RobotBASIC
Modifications History

The list of modifications and additions will be from a base reference to version 1.3.0. The help file in the latest version has all the information needed to use RobotBASIC without need to refer to or have ever used previous versions.

Things To Come

These items will be added in stages to future versions of RobotBASIC (not necessarily in this order):

- Support for Stack/Queue/List utilization.
- Database creation and manipulation system with SQL.

Versions 4.1.0 & 4.2.0

In this version the major changes are to do with:

- Fixed an error with the mPlotxy command that uses the 2D array of data where it was not calculating the correct scaling in some situations.
- Added mSortC to perform sorting on the elements of a specified Column rather than on the elements of a specified Row as in the mSortR command. The old command mSort is still available but it is better to use the mSortR command
- Added the functions SFtoDF() to convert a 32 bit fixed float format of a number (stored in the 4 bytes of an integer in RB) to a standard RB float number which is in 64 bit format. Also DFtoSF() to convert in the other direction. Also see the functions BuffWriteF32() and BuffReadF32() to do the same thing for buffers. This can be useful to read floating point data from microcontrollers such as the Propeller Chip which stores floating point numbers using the IEEE 32 bit encoding in the Little-Endian order. This way you can receive or send floating point numbers back and forth between RB and the Propeller or similar microcontrollers.
- Before this version you could not use a function call in a line by itself just like a statement and you had to use it in an assignment statement. However there are many functions whose return value is of no consequence and rather it is the action of the function that is of use. In this version you can now do a function call as a statement without assigning its return result to a value. You can do:

```
TCPS_Close()
```

  just as if it were a subroutine call. Notice how the return value is ignored and is not being assigned to a variable as you had to do before. Read more details about this in the Functions subsection in the Overview Of The Language section.
- Many new and modified Graphics Engine commands that are EASY to use but yet quite powerful.
- You can now use ++, --, +=, -=, *= and /= as assignment operators. See the Assignment Operators section under the Operators section in the Language Overview Section.
- The new command InlineInputMode allows you to make the Input and WaitKey commands prompt and accept input from the user within the Terminal Screen rather than in the input area of the Control Panel under the main screen.
- This version now allows for optional parameters to be skipped using a space (or more or none) between commas when specifying parameters for commands. See more details in the Commands subsection in the Language Overview section also see the **second** note at the top of the User Interface Commands section. So now you can say :

```
Circle 10,10,100,100, ,red  //notice the missing 5th parameter
SaveScrWH ,,70              //notice the missing 1st and 2nd parameters
```

- You can now use the \ character followed immediately by an Enter (ie CrLf) to indicate a line continuation operator that allows for a long line of code to be continued on the next line. See The Line Break Operator subsection in the Language Overview section.
- You can now use Parameterized subroutines with optional return values and by-value or by-reference parameters with optional parameter skipping and Local Scoped variables. See the Sub/Call/Return statements in the Flow Control Statements section.
- Event handlers (On[control])can now either use normal Label: subroutines (just like a gosub) with global scoped variables, or Sub subroutines with local scoped variables for their event handler subroutines. But not OnError) it is only allowed to use Label subroutines (gosub like) since it may need to use global variables during error handling.
- The new Declare statement allows you to RESTRICT RobotBASIC's variables to become Strict Typing. That is, with the Declare statement you declare variables and their types as well as an initial value. This will also make RB become a strict typing language where variables have to be declared to be of a certain type before they can be used and their type will not be changeable. A float can be assigned an integer but not a string. An integer can be assigned a float that can be truncated to an integer but not a string. As string can be assigned only strings. From the first moment the Declare statement is executed RB will perform strict typing from that point onwards. Strict typing cab be turned off if desired with the statement Undeclare. See full explanation in the Miscellaneous Commands section. Also The Input command and the Assignment Statement and the For-Next loop structure which are affected areas.
- Two new commands can be used to facilitate the creation of a pseudo-object-based actions or a simple database system. See the commands ObjectPut/RecordPut and ObjectGet/RecordGet in the Arrays Commands Section.
- With the new function VarsList() you can obtaine a list of variables (local or global) and with the new command varSet and the already existing functions varType() and varValue() that facilitate Indirection Access to variables (pointer like) you can do all sorts of tricks and even a rudimentary Debugger.
- The set of commands MediaPlay, MediaPause, MediaStop, MediaRecord, MediaSave and the function MediaState() provide the ability to play multiple media files ***simultaneously***. The files can be audio or video and of various types. Also recording is allowed of WAV Audio files. See also MediaShow, MediaReposition and MediaGetposition.
- New USBmicro functions to support the new U451 device and the I2C capability of the new firmware on later versions of the U401 and U421 with the newer USBm.DLL.
- The new mFFT and mDFT commands perform Complex Discrete and Fast Fourier Transform analysis on Real or Complex sampled data.
- Normally the Terminal Screen window is not a sizeable window and the user will not be able to modify the window width or height. However, you now can allow the user to resize the window manually or move and resize the window under program control using the commands ScrSetMetrics and ScrGetMetrics. Also with these commands you can show/hide the control panel at the bottom of the window. See these commands in the User Interface Commands section.
- Since now with the new Sub/Call routines you can have local variable scoping, this opens up the opportunity for creating Libraries of subroutines that can be distributed as standard text files or even as RB binary files (not compiled EXE...just RB binary). To facilitate the incorporation of these libraries within programs the new #Include directive is now available. You can have multiple includes and even includes within includes. The files can be normal text files or binaries created by RB. See the #Include statement under the Flow Control Commands section.
- Better Help Screen with an expandable Table of Contents that helps select topics in the help file as well as 4 list boxes with commands and functions etc. so that you can directly locate the main entry within the help file.
- A more sophisticated IDE Editor with Syntax Highlighting and ability to work on multiple simultaneously open files. Also the ability to turn on/off a line numbering feature.
- A system for configuring the IDE with your preferences for syntax highlighting and other parameters. Also these preferences will be save to disk file so as when you start RB the next time all the settings would be as you have configured them.

- The File menu option now has a list of most recently open files. The number of files in the list is configurable through a Preferences Configurations system.
- Changed the line numbering in the Editor Screen to 1-based. That is the first line is line number 1 not line number 0 as it used to be in previous versions. Also when an error is reported it will report the line number with 1-Base not 0 based as it used to be in previous versions. Additionally the GetError command will also report the line number using 1-Base. **Check that this does not affect your existing programs**.
- A new Find Dialog now allows for starting the search from the start of the file.

Altogether there are 5 new flow control statements, 1 new directive, 28 new functions, 29 new commands, 12 modified functions and 10 modified commands.

- **Added the following Directive:**

  #Include "FileName.Ext"{,...}

- **Added the following Statements:**

  Sub SubName({{&}var1{,{&}var2{,...}})
  Return {expr}
  Call SubName({expr1{expr2{,...}})
  On[control] SubName
  OnAbort SubName

- **Added the following functions:**

  SFtoDF(ExprN)
  DFtoSF(ExprN)
  BuffWriteF32(se_Buffer,ne_Position,ne_Value)
  BuffReadF32(se_Buffer,ne_Position)
  FactorColor(ne_Color,ne_Factor)
  varsList({ne_Global})
  MediaState(ne_DeviceNumber)
  MediaIsVideo(ne_DeviceNumber)
  CommandsList()
  FunctionsList()
  StatementsList()
  ConstantsList()
  mToCommaText(a_TextLines)
  ToCommaText(se_Text)
  DeskTopWidth()
  DeskTopHeight()
  usbm_InitPortsU401(ne_DeviceNumber)
  usbm_InitPortsU421(ne_DeviceNumber)
  usbm_InitPortsU451(ne_DeviceNumber)
  usbm_DirectionAIn(ne_DeviceNumber)
  usbm_DirectionAOut(ne_DeviceNumber)
  usbm_DirectionAInPullUp(ne_DeviceNumber)
  usbm_DirectionBIn(ne_DeviceNumber)
  usbm_DirectionBOut(ne_DeviceNumber)
  usbm_DirectionBInPullUp(ne_DeviceNumber)
  usbm_ReadLatches(ne_DeviceNumber)
  usbm_Wire2Control(ne_DeviceNumber,ne_Signal)
  usbm_Wire2Data(ne_DeviceNumber,se_DataBytes)

- **Added the following Commands:**

mSortC a_Name{,ne_OnColumnNumber{,ne_Descending}}
mSortR a_Name{,ne_OnColumnNumber{,ne_Descending}}
InlineInputMode {on|off}
rRelocate {ne_X{,ne_Y{,ne_Heading}}}
gePlotSurfaces
a_3DPoints,a_SurfacesSpecs{,ne_DoFilling{,ne_LineWidth{,ne_OnlyVisible{,ne_CentroidAll}}}}
geCentroids a_3DPoints,a_SurfacesSpecs,a_Centroids
ge3Dto2DVA ne_X,ne_Y,ne_Z,a_CameraSpecs,vn_ScrX,vn_ScrY
gePlotEdges a_3DPoints,a_Edges{,ne_LineWidth{,ne_LineWidth{,ne_Color}}
geRotateA a_3DPoints,ne_RotAngle,ne_AxisCode{,ne_From{,ne_To}}
Undeclare
Declare v_Name {{=}e_InitialValue}{, ...}
VarSet se_VarName,e_Value
ObjectGet a_ObjectArray,ne_ObjectNumber
ObjectPut a_ObjectArray,ne_ObjectNumber
RecordGet a_DataBaseArray,ne_RecordNumber
RecordPut a_DataBaseArray,ne_RecordNumber
MediaPlay ne_DeviceNumber,se_FileName{,ne_Loop}
MediaPause ne_DeviceNumber{,on|off}
MediaStop ne_DeviceNumber
MediaRecord ne_DeviceNumber,se_FileName
MediaSave ne_DeviceNumber
MediaShow ne_DeviceNumber{,true|false}
MediaReposition {ne_X{,ne_Y{,ne_Width{,ne_Height}}}}
MediaGetPosition {vn_X{,vn_Y{,vn_Width{,vn_Height}}}}
MediaVideoSize DeviceNumber{,vn_Width{,vn_Height}}
mDFT a_Samples{,ne_WindowFunction}
mFFT a_Samples{,ne_WindowFunction}
ScrSetMetrics {ne_X{,ne_Y{,ne_Width{,ne_Height{,ne_PanelVisible{,ne_AllowResize}}}}}}
ScrGetMetrics {vn_X{,vn_Y{,vn_Width{,vn_Height{,vn_PanelVisible{,vn_AllowResize}}}}}}

- **Modified the following commands:**

Dim a_Name1[ExprN{,ExprN...}]{ , a_Name2[ExprN{,ExprN...}] {, ...}}}...
You now can specify multiple arrays' dimensional specification using the same Dim statement.
Data a_Name;Expr{,Expr....}...
You now can use , , to create an element without a specific value. So you create a place holder but keeping the element's value and type unspecified. See the mPlotXY command to see how this ability can be useful in some cases and how it is achieved.
xyText{ne_X{,ne_Y{,e_Text{,se_FontName{,ne_FontSize{,ne_FontStyle{,ne_PenColor{,ne_BackgroundColor}}}}}}} see the command's description.
OverlayText{ne_X{,ne_Y,e_Text{,se_FontName{,ne_FontSize{,ne_FontStyle{,ne_PenColor{,ne_Quick}}}}}}}} see the command's description.
geVisibles a_3DPoints,a_CameraSpecs,a_SurfacesSpecs{,a_Edges{,ne_ColorFactor}}
see the command's description.
FloodFill {ne_X{,ne_Y{,ne_NewColor{,ne_OldColor}}}}
Now you can use -1,-1 for the X.Y coordinates to use the current pen position, or not specify the parameters.
FloodFill2 {ne_X{,ne_Y{,ne_NewColor{,ne_BorderColor}}}}
Now you can use -1,-1 for the X.Y coordinates to use the current pen position, or not specify the parameters.

Input {e_Prompt,} v_Name | a_Name[...] {,...}
Now you can have multiple variables to be inputted. See command description. Also see the new command InlineInputMode.
rLocate ne_X,ne_Y{,ne_Heading{,ne_Size{,ne_BorderColor{,ne_InsideColor{,ne_ObeyFlip}}}}}
See the command's description for details on the new parameter.
mSort a_Name{,ne_OnColumnNumber}
This command is now superceded by the new command mSortR, **but it still works**.

- **Modified the following functions:**

RotShape(se_ShapeString,ne_Direction)
Modified how the ne_Direction parameter is used. See the function's description.
Substring(se_Text{,ne_StartChar{,ne_NumCharacters}}) There are now optional parameters.
MaxDim(a_Name{,ne_Dimension}) "
StrInput({e_Caption{,e_Prompt{,e_Default}}}) "
ErrMsg(se_MessageText{,se_BoxTitle{,ne_Style}}) "
Time({ne_Type}) "
Date({ne_Type}) "
DateTime(ne_DateTimeValue{,se_Format}) "
Degrees({ne_Degrees{,ne_Minutes{,ne_Seconds}}}) "
MsgBox(a_TextLines{,e_Title{,ne_X{,ne_Y{,ne_W{,ne_H{,ne_DoWrap}}}}}})
New parameters allow for more control of the box.
StringBox(se_Text{,e_Title{,ne_X{,ne_Y{,ne_W{,ne_H{,ne_DoWrap}}}}}})
New parameters allow for more control of the box.
TextBox(se_FileName{,e_Title{,ne_X{,ne_Y{,ne_W{,ne_H{,ne_DoWrap}}}}}})
New parameters allow for more control of the box.

Versions 4.0.1 & 4.0.2

In this version the major changes are to do with:

- Fixed an error where with Windows Xp the Terminal Screen height was being sized to less than 600 even when the screen resolution was not too small.
- Fixed ScrLimits command reporting wrong height under XP as in the above problem
- Fixed a problem where Buttons Font Color was not being set.
- Fixed a problem with the usbm_SPIMaster() function sending too many bytes.
- Fixed a problem with notebooks with screen resolution less than 1024x760 showing flicker.
- Added a feature where a * is put ahead of the file name in the Editor Screen panel below the Speed buttons when the file has been modified and not yet saved. This helps in indicating if the file needs saving or not.
- In previous versions you needed to not include the ".Bmp" extension when specifying Bitmap file names. In this version you can omit the extension or include it and RB will automatically assume a ".BMP" extension regardless.
- Added the ability to have Block Comments using /* */ delimiters. Cannot be nested though. See the new Block Comments sub-section in the Language Overview section.
- A whole suite of Navigation commands and functions would allow you to do Great Circle or Rhumb Line calculations of distances and bearings and so forth. Also you can solve the Wind/Current triangle.
- New commands for saving portions of the screen to an array or to a numbered memory area and to restore portions of the saved copy back to the screen. See the new commands below. These can provide for multiple scratch pads for BitMaps so that you can do fast and powerful animations.

- New commands for manipulating and reading Serial Port Signal lines.
- During Stepping if you opened the Variables table it used to be modal and you had to close it before you could step again. Now it is modeless and it will stay active. It will go behind other windows and you may have to move it to the front.
- Now when you activate the Help Screen, any selected text in the Edit Screen will be automatically inserted into a Search dialog and the Search Dialog will be activated so that you can search for the text that was selected.
- Now if you activate the Search or Search & Replace dialogs they will be automatically filled with any selected text within the Edit Screen or the Help Screen.
- Now the rChargeLevel() function is also part of the Serial Comms protocol (# 108) and can be used to get back from the real robot the battery charge level if it has the right instrumentation.

- Added the new command rInstError to make the simulated robot return instrumentation values that are not accurate for added realism.
- Added the command rCommand() that allows you to send a general command (2 bytes) using the robot's serial comms protocol to the real robot and receive 5 bytes back. This enables more diversity in implementing more commands than are provided by the inbuilt protocol.
- A whole suite of commands to support a Regular Expressions search/replace engine.
- New Array commands that allow you to do Polynomial/Logarithmic/Exponential curve fitting to data points.
- Fixed an error where the Comms errors were not being trapped by the OnError event handling.
- Fixed an error in the CopyScr, CopyToScr and CopyFitScr commands.
- Fixed the OverlayText command to draw the text in one color so as to avoid the whitish pixels that can appear on darker backgrounds.
- Added the command SerPorts that returns a list of all currently available comm ports on the computer.

Altogether there are 25 new functions, 48 new commands, 5 modified functions and 3 modified commands.

- **Added the following functions:**

StrOfBytes(Expr{,Expr{,....}})
Map(ne_FromValue, ne_FromMin, ne_FromMax, ne_ToMin, ne_ToMax)
FileEnd(ne_FileHandle)
FileReadField(ne_FileHandle{,se_Separator})
Degrees(ne_Degrees{,ne_Minutes{,ne_Seconds}}) or **Degrees**(se_FormattedDegrees)
Lat_DM(ne_Degrees)
Lat_DMS(ne_Degrees)
Lon_DMS(ne_Degrees)
Lon_DM(ne_Degrees)
Convert(ne_ValueToConvert, ne_ConversionTypeCode)
rCommand(ne_Command,ne_Data)
rInstError {ne_PercentageLevel}
SignExtend8(ExprN)
SignExtend16(ExprN)
re_Setup(se_Template{,ne_Mode})
re_Match(se_Text)
re_Search(se_Text{,ne_SartFromPositionNumber})
re_Start({ne_GroupNumber})
re_End({ne_GroupNumber})
re_GrpNumber(se_GroupName)
re_NumOfGrps()
re_Replace(se_Text{,se_ReplaceWith{,ne_SartFromPositionNumber{,ne_NumberOfMatchesToReplace}}})

GetMemoCharPos(se_Name)
TextWidth(se_Text{,se_FontName{,ne_FontSize{,ne_FonctStyle}}})
TextHeight(se_Text{,se_FontName{,ne_FontSize{,ne_FonctStyle}}})

- **Added the following Commands:**

rSenseType {ne_NumSensors}
CommaTab {true|false}
HonorCRLF {true|false}
BmpChangeClr se_SourceFile,ne_FromColor,ne_ToColor{,ne_Tolerance}
SaveScrWH {ne_X1{,ne_Y1{,ne_Width{,ne_Height}}}}
mScrFitArray VarA{,ne_X{,ne_Y{,ne_Width{,ne_Height}}}}
mScrFromArray VarA{,ne_ScreenX{,ne_ScreenY{,ne_Width{,ne_Height{,ne_ArrayX{,ne_ArrayY}}}}}}
mScrToArray VarA,{se_FileName{,ne_X{,ne_Y{,ne_Width{,ne_Height}}}}}
CopyFitScr {ne_CopyNumber{,ne_X{,ne_Y{,ne_Width{,ne_Height}}}}}
CopyScr {ne_CopyNumber{,ne_X1{,ne_Y1{,ne_Width{,ne_Height}}}}}
CopyToScr {ne_CopyNumber{,ne_ScreenX{,ne_ScreenY{,ne_Width{,ne_Height{,ne_MapX{,ne_MapY}}}}}}}
SerPorts vs_PortsList
ReadSerSignals vn_Flags
SetSerDTR {on|off}
SetSerRTS {on|off}
mAND VarA,ExprN
mOR VarA,ExprN
mXOR VarA,ExprN
mShiftL VarA,ExprN
mShiftR VarA,ExprN
mNOT VarA
mmAND VarA1,VarA2
mmOR VarA1,VarA2
mmXOR VarA1,VarA2
mmShiftL VarA1,VarA2
mmShiftR VarA1,VarA2
SetMemoSelected se_Name{,ne_StartCharPosition{,ne_NumCharacters}}
SetListBoxItems se_Name,se_ItemsList
SetRBGroupButtons se_Name,se_ButtonsList
mExpFit a_XYdata,vn_Exponent,vn_Factor
mLogFit a_XYdata,vn_Factor,vn_Translation
mPolyFit a_XYdata,a_Specs
ngc_DistanceHeading ne_LatA,ne_LonA,ne_LatB,ne_LonB,vn_Distance,vn_Heading
ngc_FractionDistancePoint ne_LatA,ne_LonA,ne_LatB,ne_LonB,ne_FractionalDistance,vn_Lat,vn_Lon
ngc_LatFromLonCrossing ne_LatA,ne_LonA,ne_LatB,ne_LonB,ne_Lon,vn_Lat
ngc_LonFromLatCrossing ne_LatA,ne_LonA,ne_LatB,ne_LonB,ne_Lat,vn_Lon1,vn_Lon2
ngc_RadialIntersection ne_LatA,ne_LonA,ne_LatB,ne_LonB,ne_HeadingFromA,ne_HeadingFromB,vn_Lat,vn_Lon
ngc_RadialPoint ne_LatA,ne_LonA,ne_Distance,ne_Heading,vn_Lat,vn_Lon
ngc_TrackPointsFromPoint ne_LatA,ne_LonA,ne_LatB,ne_LonB,ne_LatD,ne_LonD,ne_DstnceFrmPnt,vn_Lat1,vn_Lon1, vn_Lat2,vn_Lon2
ngc_XTrackError ne_LatA,ne_LonA,ne_LatB,ne_LonB,ne_LatD,ne_LonD,vn_XTrackDistance,vn_AlongTrackDistance
nrl_DistanceHeading ne_LatA,ne_LonA,ne_LatB,ne_LonB,vn_Distance,vn_Heading
nrl_RadialPoint ne_LatA,ne_LonA,ne_Distance,ne_Heading,vn_Lat,vn_Lon
nwt_GSpeedCourse ne_WindSpeed,ne_WindDirection,ne_TrueSpeed,ne_Heading,vn_GrndSpeed,vn_CourseHeading
nwt_GSpeedHeading ne_WindSpeed,ne_WindDirection,ne_TrueSpeed,ne_CourseHeading,vn_GrndSpeed,vn_Heading
nwt_TSpeedHeading ne_WindSpeed,ne_WindDirection,ne_GrndSpeed,ne_CourseHeading,vn_TrueSpeed,vn_Heading
nwt_TSpeedWSpeed ne_V1,ne_V2,ne_V3,vn_TrueSpeed,vn_WindSpeed

nwt_WSpeedDirection ne_GroundSpeed,ne_CourseHeading,ne_TrueSpeed,ne_Heading,vn_WindSpeed,vn_WindDirection
nwt_XWind ne_WindSpeed,ne_WindDirection,ne_RunwayHeading,vn_XWind,vn_HeadWind

- **Modified the following commands:**

rLocate ne_X,ne_Y{,ne_Heading{,ne_Size{,ne_BorderColor{,ne_InsideColor}}}
Added the ability to specify the color of the inside of the robot as well as its border color.
Print {Expr,Expr;Expr...}{;|,}
Now strings with the CR or CR/LF will cause a Linefeed upon printing on the Terminal Screen.
SaveScr {ne_X1{,ne_Y1{,ne_X2{,ne_Y2}}}}
There was a problem with versions before 4.0.1 where the saved portion was 1 pixel less in both axes. It is now rectified. However, you may need to check your programs for any adverse effects since now the saved portion is 1 pixel wider in the x and y axes. This may affect your programs.
mPlotXY a_Specs,a_Xvalues,a_Yvalues OR **mPlotXY** a_Specs,a_XYvalues
This command now allows for either two separate one dimensional arrays for the X and Y values, or for one combined two dimensional array.

- **Modified the following functions:**

rSense({ne_Color})
This function now ***can*** return the status of 5 line sensors not just 3.
KeyDown({ne_ScanCode})
This function now ***can*** return the scan code of a pressed key instead of just seeing if a key with a certain scan code is pressed or not.
InString(se_Main,se_Sub{,ne_StartFrom})
The additional parameter allows you to repeatedly search the string for multiple occurrences of the sub-string.
RoundUP(ExprN{,ne_Type})
The additional parameter affects the logic for negative numbers. Read the details in the function description.
RoundDn(ExprN{,ne_Type})
The additional parameter affects the logic for negative numbers. Read the details in the function description.

Version 4.0.0

In this version the major changes are to do with:

- Compile capability. See the IDE section Editor Screen, also Sharing Your Programs section.
- The USBmicro suite of functions to support I/O with the U4x1 USB device. See the USBmicro section.
- Sending of Email.
- Buffered Asynchronous sending and receiving of TCP packets over the LAN, WAN, WiFi, or Internet.
- Buffered Asynchronous sending and receiving of UDP packets over the LAN, WAN, WiFi, or Internet.
- New suite of extensive Time and Date functions. See Now(), DateTimeStr() etc.
- Support for entering numbers in Binary (e.g. 0%011011) and Hexadecimal (e.g. 0xA45B00) formats, even in an Edit box. See IntgerEdit.
- Support for accessing the Terminal Screen text buffer. See GetTextBuff() and SetTextBuff.
- Support for sending and retrieving text from the Clipboard. See SetCBText and GetCBText().
- More Edit box and Button commands and functions. See SetEditMask, IntgerEDit and FloatEdit.
- New Check Box, Radio Button and List Box suite of functions and commands
- New Slider/Progress Bar, Spinner and Memo box suite of commands and functions.
- Timer Events. See AddTimer and OnTimer.
- Added EVENT handling for Controls, Keyboard, Mouse and for Aborting, see On[Control] and OnAbort in the Flow Control Statements section.
- Low level File I/O functions to allow for creating and reading Text or Binary files with Sequential or Random Access styles. See FileOpen(), FileCreate(), FileRead(), FileWrite(), FileSeek() etc. and FilePrintT and FilePrintB.
- Byte buffer manipulation routines that allow for manipulating BYTES in a buffer (think of it as an one-

dimensional array of bytes). See BuffWrite() and BuffRead() etc., PutStrByte(), GetStrByte() and ToByte() and BuffPrintB and BuffPrintT.
- New functions for manipulating strings including Substitute(), Encrypt() and much more as well as access to a string's individual characters as byte values (converted to integer). See PutStrByte() and GetStrByte().
- New and modified user dialog popups for presenting data and obtaining responses, see StringBox(), TextBox(), MsgBox().
- Support for POINTER style access to variables. See varType() and varValue().
- A new option on the Debug screen that allows for evaluating any RB expression. This will allow for viewing array elements. The ***View Variables Table*** option shows a table of ***all SIMPLE*** variables in the current running session. With this new ***Evaluate Expression*** option you can also view array elements and even do math and other calculations on variables and array elements. It can be any valid RB expression.
- Stepping of a program is now possible through the Stepping {On|Off} command or through a button on the Terminal Screen. It also includes an expressions evaluation system with a watch list.

Altogether there are 243 new functions, 112 new commands, 3 modified function and 4 modified commands.

- **Added the following functions:**
  - **StringBox**(se_Text{,se_Title})
  - **mToString**(VarA{,se_Separator})
  - **ProgName**()
  - **RotShape**(ExprS,ExprN)
  - **GetCBText**()
  - **GetTextBuff**()
  - **Proper**(ExprS)
  - **Substitute**(se_Text,se_TextToReplace,se_ReplaceWith)
  - **Encrypt**(se_Text,se_Key)
  - **Contains**(se_Text,se_CharList)
  - **NotContains**(se_Text,se_CharList)
  - **CrLf**()
  - **GetStrByte**(se_String,ne_ByteNumber)
  - **PutStrByte**(se_String,ne_ByteNum,ne_Val)
  - **ToByte**(Expr)
  - **MaxInteger**()
  - **MinInteger**()
  - **MaxFloat**()
  - **MinFloat**()
  - **MakeBit**(ne_Number,ne_BitPosition,on|off)
  - **MakeByte**(neNumber,ne_BytePosition,ne_ByteValue)
  - **BitSwap**(ne_Number{,ne_NumberOfBits})
  - **varValue**(se_VarName)
  - **varType**(se_VarName)
  - **Now**()
  - **Hour**(ExprN)
  - **Day**(ExprN)
  - **DayOfWeek**(ExprN)
  - **Milliscond**(ExprN)
  - **Minute**(ExprN)
  - **Month**(ExprN)
  - **Second**(ExprN)
  - **Year**(ExprN)

DateStr(ExprN)
DateTime(ExprN,ExprS)
DateTimeStr(ExprN)
DateTimeVal(se_DateTimeString)
DateVal(ExprN1,ExprN2,ExprN3)
TimeStr(ExprN)
TimeVal(ExprN1,ExprN2,ExprN3,ExprN4)
TCP_LocalIP()
TCPC_BuffCount()
TCPC_Close()
TCPC_Connect(se_IPaddress)
TCPC_ConnectHost(se_HostName)
TCPC_Peek()
TCPC_Read()
TCPC_Send(se_Data)
TCPC_Status()
TCPS_BuffCount()
TCPS_Close()
TCPS_Header({on|off})
TCPS_Peek()
TCPS_Read()
TCPS_Serve()
TCPS_Send(se_Data)
TCPS_Status()
UDP_BuffCount(se_Name)
UDP_Header(se_Name{,on|off})
UDP_Peek(se_Name)
UDP_Read(se_Name)
UDP_Send(se_Name,se_Data,se_TargetIP{,ne_TargetPort})
UDP_Start(se_Name{,ne_ListenPort})
UDP_Status(se_Name)
DiskSize(ne_DiskNumber)
DiskFree(ne_DiskNumber)
FileDate(se_FileName)
FileClose(ne_FileHandle)
FileCreate(se_FileName)
FileOpen(se_FileName,ne_Mode)
FileRead(ne_FileHandle{,ne_ByteCount)
FileReadB(ne_FileHandle)
FileReadI(ne_FileHandle)
FileReadF(ne_FileHandle)
FileSeek(ne_FileHandle,ne_FromWhere,ne_OffsetCount)
FileSize(se_FileName)
FileSize(ne_FileHandle)
FileWrite(ne_FileHandle,Expr)
FileWriteB(ne_FileHandle,Expr)
BuffRead(se_Buffer,ne_Position,ne_NumBytes)
BuffReadB(se_Buffer,ne_Position)
BuffReadF(se_Buffer,ne_Position)
BuffReadI(se_Buffer,ne_Position)
BuffWrite(se_Buffer,ne_Position,e_Value)
BuffWriteB(se_Buffer,ne_Position,ne_Value)

LastKey()
LastMouse()
ButtonEnabled(se_Name)
ButtonHasFocus(se_Name)
ButtonHidden(se_Name)
GetButtonCaption(se_Name)
GetButtonFont(se_Name)
GetButtonH(se_Name)
GetButtonW(se_Name)
GetButtonX(se_Name)
GetButtonY(se_Name)
LastButton()
EditBorder(se_Name)
EditEnabled(se_Name)
EditHidden(se_Name)
EditReadOnly(se_Name)
GetEditColor(se_Name)
GetEditFont(se_Name)
GetEditUnmasked(se_Name)
GetEditH(se_Name)
GetEditW(se_Name)
GetEditX(se_Name)
GetEditY(se_Name)
LastEdit()
CheckBoxEnabled(se_Name)
CheckBoxHidden(se_Name)
CheckBoxHasFocus(se_Name)
GetCheckBox(se_Name)
GetCheckBoxCaption(se_Name)
GetCheckBoxColor(se_Name)
GetCheckBoxX(se_Name)
GetCheckBoxY(se_Name)
LastCheckBox()
GetRBGroup(se_Name)
GetRBGroupButton(ExprS,ExprN)
GetRBGroupCaption(se_Name)
GetRBGroupColor(se_Name)
GetRBGroupFont(se_Name)
GetRBGroupH(se_Name)
GetRBGroupItems(se_Name)
GetRBGroupText(se_Name)
GetRBGroupW(se_Name)
GetRBGroupX(se_Name)
GetRBGroupY(se_Name)
LastRBGroup()
RBGroupEnabled(se_Name)
RBGroupHidden(se_Name)
RBGroupHasFocus(se_Name)
RBGroupItemsCount(se_Name)
RBGroupNumColumns(se_Name)
ListBoxEnabled(se_Name)
ListBoxHidden(se_Name)

ListBoxItemsCount(se_Name)
ListBoxSorted(se_Name)
ListBoxHasFocus(se_Name)
GetListBox(se_Name)
GetListBoxColor(se_Name)
GetListBoxFont(se_Name)
GetListBoxItem(se_Name,ExprN)
GetListBoxList(se_Name)
GetListBoxText(se_Name)
GetListBoxW(se_Name)
GetListBoxX(se_Name)
GetListBoxY(se_Name)
LastListBox()
GetSliderBarStart(se_Name)
GetSliderBarEnd(se_Name)
GetSliderMax(se_Name)
GetSliderMin(se_Name)
GetSliderPos(se_Name)
GetSliderW(se_Name)
GetSliderX(se_Name)
GetSliderY(se_Name)
LastSlider()
SliderHasFocus(se_Name)
SliderBarHidden(se_Name)
SliderDialHidden(se_Name)
SliderEnabled(se_Name)
SliderHidden(se_Name)
GetSpinner(se_Name)
GetSpinnerH(se_Name)
GetSpinnerIncr(se_Name)
GetSpinnerMin(se_Name)
GetSpinnerMax(se_Name)
GetSpinnerW(se_Name)
GetSpinnerWrap(se_Name)
GetSpinnerX(se_Name)
GetSpinnerY(se_Name)
LastSpinner()
SpinnerEnabled(se_Name)
SpinnerHidden(se_Name)
SpinnerHasFocus(se_Name)
GetMemoCharNo(se_Name)
GetMemoColor(se_Name)
GetMemoFont(se_Name)
GetMemoH(se_Name)
GetMemoLine(se_Name,ExprN)
GetMemoLineNo(se_Name)
GetMemoSelection(se_Name)
GetMemoText(se_Name)
GetMemoW(se_Name)
GetMemoX(se_Name)
GetMemoY(se_Name)
MemoChanged(se_Name)

MemoEnabled(se_Name)
MemoHasFocus(se_Name)
MemoHidden(se_Name)
MemoWrap(se_Name)
MemoReadonly(se_Name)
MemoBorder(se_Name)
MemoScrollBars(se_Name)
MemoLinesCount(se_Name)
LastMemo()
GetTimerPeriod(se_Name)
GetTimerTicks(se_Name)
LastTimer()
TimerIsOn(se_Name)
usbm_DllSpecs()
usbm_ErrorSpecs()
usbm_ClearRecentError()
usbm_FindDevices()
usbm_NumberOfDevices()
usbm_SetReadTimeout(ne_Time)
usbm_DeviceSpecs(ne_DeviceNumber)
usbm_DeviceValid(ne_DeviceNumber)
usbm_CloseDevice(ne_DeviceNumber)
usbm_DeviceCmd(ne_DeviceNumber,se_Data)
usbm_InitPorts(ne_DeviceNumber)
usbm_DirectionA(ne_DeviceNumber,ne_PinsDirection,ne_PinsFormat)
usbm_DirectionB(ne_DeviceNumber,ne_PinsDirection,ne_PinsFormat)
usbm_WriteA(ne_DeviceNumber,ne_ByteValue)
usbm_WriteB(ne_DeviceNumber,ne_ByteValue)
usbm_ReadA(ne_DeviceNumber)
usbm_ReadB(ne_DeviceNumber)
usbm_SetBit(ne_DeviceNumber,ne_PinNumber)
usbm_ResetBit(ne_DeviceNumber,ne_PinNumber)
usbm_WriteABit(ne_DeviceNumber,ne_AndingMask, ne_OringMask)
usbm_WriteBBit(ne_DeviceNumber,ne_AndingMask, ne_OringMask)
usbm_InitLCD(ne_DeviceNumber,ne_Sel, ne_Port)
usbm_LCDCmd(ne_DeviceNumber,ne_CommandByte)
usbm_LCDData(ne_DeviceNumber,ne_DataByte)
usbm_Reset1Wire(ne_DeviceNumber,ne_Specs)
usbm_Write1Wire(ne_DeviceNumber,ne_Data)
usbm_Read1Wire(ne_DeviceNumber)
usbm_Write1WireBit(ne_DeviceNumber,ne_BitValue)
usbm_Read1WireBit(ne_DeviceNumber)
usbm_InitSPI(ne_DeviceNumber,ne_Specs)
usbm_SPISlaveRead(ne_DeviceNumber)
usbm_SPISlaveWrite(ne_DeviceNumber,se_DataBytes)
usbm_SPIMaster(ne_DeviceNumber,se_DataBytes)
usbm_Stepper(ne_DeviceNumber,se_DataSpecs)
usbm_StrobeWrite(ne_DeviceNumber,se_ByteData)
usbm_StrobeRead(ne_DeviceNumber,se_ByteData)
usbm_StrobeWrites(ne_DeviceNumber,se_ByteData)
usbm_StrobeReads(ne_DeviceNumber,se_ByteData)

- **Added the following Commands:**

Stepping {On|Off}
AllowEvents {on|off}
SeedRandom ExprN
CommaTab {true|false}
mFromString VarA,se_String{,se_Separator}
PrinterSetup
ClrCB
TextBuffToCB
SetTextBuff se_Text
SetCBText se_Text
SetPromptArea se_Text
SetInputArea se_Text
BuffPrintB vs_BuffString{,Expr,Expr,Expr...}
BuffPrintT vs_BuffString{,Expr,Expr;Expr...}{;|,}
FilePrintT vn_FileHandle{,Expr,Expr;Expr...}{;|,}
FilePrintB vn_FileHandle{,Expr,Expr,Expr...}
SerialOut Expr {,Expr {, Expr ...}}
SendEmail VarA1,VarA2 {,ExprN}
EnableButton ExprS{,ExprN}
HideButton ExprS{,ExprN}
RenameButton ExprS1,ExprS2
SetButtonCaption ExprS1{,ExprS2}
SetButtonDim ExprS{,ExprN1{,ExprN2{,ExprN3{,ExprN4}}}}
SetButtonFont ExprS1{,ExprS2{,ExprN1{,ExprN2{,ExprN3}}}}
BorderEdit ExprS{,ExprN}
FloatEdit se_Name{,true|false}
HideEdit ExprS{,ExprN}
IntegerEdit se_Name{,true|false}
ReadOnlyEdit ExprS{,ExprN}
SetEditColor ExprS{,ExprN}
SetEditDim ExprS{,ExprN1{,ExprN2{,ExprN3{,ExprN4}}}}
SetEditFont ExprS1{,ExprS2{,ExprN1{,ExprN2{,ExprN3}}}}
SetEditMask se_Name,se_MaskSpecs
AddListBox ExprS1,ExprN1,ExprN2,ExprN3,ExprS2{,ExprS3}
AddListBoxItem ExprS1,ExprS2
ClearListBox ExprS
DeleteListBoxItem ExprS{,ExprN}
EnableListBox ExprS{,ExprN}
FocusListBox ExprS
HideListBox ExprS{,ExprN}
RemoveListBox ExprS
SetListBoxColor ExprS{,ExprN}
SetListBox ExprS{,ExprN}
SetListBoxDim ExprS{,ExprN1{,ExprN2{,ExprN3}}}
SetListBoxFont ExprS1{,ExprS2{,ExprN1{,ExprN2{,ExprN3}}}}
SortListBox ExprS{,ExprN}
AddRBGroup
ExprS1,ExprN1,ExprN2{,ExprN3{,ExprN4{,ExprN5{,ExprS2{,ExprS3{,ExprS4}}}}}}
AddRBGroupButton ExprS1,ExprS2
ClearRBGroup ExprS
DeleteRBGroupButton ExprS{,ExprN}

EnableRBGroup ExprS{,ExprN}
FocusRBGroup se_Name
HideRBGroup ExprS{,ExprN}
RemoveRBGroup ExprS
SetRBGroup ExprS{,ExprN}
SetRBGroupColor ExprS{,ExprN}
SetRBGroupColumns ExprS{,ExprN}
SetRBGroupDim ExprS{,ExprN1{,ExprN2{,ExprN3{,ExprN4}}}}
SetRBGroupFont ExprS1{,ExprS2{,ExprN1{,ExprN2{,ExprN3}}}}
AddCheckBox ExprS1,ExprN1,ExprN2{,ExprS2{,ExprN3{,ExprN4{,ExprS3}}}}
EnableCheckBox ExprS{,ExprN}
FocusCheckBox ExprS
HideCheckBox ExprS{,ExprN}
RemoveCheckBox ExprS
SetCheckBox ExprS{,ExprN}
SetCheckBoxCaption ExprS1{,ExprS2}
SetCheckBoxColor ExprS{,ExprN}
SetCheckBoxDim ExprS{,ExprN1{,ExprN2}}
AddSlider
ExprS1,ExprN1,ExprN2{,ExprN3{,ExprN4{,ExprN5{,ExprN6{,ExprN7{,ExprN8{,ExprS3}}}}}}}
EnableSlider ExprS{,ExprN}
FocusSlider ExprS
HideSlider ExprS{,ExprN}
HideSliderDial ExprS{,ExprN}
RemoveSlider ExprS
ShowSliderBar ExprS{,ExprN}
SetSliderPos ExprS{,ExprN}
SetSliderMin ExprS{,ExprN2}
SetSliderMax ExprS{,ExprN1}
SetSliderBarStart ExprS{,ExprN1}
SetSliderBarEnd ExprS{,ExprN2}
SetSliderDim ExprS{,ExprN1{,ExprN2{,ExprN3}}}
AddSpinner
ExprS1,ExprN1,ExprN2{,ExprN3{,ExprN4{,ExprN5{,ExprN6{,ExprN7{,ExprN8{,ExprN9{,ExprS2
}}}}}}}}
EnableSpinner ExprS{,ExprN}
FocusSpinner se_Name
HideSpinner ExprS{,ExprN}
RemoveSpinner ExprS
SetSpinner ExprS{,ExprN}
SetSpinnerMin ExprS{,ExprN}
SetSpinnerMax ExprS{,ExprN}
SetSpinnerIncr ExprS{,ExprN}
SetSpinnerWrap ExprS{,ExprN}
SetSpinnerDim ExprS{,ExprN1{,ExprN2{,ExprN3{,ExprN4}}}}
AddMemo ExprS1,ExprN1,ExprN2{,ExprN3{,ExprN4{,ExprS2{,ExprS3}}}}
AddMemoLine ExprS1,ExprS2
BorderMemo ExprS{,ExprN}
ClearMemo ExprS
DeleteMemoLine ExprS{,ExprN}
EnableMemo ExprS{,ExprN}
FocusMemo ExprS

HideMemo ExprS{,ExprN}
ReadOnlyMemo ExprS{,ExprN}
RemoveMemo ExprS
SetMemoScrollBars ExprS{,ExprN}
SetMemoColor ExprS{,ExprN}
SetMemoFont ExprS1{,ExprS2{,ExprN1{,ExprN2{,ExprN3}}}}
SetMemoDim ExprS{,ExprN1{,ExprN2{,ExprN3{,ExprN4}}}}
SetMemoSelection ExprS{,ExprN1{,ExprN2{,ExprN3}}}
SetMemoText ExprS1,ExprS2
WrapMemo ExprS{,ExprN}
AddTimer ExprS{,ExprN}
RemoveTimer ExprS
SetTimer ExprS{,ExprN}
SetTimerPeriod ExprS{,ExprN}
SetTimerTicks ExprS{,ExprN}

- **Modified the following commands:**

Print {Expr,Expr;Expr...}
Read the help on this command. It has been modified to allow for wrapping too long text and also it now allows for no line feed if terminated with a , or ;. Also see the CommaTab {On|Off} command, which allows for reversing the action of ; and ,.

SetPixel {ne_X{,ne_Y{,ne_Color}}}
This command now allows for optional parameters. No effect on old programs.

mTextFR VarA,se_FileName
This command was returning an array with an element count 1 too many. It is now corrected. You may want to check your old programs for any effects.

BmpToGray
se_SourceFileName{,se_ToFileName{,ne_RedRatio{,ne_GreenRatio{,ne_BlueRatio}}}}
This command now allows for control over how the gray scale image is formed. There is no effect on older programs.

- **Modified the following functions:**

Random(ExprN)
This function now ***can*** return a random float greater than or equal to 0 and less than 1. It still returns an integer as before too. Read the help section. There should be no effect on old programs.

MsgBox(VarA{,se_Text})
This function now allows you to specify a specific title instead of using the first element in the array to be the title. There is no effect on old programs.

ToNumber(Expr{,ne_Default})
This function now allows a default numerical value if the string cannot be converted. It also converts Hex and Binary strings to intgers if they are valid. This function now makes HexToInt() and BinToInt() obsolete. They are still available but it is better to use this function to convert hex or binary strings.There is no effect on old programs.

Version 3.2.0

This version is an errata for version 3.1.0. Additionally there are 7 new function, 6 new commands, 5 modified commands and 3 cancelled commands.

The following errors were corrected:

- Some typos and errors in the Help File.
- The Ctrl+F key combination within the Help Screen was going to the top of the help file before searching. This is now not the case. The search will always start from the current cursor position. If you do desire to search from the top of the help file then use Ctrl+Home to go to the top of the file.
- The setting of the edit box height in the AddEdit command was not working.
- A problem with the AddEdit command caused an error if you tried to create more than 10 edit boxes.
- The BMPCompare command was not working properly.
- The mReadBMP_R command was not working properly. It is now **not** available as a command, a modified version of the mReadBMP command now allows you to do the same functionality (see below).
- Added a suite of functions to allow for capturing images from TWAIN compliant devices.
- An error occurs on computers with the Spanish decimal format setup. The Spanish system uses a comma (,) as a decimal separator and a dot (.) as a thousands separator. A Spanish Windows setup causes an error in RB. This has been rectified. In RB you have to use the dot (.) as a floating point format decimal point. You have to continue to do so, but for Spanish users, you do not have to change the Windows setup in order to not cause an error. So Spanish users can keep their Windows setup as normal and use RB, but within an RB program you have to use 4.1 to define a float number of four and one tenth.

- **Added the following functions:**

  EditHasFocus(ExprS)
  NumParts(ExprS1,ExprS2)
  CaptureRdy()
  CaptureSrc()
  CaptureDlg({ExprS})
  CaptureImage({ExprS})
  PixelClr(ExprN1,ExprN2)

- **Added the following Commands:**

  CbFitBMP {ExprS{,ExprN1{,ExprN2}}
  mCombineClr VarA1,VarA2,VarA3,VarA4
  ResizeBMP {ExprS1{,ExprN1{,ExprN2{,ExprS2}}}}
  mGraphPaper VarA
  mPlotXY VarA1,VarA2,VarA3
  BmpStats VarA{,ExprS}

- **Modified the following commands:**

  mReadBMP VarA{,ExprS{,ExprN}}
  mWriteBMP VarA1{,ExprS}
  WaitKey {Expr,}Var
  Input {Expr,}Var
  Input {Expr,}VarA[ExprN{,ExprN...}]

- **The following commands are no longer available:**

  mReadBMP_B VarA,ExprS
  mReadBMP_G VarA,ExprS
  mReadBMP_R VarA,ExprS

Version 3.1.0

The additions and modifications in this version are:

- Many typos and other errors have been corrected.
- The problem with the standard 16 colors reporting different values on different machines has been corrected and now ReadPixel will report the colors correctly 0-15 on all machines.
- The LightBlue color was actually darker than the Blue color. It is now a lighter shade of blue.
- Additional color related functions have been added.
- Some new function to set/clear/get bits and bytes in a number have been added.
- An entire suite of financial functions have been added.
- A set of commands that facilitate image processing on bitmap files or images from/to the Windows Clipboard area.
- A set of commands that facilitate reading from and writing to images from the Windows Clipboard area.
- A set of commands and functions that facilitate the use of Edit boxes have been added.
- A new menu option ***Merge*** (*Ctrl+G*) under the ***File*** menu has been added. This option allows you to select a file to be merged to the end of the file you are working on. This makes it easy to incorporate a suite of subroutines written previously into your current project.
- A new menu option ***Set Directory*** under the ***File*** menu has been added. This option allows you to select a directory as a working directory. Normally when you save or open a file (.BAS) the directory where the file is becomes the working directory. So when you run the program currently in the IDE you will be running within this directory. This menu option allows you set a working directory without having to open or save any files.
- The **%** operator has been modified so that the resultant is always a floating point number regardless of the operands. So, whereas before 40%3 would result in 1 now it will result in 1.2.
- The help system has been modified to be a customizable separate file called RobotBASIC_HelpFile.RTF. This is an RTF formatted file that has to reside inside the same folder where the RobotBASIC.exe is contained. The file can be modified to include special help catered to your requirements in a section called Customized Help. RobotBASIC has a reference to this section and as long as you maintain the section name you can have it anywhere in the help file and RB will jump to that position whenever the user requests the section. This feature can be of utility to teachers. A teacher may customize the help file to the needs and level of the students. Moreover, the help file may be used to give extra information to students regarding coursework materials.

  Note: This file is best modified using the WordPad program provided in all MSWindows systems. It must be saved as an RTF format file. MSWord can read and write RTF format files but it may not save the version of the RTF format required by RobotBASIC. If you find that the file is garbled when you read it in the RobotBASIC help screen, then load and save the file again using the WorPad program. The headings in the file have the character '_' inserted within the heading. This is important and the heading must remain as is for RB to be able to locate the heading within the help file.
- The Rectangle and Circle commands had a an error where they were drawing the right and bottom edges one pixel less than specified. Previously the statement Rectangle 100,100,250,350 would (assuming a line width of 1 pixel) draw a rectangle (or circle with the circle command) that starts at 100,100 but the right and bottom edges would be at 249,349 not 250,350 as specified. This has been corrected.

  Note: Check your programs for any adverse effect due to this change. You may need to change your program to draw 1 pixel less if you want to maintain the same effect as before with the same commands.

This version has 31 new commands, 2 modified function and 27 new function.

New Commands:

AddEdit ExprS1,ExprN1,ExprN2{,ExprN3{,ExprN4{,Expr{,ExprS3}}}}
RemoveEdit ExprS
SetEdit ExprS,Expr

EnableEdit ExprS,ExprN
FocusEdit ExprS
FocusButton ExprS
RectangleWH ExprN1,ExprN2,ExprN3,ExprN4{,ExprN5{,ExprN6}}
eRectangleWH ExprN1,ExprN2,ExprN3,ExprN4{,ExprN5{,ExprN6}}
CircleWH ExprN1,ExprN2,ExprN3,ExprN4{,ExprN5{,ExprN6}}
BmpToGray ExprS1{,ExprS2}
BmpNegative ExprS1{,ExprS2}
BmpToBW ExprS1{,ExprN{,ExprS2}}
BmpEdges ExprS1{,ExprN1{,ExprS2{,ExprN2}}}
BmpRGB ExprS1,ExprN1,ExprN2,ExprN3{,ExprS2}
BmpContrast ExprS1,ExprN1{,ExprN2{,ExprS2}}
BmpFindClr ExprS1,ExprN1,VarN1{,VarN2{,ExprN2{,ExprN3{,ExprN4{,VarA}}}}}
BmpCompare ExprS1{,ExprS2{,ExprS3{,ExprN}}}
mReadBMP_R VarA,ExprS
mReadBMP_G VarA,ExprS
mReadBMP_B VarA,ExprS
PrintScr
OverlayText ExprN1,ExprN2,Expr{,ExprS{,ExprN3{,ExprN4}}}
ScrToCb {ExprN1{,ExprN2{,ExprN3{,ExprN4}}}}
ScrFromCb {ExprN1{,ExprN2{,ExprN3{,ExprN4{,ExprN5{,ExprN6}}}}}}
FitCb {ExprN1{,ExprN2{,ExprN3{,ExprN4}}}}
BmpToCb {ExprS{,ExprN1{,ExprN2{,ExprN3{,ExprN4}}}}}
CbToBMP {ExprS}
FlipCb
MirrorCb
RotateCb {ExprN}
SizeCb VarN1,VarN2

New Functions (M = Modified):
EditChanged(ExprS)
GetEdit(ExprS)
BlueValue(ExprN)
GreenValue(ExprN)
RedValue(ExprN)
ClrBit(ExprN1,ExprN2)
ClrByte(ExprN1,ExprN2)
GetBit(ExprN1,ExprN2)
GetByte(ExprN1,ExprN2)
SetBit(ExprN1,ExprN2)
SetByte(ExprN1,ExprN2)
M **Hex**(ExprN1{,ExprN2})
M **FilePrompt**({Expr})
FileSave({Expr})
FileCopy(ExprS1,ExprS2{,ExprN})
Spell(ExprN)
ProbG(ExprN1,ExprN2,ExprN3)
ProbGI(ExprN1,ExprN2,ExprN3)
ff_CIFV(PV,INTR,TERM)
ff_CII(PV,FV,TERM)
ff_CIT(PV,INTR,FV)
ff_FV(PMT,INTR,TERM,TYPE)

ff_FVP(FV,INTR,TERM,TYPE)
ff_FVT(PMT,INTR,FV,TYPE)
ff_PV(PMT,INTR,TERM,BAL,TYPE)
ff_PVP(PV,INTR,TERM,BAL,TYPE)
ff_PVT(PMT,INTR,PV,BAL,TYPE)
ff_SLN(COST,SALVAGE,LIFE)
ff_SYD(COST,SALVAGE,LIFE,PERIOD)

Version 3.0.3

This version has 3 new commands, 1 modified command (with an M),8 new function and 3 new flow control statment.

Flow Control:

OnError *Label*
OnError *Expr*
OnError

Commands:

M AddButton ExprS1,ExprN1,ExprN2{,ExprN3{,ExprN4{,ExprS2}}}
GetError *VarN1{,VarS{,VarN2{,VarN3}}}*
WaitNoKey *{ExprN}*
WaitNoKeyE *ExprN1{,ExprN2}*

Functions:

Spawn(ExprS1,ExprS2,ExprN)
ToTime(ExprN)
mSum(VarA)
mAverage(VarA)
mMin(VarA)
mMax(VarA)
mCount(VarA)
mRange(VarA)
mStdDev(VarA)
mVariance(VarA)

Version 3.0.2

This version has 1 new command and two error correction.

Errors Corrected:

The constants kc_LMouseB, kc_RMouseB and kc_MMouseB were not correctly defined.
The function acos() was not working properly (only in version 3.0.0 it worked ok before)

New Commands:

ToBMP ExprS1,ExprS2

Version 3.0.0

This version has 39 new commands, 53 new functions, 1 modified function and 3 modified commands. The major additions are to allow for:

- Bitmap manipulation.
- Animation with back buffer.
- More extensive graphics commands.
- More powerful user interface with better keyboard input and screen output.
- Support for Joystick input.
- Playing of WAV files.
- Manipulation of directories and files within your programs.

- More mathematical functions.
- More helper functions and commands.
- Support for more than 16 colors (read colors discussion in the Graphics & Screen Commands section).
- A simple 3D Graphics engine for converting 3D points to screen 2D points and to evaluate visible and invisible surfaces and some operations like rotation.
- Two new flow control statements (Terminate and Exit) that facilitate better non-programmer user interface. Also see the new command AbortMethod and the IDE section for details on how to control the termination behavior of your programs.
- The ability to create a binary version of a program file. This gives the programmer the option to distribute programs that cannot be viewed, but will otherwise run just like a normal text file version. See the IDE section for more details on how to do this.

This version supports the use of any colors supported by your version of Windows as compared to older versions that only supported 16 colors, and some new commands and functions have been added to support this. The support for the old colors is still active and programs written with the old versions will run without modifications.

A new menu item (***Colors***) has been added under the ***Help*** menu (*Ctrl+L*). This item enables the programmer to select a color at design time from a colors dialog. The color value will be copied to the clipboard to allow for pasting in your programs (also see the new PromptColor() function).

A new menu item (***Fonts***) has been added under the ***Help*** menu. This item enables the programmer to select a font name at design time from a fonts dialog. The name selected will be copied to the clipboard to allow for pasting in your programs.

A new menu option ***Auto Indent*** (*F6*) under the ***Edit*** menu has been added to allow and not allow automatic indenting of a new line to the previous line's first character level of indentation.

A new menu option ***Space Tabs*** (*F5*) under the ***Edit*** menu has been added to allow or not allow replacing of the tab key with spaces.

Two new menu items ***Indent Block*** (*F8*) and ***Un-Indent Block*** (*F7*) from the ***Edit*** menu have been added. These allow for indenting a highlighted block of text or un-indenting it one space at a time. The text will not un-indent if the first character of a line is not the space character(or tab). That means you will not lose any text. This is useful for lining up entire blocks to fit within indented formatting of program structure.

A new debugging feature has been added to help in debugging your code. In the editor screen there is now a new menu option under the ***Run*** menu called ***View Variables Table*** (*Ctrl+B*) . This enables you to view the values, types and names of all the variables at the end of the program execution, whether due to a normal terminate or a forced one due to an error or user action. This table can be a tremendous help in debugging. This table is also available from the Debug Screen by using a new button ***View Variables Table*** in that screen.

The Help screen has been modified and reorganized. Also new constants have been added to support certain commands and functions.

New Commands (Ones with an M are only modified):

AbortMethod ExprN
AddButton ExprS,ExprN1,ExprN2{,ExprN3{,ExprN4}}
Arc ExprN1,ExprN2,ExprN3,ExprN4{,ExprN5{,ExprN6{,ExprN7{,ExprN8}}}}
eRectangle ExprN1,ExprN2,ExprN3,ExprN4{,ExprN5{,ExprN6}}
FitBMP {ExprS{,ExprN1{,ExprN2{,ExprN3{,ExprN4}}}}}
Flip {ExprN}

FlipBMP {ExprS{,ExprN1{,ExprN2{,ExprN3{,ExprN4{,ExprN5{,ExprN6}}}}}}}
FloodFill2 ExprN1,ExprN2{,ExprN3{,ExprN4}}
ge3Dto2DV ExprN1,ExprN2,ExprN3,ExprN4,ExprN5,ExprN6,ExprN7,ExprN8,ExprN9,VarN1,VarN2
ge3Dto2DA VarA1,VarA2
GetCursor Var
GetKeyE Var
geRotVx ExprN1,ExprN2,ExprN3,ExprN4,VarN1,VarN2,VarN3
geRotVy ExprN1,ExprN2,ExprN3,ExprN4,VarN1,VarN2,VarN3
geRotVz ExprN1,ExprN2,ExprN3,ExprN4,VarN1,VarN2,VarN3
GetButton VarArc ExprN1,ExprN2,ExprN3,ExprN4{,ExprN5{,ExprN6{,ExprN7{,ExprN8}}}}
geVisibles VarA1,VarA2,VarA3{,VarA4}
JoyStick ExprN,VarN1,VarN2,VarN3,VarN4
JoyStickE ExprN,VarA
Line ExprN1,ExprN2,ExprN3,ExprN4{,ExprN5{,ExprN6}}
mBezier VarA{,ExprN1{,ExprN2}}
MirrorBMP {ExprS{,ExprN1{,ExprN2{,ExprN3{,ExprN4{,ExprN5{,ExprN6}}}}}}}
M mRead VarA,ExprS (see the Array Commands section for more details)
M mWrite VarA,ExprS (see the Array Commands section for more details)
mReadBMP VarA,ExprS
mTextFW VarA,ExprS
mTextFR VarA,ExprS
mWriteBMP VarA,ExprS
Pie ExprN1,ExprN2,ExprN3,ExprN4{,ExprN5{,ExprN6{,ExprN7{,ExprN8}}}}
PlaySong ExprS
PlayWav {ExprS{,ExprN1{,ExpN2}}}
ReadBMP {ExprS{,ExprN1{,ExprN2{,ExprN3{,ExprN4{,ExprN5{,ExprN6}}}}}}}
RemoveButton ExprS
RotateBMP {ExprS{,ExprN1{,ExprN2{,ExprN3{,ExprN4{,ExprN5{,ExprN6{,ExprN7}}}}}}}}
SetCursor {ExprN}
SizeBMP ExprS,VarN1,VarN2
M Swap Var1,Var2
M Swap VarA[ExprN1,ExprN,2...],Var
M Swap Var,VarA[ExprN,ExprN,...]
M Swap VarA1[ExprN1,ExprN2,...],VarA2[ExprN1,ExprN2,...]
Transparent ExprN
WriteBMP {ExprS{,ExprN1{,ExprN2{,ExprN3{,ExprN4}}}}}
xyInput Var{,ExprN1{,ExprN2{,Expr3{,Expr4{,ExprN5}}}}}
xyTtext ExprN1,ExprN2,ExprS1{,ExprS2{,ExprN3{,ExprN4}}}

New Functions (Ones with an M are only modified):

AbortFlag()
ACosH(ExprN)
ASinH(ExprN)
ATanH(ExprN)
BinToInt(ExprS)
Center(ExprS1,ExprS2,ExprN)
ConsToClr(ExprN)
CosH(ExprN)
DirCreate(ExprS)
DirCurrent()
DirCount()
DirExists(ExprS)

DirList()
DirPrompt()
DirRemove(ExprS)
DirSet(ExprS)
DtoR(ExprN)
ErrMsg(ExprS1,ExprS2,ExprN)
FilChangeExt(ExprS1,ExprS2)
FilDelete(ExprS)
FilDir(ExprS)
FilDrive(ExprS)
FilesCount({ExprS})
FilesList({ExprS})
FilExists(ExprS)
FilExt(ExprS)
FilName(ExprS)
FilPath(ExprS)
FilePrompt({ExprN})
FilRename(ExprS1,ExprS2)
FilSearch(ExprS1,ExprS2)
Insert(ExprS1,ExprS2,ExprN)
JustifyL(ExprS1,ExprS2,ExprN)
JustifyR(ExprS1,ExprS2,ExprN)
KeyDown(ExprN)
Limit(ExprN1,ExprN2,ExprN3)
Log2(ExprN)
LogB(ExprN1,ExprN2)
MaxV(ExprN1,ExprN2)
MinV(ExprN1,ExprN2)
MsgBox(VarA)
M Pi({ExprN})
PromptBMP({ExprS})
PromptColor({ExprN})
RandomG(ExprN1,ExprN2)
Replace(ExprS1,ExprS2,ExprN)
RGB(ExprN1,ExptN2,ExprN3)
RtoD(ExprN)
SinH(ExprN)
Soundex(ExprS{,ExprN})
StrInput(ExprS1,ExprS2,ExprS3)
TanH(ExprN)
TextBox(ExprS1,ExprS2)
WavBusy()

Version 2.0.3

This version has 22 new commands, 1 new function, 4 modified functions and 6 modified commands. The major additions are to add a set of commands to allow for serial and ports I/O. Some of the Robot simulator commands and functions have been modified to allow for a serial communications protocol to take place if desired. The changes are transparent to the normal operations of the simulator. They only take effect if the new rCommPort command is used to cause the communications protocol to occur instead of the normal simulated operation on the screen.

A new version of the GoSub flow control statement has been added, refer to the Flow Control Statements

section for more details.

The help screen has been modified to incorporate some new sections and some reorganized pages.

New Flow Control Statement:

GoSub Expr

New Commands:

HexToInt(Expr)
GetLineWidth Var
MicroDelay {ExprN}
Sound ExprN1,ExprN2{,ExprN3}
Speaker ExprN

New Functions:

Timer( )

New Ports & Serial I/O Commands:

CheckSerBuffer VarN
ClearSerBuffer {ExprN}
GetTimeOut VarN
InPort ExprN1,VarN
OutPort ExprN1,ExprN2
PPortIn VarN
PPortOut ExprN1
rCommPort ExprN1 {,ExprN2 {,ExprN2 {,ExprN3 {,ExprN4 {,ExprN5}}}}}
SerBytesIn ExprN,VarS,VarN
SerIn VarS
SerOut Expr {,Expr {; Expr ...}}
SetCommPort ExprN1 {,ExprN2 {,ExprN2 {,ExprN3 {,ExprN4 {,ExprN5}}}}}
SetPPortNumber {ExprN}
SetTimeOut {ExprN}
VPPortIn ExprN1,VarN
VPPortOut ExprN1,ExprN2

Modified Commands (to allow for serial I/O protocol):

rForward ExprN
rGPS VarNX,VarNY
rLocate ExprN1,ExprN2
rPen ExprN
rSpeed ExprN
rTurn ExprN

Modified Functions (to allow for serial I/O protocol):

rBeacon (ExprN)
rCompass ()
rLook ({ExprN})
rRange ({ExprN})