Intermediate Graphics

About this Documentation.

QuickCalc BASIC is designed to let you **produce graphs** of data which you calculate and analyze in your BASIC program, and to do so **easily** and **quickly**.

These goals are in conflict with "powerful" and "flexible". If you are just starting your first graph with QuickCalc, presenting you with all the options and capabilities of QuickCalc graphics could be overwhelming.

The documentation is therefore divided into **Beginning**, **Intermediate**, and **Advanced** Graphics.

Beginning Graphics (Read this one first)

Introduces you to the concepts and basic functions of QuickCalc graphics.
Allows you to produce a graph of your data with just a few statements.
Presents a **subset** of the features available, to keep it simple.

Intermediate Graphics

Shows how to tailor your graph to make it look exactly like you want it.
Describes the **SHAPE** statement to draw figures on your graph.
Describes the various ways to put **TEXT** on your graph.
Gives complete descriptions of the **GRAPH**, **SHAPE**, and **TEXT** statements.
Describes how to create and use Logarithmic and **DATETIME** scales.
Tells you how to specify colors for your graphic objects.

Advanced Graphics

Includes more advanced functions you do not normally need:

- Polylines
- Bezier Curves
- Smoothing and Averaging Plots, Polylines, and Polygons
- Complex Connected Shapes
- Outline Text Characters
- Selecting Shapes with the Mouse
- Deleting Shapes
- Changing Previously-drawn Shapes

So it is best to read these documents in order. First find out what you need to know to do the job at hand, then later read the more advanced (and more interesting) things that QuickCalc Basic Graphics can do.

Introduction to Intermediate Graphics.

Note: It is assumed that you have already read "Beginning Graphics", or have decided to skip it and dive right into the whole list of what QuickCalc BASIC Graphics can do. Good for you!

This document repeats (and expands on) many of the topics discussed in "Beginning Graphics". This document is intended to be the full set of functions (except for the ones in "Advanced Graphics").

The goal in QuickCalc Graphics is to produce graphs of data which you calculate and analyze in your BASIC program, and to do so **easily** and **quickly**. BASIC is a simple programming language, so adding graphing functionality to BASIC should be simple as well. However, just as QuickCalc BASIC adds a lot more power to the BASIC language, QuickCalc Graphics adds a lot more functionality than just simple graphs.

This document describes the ways in which you can create professional-looking graphs which can be included in a lab report, term paper, or presentation.

Note: Graphing is not supported in Console (DOS) mode.

Note: QuickCalc BASIC does not do "motion graphics".
(However, "Advanced Graphics" discusses some pseudo-motion effects).

Features

- You can produce a graph with just a few statements.

- You can see your graph immediately and then fine-tune it later, if desired.

- Most options default to reasonable values.

- QuickCalc graphs feature **automatic scaling** (which you can override). This automatically scales the graph so that all your data fits on the graph. Automatic scaling will dynamically re-scale the graph as new points or shapes are added.

- QuickCalc plots the graph **on the screen**, in a window that you can move and re-size, maximize or minimize. The on-screen graph shows all the data plotted in real time. The graph remains on the screen after the BASIC program terminates, so you can adjust its parameters and/or print it.

- You can plot data on **logarithmic scales**, if desired (See "Logarithmic Scales", below.)

- You can plot DATETIME values on a DATETIME scale.

- You can **put text on your graph** and **label the axes** (See "Putting Text On Your Graph", below.)

- You can **draw shapes** (filled or open) on your graph. Using this feature, you can draw pie charts, bar graphs, polygons, etc. (See "Drawing Shapes", below). Shapes may be **rotated** (see "Rotating Shapes", below).

Structure

QuickCalc BASIC implements Graphics with **four new statements**:

GRAPH
: Starts a graph and defines the physical attributes, scaling, axes, offsets, etc.

PLOT
: Plots collections of data values onto the graph, and describes the characteristics (line weight, color, etc.)

TEXT
: Puts text on the graph to identify points and curves, label axes, title the graph, etc. Specifies font, color, size, etc.

SHAPE
: Draws various shapes (rectangles, ellipses, etc.) and fills them. Specifies colors, line weight, location, etc.

All graphs that you can produce in QuickCalc BASIC are created using a combination of these statements.

Each of these statements and a complete list of its parameters are described below.

The Graph Window.

Graphs appear in a separate window. The Graph window is created when the **GRAPH** statement is executed.

The Graph window shows a representation of what your graph will look like when it is printed. All the statements you execute which affect your graph are reflected in the Graph window.

> The graph in the window reflects all the commands that have been processed up to that time; if you step through the program you can watch your graph being plotted.

You can re-size the Graph window, however the height and width are kept proportional. Maximizing the window will not necessarily full the screen; that depends on the height/width proportions. Maximized windows may still be moved around on the screen.

The **menu bar** on the graph window allows you to quickly do some of the graphics commands. If the option is already checked, selecting it again will turn it off.

FILE / PRINT	Prints the graph plotted so far.
FILE / METAFILE	Generates a metafile from your graph.
FILE / JPEG	Generates a JPEG file from your graph.
FILE / END	Ends the graph and closes the graph window.
FILE / PAUSE	Interrupts the BASIC program.
FILE / RESTART	Restarts the BASIC program (same as clicking on the "RUN" button). Program must be stopped for this to work. If program is in Stepping mode, it does a "Continue".
OPTIONS / GRID	Shows the grid lines.
OPTIONS / BORDER	Puts a border around your graph.
OPTIONS / AXES and NUMBERS	Displays x-y axes and numbers.
OPTIONS / MOREGRIDLINES	Increases the density of grid lines.
OPTIONS / FEWERGRIDLINES	Decreases the density of grid lines.
OPTIONS / SHOW COORDINATES	Displays the x-y coordinates of the spot on the graph to which you are pointing.
OPTIONS / CROSSHAIRS	Puts red crosshairs on the screen at the center of the physical page. Useful for centering items. Crosshairs do not print or appear in a JPEG or metafile.
ZOOM / IN or OUT	Changes the scale & offset by zooming in and out.
ZOOM / AUTO SCALE	Resets SCALE and OFFSET parameters to automatic scaling.

You can also re-position your graph (change its offsets) by clicking on the graph itself and **dragging** it. Changing the offsets this way turns off auto-scaling.

Note: Dragging the graph will not happen if the SHIFT or CONTROL key is held down. That changes the mouse function and is used for other things (See "*Advanced Graphics*").

Note: Dragging is disabled if you have specified GRAPH LOCKSCREEN=1.

Zooming and dragging the graph is limited to the range of valid DOUBLE floating-point numbers (or their logarithms).

Note: Zooming the graph changes the **scale** of the graph. It does ***not*** change the size of the printed graph. If you zoom in for a closer look, the new view is what will get printed.

One way to use the "zoom" function is to let auto-scaling scale the graph, and then zoom in or out to fine-tune the view.

Note: There are **shortcut keys** for the "zoom" functions. **Up-Arrow** zooms in 50%, **Down-Arrow** zooms out 50%. If you want to change the horizontal scale while leaving the vertical scale alone, you can use **Right-Arrow** to stretch the graph horizontally **and Left-Arrow** to compress it horizontally. These keys only work when the graph window has the focus. They are disabled if you specify **GRAPH LOCKSCREEN=1**.

Note: You can also use the **scroll wheel** on the mouse to zoom in and out. This is also disabled if you specify **GRAPH LOCKSCREEN=1**.

Note: When the graph window has the focus, you may return to the QuickCalc BASIC window by pressing the **TAB** key.

After zooming and/or dragging the graph, the scale factors and offsets will have changed. If you want to set these new scale factors into your program so the next time you run it you will get the current view, you can type **GRAPH LISTPARMS**, then code those values into your program

Showing Coordinates

When a graph is active, you can select
OPTION / SHOW COORDINATES from the drop-down menu. This causes a small window to appear displaying the coordinates on the graph to which the mouse is currently pointing. The coordinates will change as you move the mouse around. This can be useful if you want to estimate the high point on a curve, find where a function reaches a certain value, or find the intersection of two curves.

Note: The coordinates are shown as floating-point (DOUBLE) data values. They are shown with a lot of digits, however, the accuracy is typically ± 1 pixel. The graph on the screen is plotted to the nearest pixel, and the cursor can only be positioned to one pixel. Don't read too much into all those digits. Zoom in closer if you want more accuracy.

Note: Coordinates on DATETIME scales are formatted similarly to what you get from the **FORMATDATETIME$** function. The same note about accuracy still applies.

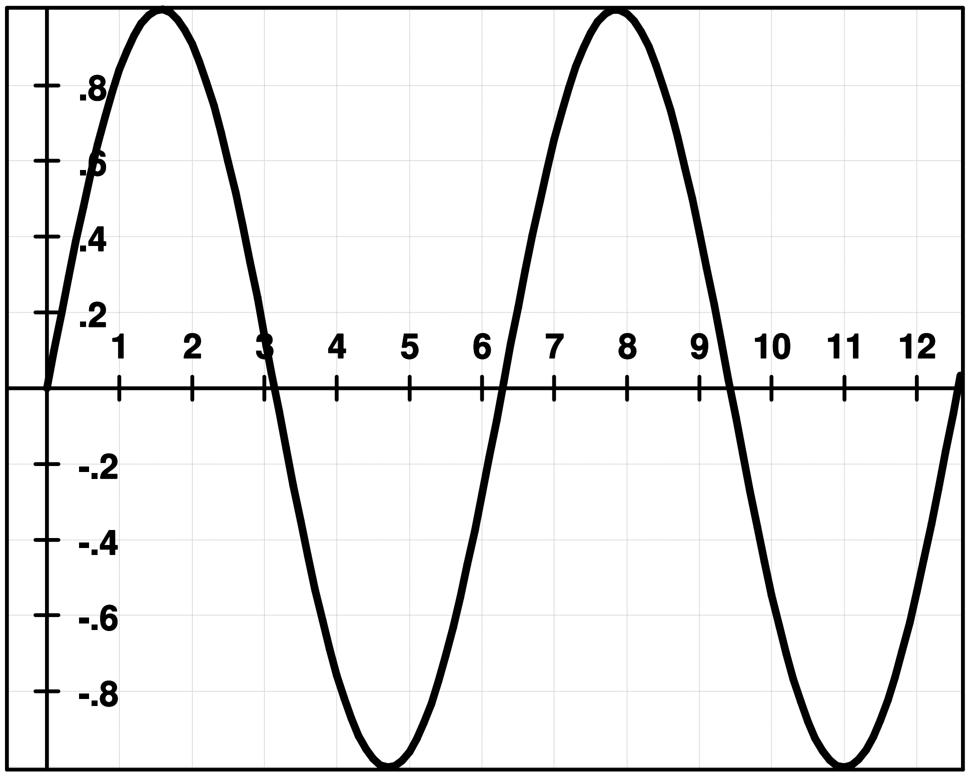
A Simple Graph Program

This example is shown here so that you can see how easy it is to produce a graph using just a few statements.

The following program will plot the sine of an angle over the range 0-720 degrees (0 to 4π radians):

```
GRAPH  WIDTH=5, HEIGHT=4
FOR angle = 0 TO 4 * PI + .1 STEP .1
  PLOT X=angle, Y=SIN (angle)
NEXT
```

This will plot the function on a graph 5” wide by 4” high, automatically scaled, using a 3-point black line, as shown below:



Note: In this example, the function was plotted incrementing by 0.1, which means that approximately 125 points were plotted. That is sufficient to produce a smooth curve at this resolution. At higher magnifications, you may want to plot more points.

Printing Your Graph.

You can print an active graph at any time with the statement

GRAPH PRINT

which may be typed into the command line or included in your program. You can also select **FILE / PRINT** from the Graph window, which is easier.

Note: The graph window will disappear from the screen until printing is complete or cancelled.

Note: You may experience some delay if you are on a network and your default printer is not currently on-line. Turning off your network adaptor when you are not connected will avoid this problem.

The graph will be printed using the default printer orientation (portrait or landscape) unless you override it in the printer properties dialog.

If your graph will not fit using the selected orientation, the program will attempt to print it in the other orientation. (A landscape (7.5 x 10) graph will not fit on the printer in portrait mode, so the program will switch the printer to landscape mode where the graph *will* fit, then print it.)

If the graph will not fit in either orientation, an error message will appear, and you can re-size your graph or select a different printer. There is no support for printing the graph on multiple pages, although some printer drivers offer a scale-to-page feature.

You can change the attributes of your graph and re-print it, if desired, as long as the graph is still active.

Creating a File Image (JPEG) of Your Graph.

You can generate a JPEG file (a graphics file image of your graph) by specifying

GRAPH JPEG

with no other parameters. (You can also select **FILE / JPEG** from the graph window menu bar.). A dialog will appear letting you select the output file, resolution (pixels per inch) and quality (compression factor).

You can then ***include the graph in another program***, like Microsoft Word, or import it into a picture editor program and edit it.

> **Note**: JPEG files may be viewed with the Windows Picture and Fax Viewer. In the Explorer window, right-click on the file and select "Preview". They may also be viewed with Internet Explorer or Mozilla Firefox.

> **Note**: WordPad doesn't have an "insert picture" function. If you want to import the JPEG file into a WordPad document, first view the file in Internet Explorer. "Copy" the picture to the clipboard. Then go into your WordPad document and "Paste" it wherever you want it

JPEG pictures are very **portable**. If you want to put the image onto the **internet,** JPEG is the best way, since the files are compressed (saving download time), and are compatible with all web browsers. JPEG files may also be taken to the camera store, drug store or copy center and printed from a thumb drive or CD, or emailed to friends, posted on Facebook, *etc*. JPEG images may be displayed on most smart phones.

JPEG images may also be edited with most picture editing software to crop, change contrast, brightness, color, etc.

Resolution and Quality

Resolution (dots per inch) determines the size of the JPEG image, and how finely you can resolve the image. A 10-inch wide graph at 100 dots per inch means the image will be 10 x 100 = 1000 dots wide.

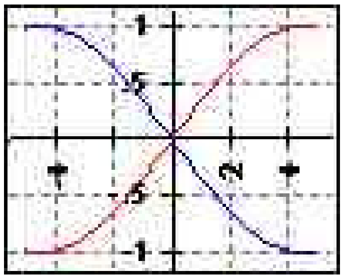
> **Note**: There are limits to how large you can make the image. For example, a 10 x 7.5-in graph maxes out at about 530 ppi, which makes a color bitmap of ***63 megabytes*** (before compression) This is on my computer and may be different on yours. If you get an error message, try a lower resolution.

If you intend to blow this image up a lot, a high resolution will give you a sharper image. A good rule-of-thumb is to use about 150 dots per inch on the printer – much more than that is wasted. If you are going to print a graph at 3-inches wide then you need a JPEG file which is 3 x 150 = 450 pixels wide. If your graph is 7.5 x 10 inches ("portrait"), then you need to make a JPEG file which is 7.5 x 150 = 1125 pixels wide by 10 x 150 = 1500 pixels high. This makes a color bitmap of around 5 megabytes (before compression).

For the internet, since the picture will probably be displayed on a screen, you can use about 90 pixels per inch as displayed on the screen. So a 3-inch wide picture on the screen requires about 3 x 90 = 270 pixels wide. If your graph page size is larger then the printed or displayed size, you can reduce the resolution so that you get the desired file size. For example, a 7.5 in. wide picture, displayed on a 90ppi screen at 3 in wide (on the screen) means the JPEG file should have a resolution of 36 pixels/in. (90 x 3 / 7.5 = 36). You can do these simple calculations for whatever size graph you are using.

These are just guidelines. You are, of course, free to use any resolution you want – experiment to see what you like best. Remember that high resolutions generate very large JPEG files.

<u>Quality</u> determines the amount of compression that is used. 100 is the best quality (least compression). Too much compression results in lower quality, which usually means fuzzier images with JPEG compression "artifacts" or garbage in the image, as shown below:



(Quality = 25)

If your JPEG image looks like this, use a higher quality setting, and perhaps a higher resolution as well.

Compression is a trade-off between picture quality and image size. JPEG images are not perfect, however with the proper quality setting, it is hard to tell. Usually a value of 80 produces acceptable results. Experiment and see for yourself.

<u>Creating a File Image (Metafile) of Your Graph.</u>

You can generate a "Windows Enhanced Metafile" (a graphics file image of your graph) by specifying

```
GRAPH METAFILE
```

with no other parameters. (You can also select **FILE / METAFILE** from the graph window menu bar.) A dialog will ask you where you want to save the file. It should have the suffix .EMF.

You can then ***<u>include the graph in another program</u>***, like Microsoft Word, or import it into a graphics design program (like Paint) and edit it. The "simple" graph, (above), as well as most other graphs in this document, were included using metafiles.

> **<u>Note</u>**: Metafiles may be viewed with the Windows Picture and Fax Viewer. In the Explorer window, right-click on the file and select "Preview".

Windows metafiles may be inserted directly into Word documents and give excellent results (no compression), and may be scaled inside Word, so you don't have to worry about resolution.

<u>Note</u>: Metafiles cannot be used on the internet, and can't be imported into many programs. They are great when they work, otherwise use JPEG.

<u>Note</u>: If you include a metafile into a Word document and later convert the Word document into a PDF file, that PDF can be put onto the internet. It will end up being viewed by Adobe Reader. This is how this document was prepared.

<u>Which is Better – METAFILE or JPEG ?</u>

First, read the notes in the previous two sections about the benefits and limitations of each.

Metafiles are resolved or rendered at the time they are printed or viewed. This means that the fonts, especially, are somewhat device-dependent. If the computer or printer does not have the exact font installed, the result could be not exactly the same as the original graph.

Sometimes this can improve your image, since the font that gets printed is rendered at print or display time, it can appear "crisper". However, it may not be exactly the right size or weight. This is especially noticeable at smaller point sizes.

On the other hand, JPEG files are resolved for a particular resolution. If the resolution you created is close to the final display resolution, it will give you the closest approximation to the original graph. Word or internet browsers will scale the image to fit the screen or printer, so small images may look blocky when enlarged. Also JPEG images can lose detail if compressed to much.

Try both. Import them into your document or web page and compare them. Print them on the same page and compare the results. Take the one you like better and delete the other one.

The GRAPH Statement

The **GRAPH** statement controls the size, shape, and scaling of the graph. It also starts the creation of the graph and can terminate the graph and print it.

Format

> **GRAPH** [*parameter* [, *parameter* …]]

You may have as many **GRAPH** statements as you want. You don't need to try to cram all the parameters onto one line.

If a parameter requires a value, *e.g.*, **HEIGHT**=*num-expression*, any valid numeric expression may be supplied, as long as it evaluates to a number with the correct range.

Starting a Graph

The **GRAPH** statement with no parameters **starts a graph** with all the defaults, if a graph is not already active. If a graph is currently active, this statement will generate an error.

The **GRAPH** statement with parameters may appear at any time, and will affect and over-ride the settings of the current graph without changing the data you have plotted so far.

Ending a Graph.

A graph remains active until

- you issue the statement **GRAPH END**,
- the BASIC program is restarted,
- another BASIC program begins executing,
- you close the graph window, or
- QuickCalc is terminated.

If your program creates a graph and leaves it active when the **BASIC** program terminates, you will then have the option of examining the graph on the screen to see if it is to your liking. If not, you can type in additional **GRAPH** commands to override the earlier settings and the displayed graph will change. You may also zoom, drag, and/or scroll the graph. When you are satisfied with the results, you may print it or create a JPEG or metafile. You may do the same thing if you place a **STOP** statement in your program, in which case you may look at and/or print your graph before it is complete.

It is usually **not** a good idea to include the **GRAPH PRINT** statement in your program, since it may print graphs before you have had a chance to review them, wasting time and paper. Similarly, if you include the **GRAPH END** statement in your program, the graph may be deleted before you get to see it. Use the **GRAPH END** statement if you need to erase this graph and then another **GRAPH** statement to start the next one.

Note: If you close the graph window before the graph is complete, you will get errors the next time you try to plot a point.

GRAPH Statement parameters.

GRAPH statement parameters override the default parameters and settings from previous **GRAPH** statements. You probably won't need to use most of them.

The statements are summarized here for reference. The meaning of each and examples of their use are described in the following sections.

AXES	Draws x-y axes and numbers. [default]
AXISCOLOR =[(*r, g, b*) | *num-expr*]	Set the color for the axes, tick marks, and numbers along the axes. Default is black.
AXISWEIGHT=*num-expression*	Sets line thickness of axes and borders, in points.
BGCOLOR=[(*r, g, b*) | *num-expr*]	Sets the background color. Default color is white. (See "Specifying Colors", below on how to specify colors.)
BORDER	Draws a border around your graph.
CLEARANCE=*num-expr*	Leaves a space of *num-expr* inches between the extremes of the plotted points and the border. *num-expr* may be 0 to .25. Default is 0.1.
END	Ends the graph and closes the graph window.
EQUALSCALES	Keeps Horizontal and Vertical scales the same.
GRID	Draws grid lines at the tick points [Default].
GRIDCOLOR=[(*r, g, b*) | *num-expr*]	Sets the grid color. Default color is gray. (See "Specifying Colors", below on how to specify colors.)
HEIGHT=*num-expression*	Sets printed graph height, in inches.
HMARGIN=*num-expression*	Sets horizontal margin, in inches.
HOFFSET=*num-expression*	Sets horizontal offset, in data units.

HSCALE=*num-expression*	Sets horizontal scale, in data units/in.
HSCALETYPE=[LOG | LINEAR | DATETIME]	Sets horizontal scale to logarithmic, linear or DATETIME scale.
INCLUDEXAXIS=[0 or **1**]	Autoscale includes [1] or does not include [0] the x-axis on the graph. [Default is 1]
INCLUDEYAXIS=[0 or **1**]	Autoscale includes [1] or does not include [0] the y-axis on the graph. [Default is 1]
JPEG	Generates a JPEG file from the graph.
LANDSCAPE	Sets **WIDTH**=10 in, **HEIGHT** = 7.5 in.
LISTPARMS	Lists all the **GRAPH** parameters currently in effect.
LOCKSCREEN=[**0** or 1]	Prevents you from accidentally re-scaling or re-positioning the graph. Useful when using mouse-click shapes or keystroke entry. When set to 1, it disables dragging the window with the mouse and zooming with the scroll wheel or arrow keys.
MAXIMIZE	Maximizes the graph window (same as the “maximize” button). This allows you to start the graph maximized. You should set the size of the graph before maximizing it.
METAFILE	Generates a metafile from the graph.
NOAXES	Suppresses x-y axes and numbers.
NOBORDER	Does not draw a border.
NOGRID	Suppresses drawing the grid.
NUMBERXAXIS=[0 or **1**]	If 0, suppresses numbers & ticks on X-axis.
NUMBERYAXIS=[0 or **1**]	If 0, suppresses numbers & ticks on Y-axis.
PORTRAIT	Sets **WIDTH**=7.5 in, **HEIGHT** = 10 in. [Default].
PRINT	Prints the graph.
VOFFSET=*num-expression*	Sets vertical offset, in data units.
VMARGIN=*num-expression*	Sets vertical margin, in inches.
VSCALE=*num-expression*	Sets vertical scale, in data units/in.
VSCALETYPE=[LOG | LINEAR | DATETIME]	Sets vertical scale to logarithmic, linear or DATETIME scale.
WIDTH=*num-expression*	Sets printed graph width, in inches.

These parameters are discussed in detail, below.

Controlling the Size and Shape of Your Graph.

The following parameters control the size and shape of your printed graph:

PORTRAIT	7.5 in. wide, 10 in. high (this is the default)

LANDSCAPE	10 in. wide, 7.5 in. high
WIDTH=*num-expression*	
HEIGHT=*num-expression*	

Height and width are given in **<u>inches</u>**, must be positive and fit within the printable area of the page (on your printer). Values below 2 inches are not allowed.

If you are going to print the graph, the page size must not be greater than the printable area on the page.

<u>Note:</u> Font sizes and line weights are set for the printed size of the graph. *If you create a metafile or JPEG file* and subsequently reduce or enlarge the image [metafile or JPEG] of the graph, the fonts and line weights will also be re-scaled, and may not give the results you want. It is best to set the width and height to match the size of the graph image when it is eventually printed or viewed. However, [if you want to] you can effect a larger or smaller font size by showing the image at a different size (see examples below).

<u>Note</u>: The graph shown on the screen is not necessarily the size it will be when printed, however the height and width are kept proportional as you re-size the screen graph window.

<u>Note</u>: If you change the page size after the graph has been plotted ***and*** automatic scaling is in effect, the graph will be re-scaled to fit on the new graph size.

<u>Automatic Scaling.</u>

Automatic scaling insures that all your data will fit on the graph. The range, on each axis, is from the lowest to the highest values plotted. The x- and y-axes are included in the plot, unless you override that (see "Including the axes in the graph", below). The scaling is not necessarily the same for x and y coordinates, and will vary if the graph size changes.

By default, automatic scaling leaves a 0.1-in clearance between the extreme data values and the border of the graph. This usually makes the graph more readable. You can change this by specifying **CLEARANCE**=*n* in the **GRAPH** statement, where n may be from 0 to 0.25. The default is 0.1 in.

<u>Note</u>: If you wish to make the horizontal and vertical scales the same, you can include the parameter **EQUALSCALES** in the **GRAPH** statement. This will cause automatic scaling to take the larger of the two automatic scales and use it for both axes. **EQUALSCALES** is not valid on log scales.

Note: Automatic scaling is disabled for the specified coordinate if you specify the horizontal or vertical scale values (see the **GRAPH** statement, below), or if you zoom or drag the graph in the window.

Scaling and Offsets.

You can **override automatic scaling** and force scaling of each axis of your graph independently.

HSCALE = *num-expression*
VSCALE = *num-expression*

The scale factors are given in **data units per inch of graph**. Data units are simply the numbers that you plot. For example, if **VSCALE**=2, each inch of the graph would represent 2 units. Higher scale factors compress the plot into less space.

Note: For logarithmic scales, the scale factor refers to the logarithm of the data value (see "Using Logarithmic Scales", below).

Setting the scale factors manually is most useful when you want the printed output to be exactly a certain scale.

The **HSCALE** and **VSCALE** values must be > 1.0e-306 and < 1.0e+307.

If you set **HSCALE**=0 or **VSCALE**=0, it restores automatic scaling for that coordinate.

You can **force both horizontal and vertical scale factors to be the same**, if desired. Just specify the **EQUALSCALES** parameter in the **GRAPH** statement. This keeps the scaling the same in both directions. Auto-scaling will use the larger of the two computed scale factors. If you specify **HSCALE** or **VSCALE**, the other scale factor will be automatically changed to the same value. This keeps squares square, circles circular, angles correct, and all other plotted values to the same scale.

Note: **EQUALSCALES** is not valid for logarithmic scales.

Setting the Offsets

If automatic scaling is not in effect, the horizontal and vertical offset of the graph are still set automatically in order to include the most negative values (and the origin of the graph). You can override this by specifying

VOFFSET=*num-expression*
HOFFSET=*num-expression*

This overrides the offset of the origin relative to the bottom or left of the graph. The offset is given in data units, (or in logarithms of data units for log scales), and corresponds to the largest **negative** value that can be displayed on that axis.

Negative offsets push the axes off the graph at the bottom or left, and start the graph at a positive value (*e.g.*, **VOFFSET** = -10 starts the graph at +10. The axis is offset 10 units below the bottom of the graph.) A positive offset raises (increases) the position of the axis, again relative to the bottom (or left side) of the graph.

If, as a result of specifying an offset, either axis is not visible on the graph, a "dummy" axis will be shown as a red line with ticks and numbers which would normally appear next to the axis.

Note: For logarithmic scales, the "axis" is defined to be at the value 1, rather than 0. This is because the logarithm of 1 is 0.

Note: The offset can only be set if automatic scaling is off. If you reset **HSCALE** or **VSCALE** to 0 (automatic scaling), the corresponding offset will also be reset.

If you specify an offset that would cause the graph range to exceed the maximum, the offset will be adjusted automatically so that the range on the graph does not exceed ± 1.79e+308 (the range of DOUBLE floating point numbers). For logarithmic scales, the range is from -323.31 to +308.25 (the common logarithms for the smallest and largest positive DOUBLE floating point numbers).

Hint: Before you try to re-position your graph using the **HOFFSET** and **VOFFSET** parameters, read the section "Including the X- and Y-Axes On Your Graph" (which is automatic), and try re-positioning and re-scaling the graph using the graph window **ZOOM** menu and dragging the graph.

Including the X- and Y-Axes On Your Graph.

Normally, graphs include the X- and Y-axes on the graph. This helps establish a frame of reference when viewing the graph. The default is to include both axes on the graph.

Sometimes, however, the data is confined to a small range away from the axis, which can result in a graph with a lot of white space and the plotted data compressed into a small part of the graph. In this case, you can specify that either axis **not** be included in the graph, with the statements:

GRAPH INCLUDEXAXIS = 0, or
GRAPH INCLUDEYAXIS = 0

(Setting the parameters back to 1 restores the defaults.)

An example of this feature is shown below:

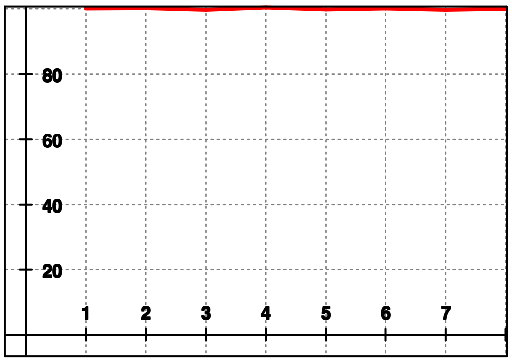
The following program:

```
GRAPH HEIGHT=3.5, WIDTH=5
DATA  100.14, 100.28, 99.75, 100.45, 99.90, 100.22, 99.83, 100.10
PLOT  COLOR=(255, 0, 0)
FOR i = 1 TO 8
READ y
PLOT  X=i, Y=y
NEXT
```

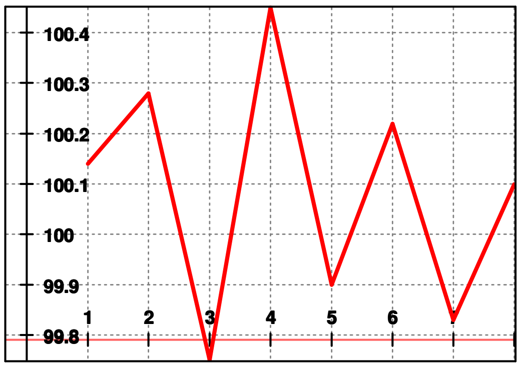
will give the graph on the left, while adding the statement

```
GRAPH INCLUDEXAXIS = 0
```

will give the graph on the right, which shows much more detail.



(**INCLUDEXAXIS**=1)



(**INCLUDEXAXIS**=0)

When these parameters are set to 0, automatic scaling scales the graph so that your data fills the graph space. Dummy (red) axes are shown and are numbered according to the calculated scales.

Note: This only works when automatic scaling is in effect. If you specify **HSCALE** the **INCLUDEYAXIS** parameter will have no effect, and if you specify **VSCALE**, the **INCLUDEXAXIS** parameter will have no effect.

Note: For logarithmic scales, the axis of the graph is the value 1, rather than 0 (since the log of 1 is 0).

Suppressing the Numbers and Tick Marks on Either Axis.

In the case where the numbers along one axis have no meaning, such as a timeline graph or comparative vertical bar graph, the numbers along the meaningless axis are unnecessary and confusing. Also, the red "fake" axis (when the axis scrolls off screen) doesn't look good on top of geometric shapes.

You can suppress the numbers and tick marks along either axis with the **NUMBERXAXIS** and **NUMBERYAXIS** parameters.

> **GRAPH NUMBERXAXIS = 0**, or
> **GRAPH NUMBERYAXIS = 0**

suppresses the numbers on the corresponding axis. **The default for both parameters is 1**, which means draw the axis, numbers and tick marks normally.

When an axis has its numbers suppressed:

- The numbers along that axis are not displayed,
- Tick marks along that axis are not displayed,
- The axis can scroll off the screen – no red "dummy" axis is shown.
- Any text that is associated with that axis is drawn, but also can scroll off screen along with the axis.

Centering the Page When Printing (Margins).

By default, QuickCalc centers the graph on the physical page, giving equal margins on both sides. If the graph will not fit in the printable area of the page when centered, you will receive the error message "Graph will not fit on page". If you get this message, try the following:

- reduce the size of your graph with **GRAPH WIDTH**=*num-expr*, **HEIGHT**=*num-expr*,
- use a different printer with a larger printable area,
- select a larger paper size, such as "legal", when printing,
- change the orientation of the printer to "landscape" if your graph is wider than it is high, or
- reposition the graph on the page with **GRAPH HMARGIN**=*num-expr*, **VMARGIN**=*num-expr* (see below).

To change the margins on your printed graph:

> **VMARGIN** = *numeric-expr*
> **HMARGIN** = *numeric-expr*

This specifies the distance, **in inches**, from the top or left side of the physical page to the edge of the graph. If *num-expr* is negative, that margin reverts to the default, which is centered.

If the new margin pushes the graph off the other side of the printable area of the page, the page may be rotated (if this fixes the problem), or else it will generate an error.

Note: A margin of less than the start of the printable area will also generate an error.

Note: Margins have no effect when viewing the graph on the screen or when creating a JPEG file or metafile.

Controlling the Appearance of the Graph.

Axis Weight.

By default, the graph is drawn with an *x*-axis, a *y*-axis, and a border surrounding the entire graph. The default weight (thickness) of these lines is 1.5 points (approximately 0.02 inches). This produces a pleasant appearance for most full-page graphs. If you desire heavier or lighter lines for the axes and border, specify the parameter:

AXISWEIGHT = *num-expression*   (in "points" = 1/72 inch)

The largest point size supported is 10. A value of zero results in a line one pixel thick.

Note: Lines on the screen may ***not*** look exactly like lines on the printer. This is because the line thickness is rounded (up or down) to the nearest pixel.

Note: Smaller lines look better in small graphs, or graphs which will be included in a paper or book. Larger graphs which will be viewed from a distance may require darker lines. The numbers on the axes will also be enlarged for larger axis weights.

Axis Color.

Normally, the *x*- and *y*-axes are black, along with the tick marks and numbers along the axes. If you want them to be a different color, specify

AXISCOLOR= (*r, g, b*), or
AXISCOLOR= *numeric-variable*

(see "Specifying Colors", below).

Axis color is different from grid color. Lightening the axis and/or grid colors can make them less obvious (or distracting). Different axis colors look better against different color backgrounds. You can also suppress the numbers, axes, and/or grid completely.

Note: White is a valid grid color, however it will disappear against a white background. However, it looks good when using a black (or other dark color) background. Experiment to see which looks best on your graph.

Specifying if you want a border:

NOBORDER	(does not print a border)
BORDER	(prints the border)

Note: You can also turn the border on or off from the graph window menus (**OPTIONS / BORDER**)

Leaving Space Between the Border and the Plotted Data.

CLEARANCE=*num-expr*	Leaves a space of *num-expr* inches between the extremes of the plotted points and the border ***when auto-scaling.*** *num-expr* may be 0 to .25. Default is 0.1 in.

Note: If you want the plotted points and shapes to touch the border, specify **CLEARANCE** = 0, or set the scale factors and offsets manually.

Displaying or Suppressing X and Y axes.

You can, optionally, choose not to show the *x-y* axes and associated ticks and numbers on the graph. This may be desirable if the graph is displaying shapes or figures that do not correspond to numerical values, such as the outline of a plot of land or shape of a part.

You can show or suppress the axes by specifying:

AXES	(displays *x-y* axes and numbers)
NOAXES	(don't display *x-y* axes and numbers)

Note: You can also display or suppress the axes from the graph window menus (**OPTIONS / AXES and NUMBERS**)

Note: **NOAXES** overrides **NUMBERXAXIS** and **NUMBERYAXIS**.

Note: When drawing shapes, it is usually best if the *x* and *y* axes are scaled identically. Use the **EQUALSCALES** parameter to accomplish this.

Grid Lines:

You can include dotted grid lines (at the tick marks) on your graph by specifying:

GRID	(prints grid lines – [default])
NOGRID	(does not print grid lines)

Note: You can also turn grid lines on or off from the graph window menus (**OPTIONS / GRID**).

The grid lines are drawn slightly smaller then the axes (but no less than 1 pixel wide), and are grey colored and dashed.

You may change the color of the grid lines to any color that you desire with the parameter **GRIDCOLOR**=[(*r,g,b*) | *num-expr*]. (See "Specifying Colors", below on how to specify colors.). By changing the color of the grid, you may make graphs that look more like those drawn on commercial graph paper. You may also use this parameter to create lighter or darker grid lines.

Grid lines are spaced based on the graph size and scale. You can change the spacing of the lines with the **OPTIONS / MOREGRIDLINES** or **OPTIONS / FEWERGRIDLINES** menu items.

Background Color

Graphs are drawn with a default background color of white. If you want to change that, use the parameter **BGCOLOR**=[(*r,g,b*) | *num-expr*] which specifies the color of the background. (See "Specifying Colors", below on how to specify colors.)

Caution: If you specify a background color other than white and then ***print*** the graph, you could use a **LOT** of ink! Some printers get the paper quite wet when printing a large area of solid color.

Colored backgrounds look really good when viewed on the screen or projected as a slide.

Note: When you change the background color, you may also need to change the color of the grid lines, axes, and any text drawn on the graph.

Displaying All the GRAPH Parameters Currently in Effect.

If you have manually changed the appearance or view of your graph (zooming, dragging, *etc*.) and want to be able to duplicate those settings later, you may print a list of the parameters currently in effect with the statement

```
GRAPH  LISTPARMS
```

This will show the parameters you have specified, plus those calculated as a result of auto-scaling or zooming and dragging the graph. For your convenience, this will also be written to the log file.

Plotting your Data.

Data plotted on the graph is controlled by the **PLOT** statement (see below).

Currently, only one type of plot is supported, which is "connected points". A series of (*x,y*) values ("points") may be sorted by their *x*- or *y*-value, then plotted with line segments connecting them (see "Sorted Points").

You may have more than one (up to 50) plots on the same graph, each with a different color or line weight, if desired.

Data plotted may be DOUBLE or LONGMATH, however, the values must be convertible to DOUBLE. Values plotted may be specified as numeric expressions.

A **PLOT** statement with a parameter other than **X**=*value* or **Y**=*value* will begin a new plot. (See the **PLOT** statement, below). The **PLOT** statement can contain parameters plus the first data point, but don't put this plot statement inside a loop, as it will try to initiate a new plot each time it is executed.

A **GRAPH** must be active before you issue a **PLOT** statement. Typically, the sequence is as follows:

```
GRAPH        [optional parameters]
…
PLOT         [optional parameters]
[controlling loop]
        …
        PLOT   X=numeric-expression, Y=numeric-expression
        …
[end controlling loop]
```

(see the "Simple" example above.)

The points do not need to be sorted, as the program will sort them by their *x*-values before plotting them (see "Sorting Data", below).

PLOT Statement Parameters.

PLOT statement parameters are:

AVERAGE=*num-expr*	"Averages" the plot by moving each point closer to the average of the points on either side of it. *num-expr* may be between 0 and 10, with 10 specifying the maximum averaging. (See "Advanced Graphics".)

COLOR=[(*r,g,b*) | *num-expr*]	Specifies the color of the line to be drawn. Default color is black. (See "Specifying Colors", below on how to specify colors.)
LINEWEIGHT=*num-expression*	Thickness of plot line, in points (Default is 3 points).Values up to 10 points are allowed. A weight of 0 means one pixel wide. **<u>Note</u>**: Line weights are rounded (up or down) to the nearest pixel, and so might appear slightly different when printed.
SMOOTH=*num-expr*	"Smoothes" the graph by connecting the points with Bezier curves. *num-expr* may be between 0 and 10, with 10 specifying the maximum smoothing. (See "Advanced Graphics".)
SORTEDX	Data points are sorted by their X-value before plotting (this is the default).
SORTEDY	Data points are sorted by their Y-value before plotting.
UNSORTED	Data points are plotted in the order they are generated.

See "Sorted Points" below, for an example of the results of sorting the data points.

X=*num-expression*,
Y=*num-expression*.

The *x*- and *y*-values must be specified together on the same **PLOT** statement. Nothing else should be on that statement, unless you wish to start a new plot.

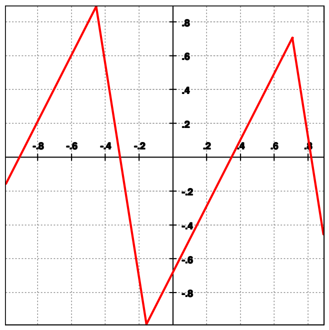
<u>Note:</u> The *x*- and *y*-values must be DOUBLE or convert correctly to DOUBLE. LONGMATH numbers will be truncated and converted, if possible. Exponent underflow will result in zero.

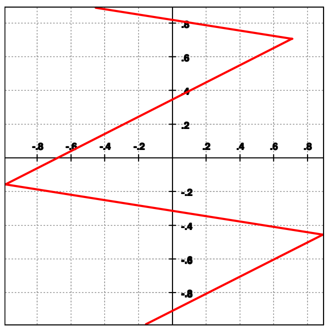
Using the PLOT Statement

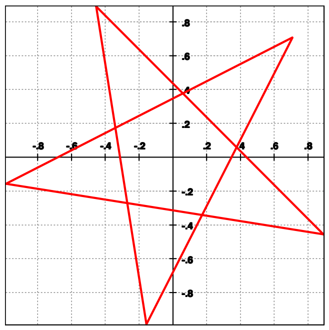
Sorted Data.

You have three choices of how to sort the data as you plot it. You can sort by ascending X-values, by Y-values, or in the order generated. The resulting plot will look different (see below).

```
GRAPH  HEIGHT=6, WIDTH=6
PLOT  COLOR=(255,0,0)  SORTEDX
//                              or SORTEDY or UNSORTED
DEGREES
FOR  i=0  TO  5
  angle = 45 + i * 2 * 360 / 5
  PLOT  X=COS (angle), Y=SIN (angle)
NEXT
```


SORTEDX


SORTEDY


UNSORTED

Note: You can also generate figures like the "star" pattern above using **SHAPE POLYLINE** (see "Advanced Graphics).

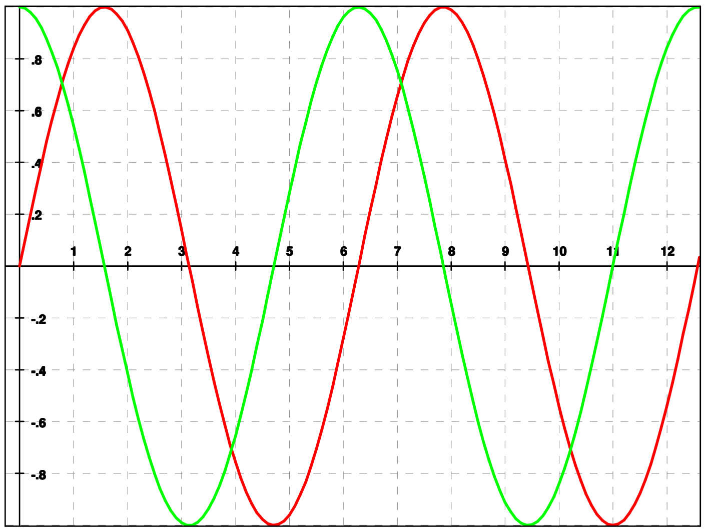
Multiple Plots on the Same Graph.

You can plot more than one set of data on the same graph. Different plots can have different line weights or colors to help identify them.

An example of multiple plots is shown in the following sample program, in which the sine is plotted in red and the cosine in green:

```
GRAPH   LANDSCAPE
PLOT COLOR = (255, 0, 0)
FOR angle = 0 TO 4 * PI + .1 STEP .1
   PLOT X=angle, Y=SIN(angle)
NEXT
PLOT COLOR = (0, 255, 0)               // starts a new plot
```

```
FOR angle = 0 TO 4 * PI + .1 STEP .1
  PLOT X=angle, Y=COS(angle)
NEXT
```



(shown reduced)

Note: The scale is the same for both plots. You can't plot multiple plots with different scales on the same graph.

Note: You must complete one plot before beginning another.

Logarithmic Scales.

You can easily plot your data on logarithmic scales, which produces a graph similar to one plotted on log-log or semi-log graph paper. Either axis or both may be logarithmic, controlled by the **GRAPH** statement parameters:

HSCALETYPE = [LOG | LINEAR | DATETIME], and / or
VSCALETYPE = [LOG | LINEAR | DATETIME]

(The default for both scale types is LINEAR.)

You can change these values at any time. Changing **HSCALETYPE** or **VSCALETYPE** will reset the corresponding scale factor to auto-scale, so if you are going to over-ride the scale factor, do so on a separate **GRAPH** statement following the one that set the scale type.

Note: If you change a scale type to logarithmic and you have already plotted or drawn something to the graph, none of the existing data or shape coordinates may be zero or negative. If that happens, the graph will be terminated and an error message issued.

The resulting plot will not be exactly like logarithmic graph paper, because to attempt to print that many grid lines would crowd the page and make the graph unreadable, and would not work on the screen. Instead, the program makes a reasonable choice as to how many grid lines it should use, based on the scale factors and page size. Grid lines will be approximately ¼ in apart, although this varies according to which portion of the graph you are viewing. You can modify the grid spacing with the **OPTIONS / MOREGRIDLINES** or **OPTIONS / FEWERGRIDLINES** menu items.

Note: Not all settings will produce desirable results, and that some settings may result in the numbers crowding together. Choose what looks best to you. More grid lines works better with lighter axis weights.

Keep in mind that the image on the screen is a reduced version of the actual (printed) graph. Grid lines that appear crowded together on the screen may look fine on the printed page.

Some functions work differently with log scales. These are described below.

Scale Factors

When a scale is logarithmic, the scale factor (**HSCALE** or **VSCALE**) refers to logarithms, rather than data values. Thus, a scale factor of 1 means that one inch of graph represents a change of 1 in the logarithm value. A change of one in the logarithm represents a factor of 10 in the data, so each inch or the graph would represent multiplying the data value by 10. This is called one “decade”.

Note: Although you can still set the scale factors manually, is easier and more intuitive to let the auto-scaling set the scale factors and then adjust the graph later using the “zoom” menu function.

Range of Values

The logarithm of a negative number is undefined. You can’t plot negative numbers or zero on a log scale, and to attempt to do so will generate an error.

The range of numbers you may plot is, therefore, $+10^{-324}$ to $+10^{+308}$, or any valid ***positive*** DOUBLE value. Numbers that are too small may underflow and revert to zero, which would generate an error. Be careful not to let that happen.

Offsets

Offsets (**HOFFSET** and **VOFFSET**) refer to the logarithms of the numbers. For example, setting **VOFFSET**=2 would start your graph at 10^{-2} or .01.

Note: Although you can still set the offsets manually, is easier and more intuitive to let the auto-scaling set the offsets and then adjust the graph later by "dragging" the graph on the screen.

Very Large or Very Small Scale Factors

You can set the scale factor (or "zoom" in or out) for log scales, just like linear scales. However some considerations apply:

When you zoom in very far on a log scale (or set the scale factor to small values), the log scale begins to look like a linear scale. The purpose of a log scale is to show a wide range of values; showing a very narrow range sort of defeats that purpose and makes the log scale unnecessary.

Zooming way out loses the detail of the numbers plotted, and shows only the powers of 10.

You can, or course, set any scale you want, or let auto-scaling make your data fit on the graph. The scale factor is limited so that the range of the graph will not exceed (the logarithms of) the range of positive DOUBLE numbers.

Interpolating values between points (on LOG Scales)

When you plot a line between two points, the line is drawn straight. On a log scale, that means that the log increases linearly along that line. Be careful not to try to interpolate values between points, as they may be different from the data values between the points.

You can get a more accurate graph by calculating and plotting more values between those points.

Date-Time Scales

Note: See the Document "*Working With Dates and Times*" for a description of the DATETIME variables and how to create and use them. If you are going to use DATETIME scales in your graph, you should first read that document.

If you want to plot a time-line chart or graph a function over time (such as stock prices), then you should use DATE/TIME variables for the values along the "time" axis (usually the *x*-axis). In order to correctly interpret the time values and properly scale the graph, specify

GRAPH HSCALETYPE (or **VSCALETYPE**) **= DATETIME**.

Just as with logarithmic scales, there is no zero, so the position of the y-axis (if the *x*-axis is the time scale) is arbitrary. By default, the axis will appear at Jan 1, 1970, local time. You can force the axis to appear anywhere along the time scale by setting the DATETIME variable ***zerodatetime*** to the date and time desired. For example,

zerodatetime = **MAKEDATETIME** (1, 1, 2011, 12, 00, 00)

positions the axis at Jan 1, 2011 at noon.

Also, the **GRAPH INCLUDEYAXIS=0** may be used if the axis position is too far from the area of interest. The "fake" red axis and the tick values will still be displayed.

Note: If you change the time zone or the daylight saving time rules (see "*Working With Dates and Times*"), this may affect the zerodatetime, so you should always set time zone and daylight saving time rules first.

Offsets

Auto-scaling works in this mode. You can still specify the offset and scale factors manually, however keep in mind that the **offset must also be a DATETIME value**. For example,

HOFFSET = **- MAKEDATETIME** (1, 1, 2000) [*note the minus sign*]

will start the graph at Jan 1, 2000.

Scale Factor

The scale factor, if set manually, must be given in **seconds per inch**. For example, to set a scale factor of 1 inch = 10 years, set **HSCALE** = 10 * 365.2425 * 86400, which is 10 years expressed in seconds.

Note: If you are setting the offset manually, be sure to set the scale factor first.

Automatic scaling is still the easiest way, and you can always adjust the scaling using the "Zoom" menu or the arrow keys.

Bar Charts and Time-Line Graphs

If you are drawing a **bar chart**, the width of the boxes (or any other measurement on the *x* (time) axis) must be specified in **seconds**. The height of the box is given in the units of the *y*-axis.

If you are doing a **time-line graph** using long rectangles to show periods in time, specify the left and right sides of the rectangle as DATETIME values.

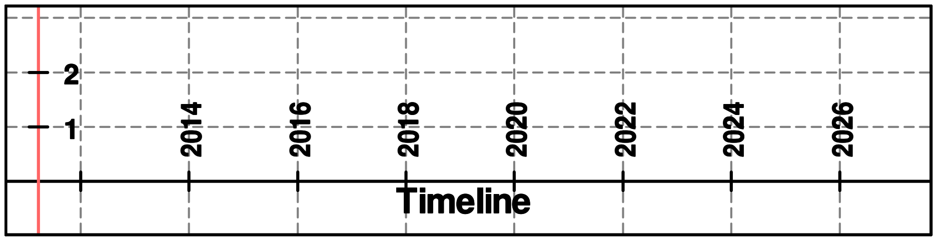
Note: When zooming a time-line graph, it is usually more effective to zoom only the "time" axis. If the time axis is the *x*-axis, you can zoom horizontally using the **left and right arrow** keys.

Note: Daylight Saving Time transitions cause an hour to be dropped or an extra hour to be added on certain days. These are shown correctly on the graph scales, however the appearance may seem strange (hours missing or extra hours inserted).

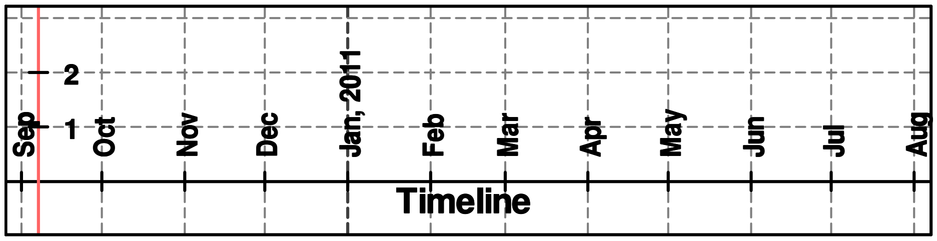
Appearance and Scales

When you zoom in or out on a DATETIME scale, the values displayed along the axis will change depending on the scale factor. As you zoom in to a given year, you will begin to see months displayed. Zooming into the month you will see days, etc. The following examples illustrate this:

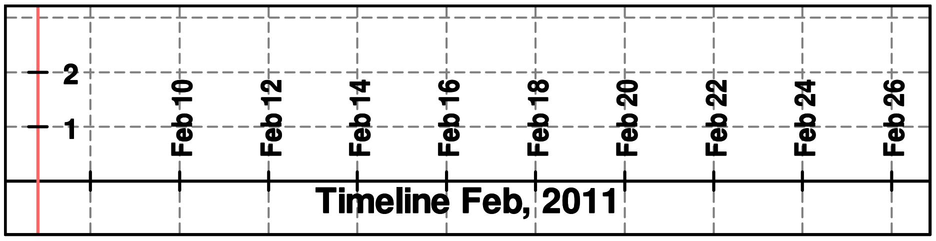
HSCALE=90000000:



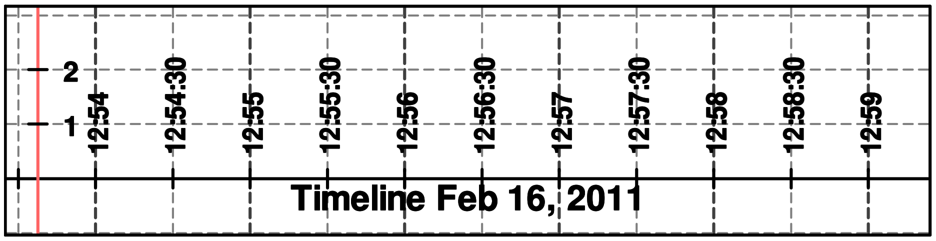
HSCALE=5000000:



HSCALE=300000:



HSCALE= 60:



Note: You may zoom in farther on a DATETIME scale than you can on LINEAR or LOG scales. The zoom limit is determined by the precision of the DATETIME value, so you can zoom in to microseconds on years closer to 1970, but at 5 billion years the limit is around 15 minutes.

Note: When you zoom in on a DATETIME scale, you may not be able to tell which day, month and year you are in. If you want that information displayed, place a title (see below) on the DATETIME axis which includes the string "**&date**". **Note**: &date must be lower-case. When the graph is displayed, this string will be replaced by the portion of the date which does not change over the range of the graph, *i.e.,* the part you can't see.

Example:

```
t$ = "Stock Prices &date"
TEXT  XAXIS,  STRING = t$
```

In the example graphs above, the string was "Timeline &date".

Daylight Saving Time

Certain default rules are set for when Daylight Saving Time occurs, and how far the clocks advance. You can change these rules, if necessary (see "*Working With Dates and Times*").

Daylight Saving Time changes are reflected in the calculations for DATETIME variables, and appear on the graph. If you zoom in on a date where Daylight Saving Time begins or ends, you will notice a missing hour or an extra hour.

Putting Text On Your Graph.

There is often a need to put text on a graph to

- label the x and y axes,
- identify objects and plots,
- title the graph,
- put date, author, copyright information on the graph, *etc*.,
- give yourself credit,
- create a legend.

QuickCalc BASIC has implemented an easy-to-use way to accomplish this.

> **Note**: If you want to draw text which is filled, outlined, rotated and scalable, you can use the more complex but versatile function **SHAPE TEXT** (see "Advanced Graphics".)

Text is placed on the graph using the **TEXT** statement. The format of the **TEXT** statement depends on how you want the text displayed.

A graph must be active (**GRAPH** statement) before the **TEXT** statement is executed.

TEXT Statement Parameters

Note: Not all combinations of these parameters are valid. Valid parameters depend on the usage. See notes below on the usage of these parameters.

ANGLE=*num-expr*	Rotates the text counter-clockwise by the angle specified in *num-expr*.
BOLD	Selects a **boldface** font.
BOXED	Draws a box around the text.
COLOR=[(*r, g, b*) | *num-expr*]]	Color of the text (see "Specifying Colors", below on how to specify colors.)
DATASIZE=*num-ex*pr	Specifies the ***height*** of the text in **data units**.
FONT=*string-expr*	Specifies the font you wish to use for the text.

	Note: You can use the function **CHOOSEFONT$** to get a valid font name.
H =*num-expr*	Horizontal location of the text relative to the **physical page**, specified as a percentage.
ITALIC	Selects an *italic* font.
JUSTIFY=[LEFT | CENTER | RIGHT]	Justification (relative to the start point). Default is CENTER.
NORMAL	Resets **BOLD** and **ITALIC**.
SIZE=*num_expr*	Specifies the **point size** of the [printed] text. Values from 6 to 100 are allowed. Default is 18 points.
STRING=[*string-var* | "*string-constant*"]	Identifies the text to be displayed.
V = *num-expr*	Vertical location of the text relative to the **physical page**, specified as a percentage.
X = *num-expr*	Horizontal location of the text in the **data space of the graph.**
XAXIS	Use this text to label the x-axis.
X2 = *num-expr*	Used along with **X** to specify an area or boundary in which the text will be centered (see "Sliding Text", below).
Y = *num-expr*	Vertical location of the text in the **data space of the graph.**
YAXIS	Use this text to label the y-axis.
Y2 = *num-expr*	Used along with **Y** to specify an area or boundary in which the text will be centered (see "Sliding Text", below).

Using The **TEXT** Statement.

<u>The **STRING** parameter</u>

(Nearly) all forms of the **TEXT** statement require the parameter **STRING=**[*string-variable* | "*string-constant*"]. The string variable or constant contains the text you want to display.

<u>Using a String Variable</u>

You can specify a string variable which contains the text to display, for example:

```
title$ = "My Graph Title"
TEXT STRING = title$, …(other parameters)
```

Each **TEXT** statement should reference a different string variable.

If the contents of the string variable change later in the program, the corresponding text on the graph will change the next time the graph is updated. If the string variable has not been defined, or is empty, no text will be displayed.

After the string variable reference by a **TEXT** statement has changed, you can force an immediate update of only the portion of the screen containing the text. Use the statement:

UPDATEREGION *left, right, top, bottom*

where *left, right, top* and *bottom* define a rectangle surrounding the text, specified in data units.

<u>Using a String Constant</u>

You can specify a <u>**quoted string constant**</u> in the **TEXT** statement instead of a variable. This constant will be assigned to a dummy variable which you cannot change.

```
TEXT STRING = "My Graph Title", …(other parameters)
```

<u>Note</u>: You may <u>***not***</u> specify a string expression like *a*$ + *b*$ or *x*$ + "abc". Instead, assign the string expression to a string variable and reference that.

<u>Note</u>: The **CLEAR** statement deletes all string variables and therefore will un-define all **TEXT** statements.

Note: If the **TEXT** statement is *only* setting the font or the style (see below), the **STRING** parameter is not required.

Note: The appearance of the text will be the same regardless of whether a string variable or string constant is specified.

If the text in the string variable contains one or more "new-line" (or \n) characters, the displayed text will be broken into **multiple lines.**

Labeling the Axes.

You can place labels on the *x*- or *y*-axis with the statement

> **TEXT XAXIS**, **STRING=**[*string-var* | "*string-constant*"]
> [,**COLOR**=[(*r, g, b*) | *num-expr*]]
> [, **H**=*num-expr*] [, **SIZE**=*num_expr*] [,**FONT**=*string-expr*]
> [,**BOLD**] [,**ITALIC**]

or

> **TEXT YAXIS**, **STRING=**[*string-var* | "*string-constant*"]
> [,**COLOR**=[(*r, g, b*) | *num-expr*]]
> [,**V**=*num-expr*] [, **SIZE**=*num_expr*] [,**FONT**=*string-expr*]
> [,**BOLD**] [,**ITALIC**]

The text string is placed, by default, centered along the x- or y-axis.

For the ***x*-axis**, **H**=*num-expr* specifies a percentage (from 0-100) for the center of the text on the screen. The text will move up or down *with the axis*, depending on the offset, but will remain at the same horizontal position on the screen.

For the ***y*-axis**, the text is rotated and is centered vertically on the page, unless you specify **V**=*num-expr* (from 0-99 percent).

SIZE=*num-expr* specifies the point size of the text. Values from 10 to 30 are allowed. Default is 18 points.

Note: The *default* point size will increase if you specify a thicker line weight on the **GRAPH** statement.

Note: The size is specified in ***points*** on the ***printed page***. It will not change if you change the graph scale or zoom into the graph.

Note: **DATASIZE, ANGLE, JUSTIFY,** and **BOXED** (see below) are not valid for text on the x and y axes.

COLOR, BOLD, **ITALIC** and **FONT** are described below.

Labeling a point or object on your graph.

Note: This section describes how to place text on the graph which is tied to a “point” (*x, y*) *i.e.*, a location in **data space**. For titles and other text tied to locations on the physical page, see “Placing Text on the (Physical) Graph”, below.

You may place text anywhere on the graph, using the statement

```
TEXT  STRING=[string-var | "string-constant"],
      X=num-expr, Y=num-expr
      [,X2 = num-expr] [,Y2 = num-expr]
      [,COLOR=[(r,g,b) | num-expr]]
      [,BOXED]
      [,SIZE=num-expr | ,DATASIZE=num-expr]
      [,FONT=string-expr]
      [,BOLD] [,ITALIC] [,NORMAL]
      [,ANGLE=num-expr]
      [,JUSTIFY = [LEFT | CENTER | RIGHT]]
```

X and **Y** refer to data values in the **data space of the graph** (*not the page*) and are given in data units. The text will appear at the desired point on the graph, and will **follow that point** if the graph is scrolled or the offset(s) changed.

The text is centered horizontally about that point, (see **JUSTIFY**), and its baseline rests vertically upon the point.

Negative values or zero are not allowed on logarithmic scales.

X2 and **Y2** (optional) are used, along with **X** and **Y**, to specify and area or boundary in which the text will be centered (see “Sliding Text”, below).

COLOR=[(*r,g,b*) | *num-expr*]
specifies the color of the text to be displayed. Default color is black. . (See “Specifying Colors”, below on how to specify colors.)

SIZE=*num-expr* specifies the **point size** of the text. Values from 6 to 100 are allowed. Default is 18 points.

<u>Note</u>: Text size is based on the printed size of the graph. Larger size graphs are shrunk to display them on the screen. Text will therefore appear smaller. Make sure to specify the text size you want on the printed graph rather than what looks good on the screen, otherwise your text may be too large on the printed graph.

<u>Note</u>: The actual, printed size of the text specified this way will ***<u>not change</u>*** if you zoom in or out.

DATASIZE=*num-ex*pr specifies the ***<u>height</u>*** of the text in **<u>data units</u>**. Any positive value is allowed. This is useful if, for example, you are placing text inside a shape and want the text to still fit inside the shape if the scale factor is changed.

<u>Note:</u> If the size is expressed in data units, it ***<u>will change</u>*** if you change the graph scale or zoom the graph. This could result in huge text that obscures everything else.

<u>Note:</u> Specify the size with either **SIZE**= or **DATASIZE**=, but not both.

<u>Note</u>: When working with **LOG** scales (particularly on the vertical axis), **DATASIZE** uses the $\log_{10}$ of the size of the font, *e.g*., **DATASIZE**=4 specifies a font with a height of 10000 (= 10^4)! Also, you can't specify a negative or zero datasize, although it can be very small, but still positive.

BOXED draws a box around the text. The line weight of the box is approximately the same as the font stroke width.

FONT specifies the font you wish to use for the text. Any font installed on your computer may be used. You must specify the name of the font exactly (except for case). Go to the Control Panel and select "Fonts" for a list. You can also use the function **CHOOSEFONT$** to select a valid font name. The default is a generic sans-serif font like "Arial".

<u>Note</u>: The font, once specified, remains in effect for all subsequent **TEXT** statements until you change it. This allows you to use the same font in multiple **TEXT** statements without having to specify it every time. To reset back to the default font, specify **TEXT FONT**="".

You may issue a **TEXT** statement which only specifies the **FONT**. This will set that font for future **TEXT** statements. If **FONT** is specified without **STRING**, all other parameters will be ignored.

BOLD selects a **boldface** font.

ITALIC selects an *italic* font.

NORMAL resets **BOLD** and **ITALIC**.

Note: The default is **BOLD** and non-italic. **BOLD**, **ITALIC**, or **NORMAL** affect only the current **TEXT** statement. If you want to change the default, issue the **TEXT** statement with only the *single parameter*, **BOLD**, **ITALIC**, or **NORMAL**. This will change the default for future **TEXT** statements so you don't have to specify them for every **TEXT** statement.

Note: If you want to specify normal-italic, give the **NORMAL** parameter first, then the **ITALIC**, since **NORMAL** resets **ITALIC**.

ANGLE=*num-expr* rotates the text counter-clockwise by the angle specified in *num-expr*. Negative angles generate a clockwise rotation. The angle is specified in radians unless the **DEGREES** statement is in effect. The angle must be between -3600 and 3600 degrees, (or -20π and 20π radians). If the text is **BOXED**, the box will also be rotated. The center of rotation is the given start point, usually the center of the baseline of the first line of text. (See **JUSTIFY**).

JUSTIFY = [**LEFT** | **CENTER** | **RIGHT**].
Specifies how the text is positioned relative to the given starting point (see **X** and **Y**, and **H** and **V**, above):

CENTER :	(default) The text is centered above the start point.
LEFT:	The left edge of the text begins at the start point and extends to the right.
RIGHT:	the right edge of the text is located at the start point and the text extends to the left.

If you rotate the text, it will be rotated about the start point, not (necessarily) the center of the text.

You may have any number of **TEXT** statements (plus the **XAXIS** and **YAXIS**) in your program.

If the text in the string variable contains one or more "new-line" (or \n) characters, the displayed text will be broken into **multiple lines**. All lines are justified and the text is positioned relative to the baseline of the first line of text. If the text is **BOXED**, the box will surround all the text lines.

Note: Depending on your scaling and offsets, the text may be off the screen and therefore not display.

Note: The color (255,255,255) is white, which will not show up on a white background.

Note: Auto-scaling does not work well if the only thing on the graph is text. This is because the size of the text will depend on the scale factor. In this case, you should set the scale factors manually, or draw an "invisible" box around all the text.

Note: This program is not a word processor. The text function was intended to allow you to annotate and title your graphs. It does not format paragraphs.

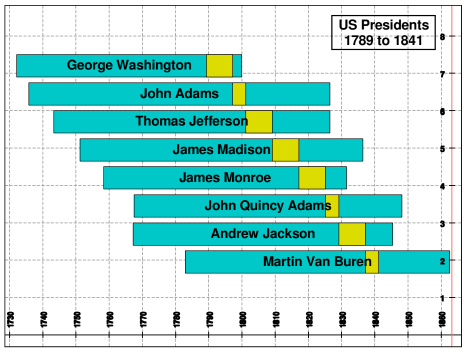
"Sliding" Text

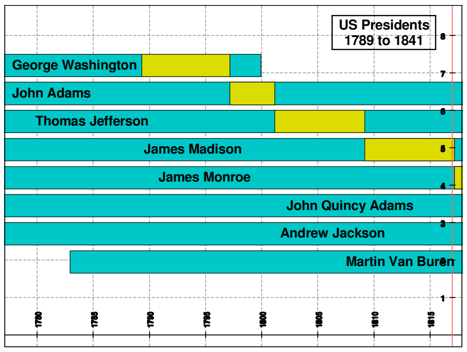
There are times when you don't want text anchored to a single point, but rather allowed to **float within a range of values**. For example, consider a time line, which consists of several wide boxes stretched out in the horizontal direction. Each box has a label, centered in the box. Now if you scroll or stretch the graph horizontally, the centers of some boxes may be off the screen, although one end or the other may still be visible. If the text also scrolls off the screen, you have lost the labeling of those boxes.

Specifying **"sliding" text** in the *x*-direction (in this example) allows the text to **slide right or left** in order to stay visible on the screen, subject to the constraints of the box.

When specifying sliding text, the **X** value is set to the left side of the boundary, and the **X2** value is set to the right side. The text is now constrained to be within those boundaries (unless it is too large). If the entire bounded range (the box, in this case) is visible on the graph, the text will be centered in the range. As the graph is scrolled left or right, the text will be slid so that it remains on the screen, constrained, of course, by the range.

Example:





Normal View:
(text centered in boxes)

Stretched horizontally:
(text slid to keep it on screen)

(See the sample file "Presidents Timeline.txt" for how this graph was produced. How to draw boxes is described below (the **SHAPE** function). Try stretching this graph horizontally (right arrow key) and then scrolling it from left to right, noting the behavior of the text items.

Note: This feature also works for vertical ranges (use **Y** and **Y2**), for example, a vertical bar graph. You can also use both horizontal and vertical sliding in the same **TEXT** statement. In this way, you can keep text inside of a rectangle.

Note: Text may be rotated, boxed, and may include multiple lines. The range will include all the lines and the boundaries of the [rotated] box.

Note: This feature is useful for vertical centering of text with multiple lines. Simply specify **Y** and **Y2** with the same value and the multiple lines of text will be centered vertically at that value.

Note: **X2** and **Y2** should only be used when using **DATASIZE**.

Superscripts and Subscripts.

You can have superscripts and subscripts in your text strings. This allows you to label points and curves with things like $y_1 = x^2$, $y = \log_{10}(x)$ and $z = 10^y$, *etc.*

You simply place a code into the string where you want the change to occur. **\+** means change from normal to superscript. **\–** switches it back.

Similarly, **\–** changes from normal to subscript, and **\+** changes back. To switch directly from subscript to superscript, you would use **\+** twice.

Example: "y\–1\+=x\+2" will draw the text $y_1 = x^2$.

Note: The sequence **\+** is translated to **CHR$** (15) and **\–** to **CHR$** (14).

Placing Text on the (Physical) Graph.

You may place text on the physical page which is anchored to a **location on the graph** and is *not* tied to any data value. Text defined in this way will not move when the screen is scrolled or re-scaled.

Note: When placing text on the physical page, you should set the page size, and plot the graph without text to see what the page layout will look like. Then you can better judge where and how large the text should be.

This type of text is useful for titles, legends, dates, author, copyright, etc. Instead of using **X** and **Y**, you specify the horizontal and vertical location on the graph as follows:

TEXT STRING=[*string-var* | "*string-constant*"],
H=*num-expr*, **V**=*num-expr*
[, *remainder of parameters are the same as above*]

H specifies a percent offset from the left of the graph (0 to 100) for the center of the text. Default is 50 % (centered).

V specifies a percent offset from the bottom of the graph (0 to 99) for the baseline of the text. Default is 85%.

Note: You can mix **H** and **V** with **X** and **Y** to specify text which is anchored to the graph in one dimension but anchored to data in the other. Just specify **X** and **V,** or **H** and **Y**. This can be useful to label a maximum or limit value on one axis.

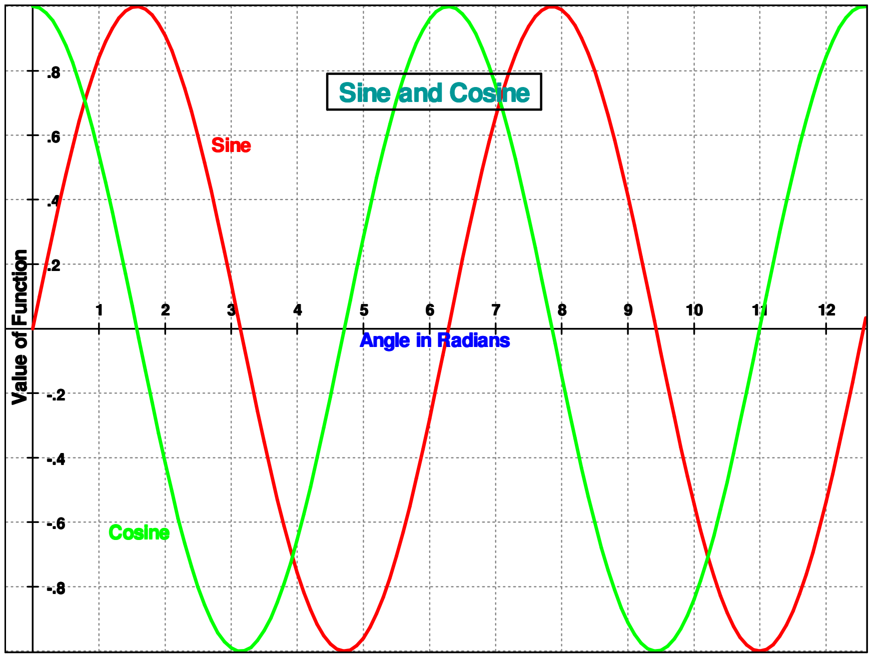
Note: If you don't specify **H** or **X,** the default is **H**=50. Similarly, if you don't specify **V** or **Y**, the default is **V**=85.

Example Graphs Using Text.

The following program illustrates how text may be applied to a graph. It is the same program we used to graph the sine and cosine, but with text added:

```
// Plots the sine (in red) and cosine (in green), with text.
GRAPH   LANDSCAPE
PLOT COLOR = (255,  0,  0)             // starts a new plot - red
FOR angle = 0 TO 4 * PI + .1 STEP .1
  PLOT X=angle, Y=SIN(angle)
NEXT
PLOT COLOR = (0, 255, 0)               // starts a new plot - green
FOR angle = 0 TO 4 * PI + .1 STEP .1
  PLOT X=angle, Y=COS(angle)
NEXT
v$ = "Value of Function"
TEXT YAXIS, STRING=v$, BOLD
a$ = "Angle in Radians"
TEXT XAXIS, STRING=a$, COLOR=(0,0,255), BOLD  // blue
s$="Sine"
TEXT STRING=s$, COLOR=(255,0,0), X=3, Y=.55, BOLD // red
c$="Cosine"
TEXT STRING=c$, COLOR=(0,255,0), X=.6, Y=-.65, BOLD // green
title$="Sine and Cosine"
TEXT STRING=title$, COLOR=(0,150,150), BOXED, SIZE=24, BOLD
```

This produces the following graph (shown reduced):

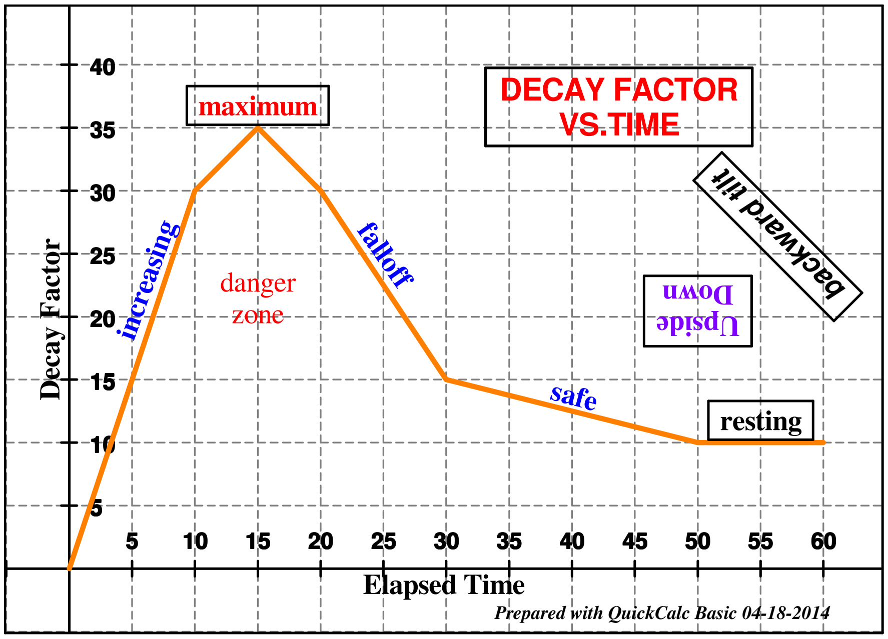


The following example program shows some of the ways in which you can use text on your graphs:

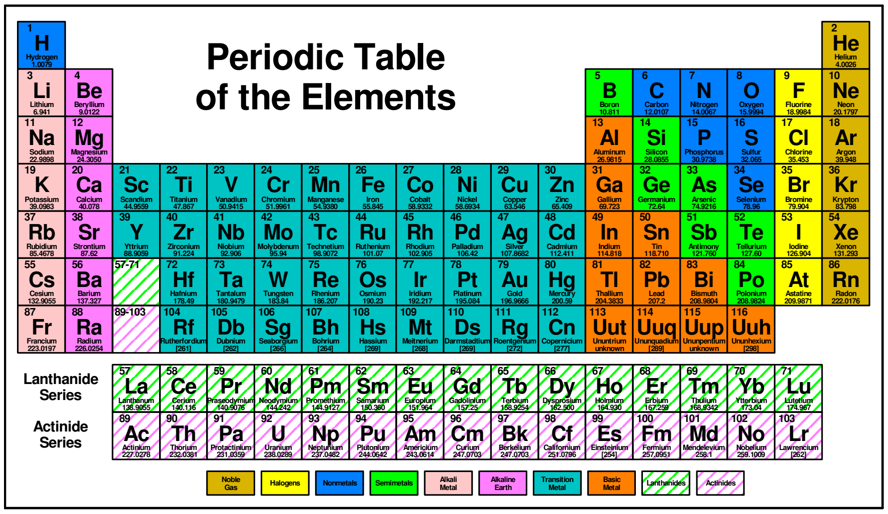
```
INCLUDE "Standard_colors.txt"
DEGREES
```

```
GRAPH HEIGHT=5, WIDTH=7, EQUALSCALES
GRAPH HSCALE=10
GRAPH HOFFSET=5,VOFFSET=5
s$="DECAY FACTOR\nVS.TIME"
TEXT STRING=s$, SIZE=20, BOXED, COLOR=red, H=70
TEXT FONT="times new roman"
TEXT XAXIS, STRING="Elapsed Time"
TEXT YAXIS, STRING="Decay Factor"
a$="Prepared with QuickCalc Basic "+date$
TEXT STRING=a$, H=75, V=2, SIZE=12, ITALIC
PLOT COLOR=orange
PLOT X=0,   Y=0:       PLOT X=10, Y=30:     PLOT X=15, Y=35
PLOT X=20, Y=30:     PLOT X=30, Y=15:     PLOT X=50, Y=10
PLOT X=60, Y=10
TEXT STRING="increasing", X=6.8, Y=22.7, ANGLE=70, COLOR=blue
TEXT STRING="maximum",  X=15, Y=36, COLOR=red,  BOXED
TEXT STRING="falloff", X=24.5, Y=24.5, ANGLE=-55, COLOR=blue
TEXT STRING="safe", X=40, Y=13, ANGLE=-15, COLOR=blue
TEXT STRING="resting", X=55, Y=11, COLOR=black, BOXED
TEXT STRING="danger\nzone", X=15, Y=22, COLOR=red, NORMAL
TEXT BOLD
x$="Upside\nDown"
TEXT STRING=x$, X=50, Y=20, COLOR=purple, ANGLE=180, BOXED
TEXT FONT=""
TEXT STRING="backward tilt", X=57, Y=27, ANGLE=135, BOXED, ITALIC
```

The graph is shown below:



Another good example of using text in a graph is given in the sample program “Periodic Table.txt”, which you can download from the website. The graph is shown below:



Drawing Shapes

QuickCalc supports **drawing shapes** (filled or open) on your graph. This is done using the **SHAPE** statement.

You can draw the following shapes:

Rectangle (which can also be a square)
Ellipse (which can also be a circle)
Arc
Chord of an arc
Pie segment
Line
Polygon (any number of sides)
Polyline (any number of points) (see "Advanced Graphics")
Bezier curve (see "Advanced Graphics")
Complex Connected Shape (see "Advanced Graphics")
Text Outlines (see "Advanced Graphics")

Each **closed** shape (*not* arc, line, polyline and Bezier curve) may optionally be filled with:

any solid color
a hatch pattern
no fill (empty and transparent)

Different shapes require different parameters in the **SHAPE** statement. These will be explained below.

The SHAPE statement.

The **SHAPE** statement has two types of parameters.

Descriptive parameters set up the sizes, positions and other characteristics of the shapes to draw. These vary depending on the shape to be drawn. There may be more than one way to specify these parameters. Descriptive parameters remain in effect until they are overridden by other parameters.

Shape drawing parameters are keywords which cause a certain shape to be drawn, using the descriptive parameters entered up to that point.

Therefore, several **SHAPE** statements may be used to specify one figure to be drawn. This avoids having really long statement lines in your program, making it more readable. It also allows for you to repeat shapes in different locations using the same size, color, etc. Just remember, a given shape will be drawn using the most recent set of descriptive parameters in effect at the time the shape is drawn.

Descriptive parameters in a **SHAPE** statement are processed first, in the order they appear. If one parameter overrides another, it will replace that value. Some parameters will force the re-calculation of others, *e.g.,* the left and right sides of a rectangle will be changed if you specify the center coordinates.

Shape drawing parameters may be intermixed with descriptive parameters. All the descriptive parameters in the statement will be processed first. The shape drawing parameters will then be processed in the order given.

Note: If you use autoscaling, the aspect ratio of shapes drawn may be distorted from what you intended. To prevent this from happening, use the **EQUALSCALES** parameter on the **GRAPH** statement. This will force the horizontal and vertical scale factors to be identical and shapes will keep their intended ratios (circles will remain circles, etc.)

Note: It is strongly recommended that you do **NOT** use logarithmic scales for graphs which include shapes. Shapes may become badly distorted and negative or zero values will cause errors.

Shape Statement Descriptive Parameters

The SHAPE statement has many descriptive parameters, not all of which apply to any given shape. First, here is a list of the descriptive parameters.

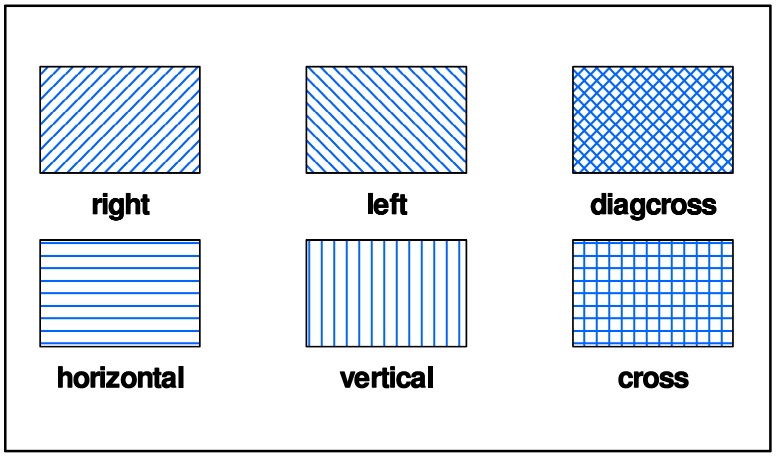
BOLD = *num-expr*
BOTTOM= *num-expr*
CENTER=(*x, y*)
CHARACTERS = *string-expr*
CTLPT1=(*x, y*)
CTLPT2=(*x, y*)
DIRECTION=[NORMAL | REVERSE]
ENDANGLE=*num-expr*
FILLCOLOR=[(*r,g,b*) | *num-expr*]
FILLTYPE=[SOLID | HATCH | NONE]
FONT = *string-expr*
HATCH=[HORIZONTAL | VERTICAL | CROSS | RIGHT | LEFT | DIAGCROSS]
HRADIUS=*num-expr*
ITALIC = *num-expr*
LEFT=*num-expr*

LINECOLOR=[(*r,g,b*) | *num-expr*]
LINEEND=(*x, y*)
LINESTART=(*x, y*)
LINEWEIGHT=*num-expr*
MOUSECLICK
POLYCOUNT = *num-expr*
POLYPOINTS = *array-name*
RADIUS=*num-expr*
RESET
RIGHT=*num-expr*
ROTATE=*num-expr*
ROTATECENTER=[(*x ,y*) | **"CENTER"**]
STARTANGLE=*num-expr*
TOP=*num-expr*
VRADIUS= *num-expr*

These parameters are described in detail, below:

BOLD = *num-expr*	Specifies the weight (from 0 to 1000) of the font used in the **SHAPE TEXT** function. Default is 0, which means use a standard or "normal" weight for the font. (See "Advanced Graphics".)
BOTTOM= *num-expr*	Sets the bottom edge of the rectangle (or bounding rectangle for **ELLIPSE**, **CHORD**, **ARC**, or **PIE**).
CENTER=(*x, y*)	Sets the center point (*x, y*) of the figure, or the center of the defining ellipse for **ARC**, **CHORD** and **PIE**.
CHARACTERS = *string-expr*	Specifies the string of characters to be used in a **SHAPE TEXT** function (see "Advanced Graphics".)
CTLPT1=(*x, y*)	Specifies the first control point for a **BEZIER** curve (see "Advanced Graphics".
CTLPT2=(*x, y*)	Specifies the second control point for a **BEZIER** curve.
DIRECTION=[NORMAL | REVERSE]	Specifies the drawing direction for shapes. Used with Complex Connected Shapes (see "*Advanced Graphics*")

ENDANGLE=*num-expr*	Specifies the ending angle for a **CHORD**, **ARC**, or **PIE**.
FILLCOLOR=[(*r,g,b*) | *num-expr*]	Specifies the color with which the shape will be filled (if **FILLTYPE** = SOLID) or the hatch pattern drawn (if **FILLTYPE** = HATCH). Default color is black. (See “Specifying Colors”, below on how to specify colors.)
FILLTYPE=[SOLID | HATCH | NONE]	Specifies how the shape will be filled. If SOLID, you must also specify **FILLCOLOR**. If HATCH, you must also specify **HATCH** (see below). If NONE, the figure will be drawn with no fill (empty or transparent). **Caution:** If a graph includes very large shapes with SOLID color fills, you can use up a **lot of ink** when printing the graph.
FONT = *string-expr*	Specifies the name of a font to be used for the **SHAPE TEXT** function. This must be an outline font (TrueType, Type 1, etc) installed on your computer, and must match the font name exactly (except for case). If there is no match, a generic sans-serif font will be used. (See “Advanced Graphics”.) **Note:** You can use the function **CHOOSEFONT$** to select a valid font name.
HATCH=[HORIZONTAL | VERTICAL | CROSS | RIGHT | LEFT | DIAGCROSS]	Specifies the hatch pattern used to fill the figure (if **FILLTYPE**=HATCH). **Note:** Hatch patterns may appear different on different printers. On some printers, hatch patterns may not work.



HRADIUS=*num-expr*	Sets the horizontal radius of an ellipse (or the defining ellipse for **ARC**, **CHORD** and **PIE**), or the distance from the center to the right or left edge of a rectangle.
ITALIC = *num-expr*	Specifies if you want normal or italic for the font used in the **SHAPE TEXT** function. Default is 0, which means normal. 1 means italic. (See "Advanced Graphics".)
LEFT=*num-expr*	Sets the left edge of the rectangle (or bounding rectangle for **ELLIPSE**, **CHORD**, **ARC**, or **PIE**).
LINECOLOR=[(*r,g,b*) | *num-expr*]	Specifies the color of the line outlining the figure, or the line used to draw **ARC**s, **LINE**s, **POLYLINE**s and **BEZIER** Curves. Default color is black. (See "Specifying Colors", below on how to specify colors.)
LINEEND=(*x, y*)	Specifies the ending point of a **LINE**.
LINESTART=(*x, y*)	Specifies the starting point of a **LINE**.
LINEWEIGHT=*num-expr*	Specifies the thickness (in points) of the line outlining the figure, or the line used to draw **ARC**s, **LINE**s, **POLYLINE**s and **BEZIER** Curves. Default is 1 point. A value of zero means a line one pixel thick.

	<u>**Note**</u>: Line weights are rounded (up or down) to the nearest pixel, and so might appear slightly different when printed.
MOUSECLICK	Allows this shape to be selected by clicking on it (See “Advanced Graphics”).
POLYCOUNT = *num-expr*	Specifies the number of points in the polygon or polyline. It must not be greater than the dimension of the array. For polygons, do not count the first point twice (since it is assumed to also be the end point.)
POLYPOINTS = *array-name*	Specifies the name of previously dimensioned <u>array</u> containing the points to be used in the polygon or polyline. The array must be dimensioned as follows: **DIM** *array-name* (*n*, 2) where *n* is greater then or equal to the number of points in the polygon or polyline. *array-name*(*n*,0) contains the x-coordinate and *array-name*(*n*,1) contains the y-coordinate of the n^{th} point. *n* is 0 for the first point. Since a <u>polygon</u> is closed, you do not specify the first point again – the program assumes that. Therefore, a 5-sided polygon will have exactly 5 points. This parameter is also used to specify a destination array for the **CHAROUTLINE** function or the source for the **PLOTCHAROUTLINE** function (see “Advanced Graphics”)
RADIUS=*num-expr*	Sets the radius of a circle (or the defining circle for **ARC**, **CHORD** and **PIE**) or the distance from the center to any edge of a square. Forces the figure to be square or circular – if you don’t want this, you may specify the horizontal and vertical radii separately.
RESET	Resets all descriptive parameters to their original default values. This includes parameters previously specified on the same

	GRAPH statement, so it should be on a statement line by itself.
RIGHT=*num-expr*	Sets the right edge of the rectangle (or bounding rectangle for **ELLIPSE**, **CHORD**, **ARC**, or **PIE**).
ROTATE=*num-expr*	Specifies the angle or rotation for any shape. *num-expr* specifies the <u>*positive*</u> angle in ***radians***, unless **DEGREES** was specified.
ROTATECENTER=[(*x* ,*y*) | **"CENTER"**]	Specifies the center of rotation for figures which are rotated (see "Rotating Shapes", below). If "center" is specified (quotes required), rotates around the center of the object.
STARTANGLE=*num-expr*	Specifies the starting angle for a **CHORD**, **ARC**, or **PIE**
TOP=*num-expr*	Sets the top edge of the rectangle (or bounding rectangle for **ELLIPSE**, **CHORD**, **ARC**, or **PIE**).
VRADIUS= *num-expr*	Sets the vertical radius of an ellipse (or the defining ellipse for **ARC**, **CHORD** and **PIE**), or the distance from the center to the top or bottom edge of a rectangle.

<u>**Note**</u>: The above parameters will <u>override each other</u>, and are calculated in the order given.

For example, if you specify the sides of a rectangle (**LEFT** and **RIGHT**) explicitly, the horizontal radius (**HRADIUS**) is defined. If you then specify the center point (**CENTER**), the figure will be re-positioned using the same dimensions but centered at the new center point. If you then specify a new horizontal radius (**HRADIUS**), the size will change, but still be centered about the new center (**CENTER**). The <u>***most recently specified***</u> parameters will be the ones in effect.

If you are not certain which parameters are in effect, you can always specify all the parameters you need over again. The reason previous values are saved and re-used is so that you won't have to specify parameters over again unless you want them to change.

If you don't know which parameters are in effect, or can't determine them at run time, you can use the **RESET** parameter. This resets all the descriptive parameters to their original, default values. Then you can start specifying them again.

Note: Figures like **ARC**, **CHORD**, and **PIE** are segments of an ellipse. The size and positioning parameters specify the size and position of that ellipse.

Note: The **LEFT** side of a figure may not be greater (farther to the right) than the **RIGHT** side. That is to say that the figure may not have a negative width. The same is true for the **TOP** and **BOTTOM**. If you set **LEFT** greater then **RIGHT**, the right side will be set equal to the left, giving a zero-width figure. You will then need to set the **RIGHT** parameter again.

Normally, you don't need to set all these parameters. Specifying **LEFT**, **RIGHT**, **TOP** and **BOTTOM** completely defines the size of a shape. Similarly, you can define it using **CENTER** and **RADIUS** (or **HRADIUS** and **VRADIUS**). Use whichever is most convenient for you.

Notes: For a **CHORD**, **ARC**, or **PIE**, The figure is drawn counter-clockwise from the **STARTANGLE** to the **ENDANGLE**.

Exception: If the **DIRECTION** parameter is set to **REVERSE,** the figure is drawn clockwise from **ENDANGLE** to **STARTANGLE**.

A zero angle represents the line extending to the right of the center point.

Angles are specified in radians unless the **DEGREES** statement was given.

Angles should be between -3600 and 3600 degrees (or -20π and $20\ \pi$ radians).

Shape Statement Shape Drawing Parameters.

The following parameters specify shapes to be drawn. Each shape is drawn **using the descriptive parameters in effect** at the time the figure is drawn. If the shape drawing parameter appears in the same **SHAPE** statement as some descriptive parameters, the descriptive parameters will be processed first.

The shapes will be drawn with the current **LINEWEIGHT** and **LINECOLOR**. They will then be filled according to the current **FILLTYPE**. If **FILLTYPE**=SOLID, they will be filled with the current **FILLCOLOR**. If **FILLTYPE**=HATCH, they will be filled with the current **HATCH** pattern. If **FILLTYPE**=NONE, they will be outlined but left unfilled.

ARCs, **CHORD**s and **PIE**s will also use the current **STARTANGLE** and **ENDANGLE**.

The shapes will be drawn at the current position and size, as calculated by the previous descriptive parameters.

RECTANGLE	Draws a rectangle (or square).
ELLIPSE	Draws an ellipse (or circle).
ARC	Draws an arc from **STARTANGLE** to **ENDANGLE**. The arc is not filled.
CHORD	Draws a chord from **STARTANGLE** to **ENDANGLE**. A chord consists of a shape bounded by an arc and a line between the endpoints of the arc. The shape is filled.
PIE	Draws a "pie segment" from **STARTANGLE** to **ENDANGLE**. A pie segment is the shape bounded by an arc and lines between the endpoints of the arc and the center of the ellipse. **Note**: If a pie chart is drawn using an ellipse rather than a circle, the angles will be adjusted so that the endpoints of each arc are proportionally scaled in the same manner as the ellipse. The figure will appear as if the entire pie chart were tilted away from the viewer. If you want the angles to be accurate, use a circular shape (a square bounding rectangle, or specify **RADIUS**) and also specify **EQUALSCALES** in the **GRAPH** statement.
LINE	Draws a line from **LINESTART** to **LINEEND**.

POLYGON	Draws a closed polygon using the points specified in the **POLYPOINTS** array. The number of points in the array is specified in the **POLYCOUNT** parameter. __Note__: Once the **POLYGON** has been specified, the values in the array may be changed, if desired, and a different polygon may reference the same array.
POLYLINE	Draws a series of connected line segments using the points specified in the **POLYPOINTS** array. The number of points in the array is specified in the **POLYCOUNT** parameter (see "Advanced Graphics").
TEXT	Draws outline text shapes (see "Advanced Graphics").
BEZIER	Draws a cubic Bezier curve from **LINESTART** to **LINEEND**, using the control points specified by **CTLPT1** and **CTLPT2** (see "Advanced Graphics").
BEGINPATH	Begins a complex connected shape (see "Advanced Graphics".)
ENDPATH	Ends a complex connected shape (see "Advanced Graphics".)
CLOSEFIGURE	Ends one part of a complex connected shape, closing it up back to its starting point. This allows you to start a new part, such as a hole cut in the first part (see "Advanced Graphics".)
STROKE	Outlines a complex connected shape (see "Advanced Graphics").
STROKEFILL	Outlines and fills a complex connected shape (see "Advanced Graphics".)
CHAROUTLINE	Generates an array of information describing the shape of a font character (see "Advanced Graphics")
PLOTCHAROUTLINE	Plots an array of information describing the shape of a font character (see "Advanced Graphics")

__**Rotating Shapes**__.

By default, all shapes are drawn with the positioning and orientation specified by the parameters in the **SHAPE** statement. You may, optionally, rotate any or all shapes using the **ROTATE** and **ROTATECENTER** parameters.

ROTATE=*angle* specifies the amount of **counter-clockwise** rotation. The angle is specified in radians, unless the **DEGREES** statement is in effect. In either case, the angle must be between -3600 and +3600 degrees (or -20π and $+20\pi$ radians). The default value is zero (no rotation).

Shapes are rotated **around the center of the shape**, it the shape has a center. Ellipses, pie segments, chords, arcs, and rectangles each have a defined or implied center. Lines, Bezier curves, polygons, and polylines do not have centers. Neither do "complex connected shapes" (see "Advanced Graphics"). If a shape does not have a center, it is rotated around the **last center specified**, *i.e.,* the center of the last shape plotted. If no shape with a center has been plotted yet, the shape is rotated around the origin of the graph.

You can specify a "center" for a line, Bezier curve, polygon, or polyline by specifying **CENTER**=(x, y) when describing the shape. Although it has no effect on the way the shape is drawn, it will define the center of rotation for that shape.

The center of rotation for a string of outlined text (See **SHAPE TEXT** in "Advanced Graphics") is defined by the **BOTTOM** and **LEFT** parameters.

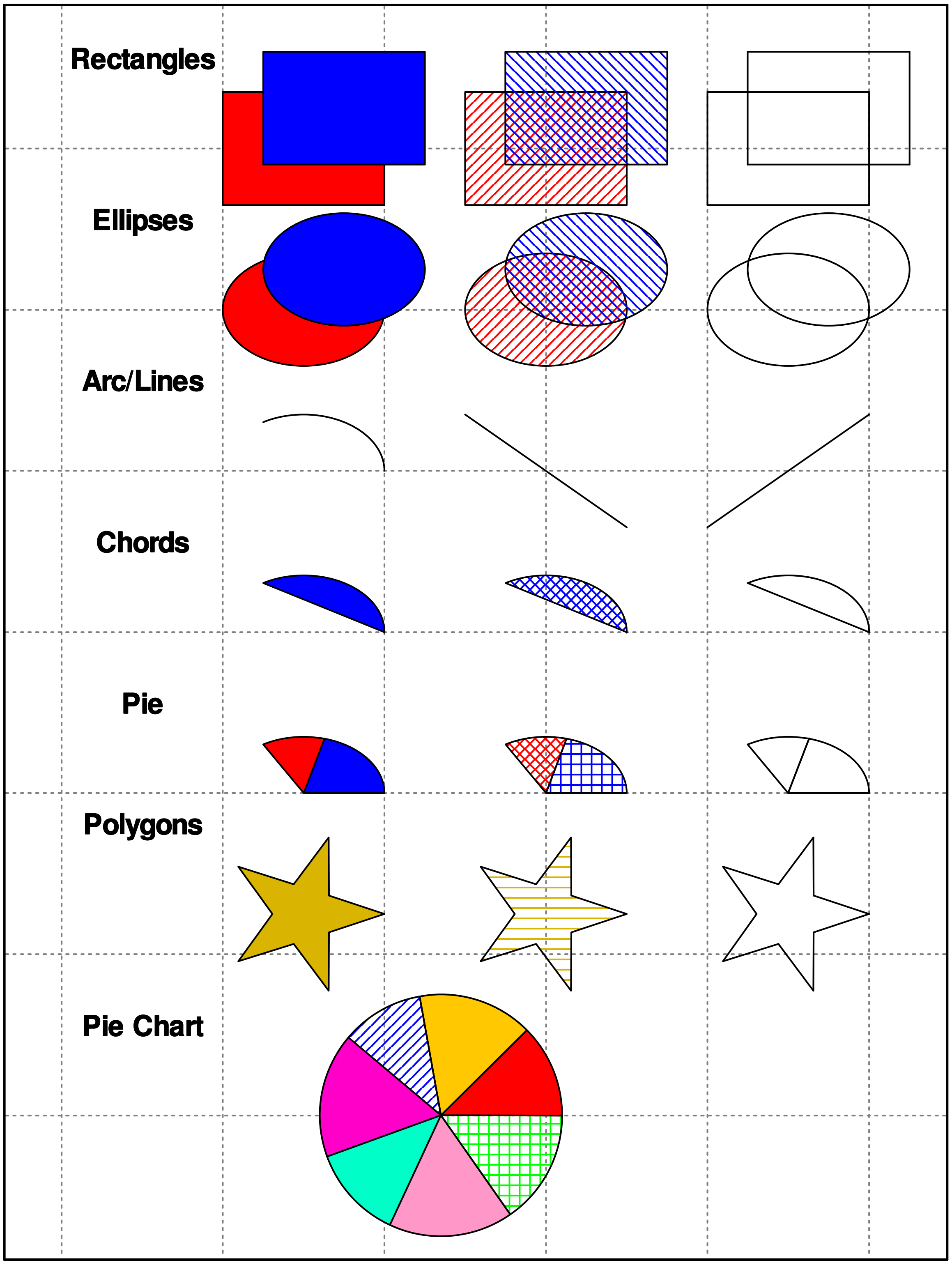
If you want to rotate a shape around a point other than its center, you may specify **ROTATECENTER**=(x, y). This overrides the center of rotation for all subsequent shapes. To reset back to using the shape's own center of rotation, specify **ROTATECENTER="center"** [quotes are required]. If no center has been specified, the center of rotation defaults to the origin of the graph.

Remember that **ROTATE** and **ROTATECENTER** remain in effect for subsequent shapes until they are changed. This can cause some strange-looking graphs if you forget to change them back. (You can use **RESET**, if necessary.) However, this can make it easy to rotate a group of objects around a common center.

Setting **ROTATE**=0 resets to no rotation, in which case **ROTATECENTER** has no effect, *but it is still remembered.*

Example Shapes.

The following chart illustrates *some of* the simple shapes that can be produced using the QuickCalc **SHAPE** statement:



The chart above was produced using QuickCalc with the sample program "graph_filled_shapes_demo.txt", which you can download from the website.

Specifying Colors.

This section is applicable to any parameter which specifies a color, in **GRAPH**, **PLOT**, **TEXT**, and **SHAPE** statements, whether specifying a line color , fill color, text color, background color or grid color.

Colors may be specified in one of two ways:

1. Colors may be specified using three values (enclosed in parentheses and separated by commas). It is specified as: **COLOR** = (*r, g, b*).

   r, *g*, and *b* represent the red, green and blue components of the desired color. Each component is a numeric expression which must evaluate to a number between 0 and 255.

   0 means none of the specified color, and 255 means the maximum of that color. (0, 0, 0) is black. (255, 255, 255) is white. You can mix over 16 million different colors this way.

2. A color may be specified by referencing a numeric variable or expression which evaluates to that color value. For example, if you have set a variable *red* to have the color value (255, 0, 0), then you can specify **LINECOLOR**=*red* instead of having to code **LINECOLOR**=(255, 0, 0). This makes it easier when you have a lot of different colors in your graph, or many elements which reference the same color, since changing the value of the color variable affects all the shapes which reference it.

   Note: Color values are stored in variables in the form:
   $r + (256 * g) + (256 * 256 * b)$.

   The function **SETCOLOR**=(*r, g, b*) is useful in defining colors. If you have the statement *cyan* = **SETCOLOR** (0, 255, 255) then you can use the variable *cyan* instead of (0, 255, 255).

   You can pre-define your favorite colors with **SETCOLOR** statements and store them in INCLUDE files. Then a simple **INCLUDE** statement pre-defines all your favorite colors.

   Note: Once a variable has been defined to be a color value, be careful not to use the same variable as a data item somewhere else in your program.

Some examples of color are (see following page):

(128,128,128) 50% Gray	(97,97,97) 62% Gray	(64,64,64) 75% Gray	(33,33,33) 83% Gray	(0,0,0) Black	(216,180,0) Gold
(128,0,0) Mahogany	(255,255,255) White	(240,240,240) 6% Gray	(224,224,224) 12% Gray	(192,192,192) 25% Gray	(161,161,161) 37% Gray
(255,128,0) Orange	(128,64,0) Chestnut	(161,97,33) Caramel	(255,128,128) Salmon	(255,0,0) Red	(194,0,0) Scarlet
(194,255,128) Lime	(255,255,128) Cream	(255,255,0) Yellow	(194,194,0) Mustard	(128,128,0) Camel	(255,255,128) Peach
(0,255,0) Green	(0,194,0) Jade	(0,128,0) Pine Green	(64,128,0) Moss Green	(97,161,33) Olive	(129,255,0) Apple Green
(0,128,128) Slate Blue	(128,255,192) Seafoam Green	(0,255,128) Aquamarine	(0,128,64) Peacock Green	(33,161,97) Emerald	(128,255,128) Ice Green
(33,97,161) Midnight Blue	(0,128,255) Copenhagen Blue	(128,194,255) Denim	(128,255,255) Sky Blue	(0,255,255) Cyan	(0,194,194) Turquoise
(97,33,161) Plum	(128,128,255) Azure	(0,0,255) Blue	(0,0,194) Sapphire	(0,0,128) Navy Blue	(0,64,128) Indigo
(255,0,255) Magenta	(194,0,194) Mulberry	(128,0,128) Maroon	(192,128,255) Lavender	(128,0,255) Purple	(64,0,128) Grape
(153,102,0) Brown	(128,0,64) Crimson	(161,33,97) Russet	(255,0,128) Carnation Red	(255,128,194) Blush	(255,128,255) Lilac

Of course, you can specify any color combination you want. If one of the colors in the chart is close to what you want, you can lighten it by increasing all three values slightly or darken it by decreasing all three values. Add more of one color and subtract the other two for a more pure primary color, or increase the other two for a more pastel look. Feel free to experiment.

The colors shown in the above chart are pre-defined in the file ***standard_colors.txt,*** which is in the sample programs on the website. You can download that file and include it into your program to get all the above colors pre-defined. You can also download and include a shorter version from the file ***basic_colors.txt***, which contains some of the more common colors.

You can, of course, create your own include file containing your favorite colors which can be any of the 16 million combinations.

Note: The above chart was produced in QuickCalc with the sample program "***color_chart.txt***", which you can download from the website. Printing this chart will give you an idea of how these colors will look on your printer.

Advanced Graphic Functions

See the separate document "Advanced Graphics".