III. More Ways to Print

Learning System/23 BASIC

III. More Ways to Print

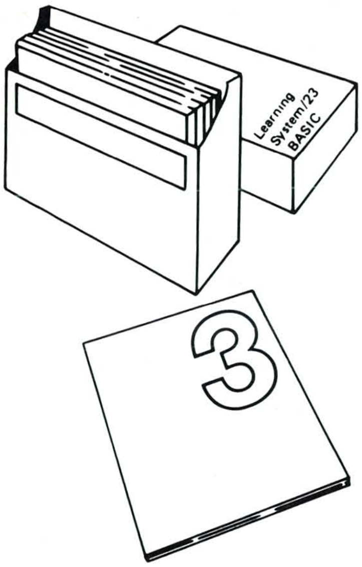
Learning System/23 BASIC

First Edition (January 1981)

III. More ways to PRINT

Contents

III. More ways to PRINT



This is the third in your series of seven books on *Learning System/23 BASIC*. You may have noticed that Book III is much shorter than the first two books. But don't let the size of this book fool you!

Book III contains important information. You will learn how to print information with your printer. You will also learn how to use your feature printer if you have one. The feature printer is the second printer attached to your System/23.

In this book, you will learn how to use spaces, blanks, and special formats when you display or print information. Because most programs are designed to display or print some results, the different ways of showing the results are important.

Again, you will need to have your diskette inserted in diskette drive 1 (or diskette drive 3 if you are using a 5322(0)). Make sure this diskette is the same diskette that you used for Books I and II.

You will also need to have your printer switched on, if your system has a printer. If you are just attaching a printer to your System/23 now, you must switch off your processor. Then, switch on your printer and your processor, in that order.

Chapter 1. Using your printer

Introduction

Up until now, you have used the PRINT statement to display information on your screen. In this chapter, you will learn how to use the PRINT #255 statement to print information with your system printer.

Even if you do not have a printer attached to your System/23, you may find this chapter interesting. We will review some of what you have already learned about PRINT and LIST. Also, you may decide to buy a printer in the future, and then you will need this information. Just don't try to enter any of the printer examples if you don't have a printer.

You will also learn how to use your feature printer, if you have a second printer attached to your System/23.

We will also show you how to print a program listing with your system printer.

Objectives

Upon completion of this chapter, you should be able to do the following:

- Print information with the printer by using the PRINT #255 statement.
- Print information with the feature printer, if you have one.
- Print a program listing by using the LISTP command.

If you are familiar with these tasks, try the exercises located at the end of this chapter. If not, read through the chapter before going on to the exercises.

Using your printer

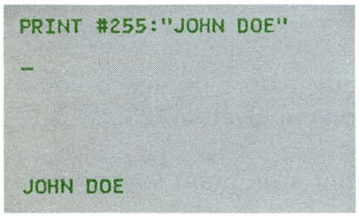
Printing information

So far, you have been using the PRINT statement to display information on the screen.

You can use the PRINT #255: statement to print information with the printer.

Before you try this, make sure that there is paper in the printer.

Now, enter the following statement, but use *your* name:



```
PRINT #255:"JOHN DOE"
_



JOHN DOE
```

```
PRINT #255:"JOHN DOE"
```

On the printer, you should see your name.

Look at the statement again:

```
PRINT #255:"JOHN DOE"
```

The #255 is called a file reference number, and it is reserved for the system printer. The #255: indicates to the System/23 that ''JOHN DOE'' should be printed by the printer, rather than displayed on the screen. You *must* enter the colon immediately following the #255.

Remember:

PRINT #255:''JOHN DOE'' means print with the printer.

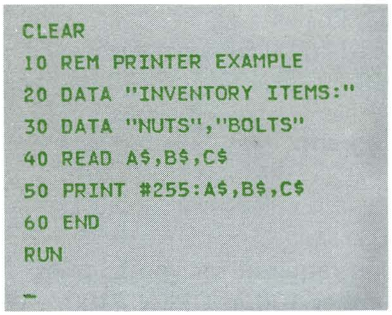
PRINT ''JOHN DOE'' means display on the screen.

Enter the following program. Remember that you can enter AUTO after you enter the CLEAR command if you want to automatically number your program statements.

```
CLEAR
10 REM PRINTER EXAMPLE
20 DATA "INVENTORY ITEMS:"
30 DATA "NUTS","BOLTS"
40 READ A$,B$,C$
50 PRINT #255:A$,B$,C$
60 END
```

Can you guess what this program will do?

Go ahead and run it:



```
CLEAR
10 REM PRINTER EXAMPLE
20 DATA "INVENTORY ITEMS:"
30 DATA "NUTS","BOLTS"
40 READ A$,B$,C$
50 PRINT #255:A$,B$,C$
60 END
RUN
_
```

```
RUN
```

```
INVENTORY ITEMS:          NUTS                          BOLTS
```

The values of A$, B$, and C$ are printed by the printer.

Notice the spacing between the items that you printed. The values of A$, B$, and C$ are printed in different print zones, because you placed commas between them in the PRINT statement. These print zones are just like the print zones on the screen.

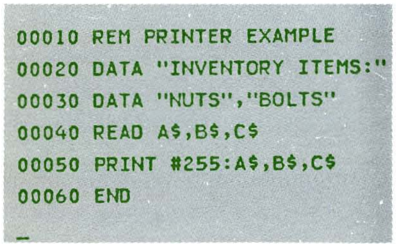
Using the feature printer

You may have a second printer attached to your System/23. We will refer to that second printer as the *feature printer*. If you do not have a feature printer, you should skip these next two pages, and go on to "Listing programs" on page 1-6.

We will show you how to change the program you just wrote, so that you print with the feature printer instead of the system printer.

First of all, list the program in the work area:

```
LIST
```



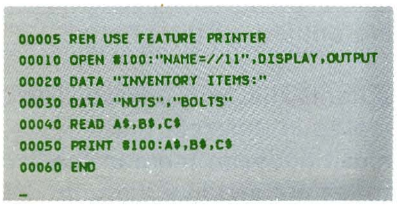
```
00010 REM PRINTER EXAMPLE
00020 DATA "INVENTORY ITEMS:"
00030 DATA "NUTS","BOLTS"
00040 READ A$,B$,C$
00050 PRINT #255:A$,B$,C$
00060 END
_
```

Now, enter the following changes:

```
5 REM USE FEATURE PRINTER
10 OPEN #100:"NAME=//11",DISPLAY,OUTPUT
50 PRINT #100:A$,B$,CS
```

List the new version of your program:

```
LIST
```



```
00005 REM USE FEATURE PRINTER
00010 OPEN #100:"NAME=//11",DISPLAY,OUTPUT
00020 DATA "INVENTORY ITEMS:"
00030 DATA "NUTS","BOLTS"
00040 READ A$,B$,C$
00050 PRINT #100:A$,B$,C$
00060 END
_
```

The OPEN statement in line 10 is required for you to print with the feature printer. The number 100 in OPEN #100: can be any number from 1 through 127.

The rest of the statement in line 10 must look exactly like we've shown it, except for the line number. This statement can be entered on any line number, as long as it is executed before the PRINT #100: statement.

The //11 tells your System/23 to use the feature printer. The 11 is the device address of the feature printer.

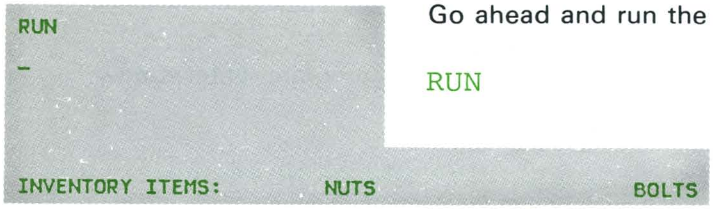
Notice the #100: in line 50.

```
50 PRINT #100:A$,B$,C$
```

This number must be the same as the number in the OPEN statement (line 10). Except for the #100, this statement is just like the statement used to print with the system printer.

Go ahead and run the program:

```
RUN
```



```
RUN
_

INVENTORY ITEMS:          NUTS                BOLTS
```

The values of A$, B$, and C$ are printed by the feature printer, just like they were by the system printer.

Remember: To print with the feature printer, you need two statements:

- `OPEN #nnn:"NAME=//11",DISPLAY,OUTPUT`
- `PRINT #nnn:`

where nnn is any number form 1 through 127.

Remember, however, that the statements in a BASIC program must be preceded by line numbers, like

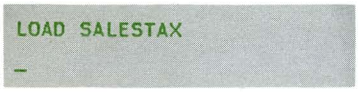
```
10 OPEN #100:"NAME=//11",DISPLAY,OUTPUT
```

Using your printer

Listing programs

In Chapter 2 of Book I, you learned how to list a program on the screen. You can also *print* a copy of your program listing. This is especially helpful when you have a program with a lot of statements.

First, load the SALESTAX program that you saved on your diskette in Chapter 6 of Book I:



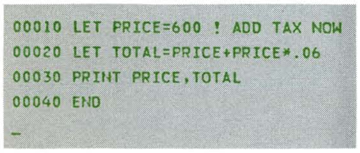
```
LOAD SALESTAX
_
```

```
LOAD SALESTAX
```

If you didn't save a copy of this program, enter it now.

```
CLEAR
10 PRICE=600 ! ADD TAX NOW
20 TOTAL=PRICE+(PRICE*.06)
30 PRINT PRICE,TOTAL
40 END
```

Now, list the program:

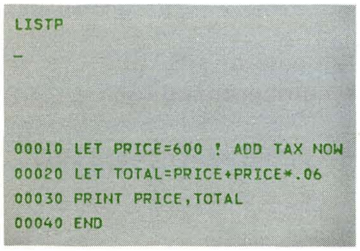


```
00010 LET PRICE=600 ! ADD TAX NOW
00020 LET TOTAL=PRICE+PRICE*.06
00030 PRINT PRICE,TOTAL
00040 END
_
```

```
LIST
```

Just as before, a listing of the program is displayed on the screen.

But watch what happens when you enter the following:



```
LISTP
_


00010 LET PRICE=600 ! ADD TAX NOW
00020 LET TOTAL=PRICE+PRICE*.06
00030 PRINT PRICE,TOTAL
00040 END
```

```
LISTP
```

Your program is listed by the printer. You now have a printed listing of the program, which you can use to help you know where to make future changes.

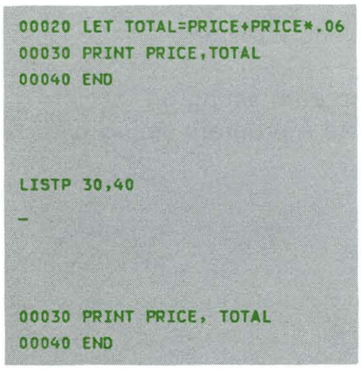
You can also print a section of a program with LISTP. For example, enter the following:

```
LISTP 20
_
```

```
LISTP 20
```

Only line 20 and any lines that follow are printed. Notice that this is different from LIST 20, where only line 20 and any *preceding* lines are *displayed*.

Now enter:



```
00020 LET TOTAL=PRICE+PRICE*.06
00030 PRINT PRICE,TOTAL
00040 END


LISTP 30,40
_


00030 PRINT PRICE, TOTAL
00040 END
```

```
LISTP 30,40
```

Only the lines from 30 through 40 are printed.

Remember:

LIST means *display* a program listing with the screen.

LISTP means *print* a program listing on the printer.

Warning: You can use the LISTP command only with the system printer, not with the feature printer. DIR and other system commands can also only be used with the system printer.

Using your printer

Chapter summary

The PRINT statement displays information on the screen. The PRINT #255: statement works like PRINT, but it prints information with the printer.

To use the feature printer, you need two statements:

- OPEN #nnn:"NAME=//11",DISPLAY,OUTPUT
- PRINT #nnn:

Where nnn is any number from 1 through 127. The statements must be preceded by line numbers.

The LIST command displays a program listing on the screen. The LISTP command prints a program listing with the printer.

Exercises

Question 1

Change line 20 of the following program so that the numbers 1 through 5 are printed by the printer.

```
10 FOR X=1 TO 5
20 PRINT X
30 NEXT X
40 END
```

Answer: ________________________________

Question 2

What command would you enter to obtain a printed listing of a program?

Answer: ________________________________

Question 3

If you do not have a feature printer, skip this question.

Using line numbers 5 and 20, change the program in Question 1 so that the numbers 1 through 5 are printed by the feature printer.

Answer: ________________________________

____

Using your printer

Answers

Question 1

```
20 PRINT #255:X
```

Question 2

```
LISTP
```

Question 3

```
5 OPEN #10:"NAME=//11",DISPLAY,OUTPUT
20 PRINT #10:X
```

The #10 could be any number from 1 through 127.

Chapter 2. Skipping spaces, lines, and pages

Introduction

Most programs are designed to produce some sort of *output*. The output is the results, which we display on the screen or print with the printer.

In this chapter, you will learn some more ways to use the PRINT and PRINT #255: statements to format your output.

Objectives

Upon completion of this chapter, you should be able to do the following:

- Use TAB with PRINT and PRINT #255: to format the output on the screen and printer.
- Use PRINT and PRINT #255: by themselves to skip lines.
- Use NEWPAGE with PRINT to clear the screen.
- Use NEWPAGE with PRINT #255: to advance to the next form on the printer.
- Use BELL with PRINT to ring the alarm.
- Use the PAUSE statement to interrupt a program while it is running.

If you are familiar with these tasks, try the exercises located at the end of this chapter. If not, read through the chapter before going on to the exercises.

Skipping spaces, lines, and pages

Using tabs

You've seen how PRINT can be used to display information on the screen. You've also seen how PRINT #255: can be used to print information with the printer.

Using commas and semicolons, you've seen how you can format the output on the screen or printer.

For example,

```
PRINT A,B,C
```

causes the values of variables A, B, and C to be displayed in three separate print zones on the screen. Also,

```
PRINT #255:A;B;C
```

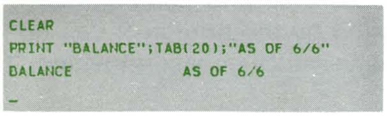
causes the values of variables A, B, and C to be printed one after another with no spacing by the printer.

Another way to control spacing is with TAB. Using TAB, you can indicate where you want an item to be displayed. Here's the way it works:

```
PRINT "BALANCE";TAB(20);"AS OF 6/6"
```

This statement causes the word BALANCE to be displayed, beginning in position 1 on the screen. TAB(20) indicates that the next item to be displayed (AS OF 6/6) should begin in position 20 of the screen.

Notice that we are using semicolons rather than commas to separate the items in the PRINT statement. If you use commas, your System/23 tries to use the 24 character print zones, and the TAB(20) would position the second item to be displayed on the *next* line. You can't use TAB to back up on the same line.



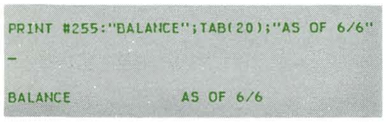
```
CLEAR
PRINT "BALANCE";TAB(20);"AS OF 6/6"
BALANCE              AS OF 6/6
_
```

Enter the PRINT statement:

```
CLEAR
PRINT "BALANCE";TAB(20);"AS OF 6/6"
```

This statement, with the addition of #255: can be used with the printer.

If you are using a printer with your System/23, enter:



```
PRINT #255:"BALANCE";TAB(20);"AS OF 6/6"
_

BALANCE              AS OF 6/6
```

```
PRINT #255:"BALANCE";TAB(20);"AS OF 6/6"
```

TAB works the same way with the printer as it does on the screen.

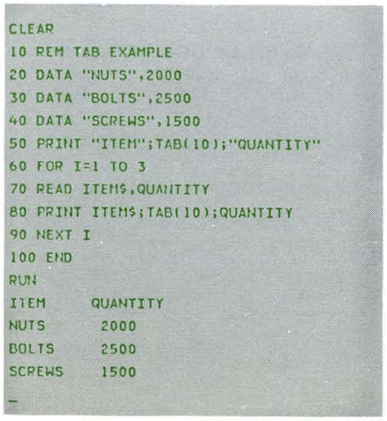
Let's use TAB in a program. Enter:

```
CLEAR
10 REM TAB EXAMPLE
20 DATA "NUTS",2000
30 DATA "BOLTS",2500
40 DATA "SCREWS",1500
50 PRINT "ITEM";TAB(10);"QUANTITY"
60 FOR I=1 TO 3
70 READ ITEM$,QUANTITY
80 PRINT ITEM$;TAB(10);QUANTITY
90 NEXT I
100 END
```

Remember from Book II that you can assign values to variables in a program by using READ and DATA statements.

Now, run this program and notice how the output is formatted:

```
RUN
```



```
CLEAR
10 REM TAB EXAMPLE
20 DATA "NUTS",2000
30 DATA "BOLTS",2500
40 DATA "SCREWS",1500
50 PRINT "ITEM";TAB(10);"QUANTITY"
60 FOR I=1 TO 3
70 READ ITEM$,QUANTITY
80 PRINT ITEM$;TAB(10);QUANTITY
90 NEXT I
100 END
RUN
ITEM     QUANTITY
NUTS      2000
BOLTS     2500
SCREWS    1500
_
```

Using tabs (continued)

Your turn!

Change the program you entered on the previous page so that the information will be printed by the printer. If you do not have a printer, write the answer in the space provided, but do not enter anything.

Answer: ______________________________________

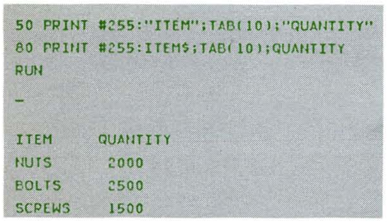
____

Here's our solution:

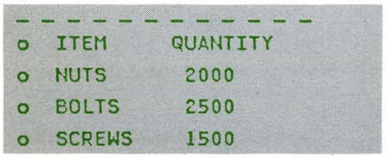
```
50 PRINT #255:"ITEM";TAB(10);"QUANTITY"
80 PRINT #255:ITEM$;TAB(10);QUANTITY
```

If you have a printer, run the program:

```
RUN
```



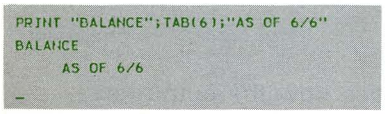
```
50 PRINT #255:"ITEM";TAB(10);"QUANTITY"
80 PRINT #255:ITEM$;TAB(10);QUANTITY
RUN
_

ITEM      QUANTITY
NUTS        2000
BOLTS       2500
SCREWS      1500
```



```
- - - - - - - - - - - - -
o  ITEM       QUANTITY
o  NUTS         2000
o  BOLTS        2500
o  SCREWS       1500
```

A note about TAB: To prevent skipping to another line, the position you indicate with TAB must be greater than the last character position displayed or printed. Remember: You can't use TAB to back up on the same line. For example, enter the following:

```
PRINT "BALANCE";TAB(6);"AS OF 6/6"
```



```
PRINT "BALANCE";TAB(6);"AS OF 6/6"
BALANCE
     AS OF 6/6

_
```

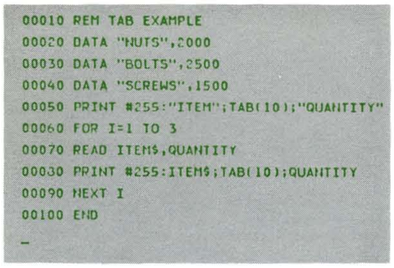
You can see that AS OF 6/6 is displayed in position 6 of the line under BALANCE.

Skipping lines

You can use the PRINT statement by itself to skip a line on the screen. Or, you can use PRINT #255: by itself to skip a line with the printer.

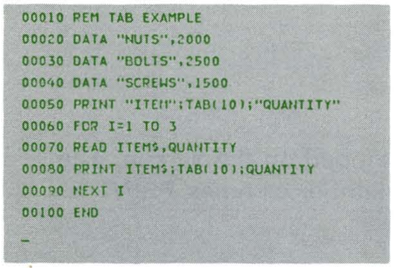
List the program in the work area:

```
LIST
```



```
00010 REM TAB EXAMPLE
00020 DATA "NUTS",2000
00030 DATA "BOLTS",2500
00040 DATA "SCREWS",1500
00050 PRINT #255:"ITEM";TAB(10);"QUANTITY"
00060 FOR I=1 TO 3
00070 READ ITEM$,QUANTITY
00080 PRINT #255:ITEM$;TAB(10);QUANTITY
00090 NEXT I
00100 END
-
```

If you do not have a printer, your program should look a little different.



```
00010 REM TAB EXAMPLE
00020 DATA "NUTS",2000
00030 DATA "BOLTS",2500
00040 DATA "SCREWS",1500
00050 PRINT "ITEM";TAB(10);"QUANTITY"
00060 FOR I=1 TO 3
00070 READ ITEM$,QUANTITY
00080 PRINT ITEM$;TAB(10);QUANTITY
00090 NEXT I
00100 END
-
```

Now, let's change this program to skip a line under the ITEM QUANTITY heading. Enter the following:

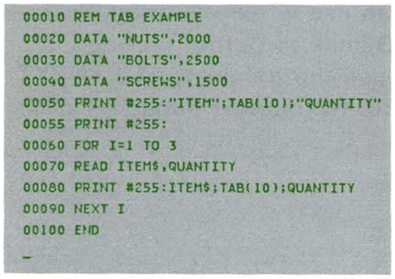
```
55 PRINT #255:
```

Or, if you don't have a printer, enter:

```
55 PRINT
```

List the new version of your program:

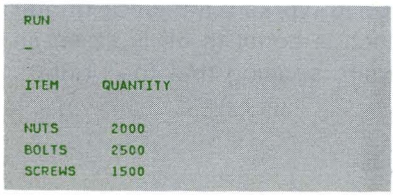
```
LIST
```



```
00010 REM TAB EXAMPLE
00020 DATA "NUTS",2000
00030 DATA "BOLTS",2500
00040 DATA "SCREWS",1500
00050 PRINT #255:"ITEM";TAB(10);"QUANTITY"
00055 PRINT #255:
00060 FOR I=1 TO 3
00070 READ ITEM$,QUANTITY
00080 PRINT #255:ITEM$;TAB(10);QUANTITY
00090 NEXT I
00100 END
-
```

Now, run your program:

```
RUN
```

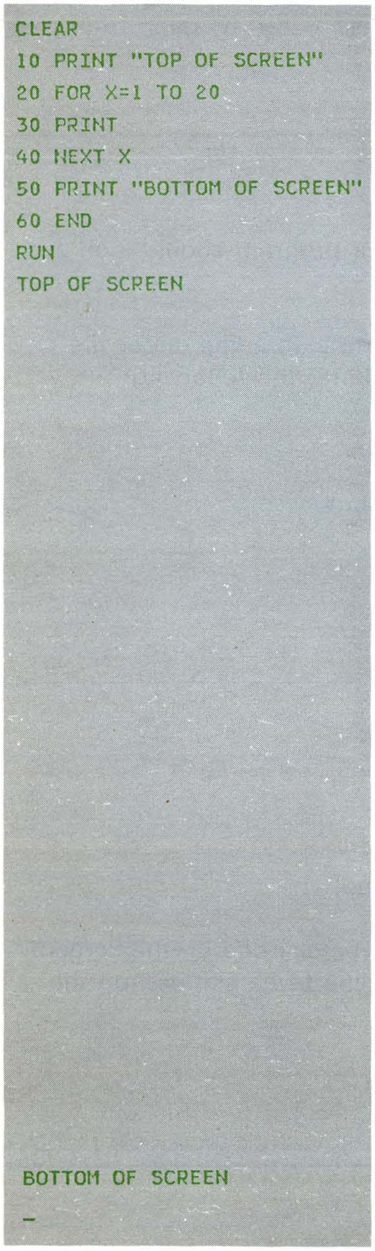


```
RUN
-
ITEM      QUANTITY

NUTS       2000
BOLTS      2500
SCREWS     1500
```

You can see that a line is skipped.

PRINT by itself causes a line to be skipped on the screen. PRINT #255: by itself causes a line to be skipped on the printer.

Skipping lines (continued)



```
CLEAR
10 PRINT "TOP OF SCREEN"
20 FOR X=1 TO 20
30 PRINT
40 NEXT X
50 PRINT "BOTTOM OF SCREEN"
60 END
RUN
TOP OF SCREEN





















BOTTOM OF SCREEN
_
```

Try this:

```
CLEAR
10 PRINT "TOP OF SCREEN"
20 FOR X=1 TO 20
30 PRINT
40 NEXT X
50 PRINT "BOTTOM OF SCREEN"
60 END
```

Now run the program:

```
RUN
```

As you can see, 20 lines are skipped on the screen. The FOR-NEXT loop controls the number of times PRINT is executed.

This is one method that you can use when you print reports. If you want to print 50 lines of output on one 11-inch page, then you could use the following FOR-NEXT loop after the fiftieth line to skip to the next page:

```
FOR X=1 TO 16
PRINT #255:
NEXT X
```

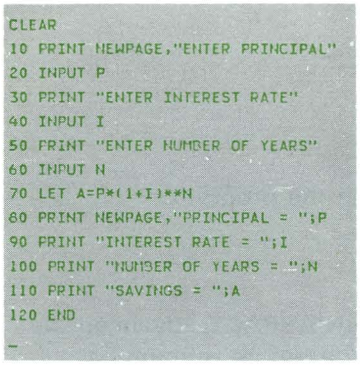
The FOR-NEXT loop shown above works when your printer is set for six lines per inch, which is equal to 66 lines on an 11-inch page. This is the standard spacing that the printer uses.

Using NEWPAGE

You can use a FOR-NEXT loop to completely clear the screen. You can also use it to skip to a new page on the printer.

But you can do both of these tasks with NEWPAGE. Here's an example of NEWPAGE on the screen. Enter:

```
CLEAR
10 PRINT NEWPAGE,"ENTER PRINCIPAL"
20 INPUT P
30 PRINT "ENTER INTEREST RATE"
40 INPUT I
50 PRINT "ENTER NUMBER OF YEARS"
60 INPUT N
70 A=P*(1+I)**N
80 PRINT NEWPAGE,"PRINCIPAL = ";P
90 PRINT "INTEREST RATE = ";I
100 PRINT "NUMBER OF YEARS = ";N
110 PRINT "SAVINGS = ";A
120 END
```



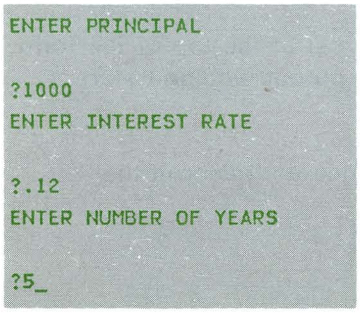
```
CLEAR
10 PRINT NEWPAGE,"ENTER PRINCIPAL"
20 INPUT P
30 PRINT "ENTER INTEREST RATE"
40 INPUT I
50 PRINT "ENTER NUMBER OF YEARS"
60 INPUT N
70 LET A=P*(1+I)**N
80 PRINT NEWPAGE,"PRINCIPAL = ";P
90 PRINT "INTEREST RATE = ";I
100 PRINT "NUMBER OF YEARS = ";N
110 PRINT "SAVINGS = ";A
120 END
_
```

Now run the program:

```
RUN
```

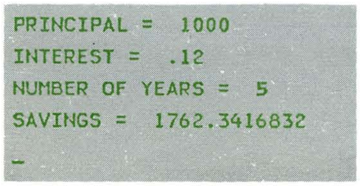
The first NEWPAGE in line 10 clears the screen. Now, enter the following values for P, I, and N:

```
ENTER PRINCIPAL
?1000
ENTER INTEREST RATE
?.12
ENTER NUMBER OF YEARS
?5
```



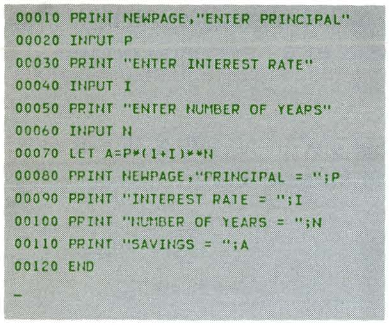
```
ENTER PRINCIPAL

?1000
ENTER INTEREST RATE

?.12
ENTER NUMBER OF YEARS

?5_
```

The second NEWPAGE in line 80 clears the screen before the output is displayed.



```
PRINCIPAL =  1000
INTEREST =  .12
NUMBER OF YEARS =  5
SAVINGS =  1762.3416832
_
```

Using NEWPAGE (continued)



```
00010 PRINT NEWPAGE,"ENTER PRINCIPAL"
00020 INPUT P
00030 PRINT "ENTER INTEREST RATE"
00040 INPUT I
00050 PRINT "ENTER NUMBER OF YEARS"
00060 INPUT N
00070 LET A=P*(1+I)**N
00080 PRINT NEWPAGE,"PRINCIPAL = ";P
00090 PRINT "INTEREST RATE = ";I
00100 PRINT "NUMBER OF YEARS = ";N
00110 PRINT "SAVINGS = ";A
00120 END
-
```

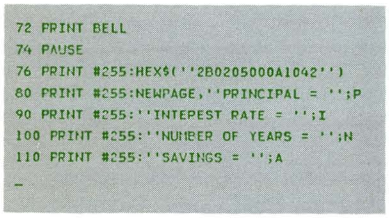
Skipping to a new page on the printer is just a little different from clearing the screen.

List the program in the work area:

```
LIST
```

If you have a printer, make the following changes:

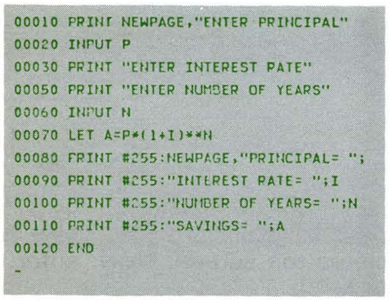
```
80 PRINT #255:NEWPAGE,"PRINICPAL = ";P
90 PRINT #255:"INTEREST RATE = ";I
100 PRINT #255:"NUMBER OF YEARS = ";N
110 PRINT #255:"SAVINGS = ";A
```



```
72 PRINT BELL
74 PAUSE
76 PRINT #255:HEX$(''2B0205000A1042'')
80 PRINT #255:NEWPAGE,''PRINCIPAL = '';P
90 PRINT #255:''INTEPEST RATE = '';I
100 PRINT #255:''NUMBER OF YEARS = '';N
110 PRINT #255:''SAVINGS = '';A
-
```

List the new version of your program:

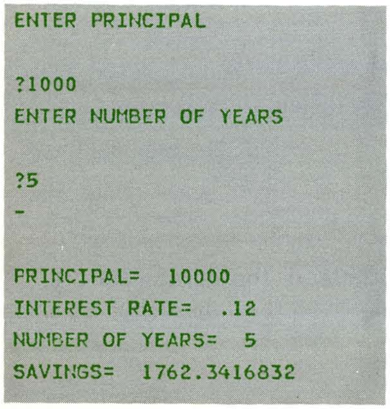
```
LIST
```



```
00010 PRINT NEWPAGE,"ENTER PRINCIPAL"
00020 INPUT P
00030 PRINT "ENTER INTEREST PATE"
00050 PRINT "ENTER NUMBER OF YEARS"
00060 INPUT N
00070 LET A=P*(1+I)**N
00080 PRINT #255:NEWPAGE,"PRINCIPAL= ";
00090 PRINT #255:"INTEREST PATE= ";I
00100 PRINT #255:"NUMBER OF YEARS= ";N
00110 PRINT #255:"SAVINGS= ";A
00120 END
-
```

And now, if you have a printer, run the program:

```
RUN
```



```
ENTER PRINCIPAL

?1000
ENTER NUMBER OF YEARS

?5
-

PRINCIPAL= 10000
INTEREST RATE= .12
NUMBER OF YEARS= 5
SAVINGS= 1762.3416832
```

You can see that the PRINT #255:NEWPAGE advanced the paper in the printer to the next page before printing.

If the paper in your printer wasn't set at the top of the form before you ran this program, your output wouldn't start at the top of the next form.

Let's look at another way to advance the paper in the printer. Enter the following:

```
72 PRINT BELL;;
74 PAUSE
80 PRINT #255:"PRINCIPAL = ";P
```

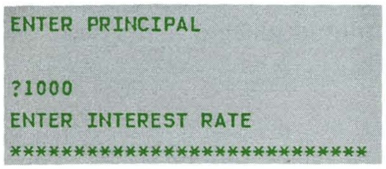
Before you run this program, look at lines 72 and 74.

Line 72 (PRINT BELL;;) causes the alarm to ring momentarily. When the alarm rings, you should advance the paper in the printer to the top of the page. PRINT BELL is a useful tool to remind you to perform some action while the program is running.

Line 74 is a PAUSE statement. It interrupts a program to let you do something while the program is running. In this example, PAUSE is giving you time to advance the paper in the printer. When your System/23 executes a PAUSE statement, you will see a line of asterisks across the screen.

When the bell rings and the asterisks appear, you will be able to adjust the paper in the printer. After you adjust the paper, you must enter GO. The GO command tells your program to resume running, starting with the next line number.

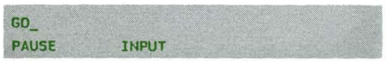
Now run the program with the following input:



```
ENTER PRINCIPAL

?1000
ENTER INTEREST RATE
****************************
```

```
RUN
ENTER PRINCIPAL
?1000
ENTER INTEREST RATE
?.12
ENTER NUMBER OF YEARS
?5
```

The asterisks should appear on the screen. Adjust the paper to the top of the next form, and then enter:



```
GO_
PAUSE        INPUT
```

```
GO
```

Skipping spaces, lines, and pages

Using NEWPAGE (continued)

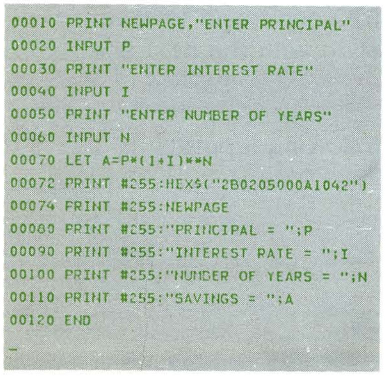
In the program you just ran, you didn't use NEWPAGE in line 80. Instead, you manually adjusted the paper in the printer.

Now let's look at one more PRINT statement. Enter the following (be sure to enter the letters in line 72 in uppercase):

```
72 PRINT #255:HEX$("2B0205000A1042")
74 PRINT #255:NEWPAGE
```

List the new version of your program:

```
LIST
```



```
00010 PRINT NEWPAGE,"ENTER PRINCIPAL"
00020 INPUT P
00030 PRINT "ENTER INTEREST RATE"
00040 INPUT I
00050 PRINT "ENTER NUMBER OF YEARS"
00060 INPUT N
00070 LET A=P*(1+I)**N
00072 PRINT #255:HEX$("2B0205000A1042")
00074 PRINT #255:NEWPAGE
00080 PRINT #255:"PRINCIPAL = ";P
00090 PRINT #255:"INTEREST RATE = ";I
00100 PRINT #255:"NUMBER OF YEARS = ";N
00110 PRINT #255:"SAVINGS = ";A
00120 END
_
```

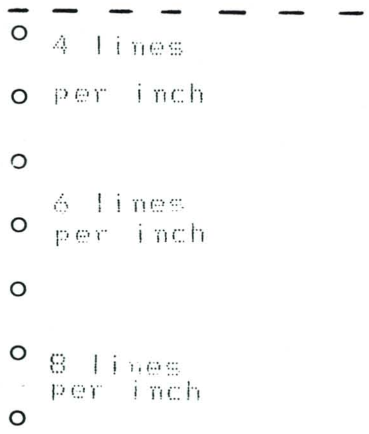
The HEX$ value in line 72 controls spacing on the printer. We are telling your System/23 to print 10 characters per inch, 6 lines per inch, and 66 lines per page. Here the statement is optional because these are the standard values that your System/23 uses.

You can indicate other values with this statement. The first eight characters inside the quotes always remain the same. The last six characters control printer spacing. Let's see.

$\texttt{2B020500}\underbrace{\texttt{0A}}\texttt{1042}$

This value controls the spacing of characters on a line. You can use 0A, which means 10 characters per inch, or 0F, which means 15 characters per inch.

```
10 CHARACTERS PER INCH
15 CHARACTERS PER INCH
```



$\texttt{2B02040A}\underbrace{\texttt{10}}\texttt{42}$

This value controls the spacing between the lines on a page. The value 10 is the HEX representation of 16 which makes the computer space down 1/6 of an inch for each line. That allows six lines of type in each inch. The following are some of the values you can enter:

For 4 lines per inch, enter 18
For 8 lines per inch, enter 0C

$\texttt{2B0205000A10}\underbrace{\texttt{42}}$

This value controls the number of lines per page. The value 42 means 66 lines per page. The following are some of the values you can enter:

For 10 lines per page, enter 0A
For 50 lines per page, enter 32

If your System/23 has a 5242 printer, you can use HEX$ to produce a high quality print with your printer. The HEX$ value you enter is:

$\texttt{PRINT \#255:HEX\$("2BD10705FF}\underbrace{\texttt{01}}\texttt{000000")}$

To return to normal printing enter:

```
PRINT #255:HEX$("2BD10705FF00000000")
```

For any other HEX$ values, refer to "Printer spacing and line control" under "PRINT statement" in your *BASIC Language Reference* manual.

Using NEWPAGE (continued)

OK? Back to the program.

List your program:

```
LIST
```

```
00010 PRINT NEWPAGE,"ENTER PRINCIPAL"
00020 INPUT P
00030 PRINT "ENTER INTEREST RATE"
00040 INPUT I
00050 PRINT "ENTER NUMBER OF YEARS"
00060 INPUT N
00070 LET A=P*(1+I)**N
00072 PRINT #255:HEX$("2B0205000A1042")
00074 PRINT #255:NEWPAGE
00080 PRINT #255:"PRINCIPAL = ";P
00090 PRINT #255:"INTEREST RATE = ";I
00100 PRINT #255:"NUMBER OF YEARS = ";N
00110 PRINT #255:"SAVINGS = ";A
00120 END
_
```

Make sure the paper is set at the top of the form in the printer, and then run the program:

```
RUN
```

```
RUN
```

Again, the PRINT NEWPAGE in line 10 clears the screen. Now, enter the following values for P, I, and N:

```
ENTER PRINCIPAL
?1000
ENTER INTEREST RATE
?.12
ENTER NUMBER OF YEARS
?5
```

```
PRINCIPAL =  1000
INTEREST =  .12
NUMBER OF YEARS =  5
SAVINGS =  1762.3416832
```

The output from the program is printed on a new page.

We did not need to include line 72 in our program to use NEWPAGE. However, if your printer is adjusted for some other spacing, the NEWPAGE may advance your paper to some place other than the top. And, if you later manually move the paper in the printer, your paper may not advance correctly with NEWPAGE.

Chapter summary

Output is the results of a program, which we display on the screen or print with the printer.

You can use TAB with PRINT or PRINT #255: to tell your System/23 in which position to start the output on the screen or printer. It works like this:

```
10 PRINT X;TAB(10);Y
or
10 PRINT #255:X;TAB(10);Y
```

You can skip a line on your screen or printer by entering:

```
10 PRINT
or
10 PRINT #255:
```

You can clear the screen or advance to a new page on the printer by entering:

```
10 PRINT NEWPAGE
or
10 PRINT #255:HEX$("2B0205000A1042")
20 PRINT #255:NEWPAGE
```

Line 10 is only necessary if you want to reset the printer spacing from some other value.

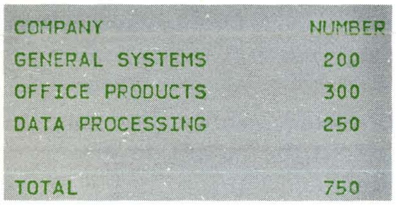
NEWPAGE is useful when you want to separate the input from the output of a program.

You can use BELL with PRINT to ring the alarm on your System/23.

The PAUSE statement interrupts a program. To resume the program, you enter GO.

Skipping spaces, lines, and pages

Exercises



```
COMPANY                 NUMBER
GENERAL SYSTEMS           200
OFFICE PRODUCTS           300
DATA PROCESSING           250

TOTAL                     750
```

Question 1

Using PRINT and END statements, write a program that will cause the screen shown at the left to be displayed.

Answer:

```
NAME                  SCORE
JACKSON                489
JOHNSON                523
JONES                  408
JUSTEN                 549

TEAM                   1969
```

Question 2

Using PRINT #255: and END statements, write a program that will cause the page shown to the left to be printed.

Answer:

Question 3

Add the necessary statements to the following program to do the following tasks:

- Ring the alarm as a signal to adjust the printer.
- Interrupt the program to allow you to adjust the printer.
- Set the printer for 10 characters per inch; 6 lines per inch, and 66 lines per page.
- Print the output on a new page.

```
10 DATA 25,50,75
20 READ NUTS,BOLTS,SCREWS
30 PRINT #255:"NUTS = ";NUTS
40 PRINT #255:"BOLTS = ";BOLTS
50 PRINT #255:"SCREWS = ";SCREWS
60 TOTAL = NUTS+BOLTS+SCREWS
70 PRINT #255:"TOTAL = ";TOTAL
80 END
```

Answer: ______________________________

Skipping spaces, lines, and pages

Answers

Question 1

```
10 PRINT "COMPANY";TAB(25);"NUMBER"
20 PRINT
30 PRINT "GENERAL SYSTEMS";TAB(25);200
40 PRINT "OFFICE PRODUCTS";TAB(25);300
50 PRINT "DATA PROCESSING";TAB(25);250
60 PRINT
70 PRINT "TOTAL";TAB(25);750
80 END
```

Question 2

```
10 PRINT #255:"NAME";TAB(27);"SCORE"
20 PRINT #255:
30 PRINT #255:"JACKSON";TAB(27);489
40 PRINT #255:"JOHNSON";TAB(27);523
50 PRINT #255:"JONES";TAB(27);408
60 PRINT #255:"JUSTEN";TAB(27);549
70 PRINT #255:
80 PRINT #255:"TEAM";TAB(27);1969
90 END
```

Question 3

```
22 PRINT BELL
24 PAUSE
26 PRINT #255:HEX$("2B0205000A1042")
28 PRINT #255:NEWPAGE
```

To resume this program, you would have to enter GO after the pause. Line 26, which indicates standard printer spacing, is optional. Line 28 is necessary to print the output on a new page.

Chapter 3. PRINT USING and FORM

Introduction

In this chapter, you will learn more ways to format the screen and printed pages. With the PRINT USING and FORM statements, you can specify the way you want data to appear.

Objectives

Upon completion of this chapter, you should be able to do the following:

- Specify the format of data to be displayed or printed by using C, N, and PIC in a FORM statement.
- Indicate the position of data to be displayed or printed by using POS, X, and SKIP in a FORM statement.

If you are familiar with these tasks, try the exercises located at the end of this chapter. If not, read through the chapter before going on to the exercises.

PRINT USING and FORM

Setting up a format

You can display or print information in a particular format with the PRINT USING statement. You indicate the format in a separate BASIC statement called FORM.

The PRINT USING and FORM statements always work together.

Here's an example of a program that includes PRINT USING and FORM. Enter:

```
CLEAR
10 COST=100
20 PRINT USING 30:COST
30 FORM N 6
40 END
```

Line 20 tells your System/23 to display the value of COST by using the format specified in line 30.

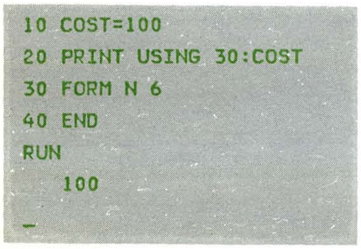
```
20 PRINT USING 30:COST

Line number of      Required(:)
the FORM statement
```

The FORM statement in line 30 tells your System/23 how you want the value to be displayed. FORM N 6 indicates that you are displaying a *number*, with up to *six* digits. The six digits include a minus sign, if present.

```
30 FORM N 6

Space is required
```

Go ahead and run this program:

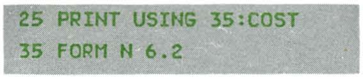
```
RUN
```



```
10 COST=100
20 PRINT USING 30:COST
30 FORM N 6
40 END
RUN
    100

_
```

As you can see, the value 100 is displayed in positions 1 through 6, with the first three positions blank.

Add these statements to your program:

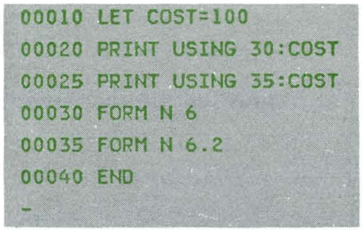


```
25 PRINT USING 35:COST
35 FORM N 6.2
```

```
25 PRINT USING 35:COST
35 FORM N 6.2
```

List the new version of your program:

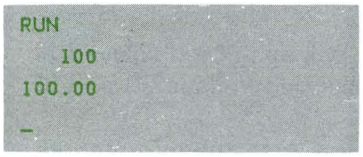
```
LIST
```



```
00010 LET COST=100
00020 PRINT USING 30:COST
00025 PRINT USING 35:COST
00030 FORM N 6
00035 FORM N 6.2
00040 END
_
```

Now run the new version of your program:

```
RUN
```



```
RUN
   100
100.00

_
```

In this program, the value of COST is displayed twice. FORM N 6 displays the value 100 in positions 1 through 6, with the first three positions blank.

FORM N 6.2 displays the value 100.00 in positions 1 through 6, allowing two digits to the right of the decimal point.

The number following N tells how many digits can be displayed. If you are displaying a negative number, the minus sign is counted as one of the digits.

Remember:

FORM N 6 allows six digits, with no decimal point.

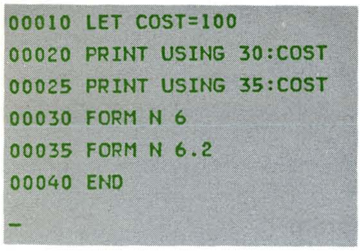
FORM N 6.2 allows five digits plus a decimal point, with two digits to the right of the decimal point.

PRINT USING and FORM

Setting up a format (continued)

List the program in the work area again:

```
LIST
```



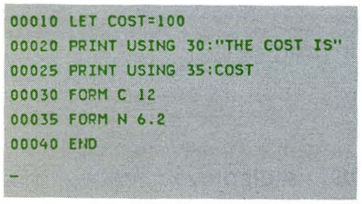
```
00010 LET COST=100
00020 PRINT USING 30:COST
00025 PRINT USING 35:COST
00030 FORM N 6
00035 FORM N 6.2
00040 END
_
```

Now make the following changes:

```
20 PRINT USING 30:"THE COST IS"
30 FORM C 12
```

Now list the new version:

```
LIST
```



```
00010 LET COST=100
00020 PRINT USING 30:"THE COST IS"
00025 PRINT USING 35:COST
00030 FORM C 12
00035 FORM N 6.2
00040 END
_
```

In this program, two pieces of information will be displayed:

- The character string THE COST IS
- The value of COST, which is assigned the value of 100.

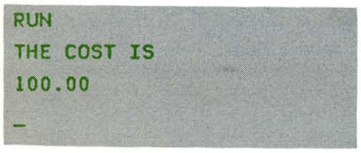
These two pieces of information will each be displayed on separate lines, in the formats described in lines 30 and 35.

The FORM statements tell how you want the information displayed.

The FORM C 12 means that a character string should be displayed, and that 12 positions should be reserved for the string. This FORM statement is used for the character string THE COST IS.

Run this program:

```
RUN
```



```
RUN
THE COST IS
100.00
_
```

Now let's change this program so that both pieces of information are displayed on the same line.

Enter the following:

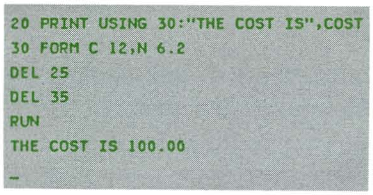
```
20 PRINT USING 30:"THE COST IS",COST
30 FORM C 12,N 6.2
DEL 25
DEL 35
```

Notice the comma in line 20.

```
20 PRINT USING 30:"THE COST IS",COST
```

This comma is used only to separate the items to be displayed, *not* to specify print zones. Print zones are used only with PRINT, not with PRINT USING.

Run the new version of your program:



```
20 PRINT USING 30:"THE COST IS",COST
30 FORM C 12,N 6.2
DEL 25
DEL 35
RUN
THE COST IS 100.00

-
```

```
RUN
```

PRINT USING and FORM must be used together, but the FORM statement can appear anywhere in the program. Note that you cannot include a remark on a FORM statement.

The formats listed in the FORM statement must be in the same order as the output items in the PRINT USING statement.

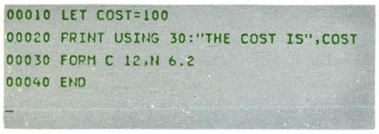
```
20 PRINT USING 30:"THE COST IS",COST

30 FORM C 12,N 6.2
```

The C 12 leaves one blank, because "THE COST IS" has only 11 characters.

PRINT USING and FORM

Setting up a format (continued)

List the program in the work area:



```
00010 LET COST=100
00020 PRINT USING 30:"THE COST IS",COST
00030 FORM C 12,N 6.2
00040 END
_
```

```
LIST
```

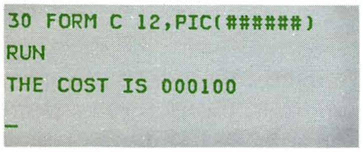
Now, change line 30 like this:

```
30 FORM C 12,PIC(######)
```

In a FORM statement, PIC shows a *picture* of the way a value should be displayed. The six pound signs (######) mean that six digits will be displayed. The value of cost is only three digits long, so three leading zeros will be displayed:

$$\begin{array}{l} 100 = \underset{\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow}{000100} \\ \text{PIC(\#\#\#\#\#\#)} \end{array}$$

Go ahead and run the program:



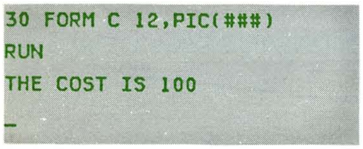
```
30 FORM C 12,PIC(######)
RUN
THE COST IS 000100
_
```

```
RUN
```

If there were three pound signs after the PIC, no leading zeros would be displayed. Change line 30 like this:

```
30 FORM C 12,PIC(###)
```

And run the program again:



```
30 FORM C 12,PIC(###)
RUN
THE COST IS 100
_
```

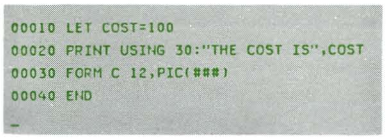
```
RUN
```

You can include decimal points in your PIC specification, too. With a decimal point and two additional pound signs, you can represent the value of COST in terms of dollars and cents.

As we told you earlier, when you change a line in a program, you do not need to reenter the entire line.

Instead, you can scroll down to the line you want to change, and move the cursor to the part that's changing.

Let's do that. List the program in the work area:

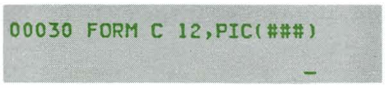


```
00010 LET COST=100
00020 PRINT USING 30:"THE COST IS",COST
00030 FORM C 12,PIC(###)
00040 END
_
```

```
LIST
```

Now, press the Scroll down key two times to get back to line 30.

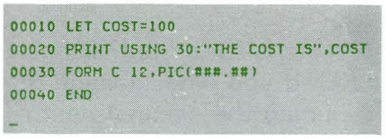
Press the Cursor right key until the cursor is under the end parenthesis. Now, enter:



```
00030 FORM C 12,PIC(###)
                       _
```

```
.##)
```

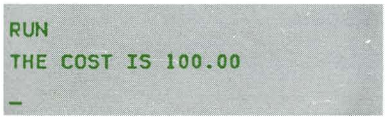
Now list the program:



```
00010 LET COST=100
00020 PRINT USING 30:"THE COST IS",COST
00030 FORM C 12,PIC(###.##)
00040 END
_
```

```
LIST
```

As you can see, the line was replaced. Now, run the program:



```
RUN
THE COST IS 100.00
_
```

```
RUN
```

Because 100 does not have a decimal portion, PIC(###.##) causes 100.00 to be displayed.

The value of COST is formatted exactly as described in the PIC.

```
100=100.00
    ↑↑↑ ↑↑
PIC(###.##)
```

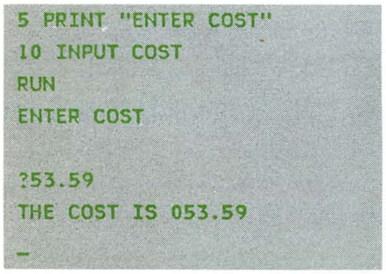
Setting up a format (continued)

Let's change the program so that you assign the value of COST with an INPUT statement. Remember that you should use prompts with INPUT statements. Enter:

```
5 PRINT "ENTER COST"
10 INPUT COST
```

Now list the program:

```
LIST
```



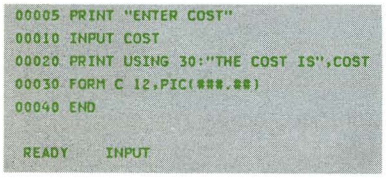
```
5 PRINT "ENTER COST"
10 INPUT COST
RUN
ENTER COST

?53.59
THE COST IS 053.59
_
```

Now, run the program:

```
RUN
```

When the question mark appears, enter 53.59 for COST:

```
ENTER COST
?53.59
```



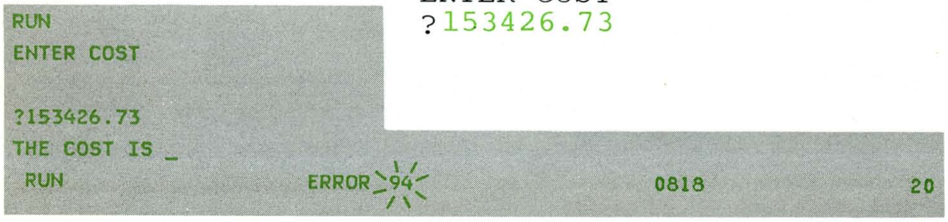
```
00005 PRINT "ENTER COST"
00010 INPUT COST
00020 PRINT USING 30:"THE COST IS",COST
00030 FORM C 12,PIC(###.##)
00040 END

 READY    INPUT
```

Look at the screen. See how the value is formatted? It has a leading zero to fit the PIC(###.##). If you enter a value of more than six digits for COST (including the decimal point), you will get an error when line 20 is executed. Let's see what happens. Run the program again:

```
RUN
```

When the question mark appears, enter:

```
ENTER COST
?153426.73
```



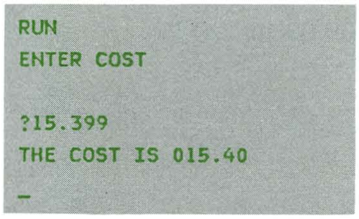
```
RUN
ENTER COST

?153426.73
THE COST IS _
 RUN                    ERROR 94                    0818                    20
```

To stop the flashing action code, press the Error Reset key. Then, enter:

```
GO END
```

Now run the program again and enter the following value for COST:

```
RUN
ENTER COST
?15.399
```



```
RUN
ENTER COST

?15.399
THE COST IS 015.40
_
```

You entered three digits to the right of the decimal point. The PIC(###.##) only shows two digits to the right of the decimal point, so your System/23 rounds to .40.

$$015.\underbrace{399}$$

$$015.40$$

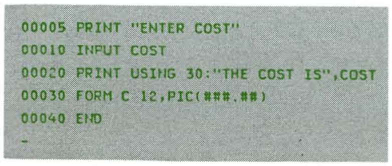
Suppose you don't want leading zeros to be displayed. Instead of using the pound sign (#) in PIC, you can use the letter Z. The letter Z means replace any leading zeros with blanks.

Let's change line 30 again to show this:

```
30 FORM C 12,PIC(ZZZZZ.##)
```

List the new version of your program:

```
LIST
```



```
00005 PRINT "ENTER COST"
00010 INPUT COST
00020 PRINT USING 30:"THE COST IS",COST
00030 FORM C 12,PIC(###.##)
00040 END
-
```

Setting up a format (continued)

Now, when you enter a value for COST, no leading zeros will be displayed. The digits to the right of the decimal point *will* be displayed, because PIC(ZZZZZ.##) shows pound signs in those positions.

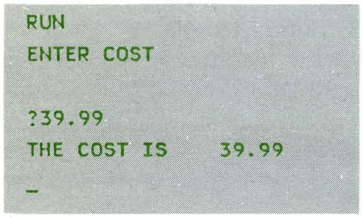
```
PIC(ZZZZZ.##)
```

Any leading zeros are not displayed.

Two digits to the right of the decimal point are displayed.

Now, run the program and enter 39.99 for COST:

```
RUN
ENTER COST
? 39.99
```

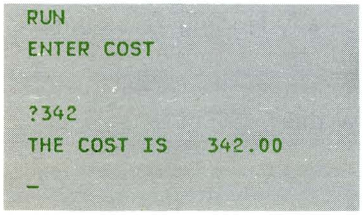


```
RUN
ENTER COST

?39.99
THE COST IS     39.99

_
```

No leading zeros are displayed.

Run the program again and enter 342 for COST:

```
RUN
ENTER COST
? 342
```



```
RUN
ENTER COST

?342
THE COST IS    342.00

_
```

No leading zeros are displayed. The zeros to the right of the decimal point *are* displayed. PIC(ZZZZZ.##) shows pound signs in those positions.

You can also include a dollar sign ($) in your PIC specification. Let's do that. Enter:

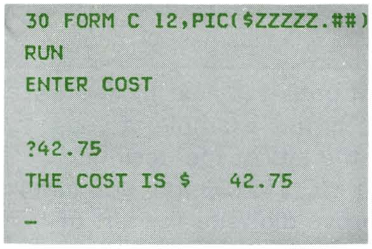
```
30 FORM C 12,PIC($ZZZZZ.##)
```

Run the program:

```
RUN
```

Now, enter a value of 42.75 for COST:

```
ENTER COST
?42.75
```



```
30 FORM C 12,PIC($ZZZZZ.##)
RUN
ENTER COST

?42.75
THE COST IS $    42.75
_
```

The value is displayed exactly as described in the PIC:

$$\begin{array}{l}\texttt{\$ZZZZZ.\#\#}\\ \quad\quad\ \downarrow\downarrow\ \downarrow\downarrow\\ \texttt{\$\ \ \ \ 42.75}\end{array}$$

Now, suppose you want the dollar sign displayed right next to the value, like this:

```
$42.75
```

You can do this by including more than one dollar sign in the PIC.

```
PIC($$$$#.##)
```

When you include more than one dollar sign, the dollar sign will immediately precede the first digit displayed. Or, the dollar sign will appear in the position of the rightmost $ in the PIC.

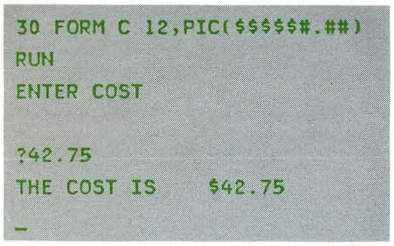
Enter:

```
30 FORM C 12,PIC($$$$$#.##)
```

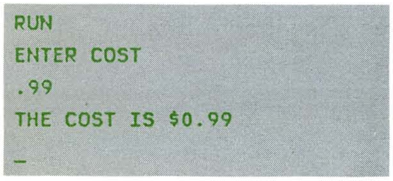
Now, run the program and enter 42.75 for COST:

```
RUN
```

Setting up a format (continued)



```
30 FORM C 12,PIC($$$$$#.##)
RUN
ENTER COST

?42.75
THE COST IS     $42.75

_
```



```
RUN
ENTER COST
.99
THE COST IS $0.99

_
```

```
ENTER COST
?42.75
```

Run the program again, and enter:

```
RUN
ENTER COST
?.99
```

The dollar sign moves to the right until it is next to the first digit when the value is displayed. In this example, if you enter a number with no digits to the left of the decimal point, a 0 is displayed. If you use PIC($$$$$#.##), you can't display a number with more than five digits to the left of the decimal point.

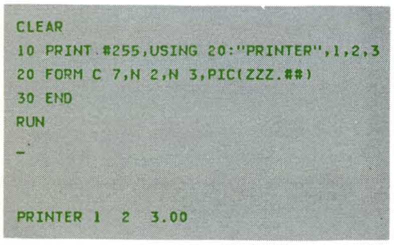
Note: All of our examples of PRINT USING with FORM have applied to displaying data on the screen. However, PRINT #255: can be used with FORM just like PRINT.

If you have a printer, enter the following:

```
CLEAR
10 PRINT #255,USING 20:"PRINTER",1,2,3
20 FORM C 7,N 2,N 3,PIC(ZZZ.##)
30 END
```

And run the program:

```
RUN
```



```
CLEAR
10 PRINT #255,USING 20:"PRINTER",1,2,3
20 FORM C 7,N 2,N 3,PIC(ZZZ.##)
30 END
RUN
_


PRINTER 1  2  3.00
```

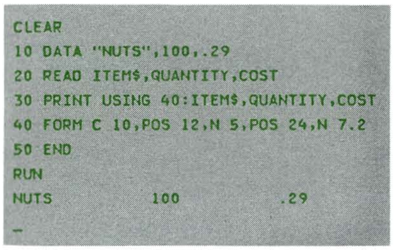
Notice that a comma is required between the #255 and the USING in line 10. The colon comes after USING 20, not after #255.

So far, you have used C, N, and PIC in the FORM statement to indicate the type of data to be displayed or printed. You can also control *where* data is to be displayed or printed by using POS, X, and SKIP.

With POS, you indicate the position of the next value you want displayed. Enter this program:

```
CLEAR
10 DATA "NUTS",100,.29
20 READ ITEM$,QUANTITY,COST
30 PRINT USING 40:ITEM$,QUANTITY,COST
40 FORM C 10,POS 12,N 5,POS 24,N 7.2
50 END
```

Run the program:



```
CLEAR
10 DATA "NUTS",100,.29
20 READ ITEM$,QUANTITY,COST
30 PRINT USING 40:ITEM$,QUANTITY,COST
40 FORM C 10,POS 12,N 5,POS 24,N 7.2
50 END
RUN
NUTS         100           .29
_
```

```
RUN
```

The C 10 reserves positions one through 10 for the variable ITEM$. The value of QUANTITY begins in position 12. However, the first two characters may be blanks.

The POS 24 means that the value of the variable COST should be displayed beginning with position 24. But notice that the number .29 actually starts in position 28, because the first four characters are blank.

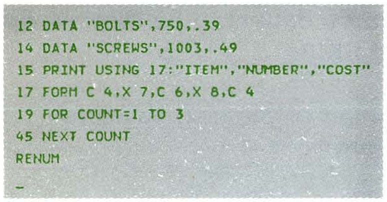
```
30 PRINT USING 40:ITEM$,QUANTITY,COST

40 FORM C 10,POS 12,N 5,POS 24,N 7.2
```

POS is similar to TAB, which you learned about in the last chapter. Remember that POS is used within a FORM statement. TAB is used directly within a PRINT statement.

PRINT USING and FORM

Setting up a format (continued)

Now let's change our program. Enter the following:



```
12 DATA "BOLTS",750,.39
14 DATA "SCREWS",1003,.49
15 PRINT USING 17:"ITEM","NUMBER","COST"
17 FORM C 4,X 7,C 6,X 8,C 4
19 FOR COUNT=1 TO 3
45 NEXT COUNT
RENUM
-
```

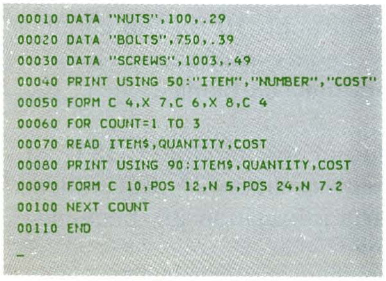
```
12 DATA "BOLTS",750,.39
14 DATA "SCREWS",1003,.49
15 PRINT USING 17:"ITEM","NUMBER","COST"
17 FORM C 4,X 7,C 6,X 8,C 4
19 FOR COUNT=1 TO 3
45 NEXT COUNT
```

These line numbers may become difficult to work with. Let's renumber the program. Do you remember how? Enter:

```
RENUM
```

Now list the program:

```
LIST
```



```
00010 DATA "NUTS",100,.29
00020 DATA "BOLTS",750,.39
00030 DATA "SCREWS",1003,.49
00040 PRINT USING 50:"ITEM","NUMBER","COST"
00050 FORM C 4,X 7,C 6,X 8,C 4
00060 FOR COUNT=1 TO 3
00070 READ ITEM$,QUANTITY,COST
00080 PRINT USING 90:ITEM$,QUANTITY,COST
00090 FORM C 10,POS 12,N 5,POS 24,N 7.2
00100 NEXT COUNT
00110 END
-
```

Notice that line 40 now reads PRINT USING 50, because line 17 was changed to line 50. Now look at line 50:

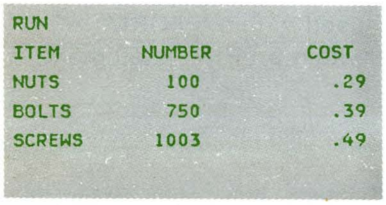
```
50 FORM C 4,X 7,C 6,X 8,C 4
```

The X 7 tells your System/23 to leave seven blanks between ITEM and NUMBER. The X 8 tells your System/23 to leave eight blanks between NUMBER and COST.

```
FORM X  7
    ↗   ↖
Leave     Seven blanks
```

Now, run the program:

```
RUN
```



```
RUN
ITEM       NUMBER       COST
NUTS          100        .29
BOLTS         750        .39
SCREWS       1003        .49
```

Now, let's expand the program to show one more thing you can do with the FORM statement. Enter:

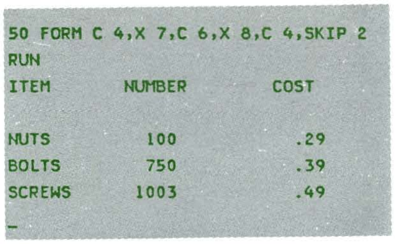
```
50 FORM C 4,X 7,C 6,X 8,C 4,SKIP 2
```

You have changed the program to skip a line between the heading and the rest of the output.

Notice that the last FORM listed in line 50 is SKIP 2.

The SKIP 2 in line 50 tells your System/23 to go to the next line and then skip a line.

Run the program:



```
50 FORM C 4,X 7,C 6,X 8,C 4,SKIP 2
RUN
ITEM       NUMBER        COST

NUTS         100          .29
BOLTS        750          .39
SCREWS      1003          .49
_
```

```
RUN
```

See how the report is formatted?

SKIP in the form statement tells your System/23 to skip lines between output. The number of lines skipped (left blank) is usually one less than the number you enter. You don't have to tell the system to go down just one line; it will do that by itself. (That is called a Skip 1.) For example:

SKIP 2 skips one line

SKIP 8 skips seven lines

But, SKIP 0 allows output on the same line.

Remember: The PRINT USING #255: and FORM statements can be used to format output on the printer.

Other data formats are available. Refer to ''FORM statement'' in the *BASIC Language Reference* manual for a complete list.

PRINT USING and FORM

Chapter summary

The PRINT USING statement tells your System/23 to *display* information according to the format described in a FORM statement.

The PRINT #255,USING statement tells your System/23 to *print* information according to the format described in a FORM statement.

The FORM statement describes the format for displaying or printing output. The format is controlled by:

- C - a character string
- N - a number
- PIC - a picture of the format
- POS - position on the line
- X - leave a blank
- SKIP - skip a line

Exercises

Question 1

What will appear on the screen if the number 34567 is displayed with the following FORM statements?

```
a.  FORM N 10
b.  FORM N 10.2
c.  FORM PIC(######)
d.  FORM PIC(ZZZZZ#)
e.  FORM PIC($#####.##)
f.  FORM PIC($$$###.##)
g.  FORM PIC(ZZZ#)
```

Answer: a. ______________________

b. ______________________

c. ______________________

d. ______________________

e. ______________________

f. ______________________

g. ______________________

Question 2

What will be displayed on the screen when the following program is run? (Use the dotted line to indicate positions.)

```
10 DATA 1001,"HAMMERS",105,9.99
20 READ N,D$,Q,C
30 PRINT USING 40:N,D$,Q,C
40 FORM N 4,X 3,C 8,X 2,PIC(ZZ#),N 8.2
50 END
```

Answer:

```
---------------------------------------
```

Exercises (continued)

Question 3

What will be printed when the following program is run? (Use the dotted lines to indicate positions.)

```
10 PRINT #255,USING 20:"NAME","SCORE"
20 FORM C 4,X 10,C 5,SKIP 2
30 PRINT #255,USING 40:"SMITH",285
40 FORM C 5,POS 15,PIC(###)
50 END
```

Answer:

```
----------------------------------------
----------------------------------------
----------------------------------------
```

Answers

Question 1

```
a.        34567

b.     34567.00

c. 034567

d.  34567

e. $34567.00

f. $34567.00
```

g. An error will occur. Too few digits are included in the PIC.

Question 2

```
1001    HAMMERS    105      9.99
---------------------------------------
```

Question 3

```
NAME             SCORE
----------------------------------------

----------------------------------------
SMITH            285
----------------------------------------
```

Cut or Fold Along Line

READER'S COMMENT FORM

SA34-0123-0

III. More Ways to Print

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or the IBM branch office serving your locality.

Corrections or clarifications needed:

Page Comment

Please indicate your name and address in the space below if you wish a reply.

____

____

____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

Cut Along Line

Reader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Systems Publications, Dept 27T
P.O. Box 1328
Boca Raton, Florida 33432

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

International Business Machines Corporation
General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150, Atlanta, Georgia 30055
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
(International)

SA34-0123-0
Printed in U.S.A.

International Business Machines Corporation
General Systems Division
4111 Northside Parkway, N.W.
P.O. Box 2150
Atlanta, Georgia 30055
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
U.S.A.
(International)

SA34-0123-0
Printed in U.S.A.