

Unofficial VRBASIC user manual

Stephen Moss

This is an unofficial manual intended to help those programming the Atari Jaguar using VRBASIC, a BASIC to C interpreter for Windows. I hope you find it useful however both it and VRBASIC are still under development and thus I do not guarantee it is 100% accurate or up to date with the latest functions.

Installation:

First you will need to install the update development system released by BelBoz available from [here](#), it is recommended that you install this to the expected default directory of Drive:\Jaguar

Next you will need to install an Atari Emulator (saves having to upload your code to a Jaguar to test it), you could probably use any emulator that supports the .64 and .BJL file types by altering the appropriate line in VRBASIC's *convert_bulid_deploy.BAT* file, however as VRBASIC has been tested in and is set up to use [Virtual Jaguar](#) it is probably best to use that. Make a note of its install folder it as you will need it later, a good place might be in the BelBoz Jaguar development system folder i.e. Drive:\Jaguar. To install Virtual Jaguar copy the Zip file to the relevant Drive/Folder, right click on it and select the "Extract Here" option.

Finally, copy your VRBASIC zip file into the folder you installed the BelBoz Jaguar development system to, i.e. Drive:\Jaguar, as with Virtual Jaguar right click on the Zip file select the "Extract Here" option, this will install VRBASIC to Drive:\Jaguar\VRBasic

Go to the Drive:\Jaguar\VRBasic folder, right click on the *convert_bulid_deploy.BAT* file and select "Edit" from the menu. Edit the fourth line down, the one that begins...

```
"set PATH_TO_VJ="
```

and enter the location of your Virtual Jaguar installations at the relevant point. Note the reference two lines below that about making changes to the *Makefile*. If the recommended installation paths and methods have been used the Makefile will be located in...

```
Drive:\Jaguar\VRBasic\BIN
```

To edit the *Makefile* right click it, select "Open" from the menu, then select "Notepad", edit as appropriate and save the file. **Note** : at the time of writing it will be necessary to make a copy your clib.a file and rename the copy as libclib.a otherwise compilation will fail.

General Information:

Capitalisation of Keywords

At the time of writing if you are not coding in an IDE that does it for you, you *must* capitalise the first character of each key word otherwise no C code will be produced. For example

Dim X As integer, will fail as would...

Dim X as Integer

dim X As Integer

Only when all key words are capitalised as in the example below will C code be produce...

Dim x As Integer

Code Comments

Most of your code will be written within subroutines (see Subroutines section), you can comment out a line of code or add a line of reference comments by starting the line with an apostrophe/single quote (') symbol, anything in a line following the apostrophe will be seen as a comment, for example...

```
Code :
'This is a comment - X holds the count for a loop
Dim X As Integer
```



*You may notice that in the example code throughout this manual that I have placed comments following the code on lines that either declare subroutines, call subroutines or start with Dim. This is done for clarity however, this **should not** be done in your program code as comments in those positions are not currently supported and will cause false compiles (see below).*

False Compiles

Sometimes the compile will appear to complete, in that Jiffi will briefly flash on the screen as it does for an actual compile, however you need to keep an eye on the command window. If there is no pause for you to see the resulting C code and subsequent pause requiring you to press any key to continue there is a problem with your code and you previously compiled code will not have been overwritten. Scroll through the text in the command window for clues to the problem.

Screen Size

The screen size used by VRBASIC is 300 x 200, with position 0, 0 being located at the top left corner of the screen. This applies to all text and graphic commands, where X is horizontal displacement and Y is vertical displacement from 0, 0.

VRBASIC code filenames and locations

The file you want to compile must be located in the Drive:\Jaguar\VRBasic\src\ folder and have the file extension VRB. Currently VRBASIC is set to convert a file called jag.VRB, to compile a different program edit the second line of the *convert_bulid_deploy.BAT* file appropriately.

Constants, Variables & Arrays:

You must **Dim**ension constants, variables and arrays before you can use them and involves giving them a Name to which they are referred throughout the program and a size/type. The format for this is...

Constants & Variables

Dim <Constant Name> As <Type> = <Initial Value>

Dim <Variable Name> As <Type> = <Initial Value>

Initial value is option and can be set later in the program. Acceptable constant and variable sizes/types are...

Boolean	A 16-bit value logical operator with only two states, True and False where 0 = False and anything else = True
Byte	An 8-bit number whole number (no decimal places) from 0 to 255
Double	An 64-bit floating point number (has decimal places) from -1.79769313486231E308 to 1.79769313486231E308
Integer	An 32-bit whole number (no decimal places) from -2,147,438,648 to 2,147,438,647
Long	An 64-bit number whole number (no decimal places) from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Single	An 32-bit floating point number (has decimal places) from -3.4028235E38 to 3.4028235E38
Sprite	A Sprite Image

Code:

```
Dim Computer_Player as Boolean      'Create a variable indicating if a computer opponent is required
Dim Teams as Byte                  'Create a variable holding the number of teams playing
Dim Colour as Integer              'Create a Variable holding the number of a Colour for Text
Dim Pi as Single                   'Create a Variable holding the value of Pi
Dim ImgWall as Sprite              'Create a #variable holding a Sprite
```

Once declared the values of constants and variables can be set during run time i.e.

Code:

```
Pi = 3.141          'Set constant Pi to a value if 3.141
Colour = 3 * Pi     'Set variable Colour to a value equal to 9.423 (3 * Pi)
```

Although it is best to assign constants their values and variables and initial value when declare them i.e.

Code:

```
Dim Computer_Player as Boolean = False  'Create & preset to No a variable for computer opponent requirement
Dim Teams as Byte = 2                  'Create & preset to 2 a variable for the number of teams playing
Dim Colour as Integer 256 AND 23       'Create & preset a variable holding a calculated text Colour
Dim Pi as Single = 3.14159             'Create & present a Constant holding the value of Pi
```

Arrays

Dim <Array Name as String>(Array length as Integer) As <Type>

An array is a collection of elements of related numerical or text data, for example a collection of player names for a team or data relating to the specifications of an object such as size, speed and location.

Arrays can be dynamic length where either the number of elements may vary or is initially unknown or they can be can be fixed length, i.e. there are 11 players in a football team so an array of player names would only need to be 11 elements long. Each element in an array can be individually accessed by specifying its index with that array, the first element in an array is always 0 not 1.

Code:

```
Dim TeamsNames() as String            'Create a dynamic size array of elements to hold team names, where each
                                      'element is addressed by an index value
Dim TeamNames(5) as String            'Create a six element (0-5) fixed length array holding team names
Player_1 = TeamNames (0)              'Get player 1 name from the first element of the TeamNames array
TeamNames(3) = "Fred Bloggs"         'Set the name stored in the fourth element of the TeamNames array

Dim PlayerScore(5) as Integer          'Create a six element (0-5) fixed length holding the players Scores
Player_2_Score = PlayerScore (1)      'Get player 2 score from the second element of the PlayerScore array
PlayerScore(3) = 25                   'Store the score or the fourth player the PlayerScore array
```

Global Constants, Variables and Arrays

Those declares that define constants (a value which will not change) should have their values assigned when declared as mentioned earlier. Variables and Arrays can have their values assigned later however, it would be best to assign them an arbitrary value of say 0 or "" (if a string) when declared otherwise you may end up working with random data.

The use of such variables, constant and arrays will generally be "local", i.e. restricted to use in the subroutines in which they were declared, unless explicitly passed to other subroutines for their use (see Subroutines) or declared within a "Globals" Module.

As you can see in the example below by declaring the lvlData array in the Globals module it is now accessible to both the DrawScreen() and Main() subroutines.

```
Code:
Module Globals
  Dim lvlData() As Byte      'Declare Global Array lvlData
  Dim ImgWall As Sprite     'Declare Global Sprite
End Module

Sub DrawScreen ()
  Dim xpos As Integer = 0   'Declare Local variable xpos and set initial value as 0
  Dim ypos As Integer = 0   'Declare Local variable ypos and set initial value as 0
  DIM idx As Integer = 4    'Declare Local variable idx as the element index reference for lvlData Array
                          'and set its initial value to 4
  If lvlData(idx) = 88 Then ShowImage (ImgWall,xpos,ypos) 'If the value of the fifth element in the
                          lvlData array = 88 then display the image.
  End If

Sub Main ()
  LoadTextData (lvlData, "D:\Jaguar\My_Game\level_1Data) 'Load level one data in ton lvldata Array
End Sub
```

General Graphic Commands:

ClearScreen()

Clear the screen

```
Code:
ClearScreen() 'Clear The screen
```

DrawLine (x1, y1, x2, y2, c)

Draw a line of colour C from x1, y1 to x2, y2.

```
Code:
DrawLine (0,0,200,120,16) 'Draw a diagonal line of colour 16 from 0,0 to 200,120
```

DrawPixel (x as Integer, y as Integer, c as Integer)

Draw a pixel of colour C at screen location X, Y.

```
Code:
DrawPixel (10,20,30) 'Draw a pixel 10 pixels left, 20 pixels down in colour 30
```

SetGraphicsMode ()

This appears to be a VRBASIC internal call that correctly sets up the Jaguar for the display mode used by the other VRBASIC commands and therefore it **must be** the first line within the Main() subroutine.



If this command is omitted everything will appear to compile ok however the graphical result will be a black screen.

```
Code:
Sub Main ()
  SetGraphicsMode ()
  Code Statements Here
End Sub
```

Palette Commands:

CopyPalette (*IndexOfPaletteColour*, *Image*)

Copies the complete colour palette of a give image into the palette at palette entry [*IndexOfPaletteColour*]

Code:
?

SetPalette565 (*ColourIndex as Byte*, *R as Byte*, *G as Byte*, *B as Byte*)

Sets the colour of the palette at the given *ColourIndex* (0-255) with the given RGB values, assume these ranges...

R 0 to 31

G 0 to 63

B 0 to 31

Code:
?

SetPalette888 (*ColourIndex as Byte*, *R as Byte*, *G as Byte*, *B as Byte*)

Sets the colour of the palette at the given *ColourIndex* (0-255) with the given RGB values. This instruction is convenient when taking values from PC image drawing software.

Assume these ranges...

R 0 to 255

G 0 to 255

B 0 to 255

Down scales them to the nearest Jaguar equivalent of...

R 0 to 31

G 0 to 63

B 0 to 31

Code:
?

SetPallete (*IndexOfPaletteColour as Integer*, *R as Byte*, *G as Byte*, *B as Byte*)

Update the palette entry [*IndexOfPaletteColour*] with the required R,G,B values

Code:
?

LoadImage (*ImageDataStructure*, *ImageFilePath*)

Code:

```
LoadImage (ImgWall, "D:\jaguar\VRBasic\src\Samples\ArtAssets\TileA2_16_Colors_Palette.gif", 8)
```

LoadBinaryData (*binDataArray*, *BinaryImageFilePath*)

Code:
?

Logic Operators:

AND	Logical Bitwise AND
OR	Logical Bitwise OR
NOT	Logical NOT
XOR	Logical Bitwise XOR

Loops and Conditional operators:

Conditional loops repeat a section of code either for the specified number of iterations or until a condition is met.

Do Until

Do Until <relational comparison>

[Statements]

Loop

The Do – Until loop will perform the code within the loop while the result of the comparison is false

Code:	
Do Until Lives = 0	'Run the code in this loop until the player runs out of lives
Code Statements here	'Code to be executed within the loop
Loop	're-execute loop code until Lives = 0

Do While

Do While <relational comparison>

[Statements]

Loop

The Do – While loop will perform the code within the loop while result of the comparison is true

Code:	
Do While Lives <> 0	'Run the code in this loop until the player runs out of lives
Code Statements here	'Code to be executed within the loop
Loop	're-execute loop code until Lives = 0

For Next

For <Variable as Type> = <Start Value as integer> **To** <End Value as Integer> **Step** <Step Value as Integer>

[Statements]

Next

The For-Next will loop for the number of iterations defined by Start and End values. The value of the Variable will normally incremented (or decremented) by one from the Start Value until it equals the End value every time the loop runs.

The optional Step value can be used to change the amount by which the variable will be incremented (or decremented) each time the loop runs.

If counting up the Start value must be greater than the end value and the Step value (if used) must be positive, if counting down the Start value must be greater than the End value and the Step value (if used) must be negative.

Variable type can be Double, Integer, Long or Single although Integer is usually sufficient and requires the least space.

Code:	
Dim C1 as Integer	
For C1 = 1 To 5	'Draw 5 lines on the screen at the specified location
DrawLine (c1*2,0,C1*2,199,C1*16)	'Code to be Executed by loop
Next	'Draw next line (C1 incremented by 1, loop exits when C1 > 6)
Dim C1 as Integer	
For C1 = 1 To 5 Step 5	'Draw 1 line on the screen at the specified location
DrawLine (c1*2,0,C1*2,199,C1*16)	'Code to be Executed by loop
Next	'Draw next line (C1 incremented by 5, loop exits when C1 > 6)

The second sample above executes the loop code once as C1 exceeds 6 on the first next.

If Then Else

If <relational comparison> **Then**

[Statements]

Else If <relational comparison> **Then**

[Statements]

Else

[Statements]

End If

The If - Then – Else conditional code structure allows you to execute different code dependent on which of the relational comparisons being tested it true. The minimum number of conditions to test it one however you can test more than one condition in a statement by combining them with logical operators. Additionally you can use optional *Else If* to add as many additional conditions as you require.

Those statements following the optional *Else* condition are executed if none of the previous conditions are true. The relational comparison functions can be mathematical or logical.

Code:	
If Count = 0 Then	'First condition to test
Message = "Less than 3 items."	'Execute this code if the first condition is true
ElseIf Count > 7 AND count <9 Then	'Second condition to test (optional)
Message = "There are 8 items."	'Execute this code if the second condition is true
Else	
Message = "Help!"	'Execute this code if the neither condition is true (optional)

```

End If

If Lives = 0 Then           'Display the Game Over message when the player loses all their lives
    Txt.DrawString (20,13, "Game Over")
End If

If A = B AND C > D Then     'Test two conditions, note use of Logical relational operator to link
                             'both conditions being tested
    Code Statements here    'Code to be executed within the loop if both conditions are true
End If

If A = B OR C > D Then      'Test two conditions, note use of Logical relational operator to link
                             'both conditions being tested
    Code Statements here    'Code to be executed within the loop if either condition is true
End If

```

While

While <relational comparison>
 [Statements]
End While

While loop will perform the code within the loop while result of the comparison is true

```

Code:
While Lives <> 0           'Run the code in this loop until the player runs out of lives
    Code Statements here   'Code to be executed within the loop
Loop                       're-execute loop code until Lives = 0

```

Mathematical Operators:

Numbers and equations can be calculated using the standard numeric symbols...

+	Add
-	Subtract
*	Multiply
/	Divide
=	Equal
<	Less Than
>	Greater Than
<>	Not Equal
=<	Equal to or Less Than
=>	Equal to or Greater Than

```

Code:
A = 2 + 20                'Add 2 to 20
A = 400 / (30 * 1.5)      'Divide 400 by 20 (30 * 1.5)
Dim B As Integer = Pi * 4 'Declare variable B and assign it the value equal to Pi multiplied by 4

If B < 40 Then A = 10      'Set the value of A to 10 if the value of B is less than 40
End If

If B > 50 Then A = 0       'Set the value of A to 0 if the value of B is greater than 50
End If

If B <> 34 Then A = 100     'Set the value of A to 100 if the value of B is not equal to 34
End If

```

Miscellaneous commands:

Wait (Value as Integer)

Do nothing for a fixed length of time where value is in milliseconds (0.001 seconds)

```

Code:
Wait (100)  'Wait for 0.1 Seconds
Wait (500)  'Wait for 0.5 Seconds
Wait (1200) 'Wait for 1.2 Seconds

```

Sprite Commands:

LoadTextData (/vdataArray, DataTextFilePath)

Level text data stored in an basic NotePad compatible text file format where for example a “X” designates a wall sprite and a space designates a walking tile. You then simple search the date for and “X” and display the relevant sprite. For example...

```
Code:
LoadTextData (lvlData, "D:\jaguar\VRBasic\src\Samples\ArtAssets\level00.txt")

    For col = 0 To 8
        ' Check for 'X' (ASCII:88) and draw the sprite
        If (lvlData(idx)=88) Then ShowImage (ImgWall,xpos,ypos)
    Else
        ShowBlankImage (ImgWall,xpos,ypos)
    End If
    ' Irrespective of the 'X' we move right in the increments of Sprite Width
    xpos = xpos + ImgWall.xl
    idx = idx + 1
Next
```

For a screen consisting of 3 row of 8 tiles the format for the Text file would be...

```
[Level 1, Screen 1, Row 1][Level 1, Screen 2, Row 1]
[Level 1, Screen 1, Row 2][Level 1, Screen 2, Row 2]
[Level 1, Screen 1, Row 3][Level 1, Screen 2, Row 3]
[Level 2, Screen 1, Row 1][Level 2, Screen 2, Row 1]
[Level 2, Screen 1, Row 2][Level 2, Screen 2, Row 2]
[Level 2, Screen 1, Row 3][Level 2, Screen 2, Row 3]
```

FlipImage (Direction)

Flip the direction of the sprite image being displayed, supported direction are Vertical, Horizontal (8-bit transfers only) and None. All subsequent draws will use the last specified direction i.e. if you use FlipImage to flip the image vertically and then call ShowImage 100 times every time image will be draw vertically flipped.

```
Code:
?
```

ShowImage (Image, x as Integer, y as Integer)

Display the image sprite [Image] at the specified location.

```
Code:
ShowImage (WallImage,xpos,ypos)
```

ShowBlankImage (Image, x as Integer, y as Integer)

Display the blank image sprite [Image] when there is no level data. Use instead of ClearScreen () when going from one level to another.

```
Code:
If (lvldata (idx)=88) Then
    ShowImage (WallImage,xpos,ypos)
Else
    ShowBlankImage (BlankWallImage,xpos,ypos)
End If
```

ShowImageSubset (Sprite, ImgXP as Integer, ImgYP as Integer xp as Integer, yp as Integer, xl as Integer, ly as Integer)

Display a small section of a larger image, could be used for scrolling or possibly sprite animation where each subset equals one sprite frame.

ImgXP & ImgYP: Defines the top left corner of the subset
 xp & yp: Defines the screen position where the subset will be drawn
 xl & yl: Define the width and height of the subset being drawn

One of three code path are used depending on amount of data transferred in one go (8-bit, 16-bit or 32-bit) with the fastest being 4 times faster than the slowest. It is advisable that ImgXP, xp and xl are multiples of 4 to achieve the highest performance.
 An 8 x 8 tile will result in a screen of 40 x 25 tiles

```
Code:
ShowImage (WallImage,xpos,ypos)
```

Subroutines:

Most of your program code will be contained within subroutines the primary one of these which is "Main ()". Main () is where your program starts and thus your main program loop should be placed here, you can branch from Main () to other subroutines however, there are some rules...

1. Other subroutines **must** be placed before Main () in your program file i.e.

<pre>Subroutine_A Code Statements here End Sub Main () Code Statements here End Sub</pre> <p>is acceptable</p>	<pre>Main () Code Statements here End Sub Subroutine_A Code Statements here End Sub</pre> <p>is not acceptable</p>
---	---

2. You **must** only call other subroutines from Main (), you cannot nest subroutine (call one from another) i.e.

<pre>Subroutine_A Code Statements here End Sub Subroutine_B Code Statements here End Sub Main () Code Statements here Subroutine_A 'Call Subroutine A from Main () Code Statements here Subroutine_B 'Call Subroutine B from Main () End Sub</pre> <p>is acceptable</p>	<pre>Subroutine_A Code Statements here Subroutine_B 'Call Subroutine B from Subroutine A End Sub Subroutine_B Code Statements here End Sub Main () Code Statements here Subroutine_A 'Call Subroutine A Code Statements here End Sub</pre> <p>is not acceptable</p>
---	---

You can declare a subroutine using as the Sub and End Sub keyword pairing as shown below. All code between this pair of keywords will be executed when the subroutine is called.

```
Sub <Name> ([ByVal | ByRef] <VariableName> As <Type>)
End Sub
```

Code:

```
Sub SubName (Argument list) 'The SubName is the name you will refer to when calling the subroutine and
                             'the argument list is an optional of list parameters passed between this
                             'subroutine and the one that called it
  Code Statements Here
End Sub
```

If you do not need to pass parameters between the calling subroutine (i.e. Main) and the subroutine being called then leave the argument list blank, otherwise you populate the argument list using *ByVal* and *ByRef* in the sub being called and just the parameter name in the Argument list of the sub that is calling it. You can use as many arguments as necessary.

ByVal: Use the ByVal argument if you are just passing the parameter to the sub being called for its own use.

Code:

```
Sub DrawScore (ByRef PlayerScore As Integer)
  Txt.DrawString (0,0,"Score: ") 'Draw Score: at screen location 0,0 in the FontColour passed by Main
  Txt.DrawNum (9,64,PlayerScore) 'Draw Score value at screen location 0,64 using the PlayerScore value
                                'passed by Main.
End Sub

Sub Main ()
  'The main Subroutine
  Dim FontColour As Integer 'Declare a Variable to hold the FontColour
  Dim PlayerScore As Integer 'Declare a Variable to hold the Players Score
  Code Statements here go here 'Do some stuff
  UpdateScore (PlayerScore) 'Call Subroutine to display the players Score and pass PlayerScore
                             'to it for modification
  DrawScore (FontColour, PlayerScore) 'Call Subroutine that displays the players Score and passed both the
                                     'score and font colour to it
  Code Statements here 'Do some more stuff
End Sub
```


ByRef: Use the ByRef argument if you the subroutine being called needs to modify the parameter and return the modified parameter to the calling subroutine.

```

Code:
Sub UpDateScore (ByRef PlayerScore as Integer)
    PlayerScore = PlayerScore + RoundScore 'Add the round score to the current player score and pass the new
                                           'total back to Main ()
End Sub

Sub Main ()
    'The main Subroutine
    Dim FontColour As Integer 'Declare a Variable to hold the FontColour
    Dim PlayerScore As Integer 'Declare a Variable to hold the Players Score
    Code Statements here go here 'Do some stuff
    UpDateScore (PlayerScore) 'Call Subroutine that updates the players Score
    Code Statements here go here 'Do some more stuff
End Sub

```



At present the terms ByVal and ByRef are not required and **only** numerical parameters can be passed from the calling subroutine.

Text Commands:

Regular text commands:

The default screen resolution for VRBASIC is 320 (0-319) pixels wide by 200 (0 -199) pixels high and text can be printed starting at any location with that space.

The default font size is 8x8, therefore if each row is spaced 8 pixel apart a screen can accommodate 25 rows of up to 40 characters, personally I think spacing rows every 10 pixels apart looks better in which case a screen will accommodate 20 rows instead of 25. Assuming you start each line of text from the left side of the screen the locations for each line will be as follows...

8x8 Row Spacing		8 x 10 Row spacing	
Row 1	0,0	Row 1	0,0
Row 2	0,7	Row 2	0,9
Row 3	0,15	Row 3	0,19
Row 4	0,23	Row 4	0,29
Row 5	0,31	Row 5	0,39
Row 6	0,39	Row 6	0,49
Row 7	0,47	Row 7	0,59
Row 8	0,55	Row 8	0,69
Row 9	0,63	Row 9	0,79
Row 10	0,71	Row 10	0,89
Row 11	0,79	Row 11	0,99
Row 12	0,87	Row 12	0,109
Row 13	0,95 (Centre Screen)	Row 13	0,119
Row 14	0,103	Row 14	0,129
Row 15	0,111	Row 15	0,139
Row 16	0,119	Row 16	0,149
Row 17	0,127	Row 17	0,159
Row 18	0,135	Row 18	0,169
Row 19	0,143	Row 19	0,179
Row 20	0,151	Row 20	0,189
Row 21	0,159		
Row 22	0,168		
Row 23	0,175		
Row 24	0,183		
Row 25	0,191		

Txt.DrawString (x as Integer, y as Integer, Text as String Literal)

Draw some text the specified location

```

Code:
Txt.DrawString (10,10,"Atari Jaguar is cool") 'Draw Atari Jaguar is cool at screen location 10,10

```



Strings **must not** contain the following characters... ! " £ \$ % ^ & * ()

You can include numbers directly into the text string however you **must not** include computed values such as those in an array, constant or variable as random text would result, for those you must use Txt.DrawNum instead, i.e.

```

Code:
Txt.DrawString (10,10,"This is the 1st line of Text") 'This is fine as is...

Dim Name As String = "Fred"
Txt.DrawString (10,10,"Player Name: " + Fred) 'This will display Player Name: Fred

Dim Score as Integer = 100
Txt.DrawString (10,10,"Player 1 " + Score) 'This will not work

```

Txt.DrawStringLn (*x as Integer, y as Integer, Text as String Literal*)

As per **Txt.DrawString** but after writing sets the location to the left hand side of the next row

Code:

```
Txt.DrawString (10,10,"Atari Jaguar is cool") 'Draw Atari Jaguar is cool at screen location 10,10
```

Txt.DrawNum (*x as Integer, y as Integer, Number as Integer*)

Draw a single numeric value at the specified location, the value can be either direct entry or computed (from a constant, variable or array).

Code:

```
Txt.DrawNum (10,10,53) 'Draw the direct entry number 53 at the specified location
```

```
Dim Score As integer = 1 'Define a Variable to hold the score and preset it to 1
```

```
Txt.DrawNum (10,10,Score) 'Draw the computed value Score at screen location 10,10. Appears as 1
```

Txt.Draw2Nums (*x as Integer, y as Integer, Num1 as Integer, Num2 as Integer*)

Draw two comma separated numeric values at the specified location, the values can be either direct entry or computed (from a constant, variable or array).

Code:

```
Txt.Draw2Nums (10,10,53,64) 'Draw the direct entry numbers 53 and 64 at the specified location.  
'Appears as 53,64
```

```
Dim Score As integer = 2 'Define a Variable to hold the score and preset it to 2
```

```
Txt.Draw2Nums (10,10,Score,64) 'Draw the computed value Score and the direct entry value 64 at screen  
'location 10,10. Appears as 2,64
```

Txt.Draw3Nums (*x as Integer, x as Integer, Num1 as Integer, Num2 as Integer, Num3 as integer*)

Draw two comma separated numeric values at the specified location, the values can be either direct entry or computed (from a constant, variable or array).

Code:

```
Txt.Draw3Nums (10,10,53,64,180) 'Draw the numbers 53, 64 and 180 at the specified location.  
'appears as 53,64,180
```

```
Dim Score As integer = 3 'Define a Variable to hold the score and preset it to 3
```

```
Txt.Draw3Nums (10,10,Score,64,180) 'Draw the computed value Score and the direct entry values 64 and 180  
'at screen location 10,10. Appears as 3,64,180
```

Txt.SetFontColor (*FrontColor as Integer, BackColor as Integer*)

Set the colour of the font used for drawing Text strings

Code:

```
Txt.SetFontColour (60,120)
```

Txt.SetFontData (*AsciiIndex as Byte, t0 as Byte, t1 as Byte, to t7 as Byte*)

Updates the character data in the current Font for the given Character (with the AsciiIndex)

Code:

```
?
```

Txt.SetFontGradientV (*Color0 as Byte, Color1 as Byte to Color7 as Byte*)

Allow each of the eight scan lines that make up each font character to be drawn in a different colour. This allows the programmer to colourise the text very easily. A animated varying colour effect can be achieved by changing the colour palette of the 8 colours used.

Code:

```
Txt.SetFontGradient (15,25,45,65,85,105,125,145) 'Set font gradient colours, Color0 is the top line,  
'Colour7, all 8 colours must be included
```

```
Txt.DrawString (0,0,"Example of Txt.SetFontGradient") 'Write in Text in gradient colours
```

Transparent text commands:

With the exception that the text is draw as transparent these commands are identical to their non transparent equivalents above.

`Txt.DrawStringT (x as Integer, y as Integer, Text as String Literal)`

`Txt.DrawNumT (x as Integer, y as Integer, Number as Integer)`

`Txt.Draw2NumsT (x as Integer, y as Integer, Num1 as Integer, Num2 as Integer)`

`Txt.Draw3NumsT (x as Integer, x as Integer, Num1 as Integer, Num2 as Integer, Num3 as integer)`

Example Code:

Not sure if this is of use however I thought I would include the code I have written while testing the VRB text and drawing commands along with the resulting screen shots in case it helps anyone.

```
Sub DrawBoxes ()
'Draw some boxes, why is bottom right corner pixel missing?

ClearScreen () 'Clear the screen of existing text/graphics

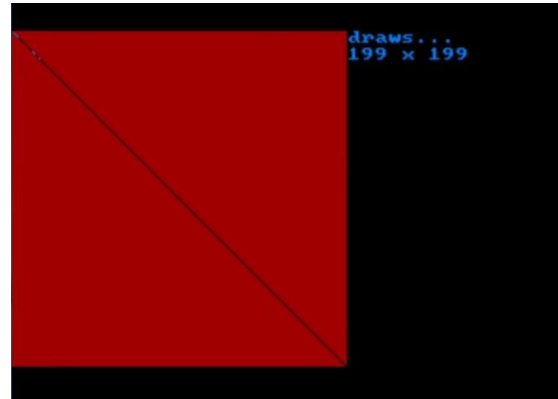
'Describe on screen what the subroutine is doing.
Txt.DrawString (0,0,"For Next Subroutine that draws...")
Txt.DrawString (0,10,"boxes in sizes of 1x1 to 199 x 199")

Wait (1500) 'Allow 1.5 seconds to read text

'Declare local variables
Dim x As Integer
Dim y As Integer
Dim Value As Integer

'Loop to draw the boxes - note how text is overwritten
For Value = 1 To 199
    x = Value
    y = Value
    DrawLine (0,0,x,0,20) 'Top Line
    DrawLine (0,y,x,y,20) 'Bottom Line
    DrawLine (0,0,y,20) 'Left side
    DrawLine (x,0,x,y,20) 'Right side
Next

Wait (1500) 'Allow 1.5 Seconds to view results
End Sub
```



```
Sub Colours ()
'Display vertical lines 50 pixels long in colours 0 to 255

ClearScreen () 'Clear the screen of existing text/graphics

'Describe on screen what the subroutine is doing.
Txt.DrawString (0,0,"Colour Draw Subroutine")
Txt.DrawString (0,10,"Draw a 50 pixel line in colours 0 to 255")

'Declare local variable
Dim Colour As Integer

'Loop drawing Coloured lines, Colour = X position and Colour
For Colour = 0 To 255
    DrawLine (Colour,50,Colour,100,Colour)
Next

Wait (2000) 'Wait 2.0 Seconds to view results
End Sub
```



```
Sub Text ()
'Display Text in different Foreground & Background colours

ClearScreen () 'Clear the screen of existing text/graphics

'Describe on screen what the subroutine is doing.
Txt.DrawString (0,0,"Colured Text Draw Subroutine")
Txt.DrawString (0,10,"Foreground & background colour equal to")
Txt.DrawString (0,20,"Line number x 10 and x 5 respectively")

Wait (1000) 'Wait 1.0 Seconds to read text

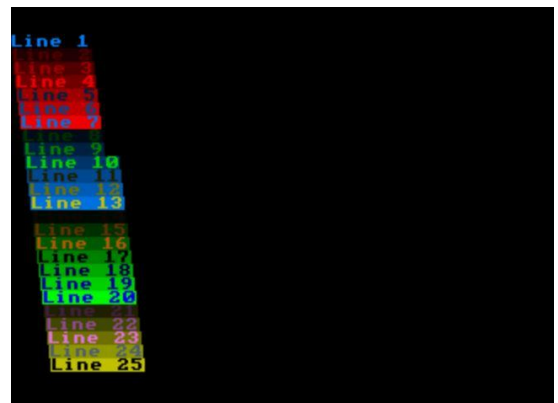
'Loop to Draw 25 lines of Text
ClearScreen () 'Clear the screen of existing text/graphics

'Declare local variable
Dim LineNum As Integer

For LineNum = 0 To 25
    If LineNum = 0 Then
        'Change first font colour from Black to readable Blue
        Txt.SetFontColor (63,0)
    Else
        'Set Font and background colours values for each line
        Txt.SetFontColor (LineNum * 10, LineNum * 5)
    End If

    Txt.DrawString (LineNum,LineNum * 8,"Line ")
    Txt.DrawNum (LineNum + 40,LineNum * 8,(LineNum + 1))
Next

Wait (2500) 'Allow 2.5 Seconds to view results
End Sub
```



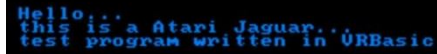
```
Sub Main ()

SetGraphicsMode () 'Sets up Jaguar Video mode.
ClearScreen () 'Clear screen of current text and Graphics
```

```
'Set colour of Font, Blue on Black
Txt.SetFontColor (63,0)

'Text Lines are font size separated (pixel 8 rows apart)
Txt.DrawString (10,80,"Hello...")
Txt.DrawString (10,88,"this is a Atari Jaguar...")
Txt.DrawString (10,96,"test program written in VRBasic")

Wait (1500) 'Wait 1.5 seconds to read text
```



```
Hello...
this is a Atari Jaguar...
test program written in VRBasic
```

```
'Text Lines are over font size separated (10 pixel rows apart)

ClearScreen () 'Clear screen of current text and Graphics

Txt.DrawString (45,80,"Basic Programming on...")
Txt.DrawString (45,90,"the Atari Jaguar.....")
Txt.DrawString (45,100,"is easy!")

Wait (1500) 'Wait 1.5 seconds to read text
```



```
Basic Programming on...
the Atari Jaguar.....
is easy!
```

```
' Draw a box around the text
DrawLine (40,75,230,75,63) 'Top Line
DrawLine (230,75,230,110,63) 'Right Line
DrawLine (230,110,40,110,63) 'Bottom Line
DrawLine (40,110,40,75,63) 'Left Line

' Draw a box around the Screen edge (320 X 200)
DrawLine (0,0,319,0,63) 'Top Line
DrawLine (319,0,319,199,63) 'Right Line
DrawLine (319,199,0,199,63) 'Bottom Line
DrawLine (0,199,0,0,63) 'Left Line

Wait (1500) 'Wait 1.5 seconds to read text
```



```
Basic Programming on...
the Atari Jaguar.....
is easy!
```

```
'Call Subroutines
'DO NOT put comments on these lines as it will cause errors
Colours ()
DrawBoxes ()
Text ()
```

```
'Set Gradient text colours
ClearScreen () 'Clear screen of current text and Graphics
Txt.SetFontGradientV (15,25,45,65,85,105,125,145)
'Write Text in gradient colours
Txt.DrawString (0,0,"Example of Txt.SetFontGradient")

Wait (1500) 'Wait 1.5 seconds to read text

'Draw a thick white bar across the screen the hard way
'Set colour at palette index 20 to white
SetPalette888 (20,255,255,255)

'Declare local variable
Dim Pixel As Integer = 0

'Loop to draw 3 white lines one pixel at a time
For Pixel = 0 To 319
    DrawPixel (Pixel,14,20)
    DrawPixel (Pixel,15,20)
    DrawPixel (Pixel,16,20)
Next

Wait (1000)
End Sub
```



```
Example of Txt.SetFontGradient
```