The assigned value of the numerical list represented by the variable will be as follows.

`A(0) = 7`	(there are seven characters in `A$`)
`A(1) = 76`	(the numerical equivalent of `L`)
`A(2) = 73`	(the numerical equivalent of `I`)
`A(3) = 66`	(the numerical equivalent of `B`)
`A(4) = 82`	(the numerical equivalent of `R`)
`A(5) = 65`	(the numerical equivalent of `A`)
`A(6) = 82`	(the numerical equivalent of `R`)
`A(7) = 89`	(the numerical equivalent of `Y`)

The numerical equivalent of any character in the string `A$` may be referred to by the corresponding subscripted variable A.

The inverse operation of a number list to the corresponding character list is also achieved by the `CHANGE` statement. The format of such a change is

```
CHANGE A TO A$
```

The variable A(0) stores the number of encoded numbers in the string list.

Conversion Using Library Functions

Another way of converting a single character to its `ASCII` numerical quantity, and vice versa, is to use the library functions `ASC` and `CHR$`.

The format of the statements are as follows.

```
A = ASC(L)      ; ASCII equivalent of L is 76. Hence A gets the value of 76
A$ = CHR$(X)    ; If X is 80, then the corresponding character is P. Hence, A$ represents P.
```

The above two library functions can be used in any type of statement (e.g. conditional branching, printing, etc.).

1.14 COMPUTER GRAPHICS

BASIC includes instructions to display the data in graphical form on the monitor.

Graphical Mode

The fundamental elements in the display of graphs are small dots, called pixels (picture elements). To bring the computer to the graphical mode, the statement to be given is

```
SCREEN 1   or   SCREEN 2
```

In `SCREEN 1`, known as medium-resolution graphics mode, there is a provision of 320 pixels in the horizontal direction and 200 pixels in the vertical direction. The position of a pixel on the monitor is governed by its coordinates. The scheme of coordinates is shown in Fig. 1.

In the scheme shown in Fig. 1, the top left-most corner of the monitor is assigned coordinates (0, 0). The bottom right-most corner of the monitor is assigned coordinates (319, 199).

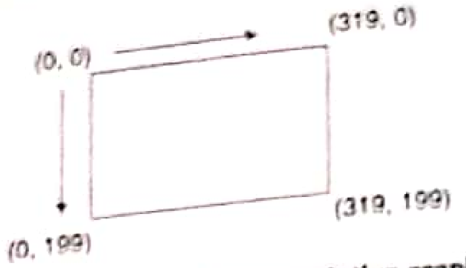


Fig. 1 Coordinate scheme in a medium-resolution graphics (SCREEN 1)

With this, the coordinates of top right-most and bottom left-most corners of the monitor have the coordinates (319, 0) and (0, 199), respectively. To display a dot at the desired position, the statement to be given is

```
PSET (X, Y)
```

where X and Y are the coordinates of the desired position of pixels on the monitor.

In SCREEN 2, known as high-resolution graphics mode, there is a provision of 640 pixels horizontally and 200 pixels vertically, with the coordinate system shown in Fig. 2.

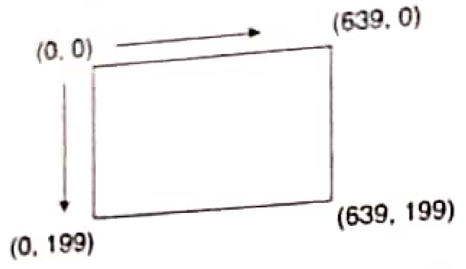


Fig. 2 Coordinate scheme in a high-resolution graphics (SCREEN 2).

Colour Scheme

The SCREEN 1 mode has a provision to display graphs in colour whereas in SCREEN 2, only black and white format is available. The choice of colour in SCREEN 1 is specified by the statement

```
COLOR X, Y
```

where X and Y are the two parameters which decide the background colour and the choice of palette, respectively. The parameter X can have any one value from 0 to 15, thus providing with a choice of 16 different colours as shown in Table 6.

Table 6 Colour variables in BASIC

Value of X	*Colour*	*Value of* X	*Colour*
0	black	8	gray
1	blue	9	light blue
2	green	10	light green
3	cyan	11	light cyan
4	red	12	light red
5	magneta	13	light magneta
6	brown	14	yellow
7	white	15	high intensity white

The parameter Y has a value of either 0 or 1. Thus, there are two schemes for the graphical display. These are shown in Table 7.

Table 7 Palletes used in Colour Scheme of BASIC

Pallete 0		*Pallete 1*	
Number	*Colour*	*Number*	*Colour*
0	background colour	0	background colour
1	green	1	cyan
2	red	2	magneta
3	brown	3	white

The choice of colour in either of the above two palettes is indicated along with the graphic statements `PSET`, `LINE` and `CIRCLE`. In the statement PSET, this number is indicated immediately after the closing parentheses with a comma in between. For example

```
PSET (X, Y), 1
```

If no number is mentioned, it is automatically taken to be 3.

Colour Scheme in Textual Mode

In textual mode (`SCREEN 0`), the COLOR statement involves three parameters specifying the foreground (text) colour, the background colour and the border colour. The COLOR statement is inserted immediately after the statement `SCREEN` which specifies text mode, with colour enabled. For example,

```
SCREEN 0
COLOR 14, 1, 4
```

These statements set yellow text against blue background with a red border.

The textual mode is a default mode when the execution of the program is started. The whole screen is divided into 25 rows and 80 columns.

LINE Statement

It is possible to draw a line between the two points on the screen. The format of the LINE statement is

```
LINE (X1, Y1)-(X2, Y2)
```

where `X1` and `Y1` are the respective horizontal and vertical pixel coordinates of the one point, and `X2` and `Y2` are those of the second point. The coordinates are enclosed in parentheses with a comma in between. The two points are separated by a dash. If the line is to be drawn in a particular colour in SCREEN 1 mode, the number from 0 to 3 of the palette chosen is mentioned immediately after the coordinates of the second point with a comma in between. For example

```
LINE (X1, Y1)-(X2, Y2), 1
```

If no number is mentioned, the computer assigns number 3 automatically.

If after drawing a line, a second line is to be drawn with the coordinates `X2` and `Y2` of the first line as the first point in the second line, this can be done by stating only the coordinates of the second point by the statement shown in the following.

```
LINE -(X3, Y3)
```

This form of the LINE statement is useful in drawing more than are interconnected lines.

It is also possible to draw a complete rectangle by a single LINE statement by adding two commas followed by inserting the symbol B as shown in the following.

```
LINE (X1, Y1)-(X2, Y2), , B
```

If this rectangle is to be drawn in a particular colour of the chosen palette in SCREEN 1 mode, the number from 0 to 3 is inserted in between the two commas such as shown in the following.

```
LINE (X1, Y1)-(X2, Y2), 1, B
```

If no number is mentioned, it is automatically taken to be number 3.

If the rectangle is to filled with the chosen colour, then the symbol BF is inserted instead of B. For example, the statement

```
LINE (X1, Y1)-(X2, Y2), 1, BF
```

generates a green rectangle against a choosen background colour if the palette 0 is mentioned in the COLOR statement.

CIRCLE Statement

It is possible to draw a circle of desired radius around a chosen point. The statement to be given is

```
CIRCLE (X1, Y1), Z, 1
```

where X1 and Y1 are the respective horizontal and vertical pixel-coordinates of the centre of the circle, Z is the radius mentioned as the number of pixels and the last number is the colour of the circle (from 0 to 3) from the palette (either 0 or 1) inserted in the COLOR statement. For example, in palette number 1, the statement

```
CIRCLE (160, 100), 60, 2
```

draw a circle around the point with horizontal and vertical pixel-coordinates equal to 160 and 100, respectively. The radius of the circle is 60 pixels and its colour is magneta. If the colour parameter is not inserted, the default value of 3 is automatically taken.

It is possible to draw an arc of a circle by stating a starting angle and an ending angle measured counterclockwise direction from the right half of the horizontal axis. These are inserted immediately after the parameter of colour separated by commas. Both the inserted angles are expressed in radians for example, the statements

```
SCREEN 1

COLOUR 14, 0

CIRCLE (160, 100), 60, 1, 0, 3.14
```

generates the upper half of a circle of radius 60 pixels around the point (160, 100) pixels in green colour.

If the angles are expressed in negative, these are interpreted as positive but with an additional effect of connecting the end points of the arc with the centre of the circle for example,

```
SCREEN 1

COLOR 14, 0

CIRCLE (160, 100), 60, 1, -3.14, -6.28
```

generates a bottom half of a circle of radius 60 pixels around the centre (160, 100) pixels in green colour with the end points joined to the centre (160, 100).

Besides drawing circles and arcs, the `CIRCLE` statement can also be used to draw ellipses and elliptical arcs. This is achieved by inserting a positive parameter immediately after the angles separated by a comma. The value of the parameter goes as follows.

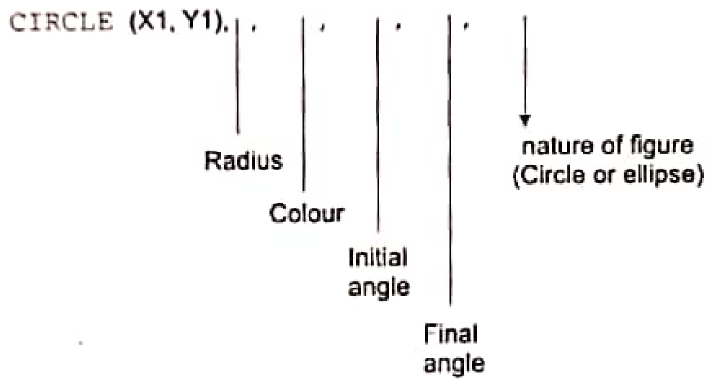
Value	*Figure*
1	nearly circular
greater than 1	vertical ellipse
less than 1	horizontal ellipse

The eccentricity of the ellipse depends on the value of this parameter. Larger its difference from unity, larger the eccentricity. The radius parameter is half of the major axis of the ellipse. For drawing a complete ellipse, either angles 0 and 2π are inserted or their locations are left blank without disturbing the placement of commas. For example, the following three statements produce the same figure.

```
CIRCLE (160, 100), 80, , , , 0.5
CIRCLE (160, 100), 80, 3, , , 0.5
CIRCLE (160, 100), 80, 3, 0, 6.28, 0.5
```

Summary of the CIRCLE Statement

The complete statement of `CIRCLE` is



PAINT Statement

A figure enclosed in a closed boundary can be filled with a colour matching with the colour of the boundary by using the statement

```
PAINT (X1, Y1), 2
```

where `X1` and `Y1` are the respective horizontal and vertical pixel-coordinates of a point within the closed boundary. The colour number should be the same as that of the closed boundary.

The colour parameter in the `PAINT` statement may be followed by one more parameter which explicitly indicates the colour of the boundary. For example, the statement

```
PAINT (X1, Y1), 3, 2
```

causes the figure with boundary colour 2 to be filled with colour number 3. This statement helps filling a desired figure if the point represented by the pixels `X1` and `Y1` lies within more than one figure.

VIEW Statement

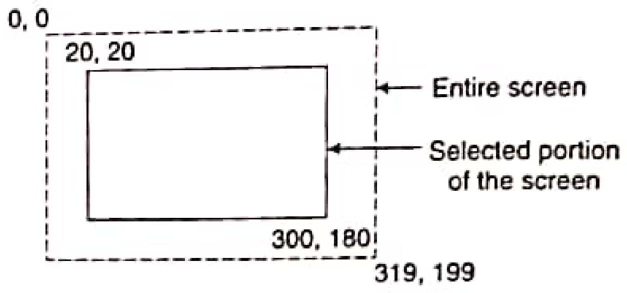
It is possible to choose a portion of the monitor within which the graphical display is to be restricted. This is achieved by the `VIEW` statement. Its format is

```
VIEW (X1, Y1)-(X2, Y2)
```

where `X1` and `Y1` are the respective horizontal and vertical pixel coordinates of the left most top corner of the selected portion of the screen and `X2` and `Y2` are those of the right most bottom corner of the selected portion. For example, the statement

```
VIEW (20, 20)-(300, 180)
```

causes the selection of the following portion of the monitor.



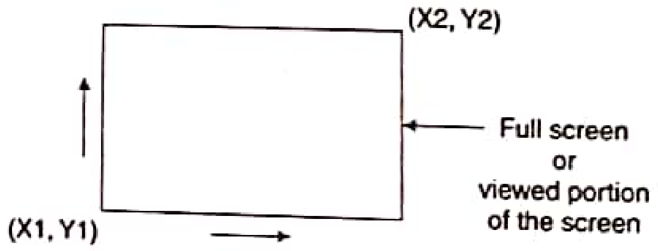
By selecting the appropriate portion of the screen, it is possible to display more than one graphical representation on the same screen.

WINDOW Statement

The `WINDOW` statement helps replacing the pixel numbering system by the new coordinate system suitable for displaying graph on the full screen or the chosen portion of the screen via `VIEW` statement. The format of the `WINDOW` statement is

```
WINDOW (X1, Y1)-(X2, Y2)
```

where `X1` and `Y1` are the respective horizontal and vertical coordinates of the bottom-left corner of the screen or view portion, and `X2` and `Y2` are those of the right-top corner. These are shown in the following.



With this coordinate system, the horizontal variation is from `X1` to `X2` and the vertical variation is from `Y1` to `Y2`. The coordinates of the left-top and bottom-right corners are `(X1, Y2)` and `(X2, Y1)`, respectively.

The `WINDOW` statement is normally inserted immediately after the `VIEW` statement. The graphic statements `PSET`, `LINE` and `CIRCLE` are governed by the coordinate system inserted in the `WINDOW` statement.

LOCATE Statement

It is possible to position the cursor in the desired location of the screen. This is achieved by the `LOCATE` statement, the format of which is

```
LOCATE X, Y
```

where `X` is the row number and `Y` is the column number. The entire screen has 25 rows (numbering starts from the top of screen) and 40 columns (numbering starts from the left of the screen).

For example, the statements

```
LOCATE 2, 10

PRINT "Plot of rate versus time"
```

cause the cursor to position in the 2nd row and 10th column. Immediately after this, the printing of the given string constant is executed as given in the next statement.

Using the `LOCATE` statement, the cursor can be placed at the desired position on the screen without disturbing any text previously written on the screen.