in column 3; and pin **1** again in columns 4 and 5. Column 6 is left blank for a space before the next character.

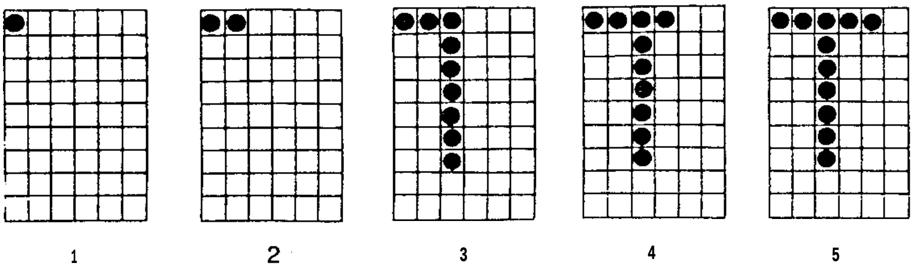


Fig. 3-2 Firing Order of Pins for Letter T

The dot matrix system is what gives the RX printer so much more flexibility over daisy wheel printers. It enables you, with various commands, to conveniently change typefaces, pitch, weight, and other character features without interrupting the printing operation.

Variations on the theme

Each RX character can be printed out in one of three different widths (also called pitches). What this means simply is that the distance traveled by the print head between firings is varied so the dot columns (and as a result, the characters) appear closer together or farther apart.

This provides you with a choice of three different character widths: pica and clite (both familiar sizes to typewriter users) at 10 and 12 characters per inch, and condcnscd, at a hair over 17 characters per inch. Examples of the threc standard cype'sizes are shown in Fig. 3-3.

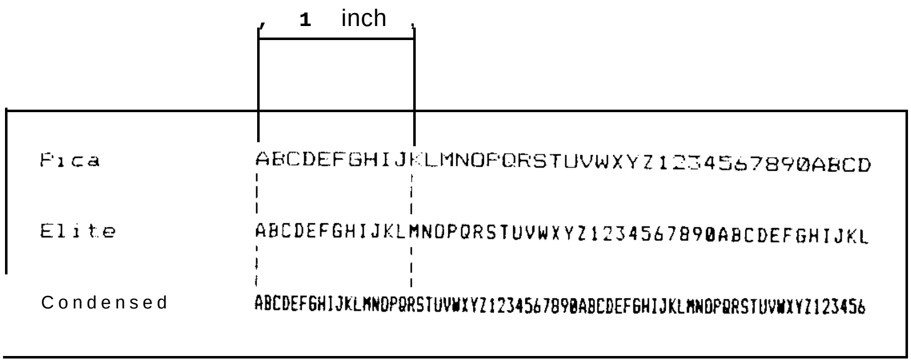

1 inch
Pica
ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890ABCD
Elite
ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890ABCDEFGHIJKL
Condensed
ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ123456

Fig. 3-3 Comparison between the Three Standard Type Widths

Some RX users find they more consistently need condensed type rather than pica or elite. To meet this requirement, rhe printer has a DIP switch that can be set so Condensed mode is automatically selected when the power is turned on. To make this printer adjustment, turn DIP switch l-l to ON. You'll then be ser for printing out financial spreadsheets and other work thar requires especially tight printing.

Later, to change the type to either of the other two pitches, you can instruct the printer with a simple command. That command and others will be explained in full later.

Overlaps

Although each RX character is designed to be five or fewer columns wide, dots can also be placed midway between each main column. This means rhat the dor pattern is printed twice, shifted one half dot to the right.

As shown in Fig. 3-4, the dots printed in these intermediate positions actually overlap with those in the main columns. You can easily see the difference in the comparison between a normal and an "emphasized" letter, as shown in Fig. 3-5.

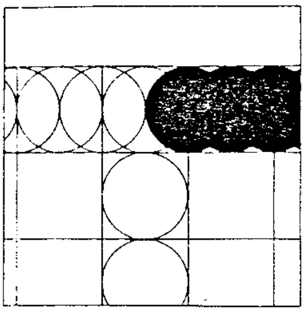


Fig. 3-4 Overlapping Dots

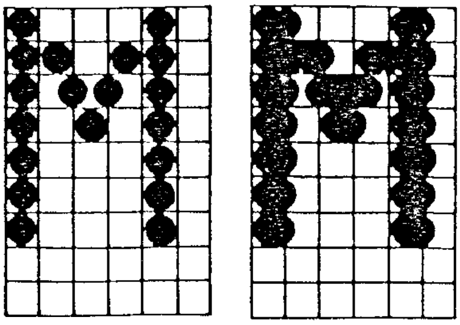


Fig. 3-5 Normal and Emphasized "M"

The print head can overlap dots both horizontally and vertically. To overlap dots vertically, the print head first prints the line of characters normally, returns to the home position, and then prints the same line again, but first advancing the paper ever so slightly (1/216 inch to be exact). This has the effect of closing the vertical spaces between dots.

This is "double strike" printing and is one of the ways that you can print distinctive, solid-looking characters with your RX printer.

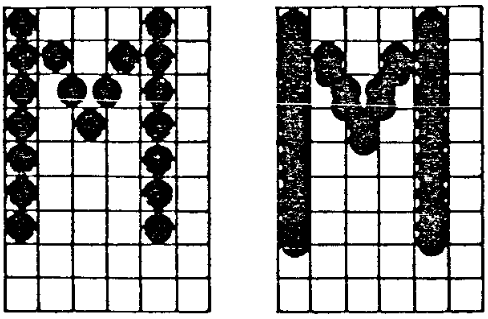


Fig. 3-6 Double-Strike Printing

The Data "Warehouse"

All the data you send to the RX from the computer spend some time in the printer's print buffer. The buffer acts as a sort of data warehouse. It holds the data until it either becomes full or until you send it one of the control codes that empties it. One such control code is carriage return (CR).

With the printer's left margin set at zero, the buffer holds up to 136 characters of pica (normal) width print and somewhat more for elite and condensed widths. (This capacity is increasingly reduced the farther the margin is set to the right.) When the buffer becomes completely full, the RX consecutively processes the line one character at a time.

Thus, a sort of assembly line process takes place, in which the raw text data are sequentially manufactured into characters according to the last control code processed. If a different manufacturing technique is indicated (such as to emphasize characters through double-striking), the printer is switched by a control code to the new task until instructed otherwise.

A Matter of Communication

If you're like many new or casual computer users, you've taken it just as a matter of faith that when you press a certain key, the character you wanted will appear on the screen. If you wish, you can continue to fall back on that faith with your new printer as well. Simplv input the correct commands, type away, and watch your printer obediently, respond.

What's going on inside isn't that difficult to understand, however.

The "ask-ee" codes

Years ago, manufacturers agreed on a system of coding, called the American Standard Code for Information Interchange (ASCII), to provide a common code base for data processing. This system is used today by most computers, printers, and software. Pressing a character key on your computer produces a bit pattern representing a particular ASCII (pronounced "ask-ee") code. The code is interpreted by the printer, which responds by printing the letter, number, or symbol desired. Or, in the case of a control code, it performs a requested function.

Most ot the 256 ASCII numbers are codes tor specific characters. Typically, codes’: 32 through 126 are reserved for the normal set of alphanumeric characters and special symbols. For instance, 65 represents capital “A”, while 90 represents capital “Z”. The italic character set is expressed by codes 160 through 254. Other numbers produce international characters, while still others concern computer and printer functions. Table 3-I summarizes the ranges of the ASCII codes used by the RX. (An itemized listing of the codes and what they do is provided in the appendix.)

ASCII code group	RX Interpretation
0 to 31, 127	Printer control codes
32 to 126	Standard (roman) character set
128 to 159	Additional printer control codes
160 to 154	Italic character set

Table 3-1 The RX and ASCII Codes

To sum up, some ASCII codes **produce standard characters** and special symbols, some **produce italic and international characters,** and others “drive” the printer.

How to Use the ASCII **Codes**

Cbviously, you can’t just type “65” to make your printer come up with an “A.” You’ve first got to tell your computer whether you want the letter to appear on your screen or be printed by the RX. This in done by inputting PRINT (for the screen) or LPRINT (for the printer). (Your computer may not use these commands. If so, check its reference manual and substitute the statements required by your computer.)

We’ll want our letter “A” to be printed out by the printer, so we use the LPRINT command.

Next, it’s necessary to unlock the door to the ASCII code table in your computer’s memory. This is done by using a special BASIC keyword. This keyword always takes the form of **CHR$(n),** where **n** represents the desired code.

* Codes can be expressed in binary, hexadecimal, or decimal form. For case of understanding, we use decimal.

For capital letter "A", the number 65 appears within parentheses, like so:

```
1Ø LPRINT (65)
```

After you type this, run your little program and you should see an "A" printed out.

```
A
```

The CHR$(n) Key

To put it simply, it helps to think of the CHR$ keyword as the "key" that unlocks the door to the ASCII table. You tell your computer which code you want by inserting the number within the parentheses following CHR$. Depending on the code, either a character will appear, or you will have instructed the printer to perform some activity, such as line feed or boldfacing.

There's a bit of a catch here, though.

Except for the standard character set, the codes used for functions and special characters (generally codes less than 32 and over 127) **may** vary from one make of computer to another. This causes some incompatibility problems between computers and printers. Fortunately, it is often possible to overcome those code inconsistencies.

You'll want to compare the codes used by your computer with those of the RX. This can be done by checking the code tables given in the manuals for both devices.

ESCaping the ASCII limitations

Even with up to 256 ASCII codes available, printers eventually had so many features that all available codes were exhausted. The solution was to string codes together, thus permitting major expansion of the control code vocabulary.

The format used for this by the RX printer is the ESCape code CHR$(27) followed by one or more of the RX's other codes. This second code can take the form of **CHR$(n)** or **"n".** (One way to think of the ESC code is that it allows an "escape" from the limitations of the too-small ASCII code table.)

Let's see how the ESC code works on two often-used KX functions —automatic underline and setting the left margin.

Example 1. Automatic underline

```
10 LF'RINT CHR$(27);"-1";
20 LPRINT "Underline"
```

Input these two lines, RUN the program, and you'll get this:

```
Underline
_________
```

As you see, the underlining occurs automatically. This mode prints by firing the ninth pin as the line of text (including the spaces) is printed. This feature is good to emphasize text and to make job application and other fill-in-the-blank forms. You can prove this to yourself by printing a couple of lines. Simply type

```
10 LF'RINT CHR$(27);"-1";
20 LPRINT "Underline"
30 LF'RINT "             "
40 LPRINT "                        "
```

and RUN your program. The result will be:

```
Underline
_________
_____________
______________________
```

Example 2. Setting the left margin

```
10 LPRINT CHR$(27);"1";CHR$(20)
20 LF'RINT "New  margin"
```

After substituting this statement line in your program as shown, go ahead and RUN it. The left margin is now set 20 character spaces to the right.

```
                    New margin
                    __________
                    _____________
                    ______________________
```

You may be surprised to see that the automatic underline is still in effect, even though the program itself no longer contains the ESC code that originally caused the underlining. This is because modes will stay in effect until they xc turned off.

To turn off the automatic underline, first delete line 10 of the Program, type

```
LPRINT CHR$(27);"-0";
```

and, instead of inputting RUN, merely press the carriage return key to send the new instruction to the printer.

Now if you RUN your program, the underline (and your form blanks) disappear, like so:

```
                    New margin
```

Change Mode Commands

What was just demonstrated is one of the three ways you can turn off a function. We gave the printer a specific change command directed at only one mode-the underline mode. You'll use this technique when you wish to reset only one or a few modes. Each mode requires its own turn-off instruction.

If you want to start from scratch-the printer's default settings-your RX comes with a convenient initialize control code: CHR$(27)"@". Thus, when you type

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

and then RUN the program, this is what happens:

```
New margin
```

A third, more drastic way to achieve the same thing is to simply turn off the printer. This also reverts the RX to its default settings but can interfere with computer-printer communication. So it's best to use this technique sparingly, if at all.

Mixing Pitch Modes

By now it should be clear that it's possible to use more than one CHR$ code setting at a time. For instance, you may wish to underline a line of italic text. Let's create a program to that effect by inputting this:

```
NEW
10 LPRINT CHR$(27) ; "4"; "These are italics ";
20 LPHINT CHR$(27);"-1";"now underlined."
30 LPRINT CHR$ (27) ; "@";
```

Okay, now RUN the program to get this:

```
These are italics now underlined.
```

In statement 30, we provided an initialize code to switch the printer back to its original settings. Now we're ready to try something even fancier. Let's see what happens when we use Superscript mode on the same line with Italic and Underline modes. Input line 20 again but with one change.

```
1Ø LPRINT CHR$ (27) ; "4"; "These are italics ";
20 LPRINT CHR$(27);"-1";"now";CHR$(27);"SØ";"
 underlined. "
30 LPRINT CHR$ (27) ; "@";
```

This is what your printer should produce:

```
These are italics now underlined.
```

Mode pecking order

Some modes were created more equal than others, with the result that if two conflicting modes are used together, the one with priority will be in effect while the other will be temporarily ignored. If the mode with priority is turned off, the other mode will then take effect.

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

Table 3-2 Mode Summary

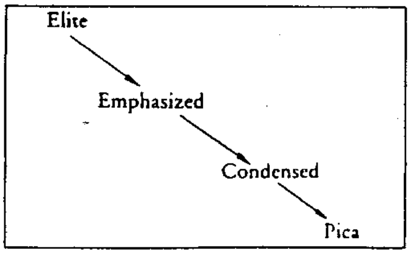


Table 3-3 Mode Priority

CONTROLLING YOUR PRINTER

As related in previous chapters, there are two types of codes-character and control. Control codes simply tell the printer what to do with the characters that follow-whether they should be elite, condensed, double-spaced, and so on.

Control codes determine what the pages coming out of your printer are going to look like.

Mixing and Matching

What makes the control codes interesting is the fact that they can be used in combination. You can print, for example, double-struck emphasized italic characters. Now not only is that a mouthful, but if this is your first encounter with a dot matrix printer, you probably can't quite imagine what such characters might look like.

The purpose of this chapter is to get you to know your control codes -what they do, what their special characteristics are, how they can be combined-so you can have all of the capabilities of the RX at your fingertips.

AU hands on

The best way to learn is to do. The different control codes will only become meaningful to you when you have used them. This chapter is designed to give you a chance to do that by providing several programs that let you "walk through" the codes.

These are simple little programs that let you try your hand at the different combinations of control codes that the printer has to offer. We also try to explain the different problems as they arise so that by the time you've finished with this chapter, you should be good friends with all the control codes.

Where to Find It

The second half of this chapter is a quick reference listing of the control codes grouped by function with a thumb index so you can get to the information you need fast. These give you the information you will need for actual use of each code. It is expected that after you've mastered the use of the printer, this manual will then spend the rest of its life next to the printer open to this section.

Spaces

The first group of control codes we look at is those that control the line spacing. They are:
ESC 0, ESC 1, **ESC 2, ESC 3, ESC A**

These simple, straightforward codes set the desired line spacing for your printer. The choices you have are 1/8 inch, 7/72 inch, 1/6 inch, n/216 inch, and n/72 inch.

The last two choices are for precise control of line spacing where **n can** be freely specified by the user. 1/72 inch corresponds to the vertical spacing between dots, and 1/216 inch to 1/3 the vertical distance between dots.

Line spacing of 7/72 inch is equivalent to seven dot spaces or one standard character. This means that with ESC 1 there is no spacing between lines.

Trying it yourself

This is the first of our walkthrough programs. Input this program exactly as shown and then RUN it. It will first ask you to input a control code. Here if you input "3" this value will be assigned to variable "CODES" in line 10. The program will then ask for a value for **n.** Input how many 216ths of an inch spacing you want. Let's say you want 20/216 inch line spacing. Input "20" and hit the RETURN key. (If the code you've chosen doesn't require a value for n, don't worry-just press the RETURN key and everything will be all right.)

ESCape to Space Program

```
10  INPUT "INPUT CONTROL CODE";CODE$
20  INPUT "VALUE FOR n PLEASE";N
30  PRINT "ANY COMMENTS ABOUT ESCAPE CODE ";CODE$;
40  INPUT COMMENT*
50  LPRINT CHR$(27);CODE$;CHR$(N);
60  LPRINT "THIS IS ESCAPE CODE ";CODE$;" WHICH ";
 COMMENT%
70  FOR X=1 TO 5
80  LPRINT "LINE SPACING --------"
90  NEXT X
100 LPRINT
110 END
```

HOW it's done

The values you input in lines **10** and 20 are used in line 40 so we have the equivalent of

LPRINT CHR$(27);"3";CHR$(20)

which is the standard escape sequence and a pretty familiar sight to us by now. In all of the walkthrough programs you will find a line like line 40. This is where we substitute the values input by the user (or generated by the program itself) to vary the operation of the printer.

Your opinion

Most of the programs will also ask for your comments. This is output on the printer for a permanent record. Input something that will help you remember later what the code in question is about. If you save all your printouts from this chapter, you'll end up with your own customized reference library for control codes.

RUNning Escape to Space

```
THIS IS ESCAPE CODE 1 WHICH SETS 7/72 INCH SPACING
LINE SPACING -------
LINE SPACING -------
LINE SPACING -------
LINE SPACING -------
LINE SPACING -------
```

```
THIS IS ESCAPE CODE 0 WHICH SETS 1/8 INCH SPACING
LINE SPACING -------
LINE SPCICING -------
LINE SPACING -------
LINE SPACING -------
LINE SPACING -------
```

```
THIS IS ESCAPE CODE A WHICH SETS 24/72 INCH SPACING

LINE SPCICING -------

LINE SPCICING -------

LINE SPCICING --w-s--

LINE SPACING --a----

LINE SPCICING -------
```

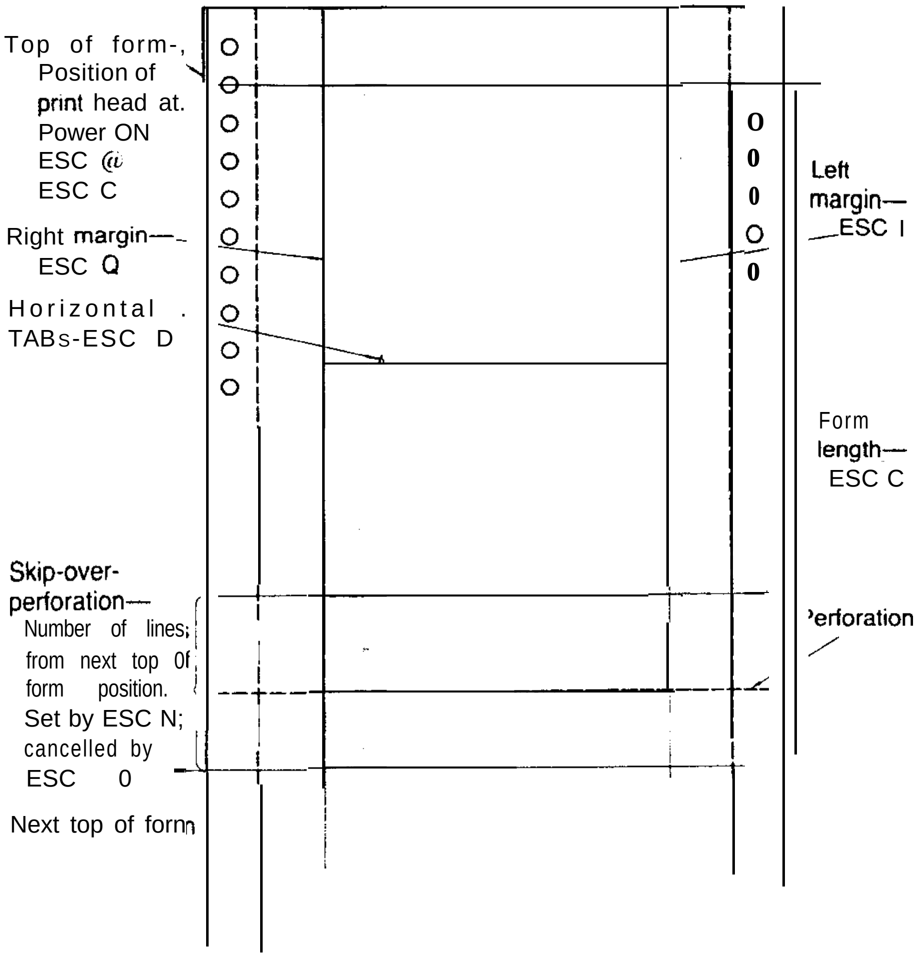
Making Page

This next group of control'codes determine the "look" of your page --its length and width, horizontal and vertical TAB settings, and so on. The codes in this group are:

ESC Q, ESC I, ESC D, HT, ESC B, VT, ESC b, ESC /, ESC C, ESC N, ESC 0

This is a list giving the role of each code.

ESC Q	Sets the right margin.
ESC 1	Sets the left margin.
ESC D	Sets horizontal TAB positions.
HT	Moves the print head to the next horizontal TAB position.
ESC B	Sets vertical TAB positions.
VT	Advances the paper to the next vertical TAB position.
ESC b	Presets vertical formats.
ESC /	Selects among vertical formats preset by ESC b.
ESC C	Sets the form length.
ESC N	Sets the number of lines to be left blank at the bottom of the page.
ESC O	Cancels the setting performed by ESC N.


Top of form-,
Position of
print head at.
Power ON
ESC @
ESC C
Right margin—
ESC Q
Horizontal .
TABs-ESC D
Skip-over-
perforation—
Number of lines;
from next top Of
form position.
Set by ESC N;
cancelled by
ESC 0
Next top of form
Left
margin—
ESC l
Form
length—
ESC C
’erforation

Print Modes

The control codes we discuss next are those that control the print mode. They are:

ESC P, ESC M, SO, ESC SO, ESC W, DC4, SI, DC2, ESC E, ESC F, ESC G, ESC H, ESC S, ESC T, ESC —, ESC 4, ESC 5, ESC R, ESC m

Great ESCape Program

The program shown below is another walkthrough program that lets you get at all of the ESC sequences for print modes. Here you can either input this program from the start or else modify the Escape to Space program so that it looks like this:

```
10 INPUT " INPUT CONTROL CODE"; CODES
20 PRINT "ANY COMMENTS ABOUT ESCAPE CODE " ; CODE$;
30 INPUT COHMENTS
40 LPRINT CHR$(27);CODE$;
50 LPRINT "THIS IS ESCAPE CODE ";CODE$;" WHICH ";
 COHMENTS
60 FOR X=32 TO 127 :' LOOP TO
70 LPRINT CHR$(X); : ' PRINT ASCII
80 NEXT X              : ' CODES 32-127
90 LPRINT
100 GOT0 10            : ' AND BACK
```

This program asks for the code you want to print and your comment. It then uses a loop (lines 60 to SO) to print ASCII codes 32 to 127 in the selected print mode.

Having done that, the program then goes back to **line 10** and starts over. You get out of this endless loop by pressing the BREAK key. This is not "elegant" programming but was opted for to keep things as simple as possible. Anyone who knows a solution to this problem should feel free to modify the program right now.

Story of n

This is probably a good place to mention something about the values that are input following the control codes. Most of the codes that control print modes do not require any data other than the code itself. Some, however, like ESC W (Enlarged mode) and ESC R (character set selection), must be followed by a value **(n)** that specifies the operation of the code. The usual format for this is:

CHR$(27);"code";CHR$(n)

Some of the codes requiring a value for **n** need only a "1" or "0" to turn the print mode on or off or to select between two options (such as Superscript and Subscript for ESC S). These codes can also take this format :

CHR$(27);"coden"; (n=0 or 1)

The codes that can do this are:

ESC W, ESC S, ESC **U, ESC —, ESC s**

Initialize

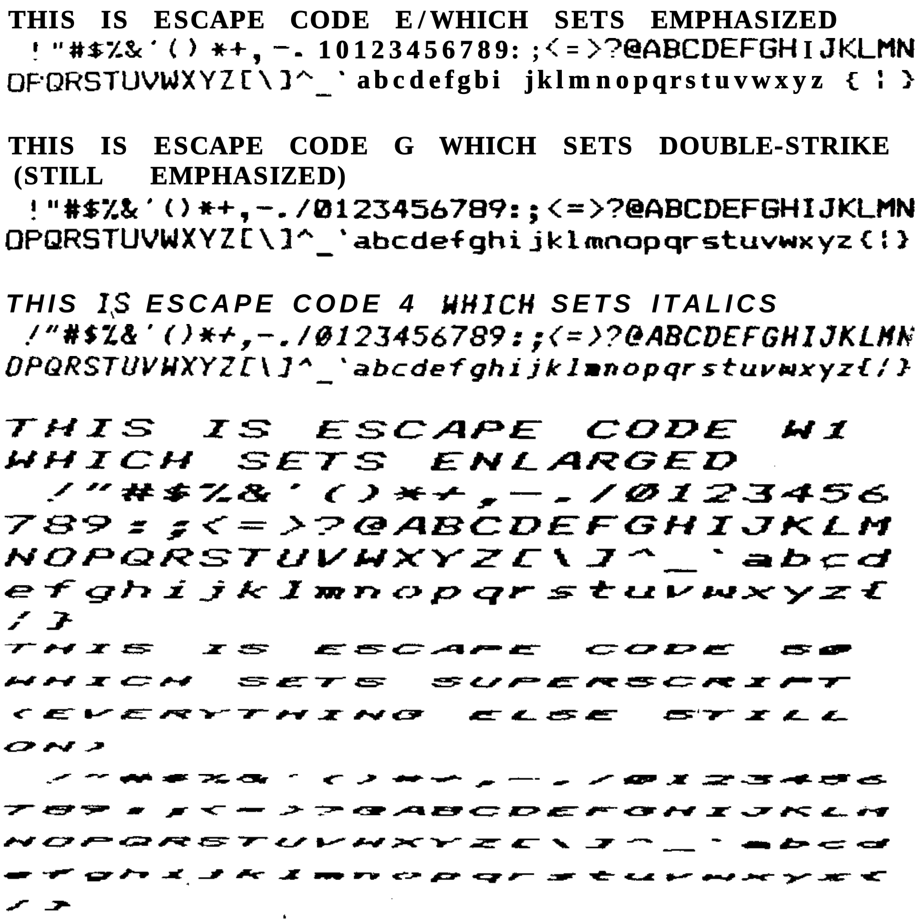
The first code you shodd try is the initialize control code "@." This will cancel any codes in effect so you can start with a "clean slate."

```
THIS IS ESCAPE CODE @ WHICH RESETS EVERYTHING
 !"#$%&'()*+,-./0123456789:;<=>?ABCDEFGHIJKLMN
```

The more the merrier

Now let's trying inputting these codes one after another: ESC E, **ESC G,** ESC **4,** ESC W1, ESC SO. These codes set emphasized, double-strike, italic, enlarged, and superscript printing.

To get the most out of the experience, be sure to get in there and look at what the print head is doing **as** it prints out your characters to order.



```
THIS IS ESCAPE CODE E/WHICH SETS EMPHASIZED
 !"#$%&'()*+,-. 10123456789:;<=>?@ABCDEFGHIJKLMN
OFQRSTUVWXYZ[\]^_`abcdefgbi jklmnopqrstuvwxyz {¦}

THIS IS ESCAPE CODE G WHICH SETS DOUBLE-STRIKE
 (STILL EMPHASIZED)
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN
OPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{¦}

THIS IS ESCAPE CODE 4 WHICH SETS ITALICS
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMN
OPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{¦}

THIS IS ESCAPE CODE W1
WHICH SETS ENLARGED
 !"#$%&'()*+,-./0123456
789:;<=>?@ABCDEFGHIJKLM
NOPQRSTUVWXYZ[\]^_`abcd
efghijklmnopqrstuvwxyz{
¦}
THIS IS ESCAPE CODE S0
WHICH SETS SUPERSCRIPT
(EVERYTHING ELSE STILL
ON)
 !"#$%&'()*+,-./0123456
789:;<=>?@ABCDEFGHIJKLM
NOPQRSTUVWXYZ[\]^_`abcd
efghijklmnopqrstuvwxyz{
¦}
```

Use this program to try out different combinations of print modes. If things get too complicated, you can either input a code to cancel a particular mode or you can input "@" and start all over again.

And when you've done all you want to, press the BREAK key.

Another Way of Doing It

Like all things in life, there's more than one way to get the print modes. Your RX printer is equipped with a powerful escape code called ESC !. This code writes the value sent after it (n) directly to the control word of the printer. The control word is shown below:

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
"1"	Underline	Italics	Enlarged	Double strike	Emphasized	Condensed	Always "0"	Elite
"0"	—	—	—	—	—	—	—	Pica

Each bit of this word controls a different aspect of the printed character. For example, when bit 7 is "I ", underline mode is selected and bit 0 selects between elite and pica characters. Let's analyze a value for **n.**

What you see

If you send this to the printer:

```
10 LPRINT CHR$(27);"!";CHR$(92);"THIS IS ! 92";
```

You will get this:

THIS IS ! 92

Which is italic double-struck emphasized condensed pica.
This is because **92,** when converted to binary a value becomes: **01011100**

	(0	1	0	1	1	1	0	0)
"1"	Underline	Italics	Enlarged	Double strike	Emphasized	Condensed	Always "0"	Elite
"0"	—	—		—	—	—	—	Pica

So we see that each bit of the control word is doing its job.

Everything

The next program we introduce is called “Everything” and uses the ESC ! code to print every kind of character your printer knows.

```
10 FOR I=Ø TO 255
20 X=I\2:IF X MOD 2=1 THEN 60
30 LPRINT CHR$(27);"!";CHR$(I);"EVERYTHING ";
40 GOSUB 100
50 LPRINT "(";R$;")"; I
60 NEXT I
100 R$=""
110 N=I
120 X=N MOD 2
130 IF X=1 THEN R$="1"+R$ ELSE R$="Ø"+R$
140 N=N\2
150 IF N>=1 THEN 120
160 R$=RIGHT$("ØØØØØØØØ"+R$,8)
170 RETURN
```

This program loops to generate all the values for **n** between 0 and 255. Line 20 uses the modulus (remainder of integer division) to skip over those codes that would be duplicated because bit 1 of the control word is always “0”

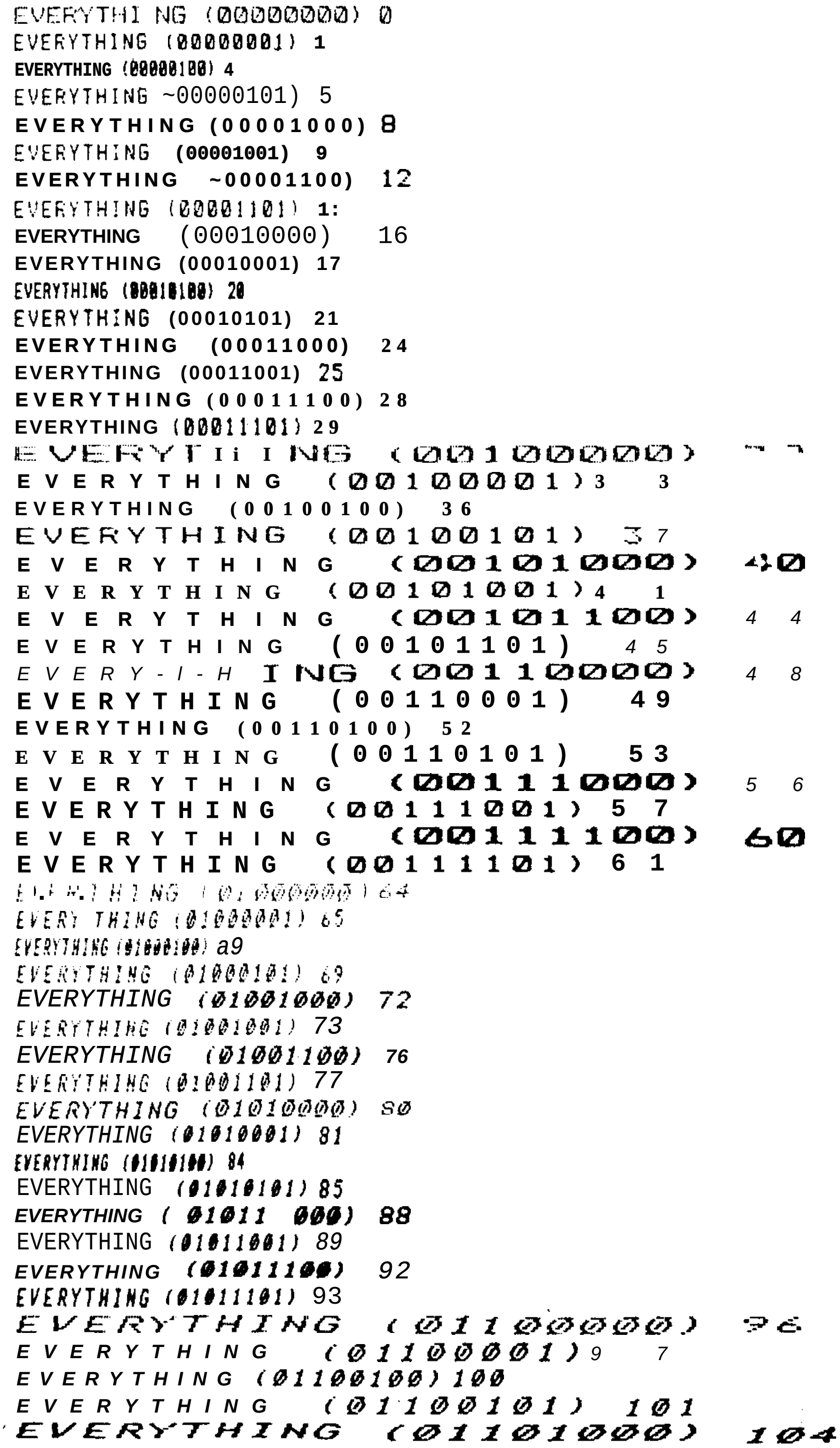
The subroutine from line 100 returns the binary equivalent of decimal numbers. This gives binary values for **n** so that we can easily compare the value of each bit against the bits of the control word.

RUNning everything

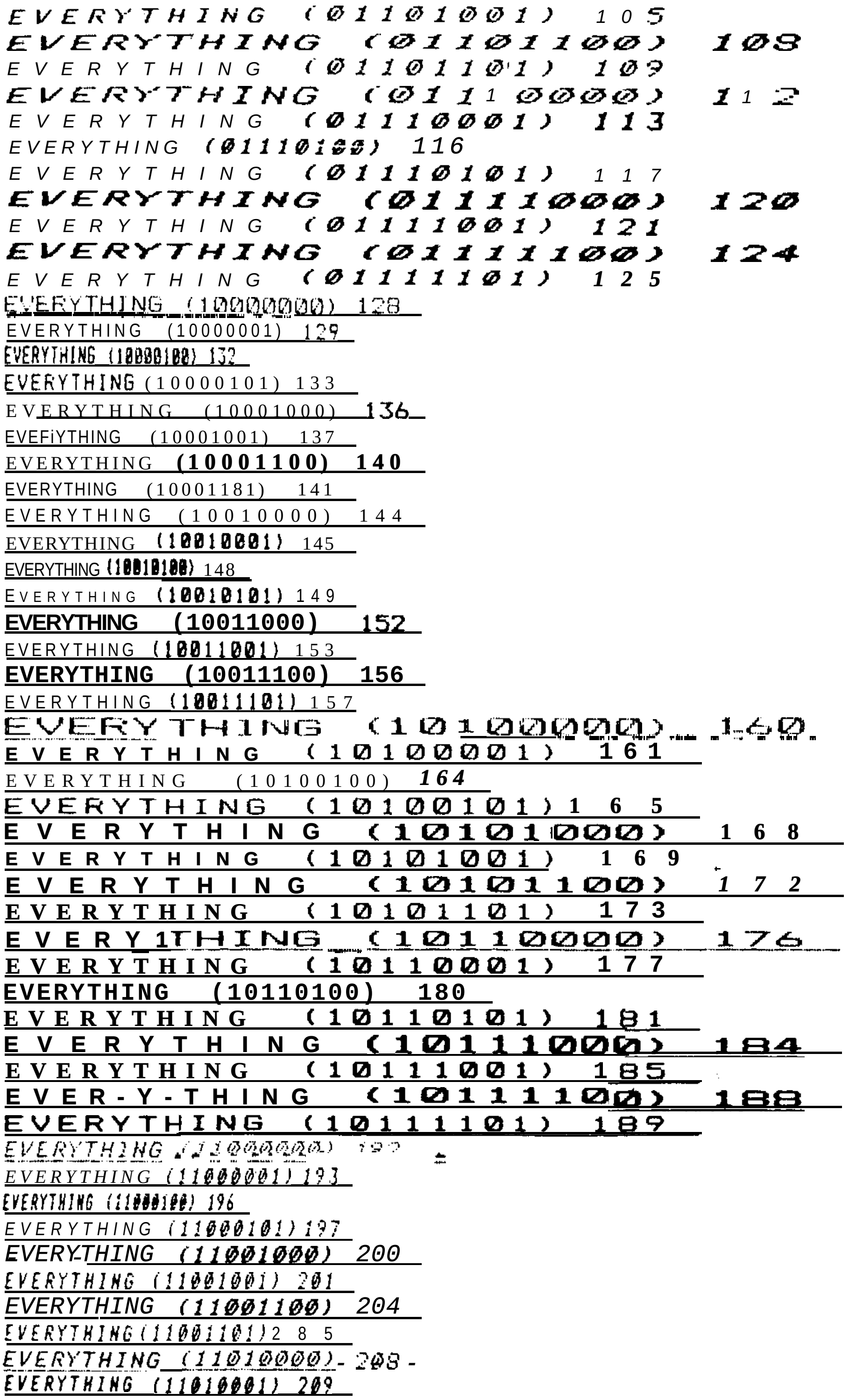
When you RUN this program, you will be treated to the sight of your printer giving you everything it has.

When Everything has finished, you should have a complete list of all the ways your printer knows of printing characters.

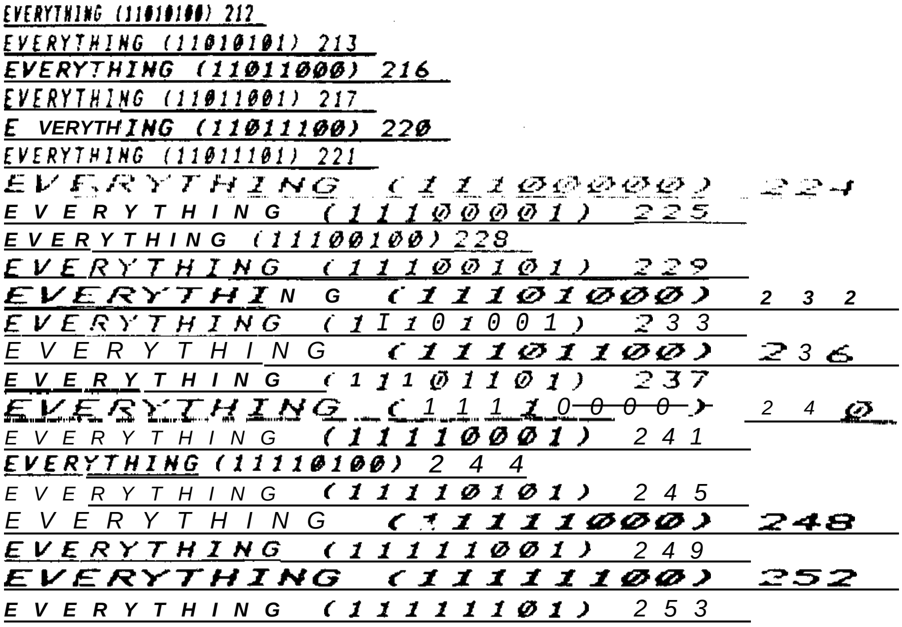
Frame this list (it’s that beautiful) and hang it on the wall next to your computer or word processor. When you want an unusual combination of print modes, check this list. When you find one that strikes your fancy, just send ESC ! to the printer followed by the decimal value at the right of the list. (The lists in this manual, for example, were all printed using **ESC ! 24.)**



```
EVERYTHING (ØØØØØØØØ) Ø
EVERYTHING (ØØØØØØØ1) 1
EVERYTHING (ØØØØØ1ØØ) 4
EVERYTHING (00000101) 5
EVERYTHING (00001000) 8
EVERYTHING (00001001) 9
EVERYTHING (00001100) 12
EVERYTHING (ØØØØ11Ø1) 13
EVERYTHING (00010000) 16
EVERYTHING (00010001) 17
EVERYTHING (ØØØ1Ø1ØØ) 2Ø
EVERYTHING (00010101) 21
EVERYTHING (00011000) 24
EVERYTHING (00011001) 25
EVERYTHING (00011100) 28
EVERYTHING (ØØØ111Ø1) 29
EVERYTHING (ØØ1ØØØØØ) 32
EVERYTHING (ØØ1ØØØØ1) 33
EVERYTHING (00100100) 36
EVERYTHING (ØØ1ØØ1Ø1) 37
EVERYTHING (ØØ1Ø1ØØØ) 4Ø
EVERYTHING (ØØ1Ø1ØØ1) 41
EVERYTHING (ØØ1Ø11ØØ) 44
EVERYTHING (00101101) 45
EVERYTHING (ØØ11ØØØØ) 48
EVERYTHING (00110001) 49
EVERYTHING (00110100) 52
EVERYTHING (00110101) 53
EVERYTHING (ØØ111ØØØ) 56
EVERYTHING (ØØ111ØØ1) 57
EVERYTHING (ØØ1111ØØ) 6Ø
EVERYTHING (ØØ1111Ø1) 61
EVERYTHING (Ø1ØØØØØØ) 64
EVERYTHING (Ø1ØØØØØ1) 65
EVERYTHING (Ø1ØØØ1ØØ) 68
EVERYTHING (Ø1ØØØ1Ø1) 69
EVERYTHING (Ø1ØØ1ØØØ) 72
EVERYTHING (Ø1ØØ1ØØ1) 73
EVERYTHING (Ø1ØØ11ØØ) 76
EVERYTHING (Ø1ØØ11Ø1) 77
EVERYTHING (Ø1Ø1ØØØØ) 8Ø
EVERYTHING (Ø1Ø1ØØØ1) 81
EVERYTHING (Ø1Ø1Ø1ØØ) 84
EVERYTHING (Ø1Ø1Ø1Ø1) 85
EVERYTHING (Ø1Ø11ØØØ) 88
EVERYTHING (Ø1Ø11ØØ1) 89
EVERYTHING (Ø1Ø111ØØ) 92
EVERYTHING (Ø1Ø111Ø1) 93
EVERYTHING (Ø11ØØØØØ) 96
EVERYTHING (Ø11ØØØØ1) 97
EVERYTHING (Ø11ØØ1ØØ) 1ØØ
EVERYTHING (Ø11ØØ1Ø1) 1Ø1
EVERYTHING (Ø11Ø1ØØØ) 1Ø4
```



```
EVERYTHING (Ø11Ø1ØØ1) 105
EVERYTHING (Ø11Ø11ØØ) 1Ø8
EVERYTHING (Ø11Ø11Ø1) 1Ø9
EVERYTHING (Ø111ØØØØ) 112
EVERYTHING (Ø111ØØØ1) 113
EVERYTHING (Ø111Ø1ØØ) 116
EVERYTHING (Ø111Ø1Ø1) 117
EVERYTHING (Ø1111ØØØ) 12Ø
EVERYTHING (Ø1111ØØ1) 121
EVERYTHING (Ø11111ØØ) 124
EVERYTHING (Ø11111Ø1) 125
EVERYTHING (1ØØØØØØØ) 128
EVERYTHING (10000001) 129
EVERYTHING (1ØØØØ1ØØ) 132
EVERYTHING (10000101) 133
EVERYTHING (10001000) 136
EVEFiYTHING (10001001) 137
EVERYTHING (10001100) 140
EVERYTHING (10001181) 141
EVERYTHING (10010000) 144
EVERYTHING (1ØØ1ØØØ1) 145
EVERYTHING (1ØØ1Ø1ØØ) 148
EVERYTHING (1ØØ1Ø1Ø1) 149
EVERYTHING (10011000) 152
EVERYTHING (1ØØ11ØØ1) 153
EVERYTHING (10011100) 156
EVERYTHING (1ØØ111Ø1) 157
EVERYTHING (1Ø1ØØØØØ) 16Ø
EVERYTHING (1Ø1ØØØØ1) 161
EVERYTHING (10100100) 164
EVERYTHING (1Ø1ØØ1Ø1) 165
EVERYTHING (1Ø1Ø1ØØØ) 168
EVERYTHING (1Ø1Ø1ØØ1) 169
EVERYTHING (1Ø1Ø11ØØ) 172
EVERYTHING (1Ø1Ø11Ø1) 173
EVERY1THING (1Ø11ØØØØ) 176
EVERYTHING (1Ø11ØØØ1) 177
EVERYTHING (10110100) 180
EVERYTHING (1Ø11Ø1Ø1) 181
EVERYTHING (1Ø111ØØØ) 184
EVERYTHING (1Ø111ØØ1) 185
EVER-Y-THING (1Ø1111ØØ) 188
EVERYTHING (1Ø1111Ø1) 189
EVERYTHING (11ØØØØØØ) 192
EVERYTHING (11ØØØØØ1) 193
EVERYTHING (11ØØØ1ØØ) 196
EVERYTHING (11ØØØ1Ø1) 197
EVERYTHING (11ØØ1ØØØ) 200
EVERYTHING (11ØØ1ØØ1) 2Ø1
EVERYTHING (11ØØ11ØØ) 204
EVERYTHING (11ØØ11Ø1) 2 8 5
EVERYTHING (11Ø1ØØØØ)- 2Ø8 -
EVERYTHING (11Ø1ØØØ1) 2Ø9
```



```
EVERYTHING (110101ØØ) 212
EVERYTHING (11Ø1Ø1Ø1) 213
EVERYTHING (11Ø11ØØØ) 216
EVERYTHING (11Ø11ØØ1) 217
EVERYTHING (11Ø111ØØ) 22Ø
EVERYTHING (11Ø111Ø1) 221
EVERYTHING (111ØØØØØ) 224
EVERYTHING (111ØØØØ1) 225
EVERYTHING (111ØØ1ØØ) 228
EVERYTHING (111ØØ1Ø1) 229
EVERYTHING (111Ø1ØØØ) 232
EVERYTHING (111Ø1ØØ1) 233
EVERYTHING (111Ø11ØØ) 236
EVERYTHING (111Ø11Ø1) 237
EVERYTHING (1111ØØØØ) 24Ø
EVERYTHING (1111ØØØ1) 241
EVERYTHING (1111Ø1ØØ) 244
EVERYTHING (1111Ø1Ø1) 245
EVERYTHING (11111ØØØ) 248
EVERYTHING (11111ØØ1) 249
EVERYTHING (111111ØØ) 252
EVERYTHING (111111Ø1) 253
```

Bit-Image **Graphics**

Your RX printer, in addition to being able to print the matrix patterns stored in its ROM memory, can also print dot patterns designed by you, the user.

This function is called dot or bit-image graphics.

"Dot graphics" refers to the fact that in this mode you choose which dots of the dot matrix are to be printed (which pins of the print head are to fire). "Bit-image" refers to the fact that a binary (bit by bit) representation is used to indicate, or address, the pins of the print head.

Bits and dots

Before giving any actual examples, let's first consider what would be the most logical way of telling the printer which pins it should fire.

At any given dot position, each pin of the print head must be in one of two states-it is either firing or it is not. In this sense, the pins can be thought of as binary digits that are either "1" (firing) or "0" (not firing). It makes sense then that an 8-bit binary number is enough to address each pin of the print head. Binary data "11111111" (255) would thus tell the printer to fire the pins and "00000000" (0) to fire none of them.

To show this in practical terms, let's bring 'back the uppercase T used in Chapter 3 to demonstrate the firing order of the pins, The data for each dot position of the uppercase T is shown below:

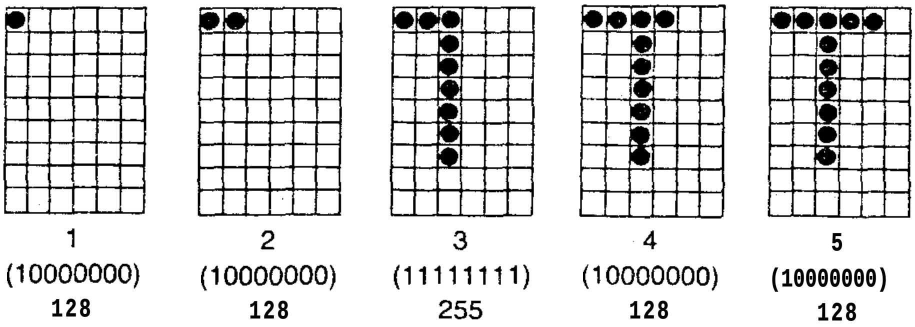


Fig. 4-l Print Data for Uppercase T

In other words, the data to print uppercase T is sent to the print head in the order of: 128, 128, 255, 128, 128.

How to do it

The preceding discussion gave an idea of how to calculate the data to be sent to the printer as bit-image data. Before you can send this kind of data, however, you must first tell the printer that you're entering Bit-image mode and that it should treat the data that follows as bit-image data. It would also be nice to tell the printer how much data to expect.

This is what we do with bit-image codes ESC **K, ESC L, ESC** Y, **ESC** Z, and **ESC * .**

Bit-image printing is performed one line at time. Since the density (dots per inch) of the graphics printed by each control code is different, the total number of data that can be sent for each code is also different.

The first type of bit-image printing is Normal-density mode. In this mode, the dot patterns are printed in a density abour equal to that for normal pica-sized characters. The maximum number of dot positions that can be printed in a single line is **S16.** The input format for the ESC K instruction that controls normal-density bit-image printing is:

CHR$(27);"K";CHR$(n1);CHR$(n2);CHR$(d_1);CHR$($d_2$);... CHR$($d_{n1,n2}$);

To find the total number of bit-image data, **n2** is multiplied by 256 and added to the value of **nl.**

For instance, let's try printing uppercase T in bit-image mode.

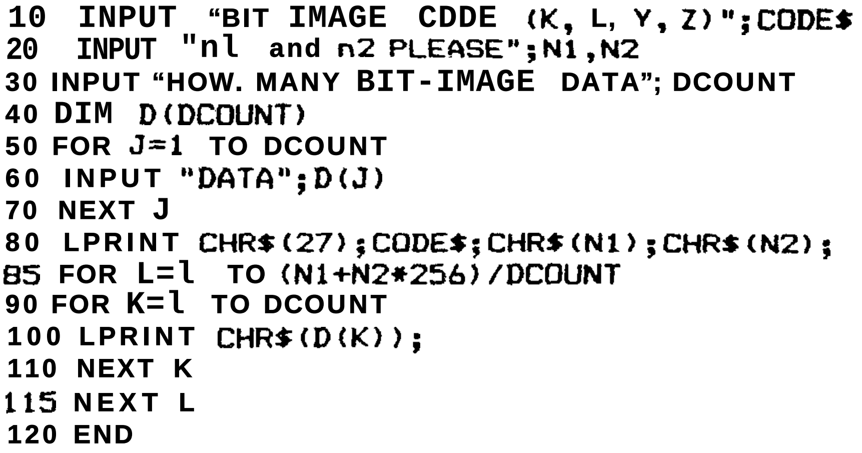
First, if we calculate the number of data we see that it is 5 with one blank for a total of 6. This translates into values of 6 and 0 (6+0×256) for **nl** and **n2.** The bit-image data used here is that shown in Fig. 4-1.

```
10 LPRXNT CHR$(27);"K";CHR$(6);CHR$(Ø);
20 LPRINT CHR$(128);CHR$(128);CHR$(255);
30 LPRINT CHR$(128);CHR$(128);CHR$(Ø);
```

Yes, it is much easier to send an ASCII code for the letter T to the printer and let it do the hard part; that's why we have the character codes. There is, however, a whole new world of computer graphics that the Bit-image mode opens up for you.

The Graphics Delight Program

In the same way that the Great Escape program let you print out data in each of the different print modes, this program lets you do the same thing for graphics.



```
10 INPUT "BIT IMAGE CDDE (K, L, Y, Z)";CODE$
20 INPUT "nl and n2 PLEASE";N1,N2
30 INPUT "HOW. MANY BIT-IMAGE DATA"; DCOUNT
40 DIM D(DCOUNT)
50 FOR J=1 TO DCOUNT
60 INPUT "DATA";D(J)
70 NEXT J
80 LPRINT CHR$(27);CODE$;CHR$(N1);CHR$(N2);
85 FOR L=l TO (N1+N2*256)/DCOUNT
90 FOR K=l TO DCOUNT
100 LPRINT CHR$(D(K));
110 NEXT K
115 NEXT L
120 END
```

Input this program exactly as it appears above. When you RUN the program, it will first ask you to input the control code that you want to execute for bit-image printing.

Your choices here' are: ESC K, ESC L, ESC Y, and ESC Z which select Normal, Double-density, Double-density double-speed, and Quadruple-density bit-image print modes.

As the names of each of these modes suggest, the maximum 816 dot positions per inch of the Normal bit-image mode doubles to 1,632 for Double-density, and doubles again to 3,264 positions per line in Quadruple mode.

The higher the dot density used, the higher the resolution (quality) of your graphics. The price you pay is speed-in double- and quadruple-density graphic printing, the print speed drops to about half that for normal density.

As a compromise solution, there is also a Double-density double-speed mode that gives you extra speed but cannot print horizontally adjacent dots.

We will execute our first RUN through the program in Normal-density mode, so input "K."

We are not alone

Let's say you want to bring not just one, but a whole attack squadron of space invaders down upon yourself. We show you how to do this.

Here we'll assume that the avcrage width of a space invader is 12 dot positions (plus one so their wings don't brush). This means that 13 bit-image data will be enough to describe a single invader.

If we feel that a squadron of 30 should **do** the trick, we get a total of 390 (30*1 I) bit-image data.

To find **nl** and **n2,** we then divide 390 by 256 and get a quotient of 1 with remainder of 134. These become the values for **n2** and **nl,** respectively.

Input these values with a comma between them. The program will then ask you how many data per space invader. Input "13" and press the RETURN key.

The program will now ask in order for the data for a single invader. Input:

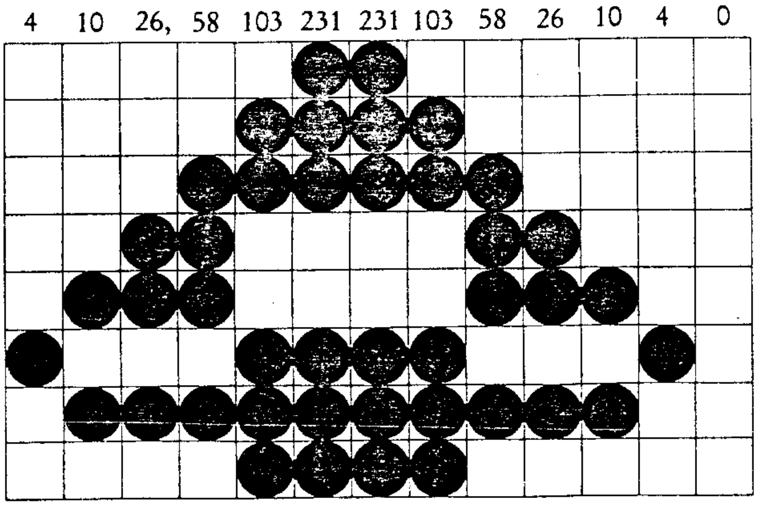


Fig. 4-2 Bit-Image Space invader

When you input the last data, your computer will think a while (quite a while when the bit-image to be printed is genuinely complicated) then, and then, AAARGH !!!

Let's try that once more, this time in the Double-density mode. This time when the program asks for a control code, give it an "L." Input the rest of the data exactly as in Normal density mode.

When the program prints, you'll find that the attack squadron is bearing a hasty retreat across the universe.

The main thing to watch when executing this program is to get the right values for nl and n2. If they're not correct, you end up sending the printer more data than it was told to expect for bit-image data. When it gets these data, it has no choice but to print them as character codes.

With that one word of caution, we leave you to get some graph paper, figure out where you want your dots to go and have yourself a good time.

CR CARRIAGE RETURN

Format CHRS(13);

Function This code starts the printing.

When all the characters in the print buffer have been printed, the print head returns to the left margin.

You can set your printer with a DIP switch setting so when a carriage return is sent, it also automatically performs line feed. See Chapter 3 for details.

If the printer has no characters to print (or if they are all Space code), the print head will not move when you send it a CR code. If the printer has been set to perform line feed, however, it will faithfully do so even if it has nothing to print.

LF

LINE FEED

Format **CHR$(10);**

Function This code advances the paper set in the printer the equivalent of one line. You can set the amount of the line feed (the distance the paper advances) using the control codes detailed elsewhere in this chapter.

Format **CHR$(12);**

Function This code advances the paper to the next top of form position. The top of form position is the vertical position of the print head when you rum the power to the printer on, send an ESC @ to initialize it, or set the page length with ESC C. The printer remembers this position and advances the paper to that spot on the next page when it receives this code.

ESC @ INITIALIZES THE PRINTER

Format **CHR$(27);"@";**

Function This code initializes the printer. In other words, it returns the printer to the same condition as when power has just been turned on. Any print modes or margins that have been set will be cleared and only those conditions indicated by the settings of the DIP switches will be in effect.

ESC M E m PITCH

Format **CHR$(27);“M”;**

Function This code causes characters to be printed in elite pitch. Elite type fits 12 characters to an inch.

Because this control code takes precedence, if a code that selects emphasized characters is also input, the new code will be seemingly ignored. What has actually happened is that the printer has set an internal flag to take note of the fact that Emphasized mode has been requested; you will get your Emphasized characters as soon as you return to printing pica characters (ESC P).

Example

Elite characters
Pica characters
Condensed characters

ESC P

PICA PITCH

Format CHR$(27);"P";

Function This code cancels the elite print set by ESC M.

When this code is input, the printer returns to pica print. This code cancels only elite print and leaves all other modes, such as Enlarged or Condensed, in effect.

Example

```
Pica characters
Elite characters
Condensed characters
```

SO

ENLARGED PRINT (ONE LINE)

Format CHR$(14)

Function This code prints enlarged type for one line. That is, the characters starting from CHRS(14) to the end of the line (line feed or carriage return) are printed as enlarged characrcrs.

ESC SO is another way of writing this code; the function is identical either way. A line feed, DC4, or ESC W0 can all be used to end Enlarged mode set by this code. This code is useful for printing section titles and other one-line captions.

Example

```
Pica        Pica enlarged
Elite       Elite enlarged
Condensed   Condensed enlarged
```

DC4 CANCELS ENLARGED PRINT

Format **CHR$(20);**

Function This code cancels Enlarged print mode set by the SO (or ESC SO) codc but not that set by the ESC W1 code.

As shown, thcre are a number of codes that can be used to set and cancel the Enlarged print mode. How these codes relate is summarized below.

Sets	Cancelled by
so	LF, DC4, ESC WO
ESC SO	LF, DC4, ESC WO
ESC W1	ESC wo

We should note here that the LF code also contains a CR code. This is why codes like ESC SO and SO that are cancelled by CR are also cancelled by LF.

ESC W

Format **CHR$(27);"W";CHR$(n);**

n=1 or 49 Sets Enlarged print mod;

n=0 or 48 Cancels Enlarged print mode

Function This code sets (n=1 or 49) and cancels (n=0 or 48) the Enlarged print mode. The difference between this code and the SO code is that because it is not cancelled by line feed, ESC W allows you to print Enlarged print over several lines.

ESC W1 can only be cancelled by ESC W0.

SI — CONDENSED PITCH

Format **CHR$(15);**

Function This code causes characters to be printed as condensed characters (17 character per inch).

This code can be used in combination with SO (Enlarged mode) to print condensed enlarged characters.

Condensed emphasized characters, however, cannot be printed. If the printer is already in Emphasized mode when you instruct it to print condensed characters, this setting will seemingly be ignored. You will get your condensed characters, however, as’soon as you leave Emphasized mode.

The exact same thing happens if you tell the printer to print condensed characters when it is already set for elite characters.

SI can also be expressed as ESC SI. The effect is exactly the same.

Example

```
Condensed characters
Pica characters
Elite characters
```

DC2 CANCELS CONDENSED MODE

Format **CHR$(18);**

Function If you are printing condensed characters when you send this code to the printer, it cancels the Condensed mode and returns to printing normal characters.

If you are printing elite-sized or emphasized characters with the Condensed mode waiting its turn (see SI for details), this code will cancel the waiting condensed mode.

ESC E — EMPHASIZED PRINT

Format **CHR$(27);“E”;**

Function This code sets the Emphasized character mode. This code causes the print head to overprint the dot pattern for a given character, shifting it slightly to the right. Emphasized mode gives you higher-quality, more distinctive printing.

Because this mode prints two dots for each dot of the pattern, printing speed drops to approximately 50 cps (characters per second)-about half the normal printing speed. Emphasized mode is cancelled by sending ESC F to the printer.

Emphasized elite characters cannot be printed. If you send ESC. E to the printer when it is printing elite characters, it will put the Emphasized mode on hold (see ESC M for details).

Condensed emphasized characters also cannot be printed. In this case Emphasized mode takes precedence (see SI for details).

Example

Pica	**Pica emphasized**
Elite	Elite emphasized
Condensed	**Condensed emphasized**

ESC F CANCELS EMPHASIZED PRINT

Format **CHR$(27);"F";**

Function This code cancels the Emphasized print mode set by ESC E.

ESC G DOUBLE-STRIKE PRINT

Format **CHR$(27);"G";**

Function This code overprints characters with two passes of the print head. After rhe first pass. rhe printer advances rhc paper by about one third of a dor. This fills **in** the vertical gap between dors for higher-quality printing.

Double-strike **mode** can be **used** in combination with any of the other print modes.

Example

Pica	**Pica double-strike**
Elite	**Elite double-strike**
Condensed	**Condensed double-strike**

ESC H CANCELS DOUBLE-STRIKE PRINT

Format **CHR$(27);"H";**

Function This **code** cancels the double-strike mode set by ESC G.

ESC S SUPER/SUBSCRIPT

Format **CHR$(27);"S";CHR$(n);**

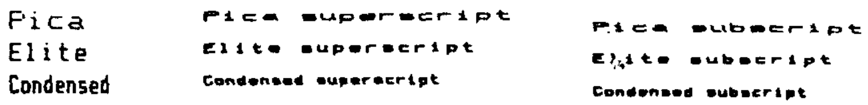
n=O or 48	Sets Superscript mode
n=1 or 49	Sets Subscript mode

Function This code sets the Superscript or Subscript mode. Superscript mode prints 1.6mm characters in the upper half of the print line and Subscript mode prints 1.6mm characters in the lower half of the print line.

These characters are formed using unidirectional double-strike printing.

Cancel this mode using ESC T.

Example



Pica Pica superscript Pica subscript
Elite Elite superscript Elite subscript
Condensed Condensed superscript Condensed subscript

ESC T CANCEL SUPER/SUBSCRIPT

Format **CHR$(27);"T";**

Function **This code cancels the Super/Subscript mode set by the ESC S code.**

ESC — UNDERLINE

Format CHR$(27);"–";CHR$(n);

n=l or 49	Sets Underlined print mode
n=0 or 48	Cancels Underlined print mode

Function This code sets or cancels the Underlined print mode. The underline is printed as a continuous line in all print modes; this code is ignored, however, in Bit-image mode.

This is underlininq code ESC -1

ESC 4 ITALIC PRINT

Format **CHR$(27);"4";**

Function This code sets Italic **mode. Italic mode can** be combined with any of the other print modes. It does this by selecting the alternate character set in ROM codes **160** to ***254*** and printing them instead of codes ***32*** to ***126.***

Pica	*Pica italic*
Elite	*Elite italic*
Condensed	Condensed *italic*

ESC 5 CANCELS ITALIC PRINT

Format CHR$(27);"5";

Function This code cancels the Italic mode set by ESC 4.

ESC R Selects Character Set

Format ***CHR$(27);"R";CHR$(n);***
n=O to 10

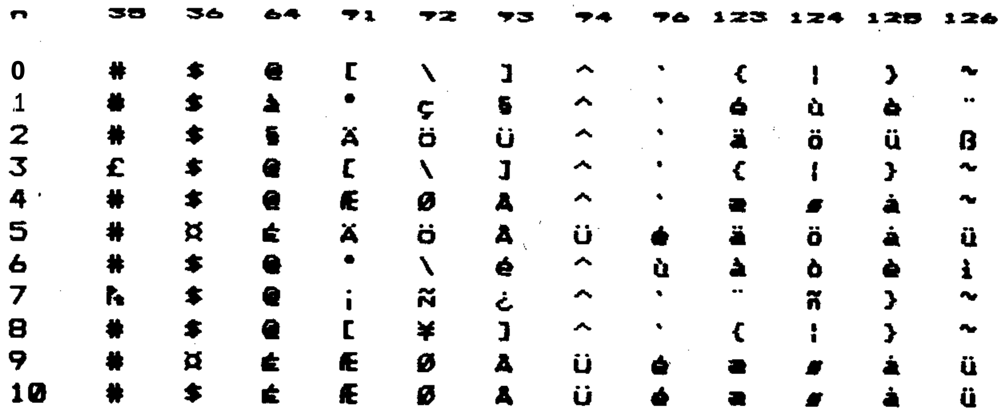
Function This code lets you select among international character sets. The RX printer can "speak" 10 languages using the codes stored in its ROM memory. The character set that you select here (using **n)** will remain in effect until another ESC R code is sent to the printer.

n	Country
0	U.S.A
1	France
2	Germany
3	England

n	Country
4	Denmark I
5	Sweden
6	Italy
7	Spain

n	Country
8	Japan
9	Norway
10	Denmark II

Example



n	35	36	64	91	92	93	94	96	123	124	125	126
0	#	$	@	[	\	]	^	`	{	|	}	~
1	#	$	à	°	ç	§	^	`	é	ù	è	¨
2	#	$	§	Ä	Ö	Ü	^	`	ä	ö	ü	ß
3	£	$	@	[	\	]	^	`	{	|	}	~
4	#	$	@	Æ	Ø	Å	^	`	æ	ø	å	~
5	#	¤	É	Ä	Ö	Å	Ü	é	ä	ö	å	ü
6	#	$	@	°	\	é	^	ù	à	ò	è	ì
7	₧	$	@	¡	Ñ	¿	^	`	¨	ñ	}	~
8	#	$	@	[	¥	]	^	`	{	|	}	~
9	#	¤	É	Æ	Ø	Å	Ü	é	æ	ø	å	ü
10	#	$	É	Æ	Ø	Å	Ü	é	æ	ø	å	ü

Format **CHR$(27);"m";CHR$(n);**

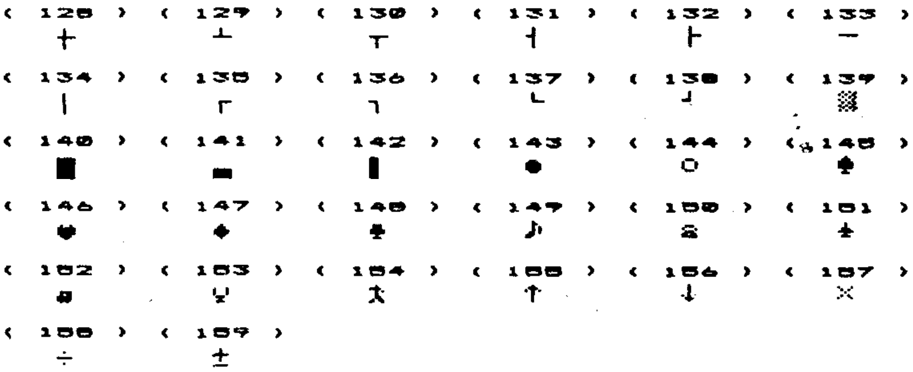
n=4 Selects graphic characters

n=0 Selects control codes

Function This code tells the printer how you want it to treat ROM **codes** 128 to 159. These codes can either be used as standard control codes (n=0) or for graphic characters (n=4).

This selection can also be performed by DIP switch l-2. See Appendix for details.

Example



```
< 128 >  < 129 >  < 130 >  < 131 >  < 132 >  < 133 >
   ┼        ┴        ┬        ┤        ├        ─
< 134 >  < 135 >  < 136 >  < 137 >  < 138 >  < 139 >
   │        ┌        ┐        └        ┘        ▒
< 140 >  < 141 >  < 142 >  < 143 >  < 144 >  < 145 >
   █        ▄        ▌        ●        ○        ♠
< 146 >  < 147 >  < 148 >  < 149 >  < 150 >  < 151 >
   ♥        ♦        ♣        ♪        ☎
< 152 >  < 153 >  < 154 >  < 155 >  < 156 >  < 157 >
                              ↑        ↓        ×
< 158 >  < 159 >
   ÷        ±
```

These characters are the same as those used with the Epson HX-20 Notebook Computer.

ESC 0 1/8 INCH LINE SPACING

Format **CHR$(27);"0";**

Function **This code changes** the line spacing (the amount that the printer advances the paper when it performs line feed) to 1/8 inch. This distance corresponds exactly to the height of a 6-by-9 dot matrix pattern, so you can print 9-pin bit-image graphics and not leave gaps between lines.

Example

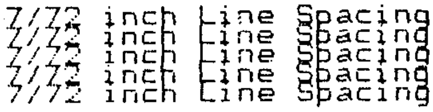
```
1/8 inch Line Spacing
1/8 inch Line Spacing
1/8 inch Line Spacing
1/8 inch Line Spacing
1/8 inch Line Spacing
```

ESC 1 — 7/72 INCH LINE SPACING

Format **CHR$(27);"1";**

Function This code changes the line spacing to 7/72 inch, equivalent to the vertical spacing for seven dots. Since standard characters are printed in a 5-by-7 dot matrix, this setting leaves no space between lines, which comes in handy for 8-pin bit-image graphics.

Example



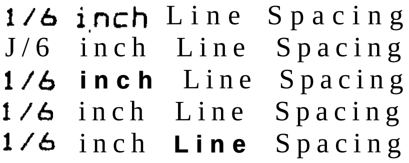
```
7/72 inch Line Spacing
7/72 inch Line Spacing
7/72 inch Line Spacing
7/72 inch Line Spacing
7/72 inch Line Spacing
```

ESC 2 — 1/6 LINE SPACING

Format CHR$(27);"2";

Function This code changes the line spacing to 1/6 inch, the height of 12 dots. This setting is the default setting and is automatically selected when the printer is switched on.

Example



```
1/6 inch Line Spacing
J/6 inch Line Spacing
1/6 inch Line Spacing
1/6 inch Line Spacing
1/6 inch Line Spacing
```

ESC J

Format **CHR$;"J";CHR$(n);**

Function This code lets you execute line feed for **n/216** inch for ultra-precise control of line spacing.

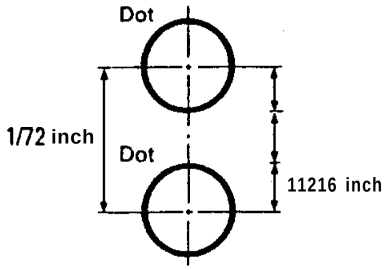
The size of the line feed you want to perform is specified by **n** as shown above. Because this code includes a line feed (LF), the printer advances the paper when you send it this code. This contrasts with ESC 3 that sets, but does not execute, line feed.

Because of the minute distances involved, the accuracy of paper feed performed when **n=l** or **n=2** is not assured.

ESC 3 n/216 INCH LINE SPACING

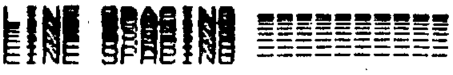
Format **CHR$(27);"3";CHR$(n);**
n=l to **255**

Function This code sets the line spacing to n/2 16 inch. A line spacing of 1/216 is equivalent to 1/3 the vertical spacing between dots.



Example

```
1Ø FOR I=1 TO 1Ø
28 LPRINT CHR$(27);"3";CHR$(I);
38 LPRINT "LINE SPACING -----------"
4Ø NEXT
```



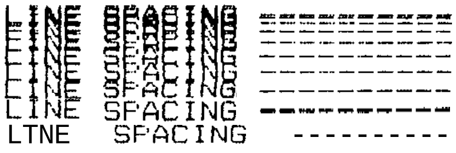
ESC A n/72 INCH LINE SPACING

Format CHR$(27);"A";CHR$(n);
n=O to 85

Function This code sets the line spaçing to **n/76** inch. Linc spacing of 1/76 inch is the amount of vertical spacing bctwccn two dots.

Example

```
10  FOR I-l TO 10
20 LPRINT CHR$(27);"A";CHR$(I);
30 LPRINT "LINE SPACING ----------"
40 NEXT
```



```
LINE SPACING ----------
LINE SPACING ----------
LINE SPACING ----------
LINE SPACING ----------
LINE SPACING ----------
LINE SPACING ----------
LTNE  SPACING  ----------
```

ESC l (el) — SETS LEFT MARGIN

Format **CHR$(27);"l"; CHR$(n);**

n=0 to 78 Pica and emphasized characters

n=0 to 93 Elite characters

n=0 to 128 Condensed characters

Function This code sets the left margin according to the currently selected character width. This code is used in combination with ESC Q to set the left and right margins of the print area.

Since the margin is set as the number of characters, the location of the left margin will depend on the current character width. The range of values that can be specified for **n** is therefore different for each print pitch.

In Enlarged mode, the values that can be specified for each type pitch will be half of those shown above.

The horizontal TAB positions are calculated from the left margin. Sending ESC l to the printer to set a new left margin will cancel the existing TABs; TABs specified after that will be based on the new left margin.

ESC Q — SETS RIGHT MARGIN

Format **CHR$(27);"Q";CHR$(n);**

n=2 to 80 Pica-sized and emphasized characters
n=3 to 96 Elite-sized characters
n=4 to 132 Condensed characters

Function This code sets the right margin.

The value specified here for n must be at least 2 greater than the left margin setting in Pica or Emphasized mode, 3 greater in Elite mode, and 4 greater in Condensed mode.

In Enlarged mode, the values that can be specified for n for each pitch will be half of those shown in the format above.

You should place this code on a program line by itself or else at the start of the line. Otherwise, data appearing before rhe code in the line may be lost.

ESC D — SETS HORIZONTAL TAB

Format **CHR$(27);"D";CHR$(n_1);CHR$($n_2$);...CHR$(n_k);CHR$(0);**
n=1 to 132
k=1 to 32

Use this code to specify horizontal TAB positions. You can indicate as many as 32 horizontal TABs by specifying values for n_1...n_k as shown above. These values should be specified from smallest to largest.

You must input CHR$(O) **at the** end of **the** TAB specifi**cation. Otherwise, the TABs will not be set correctly.**

Once you've set the TABs, the print head will move to the next TAB each time you send it an HT **code.**

The printer calculates these positions by multiplying the number of characters (specified as **n**) by the current character width. These distances are then remembered as absolute values which means that they will not change even if you start printing characters of a different pitch.

You can specify TAB positions up to the 80th character position for pica-sized characters, to the 96th position for elite, and to the 132nd position for condensed.

The default setting is a TAB every eight character positions.

HT

Format **CHR$(9);**

Function Sending this code to the printer moves the print head to the next horizontal TAB position. The ESC D code is used to set these positions. When you turn the power for your printer on, horizontal TABs are set at every eight characters. If more than one HT is input, the print head moves that many TAB positions to the right. If you input SO many HTs that the print head moves past the right margin, it will return to the left margin of the next line and continue printing from there.

Note that you may not be able to use this code in the version of BASIC used by your computer. In this case, try sending CHR$(137). It should work fine.

ESC B

Format CHR$(27);"B";CHR$(n_1);CHR$($n_2$);...CHR$(n_k);CHR$(0);
n=l to **254**
k=1 to 16

Use this code to set the vertical TAB positions.

You can indicate as many as **16** vertical TAB positions by specifying values for **n** as shown above. These values should be specified from smallest to largest.

You must input CHR$(0) at the end of the TAB specification. Otherwise, the TABs will not be set corrcctly.

Once you've set the vcrtical TABs, the paper will advance to the next vertical TAB every time you send the printer a VT code (explained next). The printer calculates these positions by multiplying the number of lines **(n)** by the current iine spacing. This position is then remembered as an absolute value, which means it will not change even if the line spacing itself is subsequently changed.

VT EXECUTES VERTICAL TAB

Format CHR$(11);

Function Sending this code to the printer causes the paper to advance to the next vertical TAB position (set by ESC B or ESC h).

Since this code contains an LF, it cancels the Enlarged print mode set by SO.

ESC b PRESETS VERTICAL FORMAT

Format CHR$(27);"b";CHR$(m);CHR$($n_1$);...CHR$(n_k);CHR$(0);
m=O to 7
n=1 to 254
k=l to 16

Function Your printer can remember up to eight different sets of vertical **TAB** settings. This code is used to program these formats, that is, to preset the vertical TABs for each page format.

For example, you could set TABs at the 3rd, 5th, and **15th** lines for format 1 and at the 6th, 10th, and 30th lines for format 2.

To set the vertical TABs, specify the format number as **m as** given above. Then specify the vertical TAB positions as n_1...n_k. Values for **n** are specified in exactly the same manner as they are with the ESC B code.

You must input CHR$(O) at the end of the TAB specification. Otherwise, the TABs will not be set correctly.

The printer automatically selects format 0 when power is turned on. This is also tine format set by ESC B.

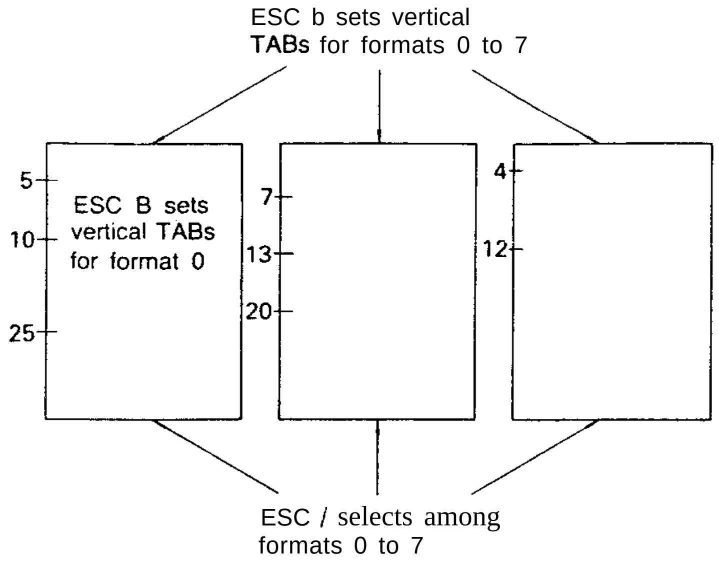
ESC / SELECTS VERTICAL FORMAT

Format **CHR$(27);"/";CHR$(n);**
n=0 to 7

Function This code selects from among the vertical TAB formats set by ESC b.

Send this code to the printer with the desired format number specified as **n** above.

If format **0** is selected, the vertical TABs set by ESC B will be in effect.



ESC e HORIZONTAL/VERTICAL TAB

Format CHR$(27);"e";CHR$(m);CHR$(n);
m=O or 1

Function This code sets vertical (m= 1) or horizontal (m=O) TABs every n lines or characters. For vertical TABs, a TAB position that exceeds the length of the page will be ignored and for horizontal TABs, the minimum value for n is 2 (pica characters).

ESC f PAPER FEED/TAB EXECUTE

Format **CHR$(27);"f";CHR$(m);CHR$(n);**
n=O to 127
m=O or 1

Function This code can be used to advance the paper **(m=** 1) or move the print head to the right **(m=O).** The units for these operations are lines and characters spaces, respectively, specified by **n.**

ESC C

Format **CHR$(27);"C";CHR$(n);**
n=1 to 127 (number of lines)
CHR$(27);"C";CHR$(0);CHR$(n);
n=1 to 22 (inches)

Function This code sets the form length, or how much the paper is advanced when FF (form feed) is performed. As shown above, the form length can be specified either as the number of lines or in inches.

If the form length is specified in lines, the printer multiplies the value specified as n by the current line spacing.

The printer remembers the form length as an absolute value. This means that it will not change even if the line spacing is changed.

The position of the print head when this code is input becomes the new top of form position.

DIP switch pin 1-4 is used to set either 11- or 12-inch form length for the default value.

Note that input of this code cancels the amount that the paper automatically advances to skip over the perforation at the end of the form (set by ESC N code, described next).

ESC N

Skip-Over-Perforation

Format **CHR$(27);"N";CHR$(n);**
n=1 to 127

Function This code is used to set the number of lines that will be left blank at the bottom of the form when printing on continuous-feed paper.

For example, to leave the last five lines of the form blank, send this code to the printer with a value of 5 for **n.**

Then when the prinrer reaches the fifth line from the bottom of the form (set by ESC C), it stops printing and advances the paper to the next top-of-form position. Any setting that exceeds the form length is ignored.

The setting performed by this code is cancelled if ESC C is sent to the printer to set a new form length.

You can set a I-inch "skip-over" by setting DIP switch pin **2-4** to ON.

ESC 0 — Cancels Skip-Over-Perforation

Format CHR$(27);"O";

Function This code cancels the skip-over-perforation set by ESC N. It does not affect any other aspect of the page format, such as line spacing, left and right margins, etc.

AIRE AND ANGELS

Twice or thricet-bad I lov-
ed thee,
Be-Fore I knew thyfa
ce or name;
SO in a voice,
so in a shapel
ess flame,
Angells af
fect us of
t, and wor-
ship'd bee
i
Still
when
to
where
thou
wert
I c
ame,
Some lovely gloriousnoth
ing I did see,
But since my so
ule, whose child lov
e is,
Takes limmes of
flesh, and els
e could nothing
doe,
More
subtile th
an the par
ent is,
Love
must
not b
e, bu
t tak
e a b
ody t

ESC K NORMAL-DENSITY BIT-IMAGE

Format **CHR\$(27);“K”;CHR\$(n1);CHR\$(n2);CHR\$(d_1);... CHR\$($d_{n1,n2}$);**

Function This code sets Normal-density bit-image mode. In this mode, data (**d_1...$d_{n1,n2}$**) sent to the printer are treated as bit-image data that direct the firing of each pin of the print head.

nl and **n2** together specify the total number of bit-image data to be sent to the printer.

nl is the modulus (remainder of integer division) of the total number of data by 256; **n2** is the quotient of this division.

Refer to Chapter 4 Bit-Image Graphics for details.

The maximum number of data that can be specified for one line of printing in this mode is 480.

Bit-image and Text modes can be mixed on the same line. In this case, however, the total number of bit-image data that can be specified is decreased for each character of text data; by 6 data for each pica or emphasized character, 3.5 for each condensed, and 5 for each elite character. These values are doubled for Enlarged mode.

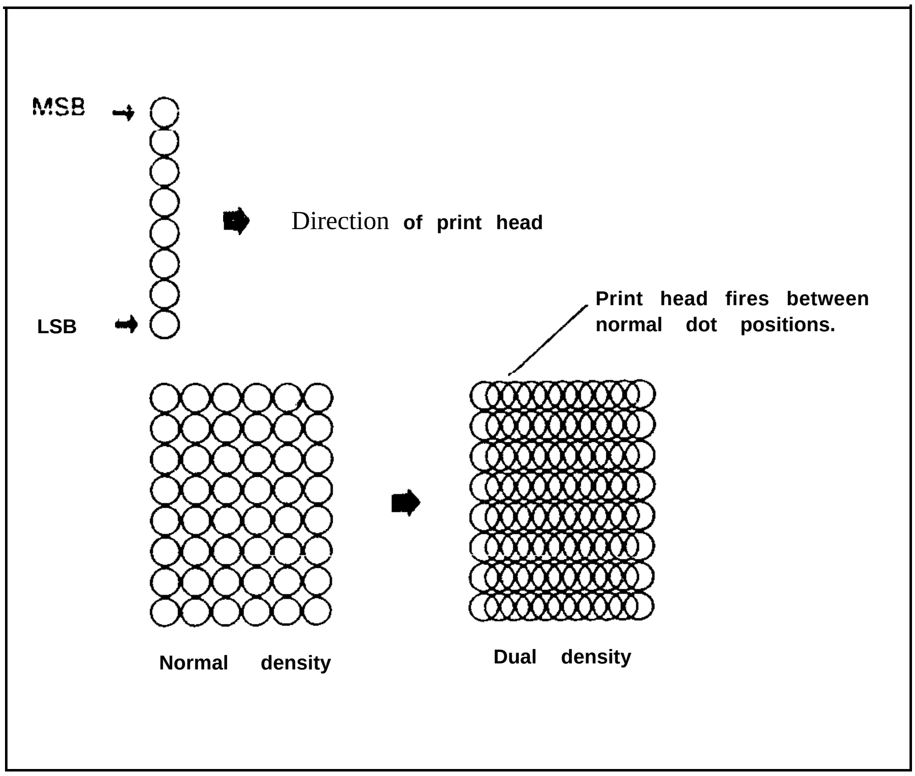
ESC L DUAL-DENSITY BIT-IMAGE

Format CHR$(27);"L";CHR$(n1);CHR$(n2);CHR$(d_1);... CHR$($d_{n1,n2}$)

Function This code sets Dual-density bit-image mode. The function of this mode is identical to that of Normal-density mode except that twice the number of data can be printed per line. The total number of bit-image data per line in this mode is 960.

Different bit-image modes as well as bit-image and text printing can be mixed on the same line.

If dual-density bit-image printing is mixed with text data, the total number of bit-image data that can be specified is decreased for each character of text data; by 12 for each pica-sized or emphasized character, 7 for each condensed, and **10** for each elite-sized character. These values are doubled for Enlarged mode.



Printing speed in this mode decreases from 10 II'S (inches per second) to 5 IPS.

ESC Y DOUBLE-SPEED, DUAL-DENSITY BIT-IMAGE

Format **CHR$(27);"Y";CHR$(n1);CHR$(n2);CHR$(d_1);CHR$($d_2$)... CHR$($d_{n1,n2}$);**

Function This code sets Double-speed dual-density bit-image mode.

The function of this mode is identical to that of ESC L except that, because horizontally adjacent dots are not printed, a print speed of 10 IPS is maintained.

ESC Z

Format **CHR$(27);"Z";CHR$(n1);CHR$(n2);CHR$(d_1);... CHR$($d_{n1,n2}$)**

Function This code sets Quadruple-density bit-image mode.

The function of this mode is identical to that of Normal-density mode except that four times the number of data can be printed per line. The total number of bit-image data per line in this mode is 1,920.

In this mode, horizontally adjacent dots are not printed. The print speed is 5 IPS, the same as that of Dual-density mode.

ESC *

Format **CHR$(27);“*”;CHR$(m);CHR$(n1);CHR$(n2);CHR$(d$_1$);...**
CHR$(d$_{n1,n2}$);

Function This code selects the bit-image mode.

The value of m selects the mode as shown in the table below.

m	Mode	Dot/inch	Head Speed (inch/sec)	Equivalent code
0	Normal density	60	16	ESC K
1	Dual density	120	8	ESC L
2	Double-speed Dual density	120	16	ESC Y
3	Quadruple density	240	8	ESC Z
4	CRT Graphics	80	8	—
6	CRT Graphics II	90	8	—

For details of operation, refer to the description of each code. m=4 selects CRT graphics mode which is optimal for making hard copies of the CRT display.

ESC ? REDEFINES BIT-IMAGE MODES

Format **CHR$(27);“?”;“n$”;CHR$(m);**
n$=K, L, Y, or Z
m=1 to **4,** or 6

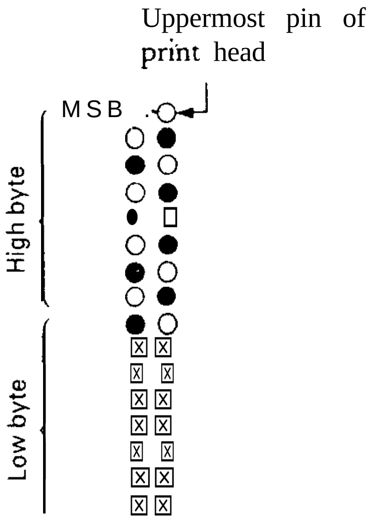
Function This code lets you redefine the bit-image modes.

For example, ESC L, which normally selects Dual-density mode, could be redefined to select Quadruple-density mode. By using this code at the beginning of a program that has already been developed, you could change the result of program execution without having to rewrite ESC L every time it occurs.

Format **CHR$(27);"^";CHR$(a);CHR$(n1);CHR$(n2);CHR$($d_{1H}$); CHR$(d_{1L});CHR$($d_{2H}$);CHR$(d_{2L});...CHR$($d_{n1,n2L}$);**
a=0 or **1**
H: High byte, **L:** Low byte

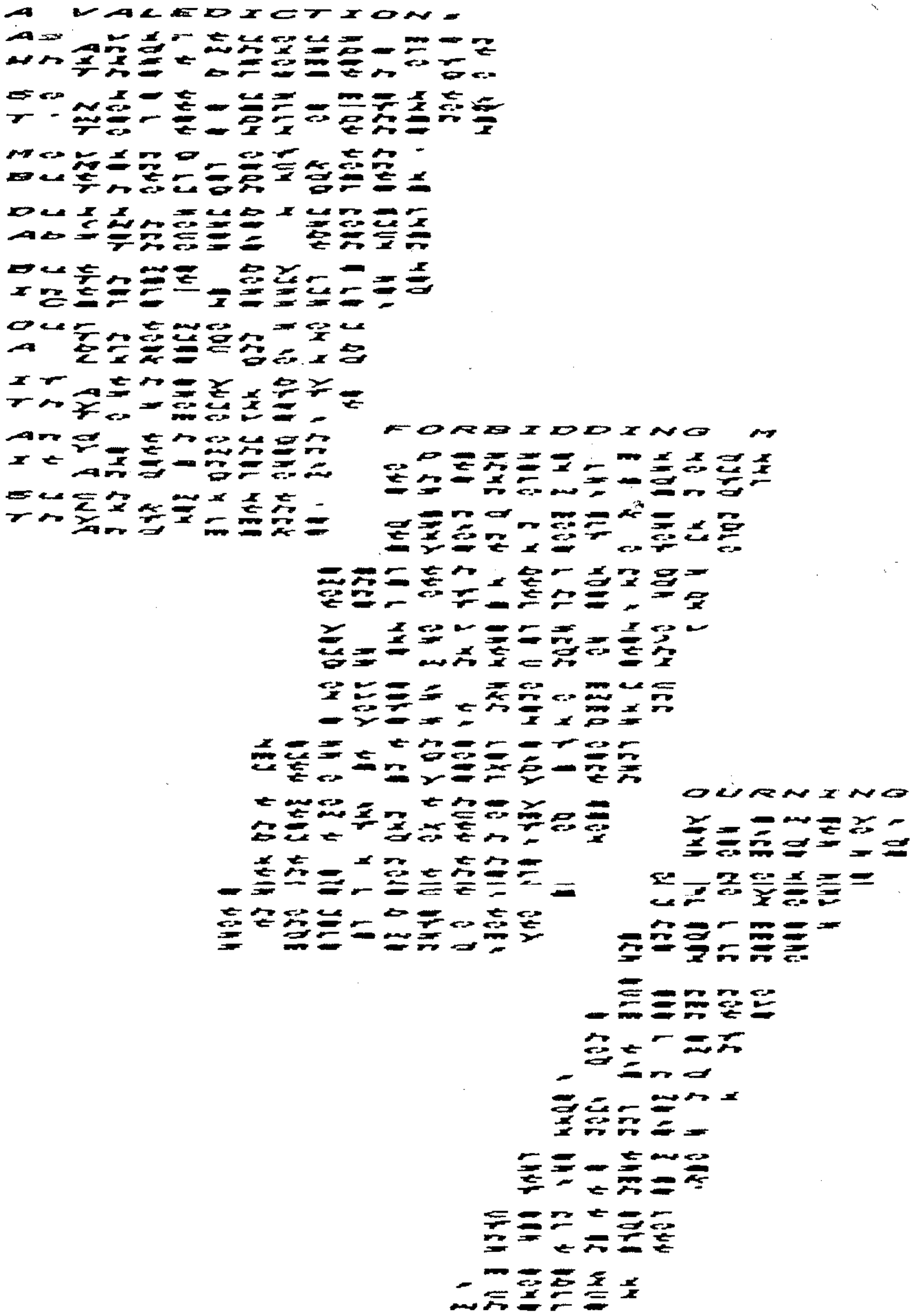
Function This code sets the 9-pin bit-image mode.

a specifies the dot density. When **a**=0, normal density (60 dots/inch) is specified. When **a**- 1, dual density (120 dots/ inch) is specified.



As shown in the figure, the high byte specifies which of the high-order 8 pins of the print head are to fire and the MSB of the low byte determines whether or not the 9th pin of the print head is to fire.

Two bytes must therefore be used to print each dot position and the number of dot positions is half of the value specified by **nl** and **n2.**


A VALEDICTION:
FORBIDDING M
OURNING

BEL — SOUNDS BUZZER

Format CHR$(7);

Function This code causes the buzzer on the printer to sound for approximately 0.2 second.

CAN — CANCELS DATA IN THE PRINT BUFFER

Format **CHR$(24);**

Function This code cancels all data on the line (clears the print buffer).

BS

BACKSPACE

Format **CHR$(8);**

Function This code moves the print-position back one character space. The size of the space depends on the pitch (width) mode being used. For instance, if you are printing in Enlarged mode, BS will move the print position back the equivalent of two pica spaces.

DEL

CANCELS LAST CHARACTER

Format **CHR$(127);**

Function This code cancels the last printable character (code other than a control code) sent to the printer. DEL is ignored when printing bit-image graphics.

ESC 9 PAPER END DETECTOR-ON

Format CHR$(27);"9";

Function Your RX printer is provided with a function to switch the printer off line when it detects the end of the paper (you can resume printing by placing new paper in the printer and pressing the ON LINE switch). This code is used to enable this function.

ESC 8 Paper End Detector—Off

Format CHR$(27);"8";

Function The ESC 8 code disables the paper end detector to allow you to print up to the end of the last page or when printing single sheets. The paper end detector can also be disabled by setting DIP switch pin 1-5 to ON.

Note that this code is not effective if sent to the printer after the printer has already detected the end of the paper and gone off line.

ESC < UNIDIRECTIONAL PRINT (ONE LINE)

Format **CHR$(27);"<";**

Function This code sets unidirectional printing for the current line only. In this mode, printing is performed only when the print head is moving from left to right and results in improved vertical alignment of dots.

ESC U UNIDIRECTIONAL PRINTING

Format CHR$(27);"U";CHR$(n);
n=1 or 49 Unidirectional printing
n=0 or 48 Bidirectional printing (except bit-image graphics)

Function This code sets unidirectional printing. In contrast with ESC <, this setting remains in effect after the first line has been printed.

Unidirectional printing is useful mainly to improve the vertical alignment of dots. Normally, the quality of bidirectional printing is more than enough. There may be occasions, however, when you want the extra-precise vertical alignment of unidirectional printing.

ESC s

HALF SPEED PRINTING

Format CHR$(27);“s”;CHR$(n);
n=1 or 49 Sets half speed printing
n=O or 48 Cancels half speed printing

Function This codes sets or cancels half speed printing.

Half speed printing reduces the normal printing speed of 100 cps to 50 cps. This reduces printer noise for a quieter home or office environment.

THE UNDERTAKING

I have done one braver thing
Than all the Worthies did
And yet a braver thence doth spring,
Which is, to keepe that hid.

It were but madnesse now t'impart
The skill of specular stone
When he which can have learn'd the art
To cut it, can find none.

So, if I now should utter this,
Others (because no more
Such stuffe to worke upon, there is,)
Would love but as before.

But he who lovelinesse within
Hath found, all outward loathes,
For he who colour loves, and skinne,
Loves but their oldest clothes.

If, as I have, you also doe
Vertue'attir'd in woman see,
And dare love that, and say so too,
And forget the Hee and Shee;

And if this love, though placed so,
From prophane men you hide,
Which will no faith on this bestow,
Or if they do, deride:

THen you have done a braver thing
Than all the Worthies did
And a brave thence will spring,
Which is, to keepe that hid.

John Donne