

RedCrab

The Calculator

Version 4.40 News

copyright © by Redchillicrab, Singapore 2009 - 2013

Version 4.40 news (User Manual)

13.9 Plot Box

The menu ***Plotbox*** opens a plot box on the worksheet, where you can display calculation results graphically. The initialisation handling is like result- and chart boxes. Click the right mouse button on the plot box to open the popup menu. The menu ***References*** opens a dialog window for the plot box reference variable input.

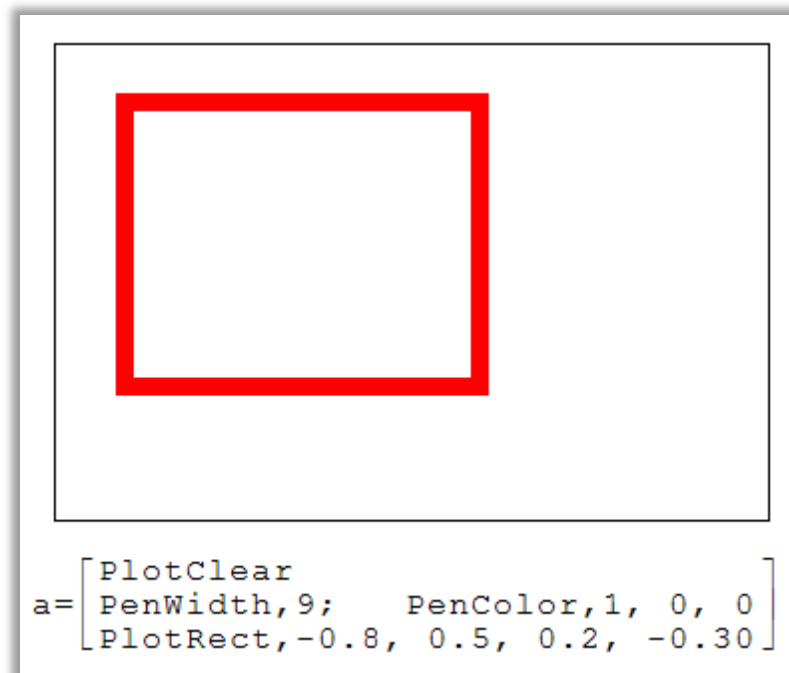
The difference to chart boxes is that plot boxes works with low level graphic commands. The chart box shows data of a variable directly in lines or bars. The handling is easy, but the possibility is limited to the chart box series and options.

The plot box works with data array that contains a series of graphic commands and parameters. That means the results cannot be displayed directly like in chart boxes. The result data must first be converted in a series of graphic commands. The design of the graphic is not limited.

Plotbox Data Format

The reference variable of a plot box must contain a two dimensional data field. The first cell contains the graphic command, the following cells contain different numbers of parameters.

Example:

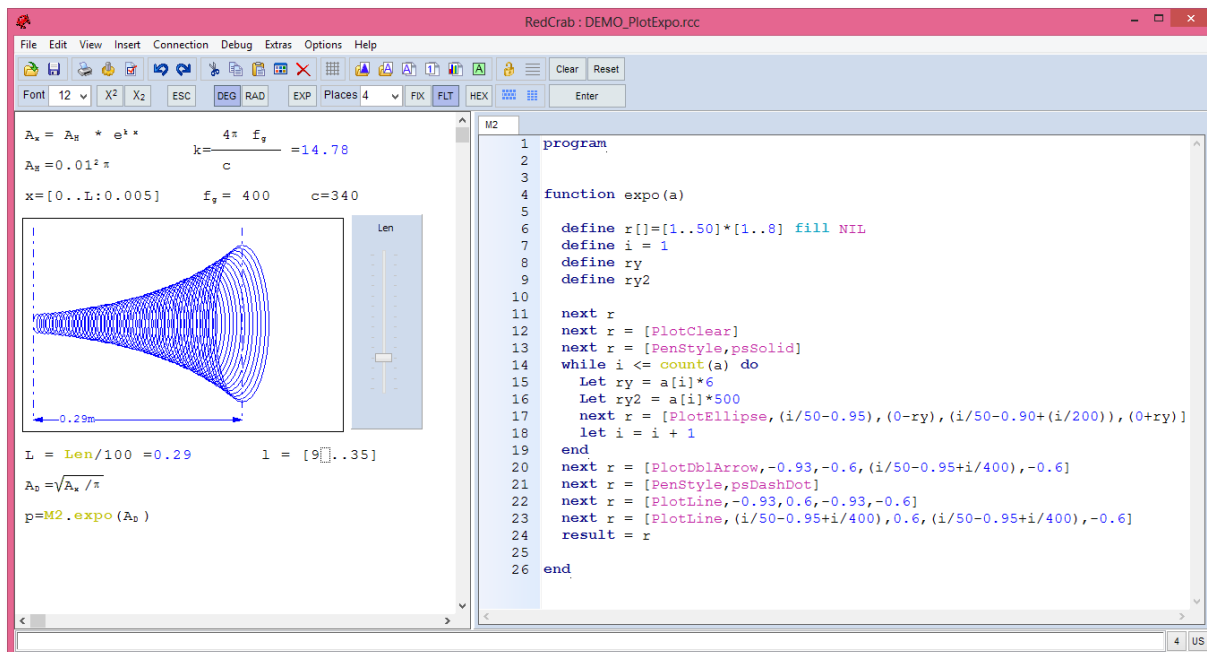


In the example above, the reference variable '*a*' contains a four row command sequence, which draw a red rectangle.

1. **PlotClear** fills the box with the preselect background colour.
2. **PenWidth** set the pen width to 9 pixels. The semicolon next to the 9 starts a new row.
3. **PenColor** sets the colour to red.
4. **PlotRect** draws the rectangle.

The integration of graphic commands in data fields provides the import of graphics or parts of them in functions, or export as result of functions.

The example below shows the calculation of a loudspeaker exponential horn. The worksheet sends the dimensions data for the horn to the extern function **M2.expo**. **M2.expo** returns a data field, which contains the graphical description of the horn, to the variable *p*. The variable *p* is the reference of the plot box.



13.9.1 Colour Components

In RedCrab, a single colour is represented as a mixture of red, green, and blue components. The range for each component is preset from 0.0 to 1.0. The values for each component can be any floating-point value between 0 and 1.

Syntax: [SetPenColor, red, green, blue]

Example: [SetPenColor, 0.98, 0.625, 0.12]

The example above sets pen colour to orange. The table below lists some common colours and their component values. These values can be used with any RedCrab colour-related functions.

Composite Colour	Red Component	Green Component	Blue Component
White	1.0	1.0	1.0
Black	0.0	0.0	0.0
Red	1.0	0.0	0.0
Green	0.0	1.0	0.0
Blue	0.0	0.0	1.0
Yellow	1.0	1.0	0.0
Magenta	1.0	0.0	1.0
Cyan	0.0	1.0	1.0
Dark gray	0.25	0.25	0.25
Light gray	0.75	0.75	0.75
Brown	0.60	0.40	0.12
Orange	0.98	0.625	0.12
Pink	0.98	0.04	0.70
Purple	0.60	0.40	0.70

RGB components are similar to Windows specifications. The difference is that in Windows, each colour component range from 0 to 255. If you prefer this component range, you can toggle the range with the statement: [AbsColor,TRUE].

13.9.2 Plot Coordination

Similar to the colour components, the plot X- and Y-coordination can be defined in floating point or absolute pixel values.

The floating point mode is preset. You can use any valid positive or negative floating point value to determine a position. The lowest values represent the lower left corner.

RedCrab set axis limits automatically. You can specify the axis limits yourself, with use of **PlotRange**.

Instead floating point values, you can use Windows pixel coordinates. Before you can use pixel coordinates, you must write the statement: [AbsPosition,TRUE]. According to Windows system, the pixel position x=1 / y=1 is the plot box upper left corner.

The graphic size is absolute and does not correspond to the plot box size.

13.9.3 Plot Commands

The table below shows the valid graphic command

PlotClear	Fills the plot box using the current clear colour.
ClearColor	Specify the clear colour.
PlotSize	Specify the width and height of the plot box in pixels.
AbsPosition	Determine the X- and Y-coordination range.
AbsColor	Determine the colour component range.
PlotRange	Specify the X/Y axis minimum and maximum range.
PlotBorder	Switch the border line around the box on or off.
BorderColor	sets the border lines colour
MoveTo	Changes the current drawing position.
LineTo	Draws a line from the current drawing position to specified point.
PlotLine	Draws a line between two specified points.
PlotArrow	Draws a line with an arrow head.
PlotDbIArrow	Draws a line with an arrow head on both tails.
FrameRect	Draws a rectangle using the current brush to draw the border.
FillRect	Fills the specified rectangle using the current brush.

PlotRect	Draws a rectangle using the current pen to draw the border.
RoundRect	Draws a rectangle with rounded corners.
PlotEllipse	Draws an ellipse using the current pen.
PlotArc	Draws an arc along the perimeter of an ellipse.
PlotChord	Draws a closed figure represented by the intersection of a line and an ellipse.
PlotPie	Draws a pie-shaped section of an ellipse
PolyBezier	Draws a Bezier curve.
PlotPolygon	Draws a closed, multi-sided shape.
KeepPolygon	Is an extension of <i>PlotPolygon</i> .
PenColor	Determines the colour used to draw lines
PenWidth	Specify the width of the pen in pixels.
PenStyle	Determines the style in which the pen draws lines.
PenMode	Indicates how the pen colour interacts with the colour of the plot box.
BrushColor	Determines the colour of the brush.
BrushStyle	Specify the pattern of a brush.
PlotText	Write a string into the plot box.
FontSize	Specify the point size of the font.
FontStyle	Specify style characteristics of a font.
FontColor	Specify the colour of the text.
FontName	Identifies the typeface of the font.

13.9.4.1 PlotClear

PlotClear fills the plot box using the current clear colour

Syntax: [PlotClear]

13.9.4.2 ClearColor

ClearColor specifies the clear colour

Syntax: [ClearColor, red, green, blue]

The ***ClearColor*** property determines the background colour. This is the colour that used ***PlotClear*** to fill the background.

13.9.4.3 PlotSize

PlotSize specifies the width and height of the plot box in pixels.

Syntax: [PlotSize, width, height]

PlotSize resize the plot box to the specified width and height and fills the background using the **ClearColor** specified colour.

13.9.4.4 AbsPosition

AbsPosition toggles position coordination from floating point to Windows specifications

Syntax: [AbsPosition, {TRUE | FALSE}]

Preset for coordination are floating point values. [AbsPosition, TRUE] toggles to integer pixel coordination, similar to the Windows specifications. [AbsPosition, FALSE] toggles back to floating points. For more information read the description **Plot Coordination** above.

13.9.4.5 AbsColor

AbsColor toggles the colour component range.

Syntax: [AbsColor, {TRUE | FALSE}]

AbsColor toggles the colour component range. Preset is floating point values from 0.0 to 1.0. [AbsColor, TRUE] toggles the range to integer values from 0 to 255,

similar to the Windows specifications. [AbsColor, FALSE] toggles back in floating point mode.

13.9.4.6 PlotRange

PlotRange specifies the X/Y axis minimum and maximum range.

Syntax: [PlotRange, minX, maxX, minY, maxY]

The X and Y axis has preset a variable range. **RedCrab** sets the limits automatically according to the graphic size. **PlotRange** sets the axis to fixed limits. You can set the limits to any valid positive or negative floating point values.

13.9.4.7 PlotBorder

Switch the border line around the box on or off

Syntax: [PlotBorder, {TRUE | FALSE}]

PlotBorder toggle the one pixel width borderline on or off. Preset is on.

13.9.4.8 BorderColor

BorderColor sets the border lines colour

Syntax: [BorderColor, red, green, blue]

13.9.5.1 MoveTo

MoveTo changes the current drawing position to the point *X, Y*.

Syntax: [MoveTo, X, Y]

Use **MoveTo** to set the current drawing position before calling **LineTo**.

13.9.5.2 LineTo

LineTo draws a line on the plot box from the current drawing position to the point specified by *X, Y*.

Syntax: [LineTo, X, Y]

LineTo draws a line from current drawing position up to, but not including the point *X, Y*. **LineTo** changes the current drawing position to *X, Y*. The line is drawn using **PenColor** and **PenStyle**.

Note: if the current Pen does not have a style of **psSolid**, the line is drawn with a background specified by the current brush.

13.9.5.3 PlotLine

PlotLine draws a line on the plot box from the point specified by *X1, Y1* to the point specified by *X2, Y2*.

Syntax: [PlotLine, X1, Y1, X2, Y2]

PlotLine draws a line from the point specified by *X1, Y1* up to, but not including the point *X2, Y2*. **PlotLine** changes the current drawing position to *X2, Y2*. The line is drawn using **PenColor** and **PenStyle**.

Note: if the current Pen does not have a style of *psSolid*, the line is drawn with a background specified by the current brush.

13.9.5.4 PlotArrow

PlotArrow draws a line with an arrow head on the plot box.

Syntax: [PlotArrow, topX, topY, tailX, tailY]

PlotArrow draws a line with an arrow head from the point specified by *topX*, *topY* up to, but not including the point *tailX*, *tailY*. **PlotArrow** changes the current drawing position to *tailX*, *tailY*. The line is drawn using *PenColor* and *PenStyle*.

Note: if the current Pen does not have a style of *psSolid*, the line is drawn with a background specified by the current brush.

13.9.5.5 PlotDblArrow

PlotDblArrow draws a line with an arrow head on both tails.

Syntax: [PlotDblArrow, X1, Y1, X2, Y2]

PlotDblArrow draws a line with an arrow head on both tails from the point specified by *X1*, *Y1* up to the point *X2*, *Y2*. **PlotDblArrow** changes the current drawing position to *X2*, *Y2*. The line is drawn using *PenColor* and *PenStyle*.

Note: if the current Pen does not have a style of *psSolid*, the line is drawn with a background specified by the current brush.

13.9.5.6 FrameRect

FrameRect draws a 1 pixel wide border around a rectangular region.

Syntax: [FrameRect, X1, Y1, X2, Y2]

Use **FrameRect** to draw a 1 pixel wide border around a rectangular region specified by *X1*, *Y1* and *X2*, *Y2*. **FrameRect** draws the rectangle using **BrushColor**. **FrameRect** does not fill the interior of the rectangle with the Brush pattern. To draw a boundary using the Pen instead, use the **PlotPolygon** method.

13.9.5.7 FillRect

FillRect fills the specified rectangle on the plot box using the current **FillColor**.

Syntax: [FillRect, X1, Y1, X2, Y2]

Use **FillRect** to fill a rectangular region using the current **Brush**. The region is filled including the top and left sides of the rectangle, but excluding the bottom and right edges.

13.9.5.8 PlotRect

PlotRect draws a rectangle on the plot box.

Syntax: [PlotRect, X1, Y1, X2, Y2]

Syntax: [PlotRect, X1, Y1, X2, Y2, 1]

Use **PlotRect** to draw a rectangle using **Pen**. Specify the rectangle's coordinates in one this way: Giving four coordinates that define the upper left corner at the point (X1, Y1) and the lower right corner at the point (X2, Y2). An optional value, not equal to 0 in the sixth field, fills the rectangle using Brush colour and style.

To fill a rectangular region without drawing the boundary in the current pen, use ***FillRect***. To draw a rectangle with rounded corners, use ***RoundRect***.

13.9.5.9 RoundRect

RoundRect draws a rectangle with rounded corners on the plot box.

Syntax: [RoundRect, X1, Y1, X2, Y2, X3, Y3]

Syntax: [RoundRect, X1, Y1, X2, Y2, X3, Y3, 1]

Use ***RoundRect*** to draw a rounded rectangle using Pen. The rectangle will have edges defined by the points (X1,Y1), (X2,Y1), (X2,Y2), (X1,Y2), but the corners will be shaved to create a rounded appearance. The curve of the rounded corners matches the curvature of an ellipse with width X3 and height Y3.

An optional value, not equal to 0 in the eighth field, fills the rectangle using Brush colour and style.

13.9.5.10 PlotEllipse

Draws the ellipse defined by a bounding rectangle on the canvas

Syntax: [PlotEllipse, X1, Y1, X2, Y2]

Syntax: [PlotEllipse, X1, Y1, X2, Y2, 1]

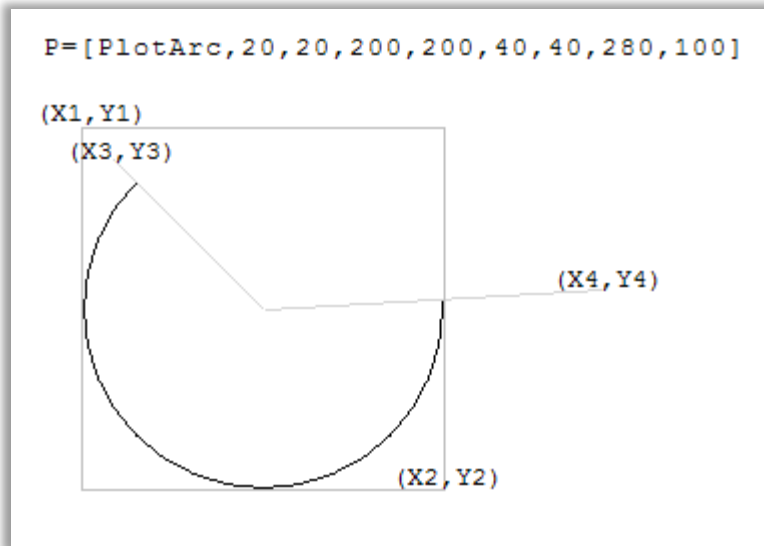
Use ***PlotEllipse*** to draw a circle or ellipse on the plot box. Specify the bounding rectangle either by giving the top left point at pixel coordinates (***X1***, ***Y1***) and the bottom right point at (***X2***, ***Y2***). If the bounding rectangle is a square, a circle is drawn. The ellipse is outlined using the value of ***PenColor***. An optional value, not equal to 0 in the sixth field, fills the ellipse using Brush colour and style.

13.9.5.11 PlotArc

PlotArc draws an arc on the plot box along the perimeter of the ellipse bounded by the specified rectangle.

Syntax: `[PlotArc, X1, Y1, X2, Y2, X3, Y3, X4, Y4]`

Use **PlotArc** to draw an elliptically curved line with the current **PenColor**. The arc traverses the perimeter of an ellipse that is bounded by the points (X1, Y1) and (X2, Y2). The arc is drawn following the perimeter of the ellipse, counterclockwise, from the starting point to the ending point. The starting point is defined by the intersection of the ellipse and a line defined by the center of the ellipse and (X3, Y3). The ending point is defined by the intersection of the ellipse and a line defined by the center of the ellipse and (X4, Y4). The arc is drawn using the value of **PenColor**



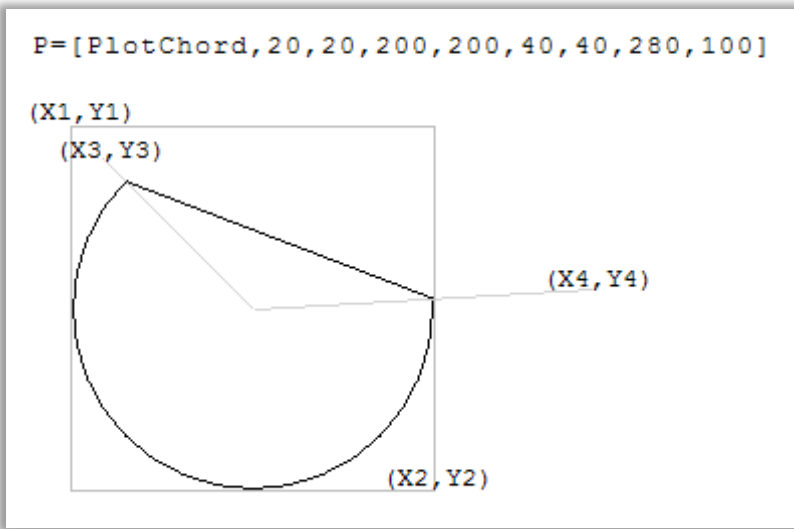
13.9.5.12 PlotChord

PlotChord draws a closed figure represented by the intersection of a line and an ellipse.

Syntax: `[PlotChord, X1, Y1, X2, Y2, X3, Y3, X4, Y4]`

Syntax: `[PlotChord, X1, Y1, X2, Y2, X3, Y3, X4, Y4, 1]`

Use **PlotChord** to create a shape that is defined by an arc and a line that joins the endpoints of the arc. The chord consists of a portion of an ellipse that is bounded by the points $(X1, Y1)$ and $(X2, Y2)$. The ellipse is bisected by a line that runs between the points $(X3, Y3)$ and $(X4, Y4)$. The perimeter of the



chord runs counterclockwise from $(X3, Y3)$, counterclockwise along the ellipse to $(X4, Y4)$, and straight back to $(X3, Y3)$. If $(X3, Y3)$ and $(X4, Y4)$ are not on the surface of the ellipse, the corresponding corners on the chord are the closest points on the perimeter that intersect the line.

The outline of the chord is drawn using the value of **PenColor**. An optional value, not equal to 0 in the tenth field, fills the chord using Brush colour and style.

13.9.5.13 PlotPie

Draws a pie-shaped the section of the ellipse bounded by the rectangle $(X1, Y1)$ and $(X2, Y2)$ on the plot box.

Syntax: `[PlotPie, X1, Y1, X2, Y2, X3, Y3, X4, Y4]`

Syntax: `[PlotPie, X1, Y1, X2, Y2, X3, Y3, X4, Y4, 1]`

Use **PlotPie** to draw a pie-shaped wedge on the plot box. The wedge is defined by the ellipse bounded by the rectangle determined by the points $(X1, Y1)$ and $X2, Y2)$. The section drawn is determined by two lines radiating from the center of the ellipse through the points $(X3, Y3)$ and $(X4, Y4)$. The wedge is outlined using Pen and optionally brush-filled, if the tenth field not equal to 0.

13.9.5.14 PolyBezier

PolyBezier draws a Bezier curve.

Syntax: [PolyBezier, X1, Y1, X2, Y2, X3, Y3]

Use *PolyBezier* to draw a cubic Bezier curve from the current *Pen* position to the endpoints specified by the parameter *X3, Y3*. The first and second control points are the parameters *X1, Y1* and *X2, Y2*.

This procedure draws lines by using the current pen.

13.9.5.15 PlotPolygon

PlotPolygon draws a closed, multi-sided shape.

Syntax: [PlotPolygon, X1, Y1, X2, Y2, ... Xn, Yn]

Use *PlotPolygon* to draw a closed, multi-sided shape on the plot box, using the value of *Pen*. After drawing the complete shape, *PlotPolygon* fills the shape using the value of *Brush*. The parameter is a series of X/Y-positions that give the vertices of the polygon. For more information read the description of *KeepPolygon*.

13.9.5.16 KeepPolygon

KeepPolygon is an extension of *PlotPolygon*.

Syntax: [KeepPolygon, X1, Y1, X2, Y2, ... Xn, Yn]

If you have more ***PlotPolygon*** parameters than space in the row, use ***KeepPolygon*** to continue the parameter list in the next row. You can use any numbers of ***KeepPolygon*** to extend one ***PlotPolygon*** command.

13.9.6.1 PenColor

Determines the colour used to draw lines on the plot box.

Syntax: `[PenColor, red, green, blue]`

13.9.6.2 PenWidth

Specify the width of the pen in pixels.

Syntax: `[PenWidth, w]`

Example: `Next r = [PenWidth, 3]`

Use ***PenWidth*** to give the line greater width. The default width is 1.

Note: The value of Width influences which values of PenStyle are valid.

13.9.6.3 PenStyle

PenStyle determines the style in which the pen draws lines.

Syntax: `[PenStyle, ps]`

Example: `Next r = [PenStyle, psSolid]`

Use *PenStyle* to draw a dotted or dashed line, or to omit the line that appears as a frame around shapes.

Note: Dotted or dashed pen styles are not available when the *SetWidth* property is not 1.

The following lists the possible predefined values of *ps*. The default value is *psSolid*.

psSolid	Draw a solid line
psDash	A line made up of a series of dashes
psDot	A line made up of a series of dots
psDashDot	A line made up of alternating dashes and dots
psDashDotDot	A line made up of a series of dash-dot-dot combinations
psClear	No line is drawn (used to omit the line around shapes that draw an outline using the current pen)
psInsideFrame	A solid line, but one that may use a dithered color if Width is greater than 1

13.9.6.4 PenMode

Use *PenMode* to determine how the colour of the pen interacts with the colour on the plot box.

Syntax: [PenMode, pm]

Example: Next r = [PenMode, pmBlack]

The following lists the possible predefined values of *pm*. The default value is *pmCopy*.

pmBlack	Always black
pmWhite	Always white
pmNop	Unchanged
pmNot	Inverse of plot box background colour
pmCopy	Pen colour specified with <i>PenColor</i>
pmNotCopy	Inverse of pen colour
pmMergePenNot	Combination of pen colour and inverse of plot box background
pmMaskPenNot	Combination of colours common to both pen and inverse of plot box background

pmMergeNotPen	Combination of plot box background colour and inverse of pen colour
pmMaskNotPen	Combination of colours common to both plot box background and inverse of pen colour
pmMerge	Combination of pen colour and plot box background colour
pmNotMerge	Inverse of pmMerge: combination of pen colour and plot box background colour
pmMask	Combination of colours common to both pen and plot box background
pmNotMask	Inverse of pmMask: combination of colours common to both pen and plot box background
pmXor	Combination of colours in either pen or plot box background, but not both
pmNotXor	Inverse of pmXor: combination of colours in either pen or plot box background, but not both

13.9.7.1 BrushColor

BrushColor determines the colour of the brush.

Syntax: [BrushColor, red, green, blue]

The brush colour is the colour that is used to draw the pattern represented by the **BrushStyle** command, not the background colour of the brush (unless **BrushStyle** is *bsSolid*).

Note: If the value of **BrushStyle** is *bsClear*, the **BrushColor** command is ignored. Furthermore, any value assigned to **BrushColor** is lost when **BrushStyle** is set to *bsClear*.

13.9.7.2 BrushStyle

BrushStyle specifies the pattern on a brush.

Syntax: [BrushStyle, bs]

Example: Next r = [BrushStyle, bsSolid]

The following lists the possible predefined values of *bs*. The default value is *bsSolid*.

bsSolid	bsClear	bsBDiagonal	bsFDiagonal
bsCross	bsDiagCross	bsHorizontal	bsVertical

The pattern represented by this *BrushStyle* property shows the picture below.



13.9.8.1 PlotText

PlotText write a string into the plot box.

Syntax: [PlotText, X, Y, TextString]

Use ***PlotText*** to write a string into the plot box at the given X/Y - position. The string will be written using the current value of ***Font***. After a call of ***PlotText***, the pen position indicates the point at the top right of the text on the plot box.

13.9.8.2 FontSize

Use ***FontSize*** to specify the point size of the font.

Syntax: `[FontSize, fs]`

Use ***FontSize*** to specify the point size of the font. If the value is negative, the internal leading that appears at the top of each line of text is included. If the value is positive, ***FontSize*** represents the height of the characters but not the internal leading.

13.9.8.3 FontStyle

FontStyle specify style characteristics of a font.

Syntax: `[FontStyle, fs]`

Example: `Next r = [FontStyle, (fsBold or fsItalic)]`

FontStyle sets the font style values. Use the ***or*** operator to set more as one value. The following table lists the possible values of font style:

<code>fsBold</code>	the font is boldfaced.
<code>fsItalic</code>	the font is italicized.
<code>fsUnderline</code>	the font is underlined.
<code>fsStrikeOut</code>	the font is displayed with a horizontal line through it.

13.9.8.4 FontColor

Specify the colour of the text.

Syntax: [FontColor, red, green, blue]

Use **FontColor** to specify the colour of the text characters (as opposed to the background colour). If **FontColor** is set to a dithered colour, the text appears in the next closest non-dithered value.

13.9.8.5 FontName

Identify the typeface of the font.

Syntax: [FontName, NameString]

Use **FontName** to specify the typeface of the font. If the font family described by **FontName** includes multiple character sets, be sure to set the **Charset** property as well.

Note: If the combination of font family (typeface) and attributes (such as **bold** or **italic**) specifies a font that is not available on the system, the system substitutes a different font.

Version 4.34 news (User Manual)

5.9 Sqr

The *Sqr* function returns the square of the argument.

Example: `Sqr(4)` = 16

5.10 Sqrt

The result of *Sqrt* is the square root of the argument.

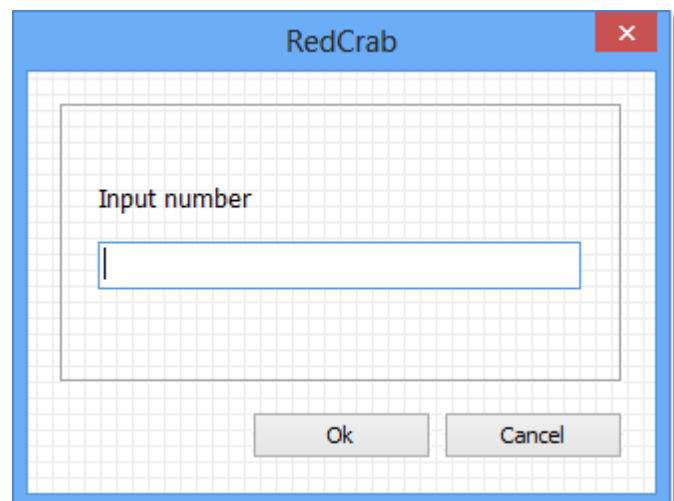
Example: `Sqrt(4)` = 2

Version 4.34 news (Programmers Manual)

3.1 Input

The *input* statement opens up a request window for input of a numeric value. The *input* statement follows two or three parameters:

1. A text string, which is shown above the input line.



2. The name of the variable, whose input value will assign to.
3. With a third string parameter, you can define an optional preset value, which is displayed in the input line when the window is shown.

Example 1: `input "Input number", x, "123"`

Example 2: `let a = "Input number"`
 `Let b = "123"`
 `Input a, x, b`

3.2 Display

Use the ***display*** statement to display a numeric value in a message window. The ***display*** statement follows two parameters:

1. The variable, whose value will be displayed in the window.
2. A text string, which can be displayed in addition to the value. The string can contains format instructions.

Example 1: `Let x = 471.2`
 `Display x, "The result is: "`

The message window show: „The result is: 471.2“

Example 2: `Let x = 471.2`
 `Display x, "The result is: #.##"`

The message window show: „The result is: 471.20“

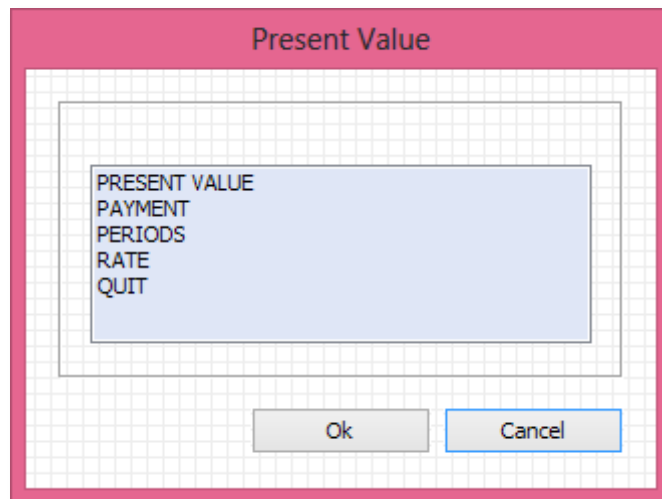
The format instructions are conform to the result box formatting. For more information, read the description in the user manual: ***Result Box / Formatting***.

3.3 Menu

With the *menu* statement you can open a menu window. The statement follows different numbers of text string parameters.

The first parameter is the menu title and follows by the name of the first menu item and the function, which is to call, if the user clicks this item. It follows the name of the second menu item and the calling function, and so on. All parameters are separated by comma and included in quotation marks.

Example: Menu "Present Value",
 "PRESENT VALUE", "PresentVal",
 "PAYMENT", "PresentPayment",
 "PERIODS", "PresentPeriods",
 "RATE", "PresentRate",
 "QUIT", "Quit"



Version 4.31 news (Free + Shareware)

The reference variable of result boxes can now be input with keyboard.

The reference variable of result boxes can now be inserted with keyboard instead the ***Paste*** button. Please note that the keyboard provides ***ANSI*** symbols only. Names which include Greek characters must be inserted per ***Paste*** button.

Undo function improved

Undo deletes function names, when inserted by button click, completely (not character by character).

Bug fixes

Chart box becomes invisible, when toggled between ***NonSync*** and ***GridSync*** mode.

After loading of ***RedCrab***^{PLUS} demo files in freeware mode, there was a problem with saving of freeware files.

The definition of complex data field which includes data ranges could cause a runtime error.

New features in Version 4.31 (RedCrab^{PLUS} only)

3.4 Menu View Program Panel

When the option Program Panel in menu View is switched on, **RedCrab** creates a button for any function in the program editor. A single click on the button inserts the function name in the worksheet on the actual cursor position. A double click inserts the name and the parameter list.

If the program changed, you can refresh the button list with Refresh in the popup menu. To open the popup menu, click with the right mouse button in the button area.

If the button panel too large for the window, you can move it with the left mouse button.

