Beginner’s Guide

Copyright: © 2014, 2015, 2016 Anywhere Software Edition 1.2

Last update: 2016.10.26

Contributors:

Klaus Christl
Erel Uziel

All the source code and files needed (layouts, images etc.) of the example projects in this guide are included in the SourceCode folder.

Updated for B4i version 3.0.

1 Getting Started

B4i is a simple yet powerful development environment that targets Apple devices (iPhone, iPad etc.).
Basic4i language is similar to Visual Basic and B4A language.
Basic4i compiled applications are native iOS applications; there are no extra runtimes or dependencies.
Unlike other IDE's, B4i is 100% focused on iOS.
B4i includes a powerful GUI designer, built-in support for multiple screens and orientations.
You can develop and debug with a real device

iOS 7 and above are supported.

What you need:

- The B4i program, this is a Windows program running on a PC.
- The Java SDK on the PC, free
- An Apple developer license, cost 99$ per year.
- A device for testing.
- The Basi4i-Bridge program on the device, free
- A Mac builder to compile the program, this be either
    - A Mac computer with the Mac Builder program, on local wifi.
    - The hosted Mac Builder service over Internet, cost 26$ per year
- A Mac computer or a MacInCloud service to distribute the program

Links to tutorials in the forum:
Local Mac Builder Installation
Creating a certificate and provisioning profile
Installing B4i-Bridge and debugging first app

1.1 Installing B4i

1.1.1 Installing Java JDK

B4i depends on the free Java JDK component

If you are already using B4A you can skip this chapter.

Installation instructions:

The first step should be to install the **Java JDK**, as B4i requires it.
Note that there is no problem with having several versions of Java installed on the same computer.

- Open the Java 8 JDK download link.
- Check the Accept License Agreement radio button.
- Select **"Windows x86"** in the platforms list (for 64 bit machines as well).
You should install the regular JDK for 64-bit computers as well.

- Download the file and install it.

1.1.2 Installing B4i

Download and install the B4i file on your computer.

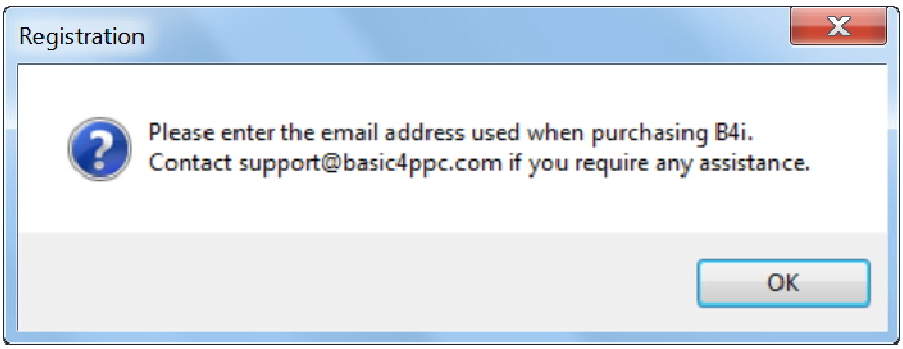
iOS compilation requires an Apple Mac computer. Developers have two options with B4i:

- Use a local Mac machine connected over the local network.
  For this you should also download the Mac builder and install it.

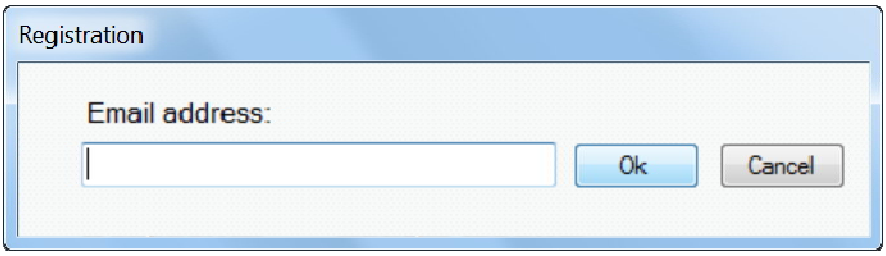
- Use our hosted builder rental service. Hosted Mac Builder installation.
  The builder service allows you to develop iOS applications without a Mac computer.
  All of the development steps can be done with the builder service except of the final step which is uploading the application to Apple App Store. This step requires a Mac or a service such as MacInCloud.
  Note that the builder is currently limited to projects of up to 15mb.

Copy the license file *b4i-license.txt* to the B4i folder and to a safe place on the computer for backup.
Note that this is not a text file, do not open it with a text editor.

When you first run B4i you will be asked to enter your e-mail address, the one you used when you purchased it B4i.
You find it also in the mail you received with the B4i file.



Enter the e-mail address.



You get a confirmation window that B4i is registered.

Contact support@basic4ppc.com if you require any assistance.

1.1.3 Mac Builder installation

iOS compilation requires an Apple Mac computer. Developers have two options with B4i:
- Use a local Mac machine connected over the local network.
- Use our hosted builder rental service.

Link to the tutorial in the forum: Local Mac Builder Installation.

These instructions explain how to install the builder on a local Mac machine.

1. Install Java JDK 8: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
2. Install Xcode 6.
3. Download and unzip the B4i-Builder.
4. Open a terminal and navigate to B4i-Builder folder.
5. Run it with: java -jar B4iBuildServer.jar
6. Set the builder IP address in the IDE under Tools - Build Server - Server Settings

Notes & Tips

- By default ports numbers 51041 (http) and 51042 (https) are used.
- The firewall should be either disabled or allow incoming connections on these two ports.
- You can test that the server is running by going to the following link: http://<server ip>:51041/test
- You can kill the server with: http://<server ip>:51041/kill
- It is recommended to set your Mac server ip address to a static address. This can be done in your router settings or in the Mac under Network settings.
- A single Mac builder can serve multiple developers as long as they are all connected to the same local network. Note that you are not allowed to host builders for developers outside of your organization.

Multiple IPs.

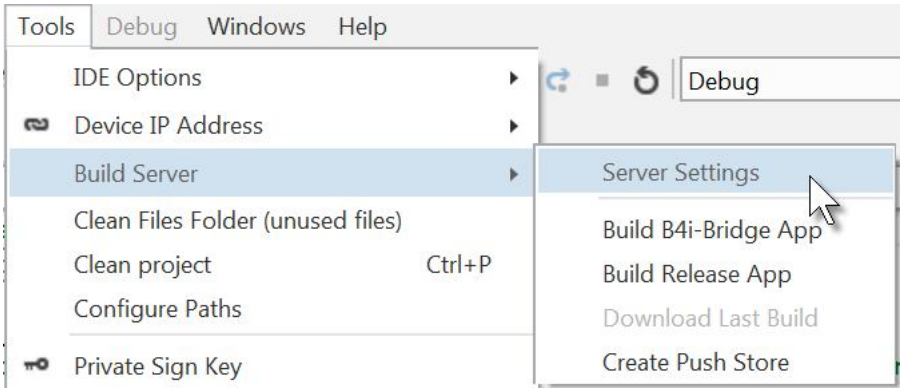
When the server is started it takes the first IP address reported by the OS and uses it as its own IP address. You can see this address in the server messages.
In most cases this is the correct address. However if it is not the correct IP address then the server will not be usable.
In that case you need to explicitly set the correct address:
- Open the key folder and delete all files.
- Edit key.txt and change it to:

1.1.4 Hosted Mac builder service (optional)

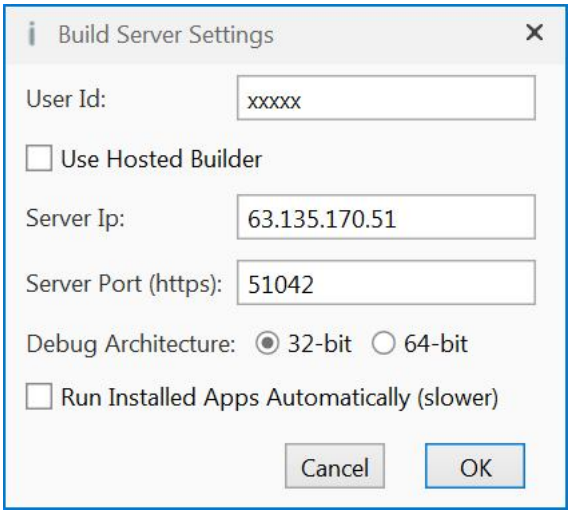
If you have bought the *hosted Mac builder service* you got a mail with your user ID.

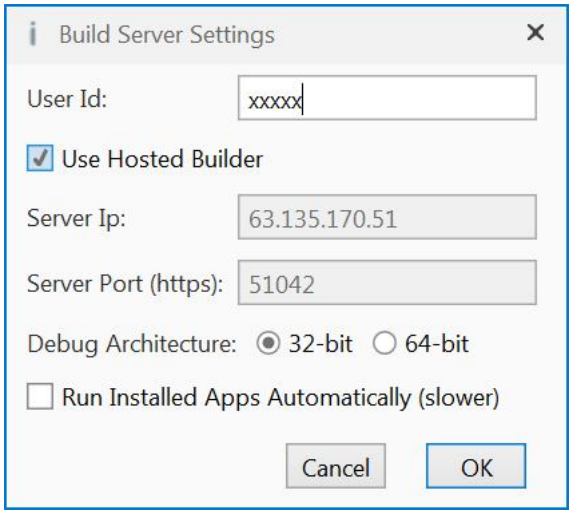
Link to the tutorial in the forum:

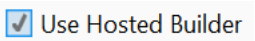
You must enter it in the IDE.



Enter the ID.



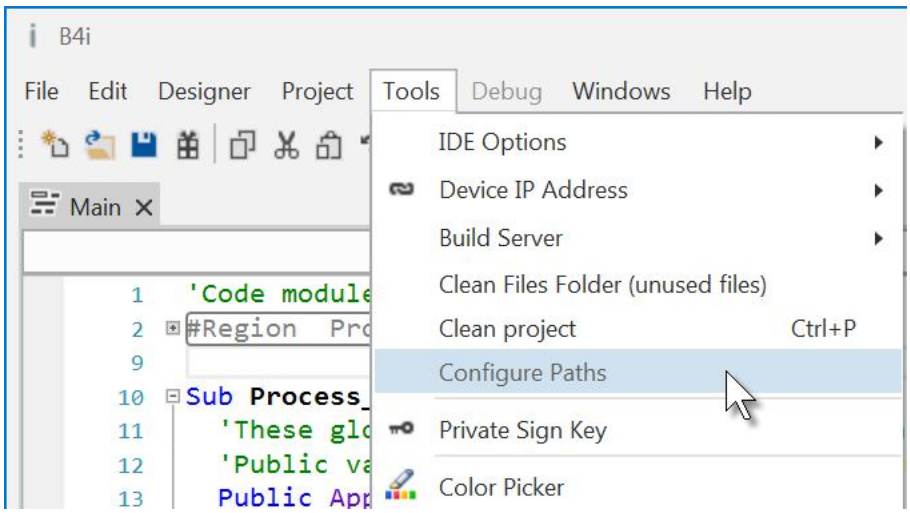


Don’t forget to check 

if you use the Hosted Builder Service!

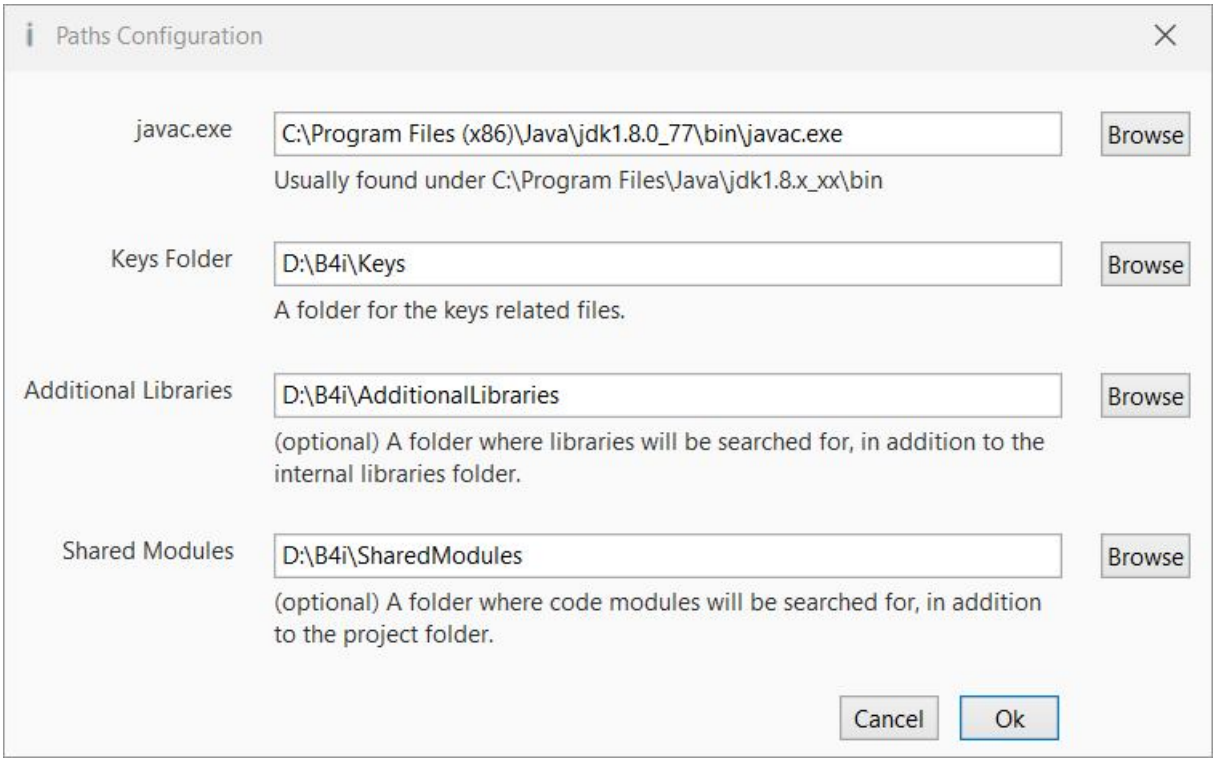
1.2 Configure Paths in the IDE

Then you need to configure the different paths in the IDE.



Run the IDE.

In the Tools menu click on Configure Paths.



javac.exe:
Enter the folder of the javac.exe file.

Keys folder:
Create a special folder for the Keys, for example C:\B4i\Keys.

Additional libraries:
Create s specific folder for additional libraries, for example C:\B4i\AdditionalLibraries.

Shared Modules:
Create s specific folder for shared modules, for example C:\B4i\SharedModules.

1.3 Creating a certificate and provisioning profile

Don't panic!
While this process can be a bit annoying it is not too complicated and you can always delete the keys and start from scratch (which is not always the case in Android).

Note that you must first register with Apple as an iOS developer (costs $99 per year).
The whole process is done on a Windows computer.

In order to install an app on an iOS device you need to create a certificate and a provisioning profile.

The certificate is used to sign the application. The provisioning profile, which is tied to a specific certificate, includes a list of devices that this app can be installed on.

The video shows the steps required for creating and downloading a certificate and provisioning profile.
There are two steps which are not shown in the video and are also required before you can create a provisioning profile:

- Create an App ID. This step is quite simple. Just make sure that you create a wildcard id.
- Add one or more devices. You will need to find the devices UDID for that.

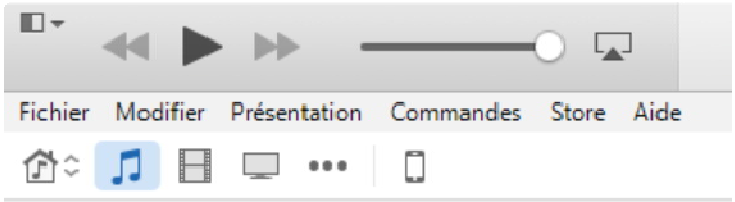
Link to the tutorial in the forum: Creating a certificate and provisioning profile.

1.3.1 UDID

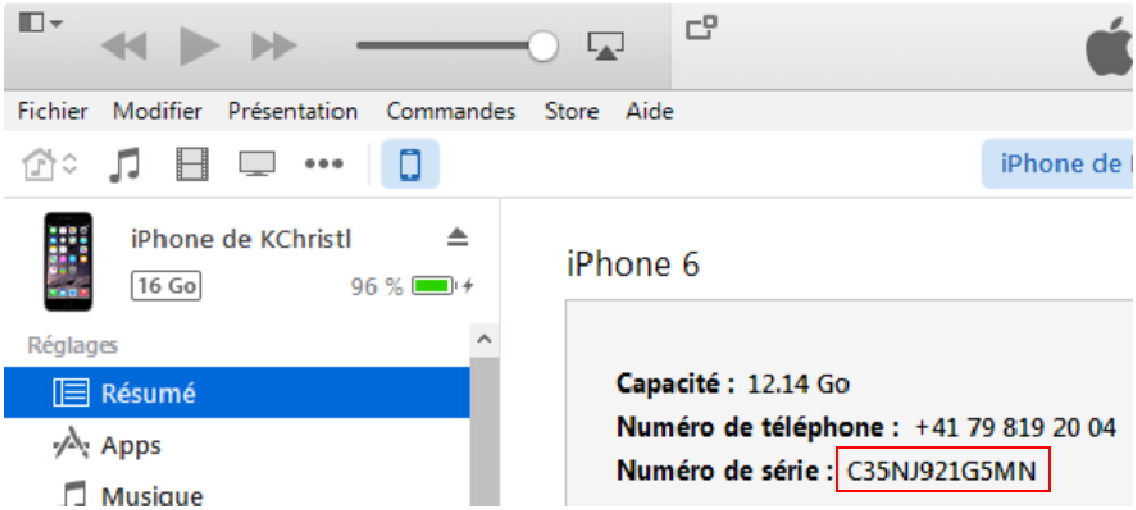
Devices are recognized by their UDIDs. There are two ways to get the device UDID:

1. If iTunes is installed then you can find it in iTunes.
   The first time, connect your device with the USB cable to the computer.

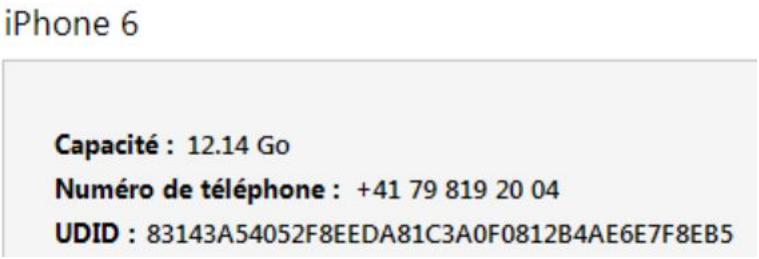
   Run iTunes, you should see on top this icon . It can take a while before you see it.



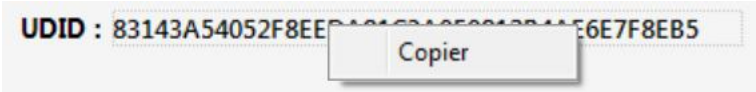
   Click on and you get this screen:



Now click on C35NJ921G5MN to get the UDID.



Right click on 83143A54052F8EEDA81C3A0F0812B4AE6E7F8EB5 to copy the UDID.



2. Use an online service such as this one: http://get.udid.io/

1.3.2 Certificate and Provisioning Profile

Main steps:

1. Set a new keys folder in the IDE.
2. Create a key by choosing Tools - Private Sign Key
3. Create and download the certificate as demonstrated in the video. You will need to upload the CSR file that was created in step 2.

   Note that you can choose either **iOS App Development** or **App Store and Ad Hoc** in the certificate page.
4. Create and download a provisioning profile.

1.4 Installing B4i-Bridge and debugging first app

B4i-Bridge is an application that you install on the device.
It has three purposes:

1. Launch the installation process when needed.
2. Run the installed app (when installation is not needed).
3. The bridge is also the WYSIWYG visual designer.

You need to install B4i-Bridge once. It is done from the device browser.

Link to the tutorial in the forum: Installing B4i-Bridge and debugging first app.

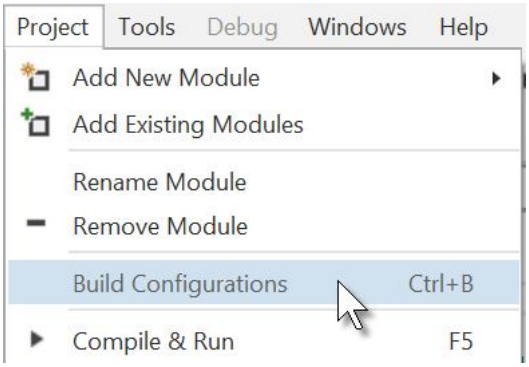
1.4.1 Install the B4I certificate

Open Safari (device browser) and navigate to: www.b4x.com/ca.pem
Follow the instructions.
You can see at any time the profile in Settings / General / Profile.

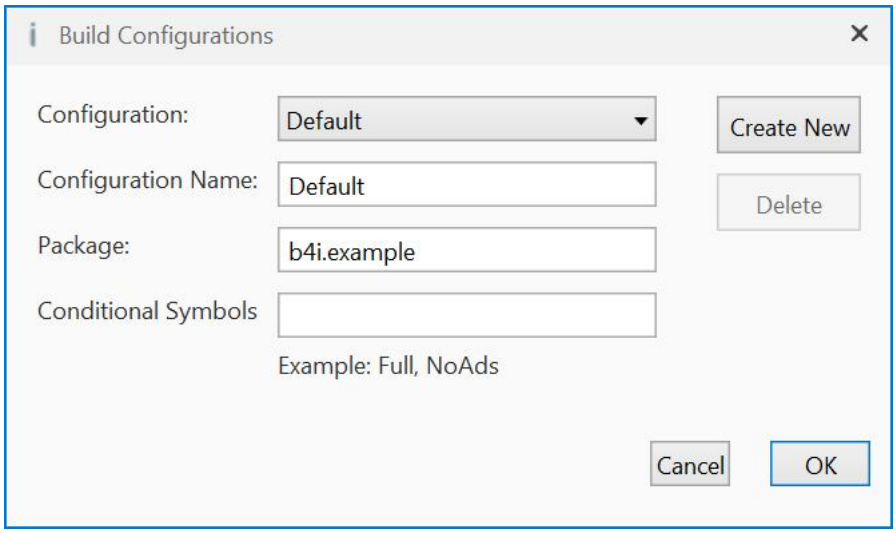
1.4.2 Set the package name based on the provision app id

Run B4i, load a project or use the default project and set the package name based on the provision app ID.

In the Project menu click on Build Configurations.

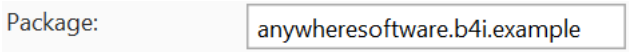


The window below is shown:



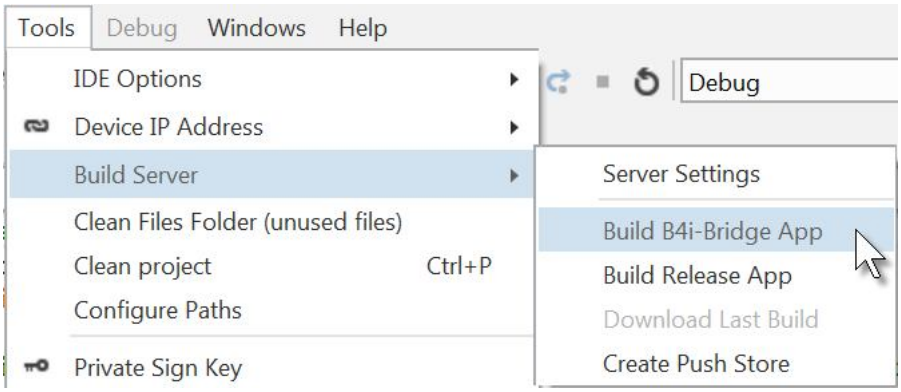
Change the Package name according to the provision app ID.

Example in my case:

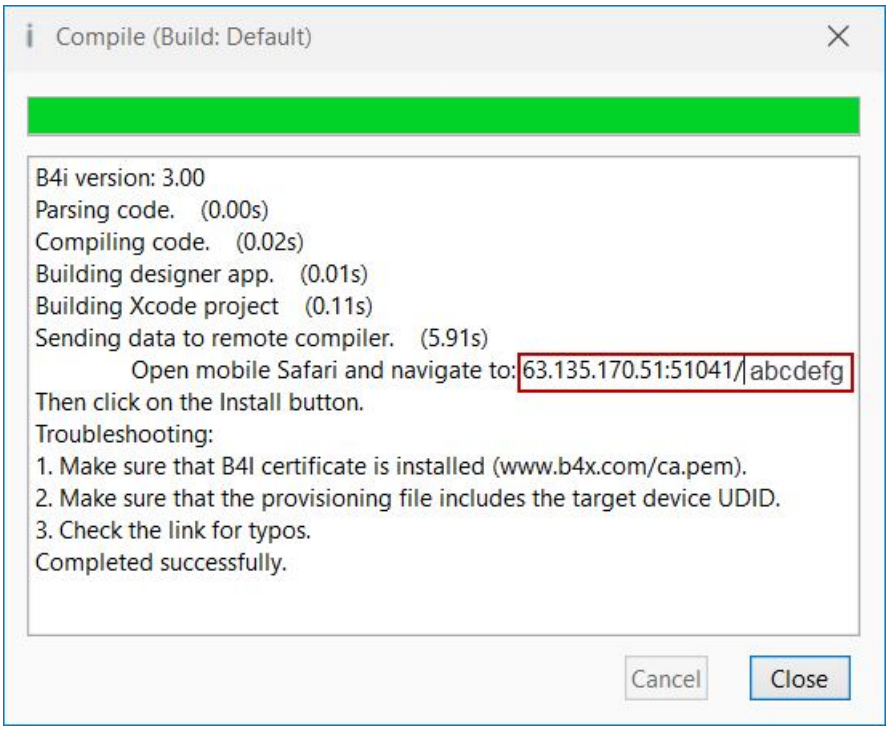


1.4.3 Install Build B4i-Bridge

In the Tools menu click on Build Server and click on Build B4i-Bridge App :



You get the compilation window.



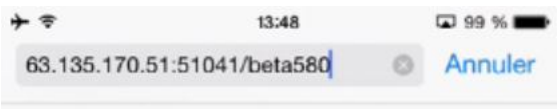
The code you see will be different!

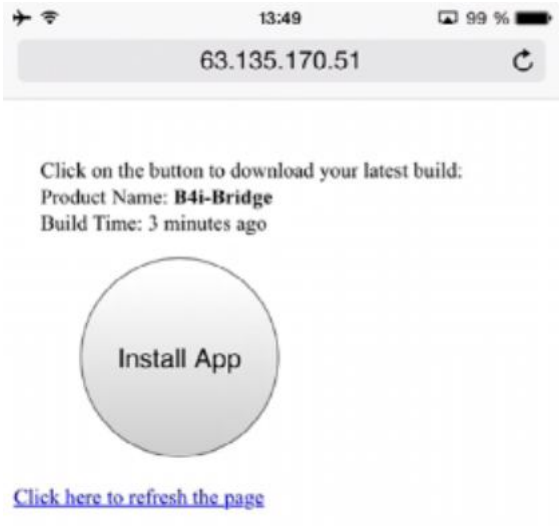
1.4.4 Load B4i-Bridge

Open Safari on the device 

.

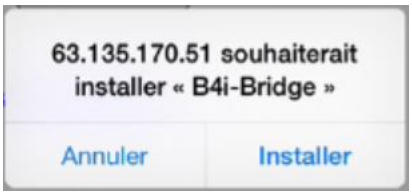
Enter the code in the search box on top.





This screen will appear.

Click on



Close Safari.

1.4.5 Install B4i-Bridge and run it

Click on this B4i-Bridge icon 

on the device, you will see the installing animation and finally the B4i-Bridge icon 

.

Tips:
- You don't need to wait for the installation animation to complete. Once the animation starts you can click on the app icon.
- If the installation is stuck in the "waiting" step for more than 10 or 15 seconds then uninstall it and install it again.
- B4i-Bridge must be in the foreground for it to be able to start an installation or to run the application. In most cases it will be in the foreground automatically. If it is not in the foreground then you need to click on it to bring it to the foreground.

2 My first program (MyFirstProgram.b4i)

Let us write our first program. The suggested program is a math trainer for kids.

The project is available in the SourceCode folder shipped with the Beginner's Guide:
SourceCode\MyFirstProgram\MyFirstProgram.b4i

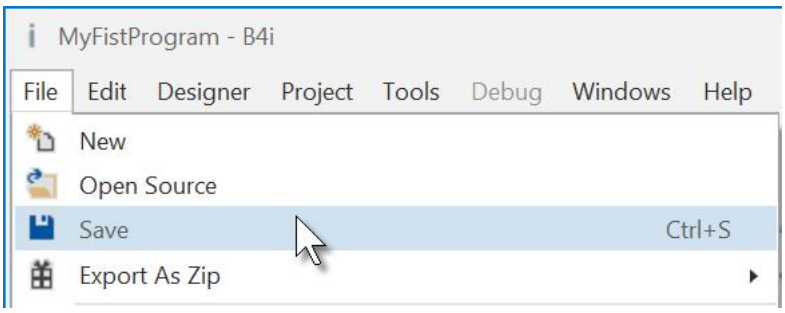
On the screen, we will have:
- 2 Labels displaying randomly generated numbers (between 1 and 9)
- 1 Label with the math sign (+)
- 1 TextField where the user must enter the result
- 1 Button, used to either confirm when the user has finished entering the result or generate a new calculation.
- 1 Label with a comment about the result.

In iOS:
- Label is an object to show text.
- TextField is an object allowing the user to enter text.
- Button is an object allowing user actions.

We will design the layout of the user interface with the Designer, the Abstract Designer and on a Device and go step by step through the whole process.
The Designer manages the different objects of the interface.
The AbstractDesigner shows the positions and sizes of the objects and allows moving or resizing them on the screen.
On the Device we see the real result.

Run the IDE 

Save the project.

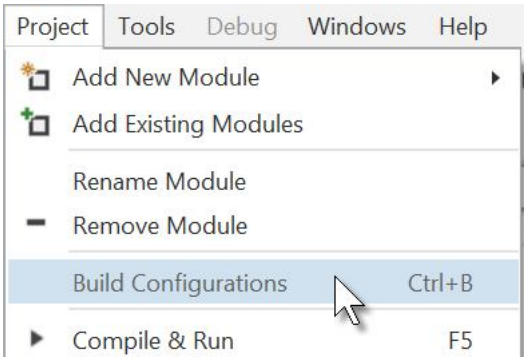


You must save the project before you can run the Designer.

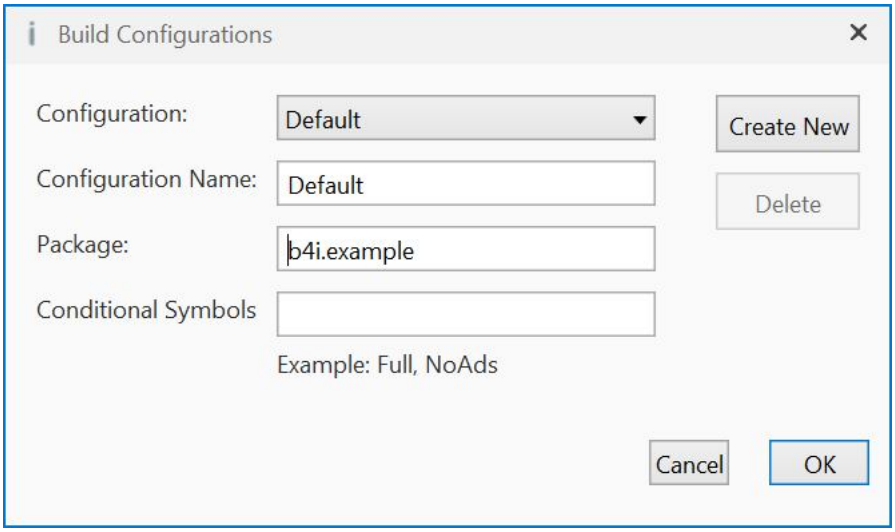
Create a new folder MyFirstProgram and save the project with the name MyFirstProgram.

Set the Package Name.
Each program needs a package name.

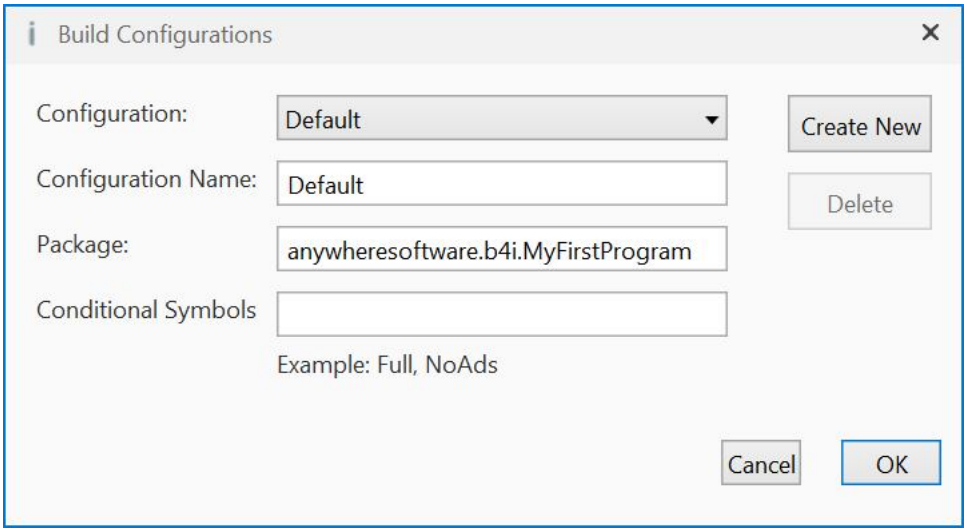
In the menu Project click on Build Configurations.



This window appears:



The default name is b4i.example. We will change it to anywheresoftware.b4i.MyFirstProgram.



Set the Application Label.

The Application label is the name of the program that will be shown on the device.

On top of the code screen you see the 'region' Project Attributes.

Regions are code parts which can be collapsed

```
1  'Code module
2  #Region  Project Attributes
9
```

or extended at the right.
Clicking on ⊞ will expand the Region.
Clicking on ⊟ will collapse the Region.
Regions are explained in Collapse a Region.

```
1  'Code module
2  #Region  Project Attributes
3    #ApplicationLabel: B4i Exa
4    #Version: 1.0.0
5    'Orientation possible valu
6    #iPhoneOrientations: Portr
7    #iPadOrientations: Portrai
8  #End Region
```

```
#Region  Project Attributes
      #ApplicationLabel: B4i Example
      #Version: 1.0.0
      'Orientation possible values: Portrait, LandscapeLeft, LandscapeRight and
PortraitUpsideDown
      #iPhoneOrientations: Portrait, LandscapeLeft, LandscapeRight
      #iPadOrientations: Portrait, LandscapeLeft, LandscapeRight, PortraitUpsideDown
#End Region
```

The default name is `B4i Example`, but we will change it to `MyFirstProgram` for naming consistency.

Change this line:

```
#ApplicationLabel: B4i Example
```

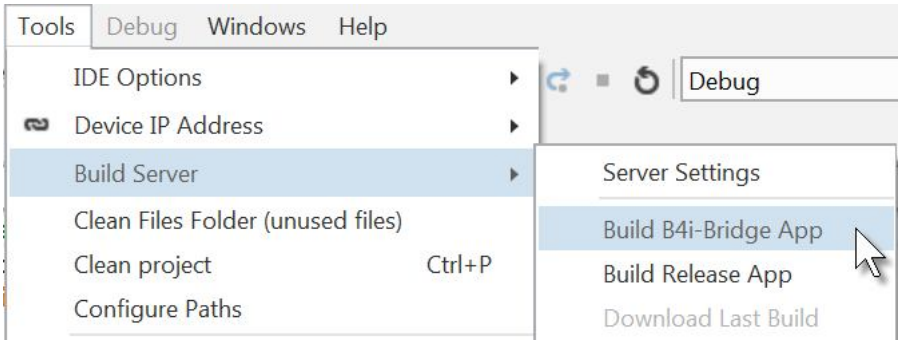
to

```
#ApplicationLabel: MyFirstProgram
```

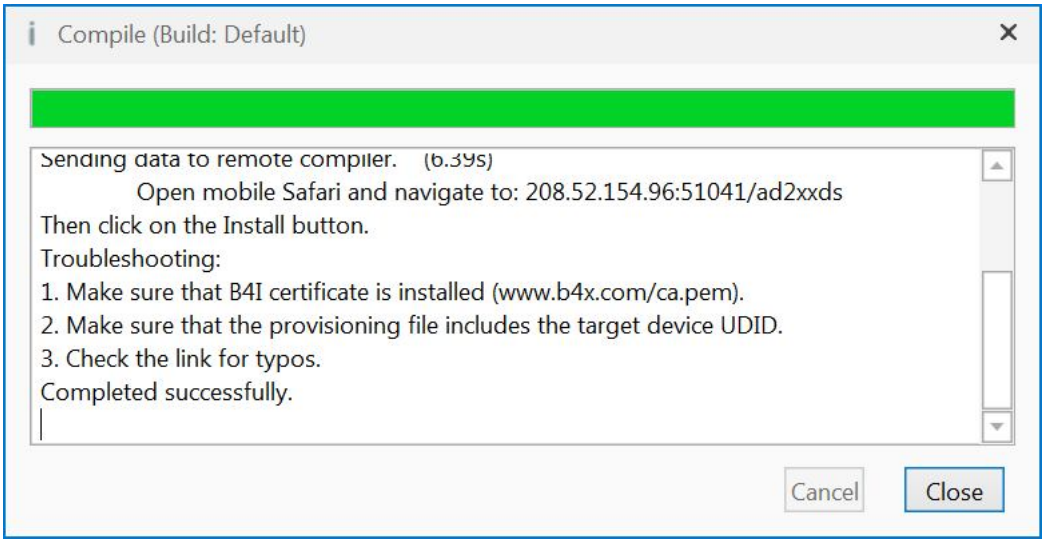
The other lines are explained in Code header Project Attributes / Activity Attributes.

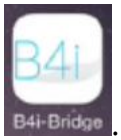
In the IDE run Build B4i-Bridge App.

IDE menu Tools / Build Server / Build B4i-Bridge App



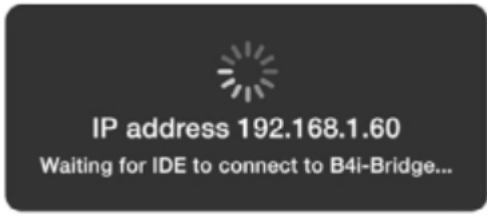
You get this screen.



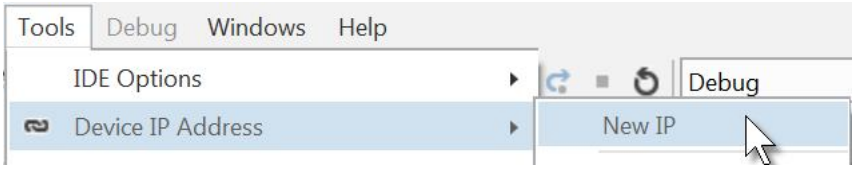
On the device run B4i-Bridge 

.

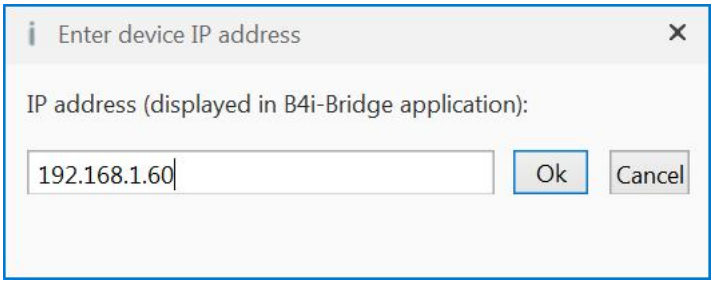
On the screen you see the IP address of the device.



In the IDE click on Tools / Device IP Address / New IP



Enter the IP address:

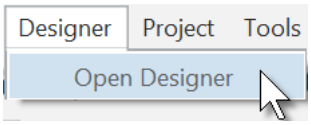


Click on 

.

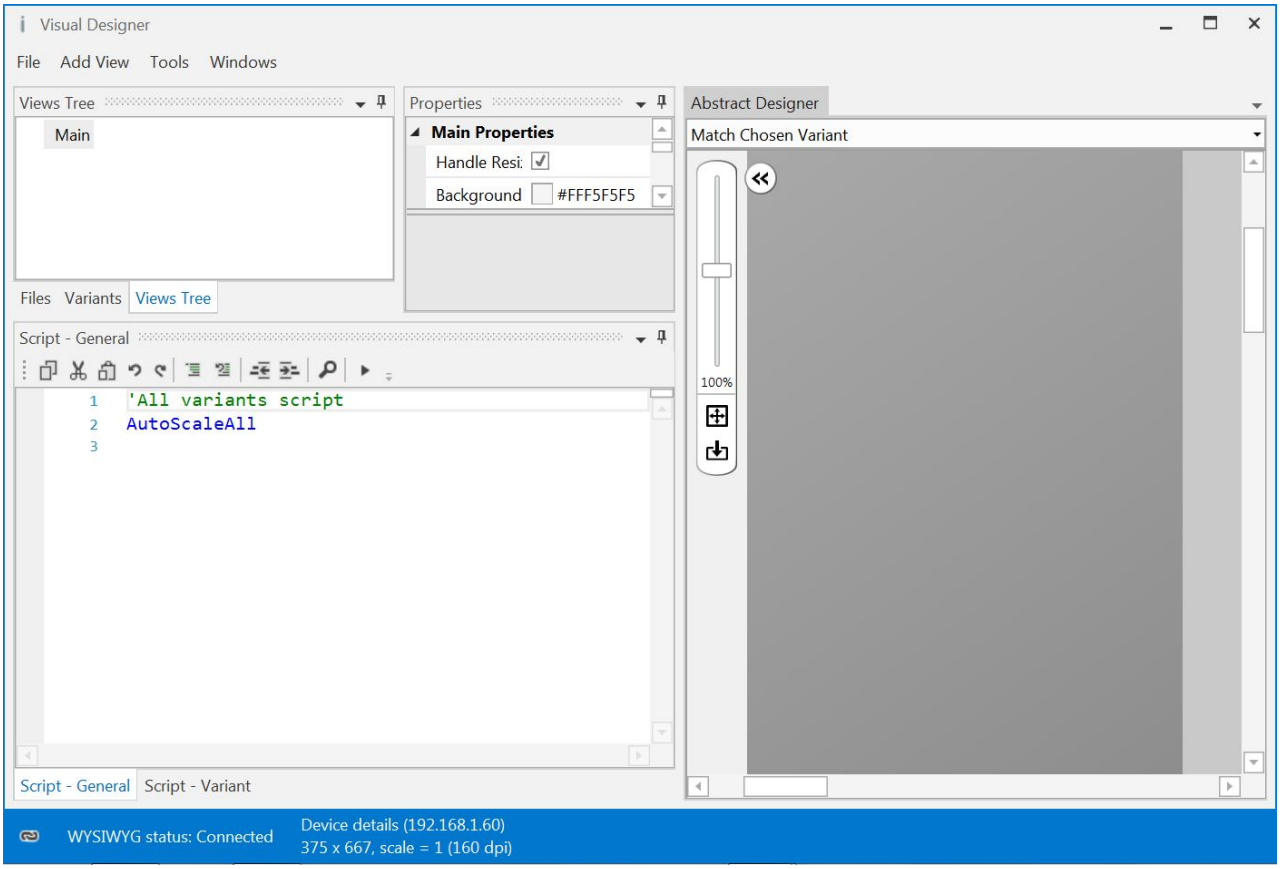
You will see this screen on the device (only the upper part is shown).

In the IDE open the Designer.



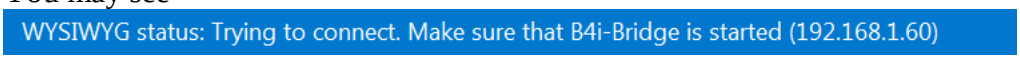
Wait until the Designer is ready.

The Designer looks like this.



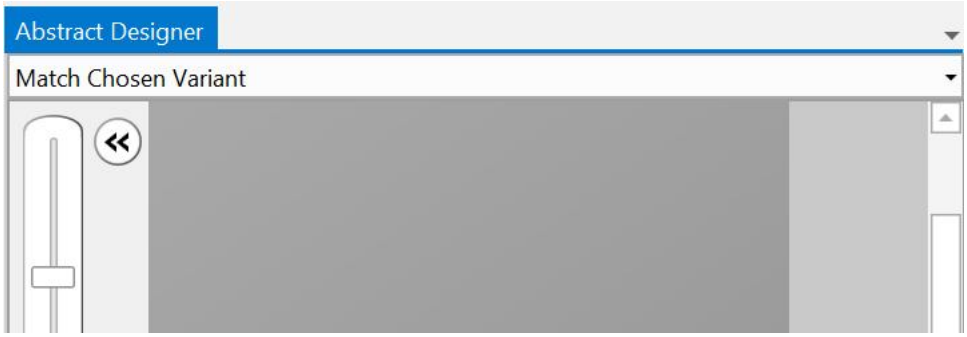
Note that in the bottom left of the Designer window you see the connection status:

You may see



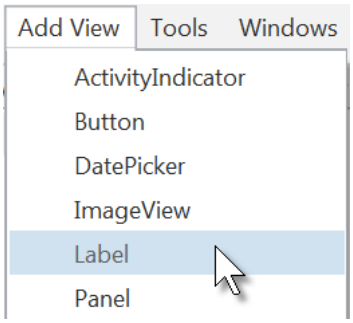
if the device is not connected.

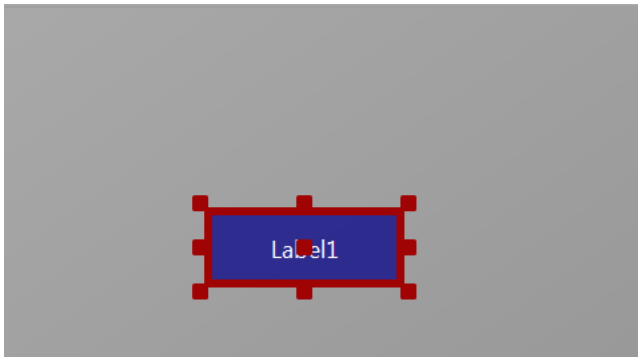
With the Designer we have also the Abstract Designer which shows the layout not exactly WYSIWYG but the positions and size of the different objects.
Only the top of the image is shown.



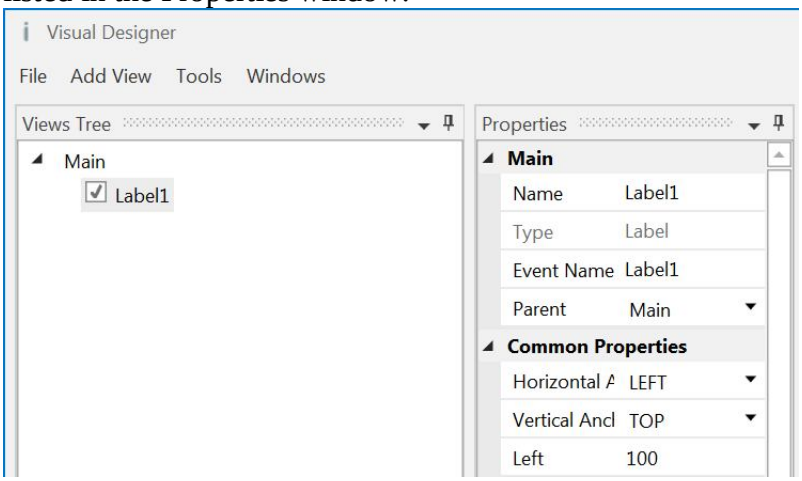
The dark gray area represents the screen area of the connected device.

Now we will add the 2 Labels for the numbers.
In the Designer, add a Label.





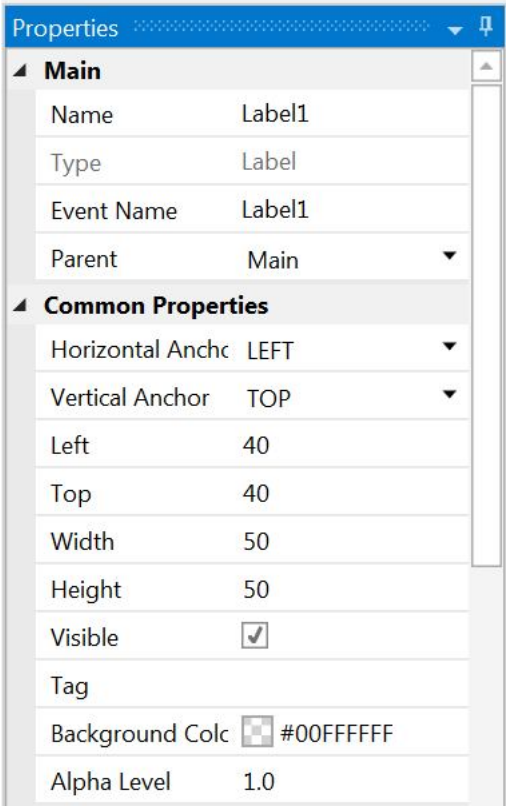
The label appears in the Abstract Designer, in the Views Tree window and its default properties are listed in the Properties window.



Resize and move the Label with the red squares like this.

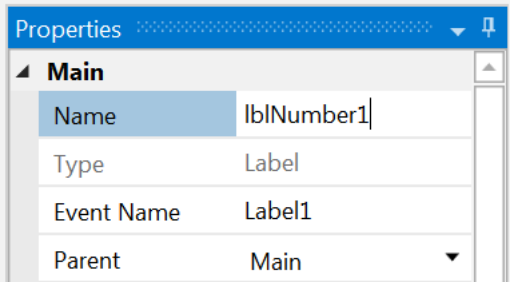
You can follow the layout on the device.

At the moment we see only Lab…
The background color is by default transparent.
Lab… stays for Label1



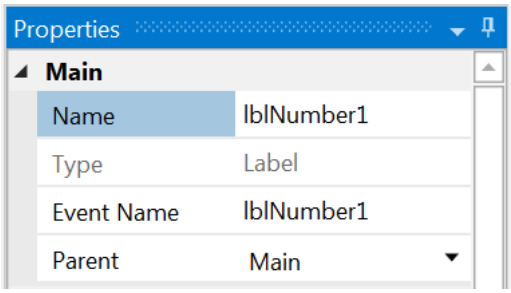
The new properties Left, Top, Width and Height are directly updated in the Properties window.
You can also modify the Left, Top, Width and Height properties directly in the Properties window.

Let us change the properties of this first Label according to our requirements.
By default, the name is Label with a number, here Label1, let us change its name to lblNumber1.
The three letters 'lbl' at the beginning mean 'Label', and 'Number1' means the first number.
It is recommended to use significant names for views so we know directly what kind of view it is and its purpose.



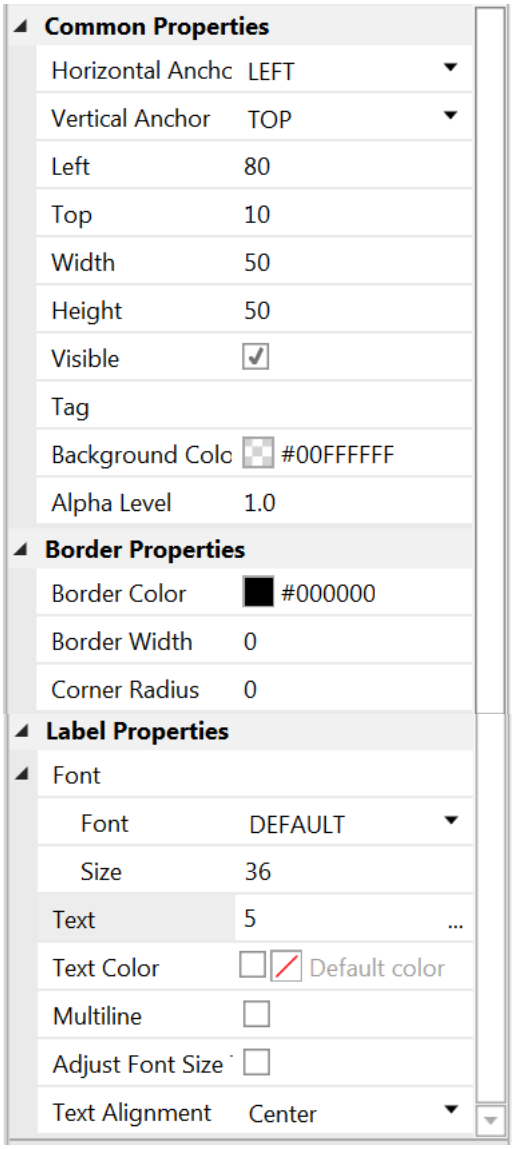
Pressing the 'Return' key or clicking elsewhere will also change the Event Name property.

We have now:



Main : main module.
Name : name of the view.
Type : type of the view. In this case, Label, which is not editable.
Event Name : generic name of the routines that handle the events of the Label.
Parent : parent view the Label belongs to.

Let us check and change the other properties:



Set Left, Top, Width and Height to the values in the picture.

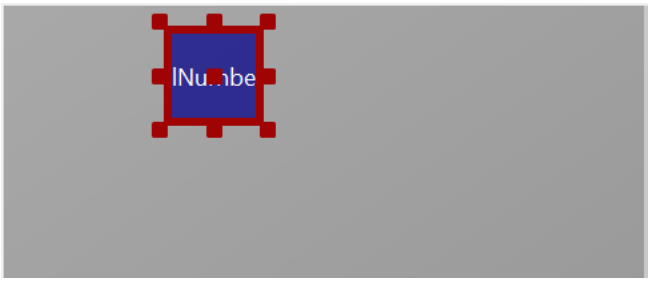
Visible is checked.

We leave the default colors.

We leave the default Font.
Text Size, we set it to 36.

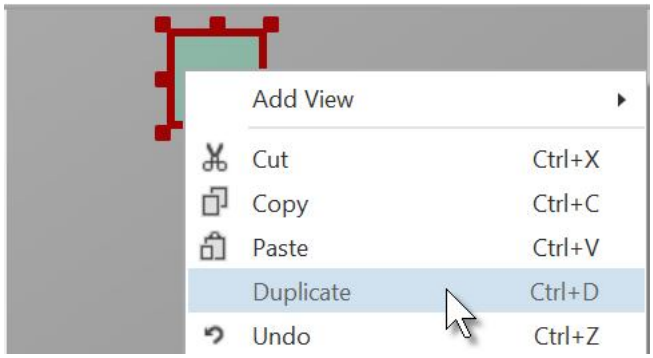
Text set to 5

Set Text Alignment to Center.



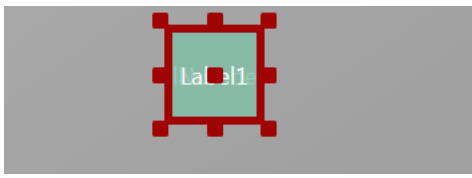
And the result in the Abstract Designer and on the device.

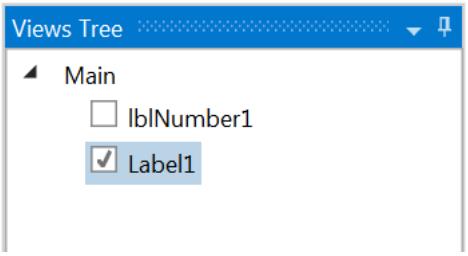
We need a second Label similar to the first one. Instead of adding a new one, we copy the first one with the same properties. Only the Name and Left properties will change.



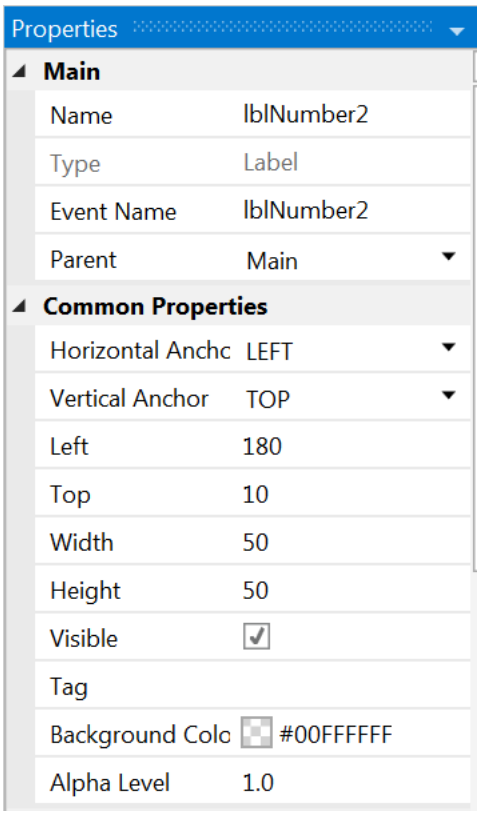
Right click on lblNumber1 and click on Duplicate in the popup menu.

The new label covers the previous one.





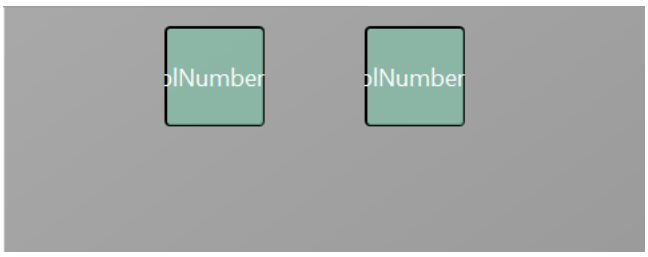
In the left part, in the Views Tree, you see the different views. The new label Label1 is added.



Let us position the new Label and change its name to lblNumber2.

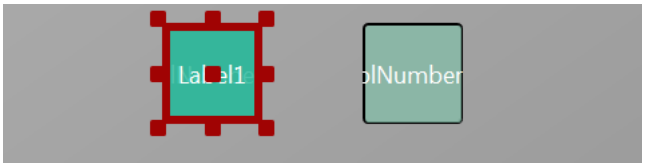
Change the name to lblNumber2.

The Left property to 180.



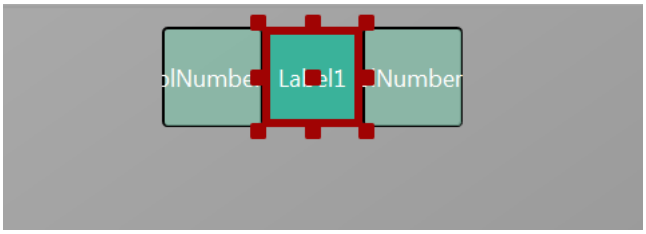
And the result in the Abstract Designer and on the device.

Let us now add a 3rd Label for the math sign. We copy once again lblNumber1.
Right click on lblNumber1 in the Abstract Designer and click Duplicate on in the popup menu.

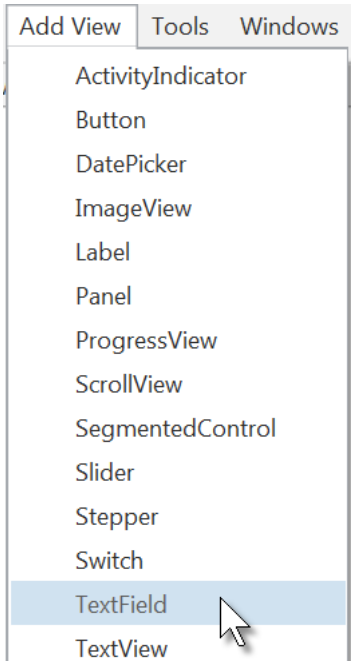


The new label covers lblNumber1.

Position it between the first two Labels and change its name to lblMathSign, its Text property to '+'.

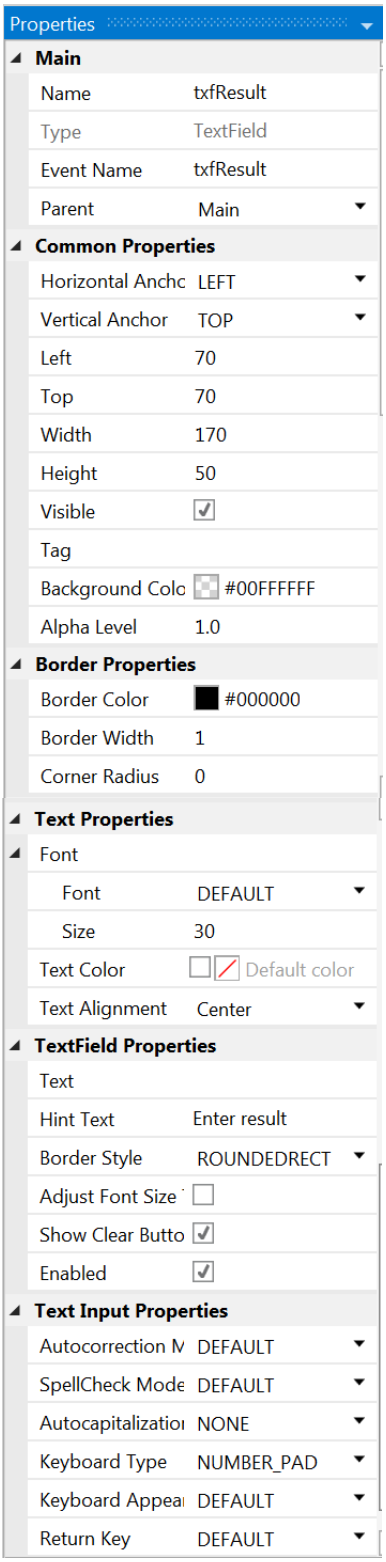


And the result in the Abstract Designer and on the device.



Now let us add a TextField view.
In the Designer Add View menu click on TextField.

Position it below the three Labels and change its name to txfResult.
'txf' means TextField and 'Result' for its purpose.



Change these properties.

Name to txfResult

Left, Top, Width and Height.

Border Width to 1

Text Size to 30

Text Alignment to Center

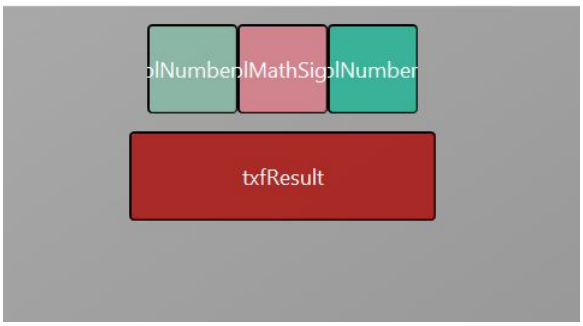
Hint Text to Enter result

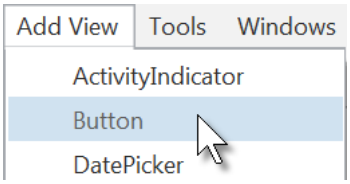
Hint Text represents the text shown in the TextField view if no text is entered.

Keyboard Type to NUMBER_PAD

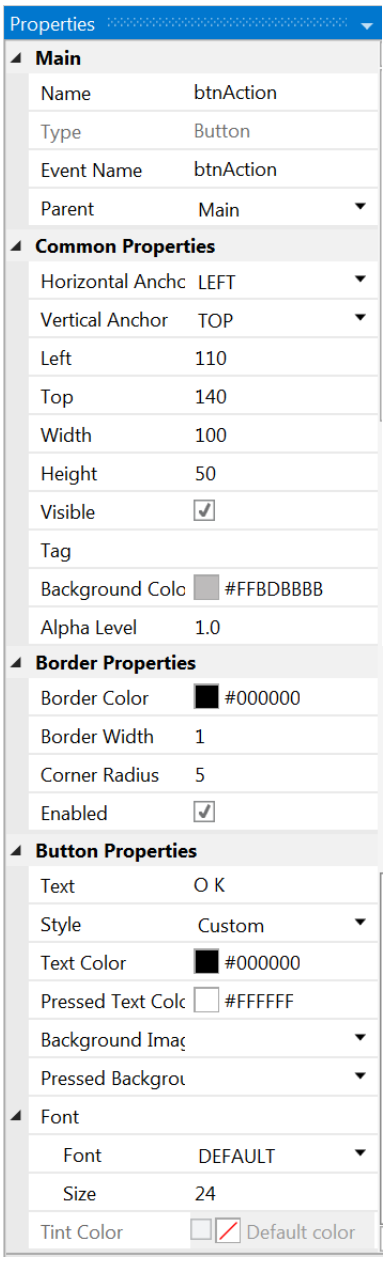
Setting Input Type to NUMBER_PAD lets the user enter only numbers.

After making these changes, you should see something like this.





Now, let's add the Button which, when pressed, will either check the result the user supplied as an answer, or will generate a new math problem, depending on the user's input.



Position it below the TextField view. Resize it and change following properties:

Name to btnAction

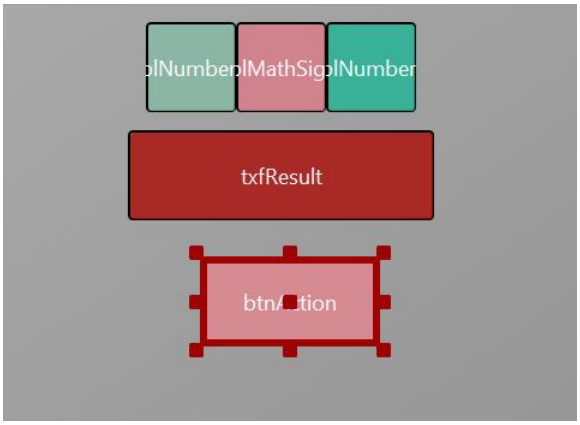
Left, Top, Width and Height.

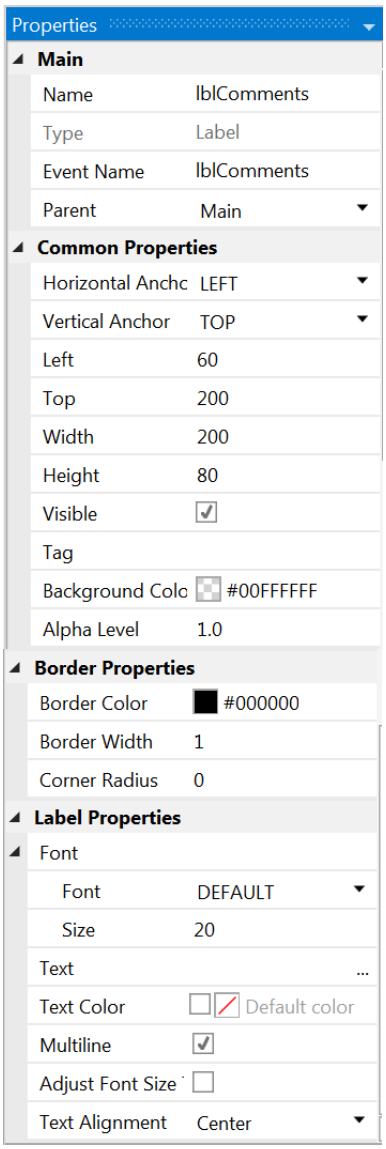
Background Color to #FFBDBBBB

Border Width to 1

Text to O K (with a space between O and K)

Text Size to 24





Let us add the last Label for the comments. Position it below the Button and resize it.

Change the following properties:
Name to lblComments

Left, Top, Width and Height.

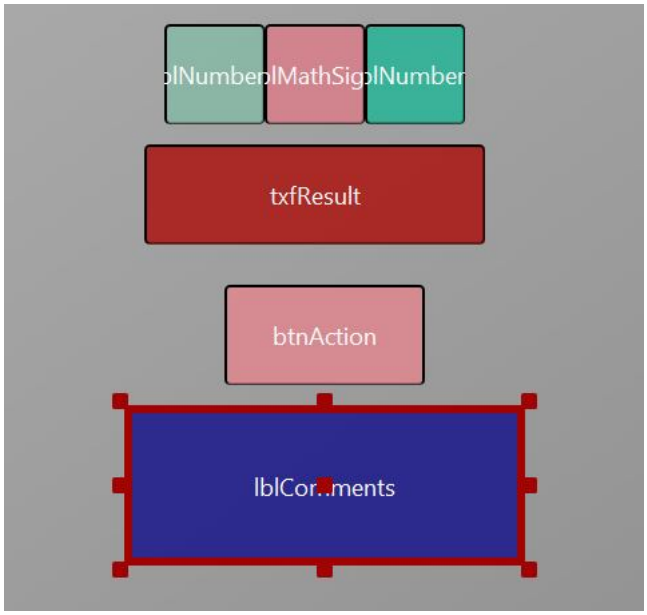
Border Width to 1

Text Size to 20

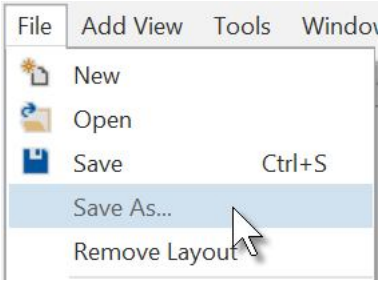
Multiline to True (checked)

Text Alignment to Center

And the result.



Now we save the layout in a file.

Click on 

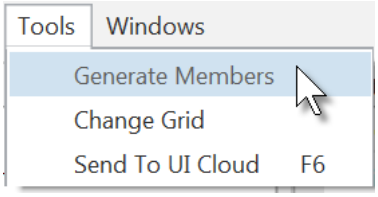
and save it with the name 'Main'.

Click on 

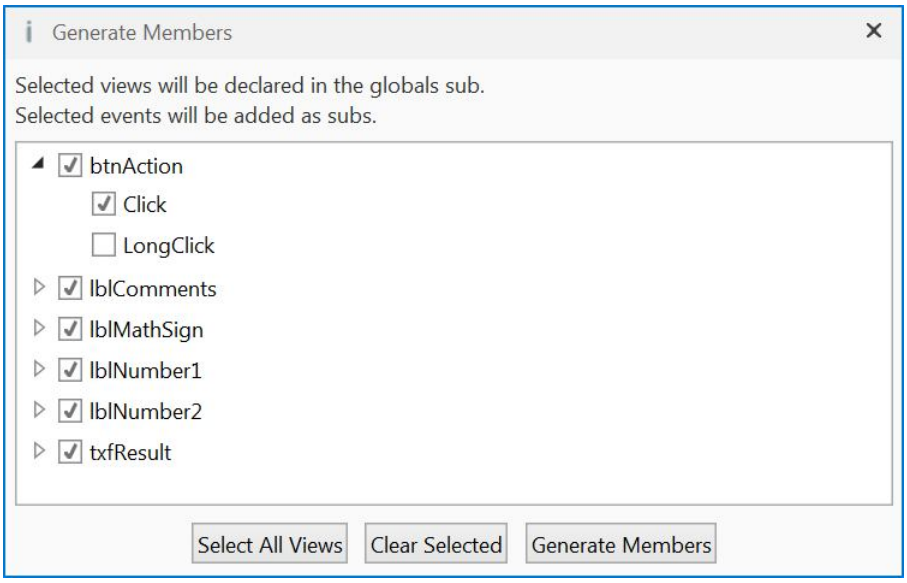
.

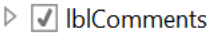
To write the routines for the project, we need to reference the Views in the code.
This can be done with the *Generate Members* tool in the Designer.

The *Generate Members* tool automatically generates references and subroutine frames.

Click on 

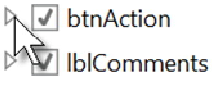
to open the generator.

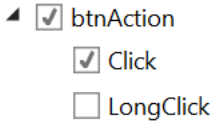


Here we find all the views added to the current layout.
We check all views and check the Click event for the btnAction Button.
Checking a view 

generates its reference in the Globals Sub routine in the code.
This is needed to make the view recognized by the system and allow the autocomplete function.

```
Private btnAction As Button
Private lblComments As Label
Private lblMathSign As Label
Private lblNumber1 As Label
Private lblNumber2 As Label
Private txfResult As TextField
```

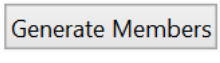
Clicking on 

shows all events for the selected view 

.
Clicking on an event of a view 

generates the Sub frame for this event.

```
Sub btnAction_Click

End Sub
```

Click on 

to generate the references and Sub frames, then close the window 

.

Now we go back to the IDE to enter the code.

On the top of the program code we have:

```
Sub Process_Globals
   'These global variables will be declared once when the application starts.
   'Public variables can be accessed from all modules.
   Public App As Application
   Public NavControl As NavigationController
   Private Page1 As Page

   Private btnAction As Button
   Private lblComments As Label
   Private lblMathSign As Label
   Private lblNumber1 As Label
   Private lblNumber2 As Label
   Private txfResult As TextField
End Sub
```

These lines are automatically in the project code.

```
   Public App As Application
   Public NavControl As NavigationController
   Private Page1 As Page
```

iOS needs an Application, a NavigationControl and at least one Page, the details are explained in the chapter Process life cycle.

Below the code above we have the `Application_Start` routine which is the first routine called when the program starts.
The content below is also added automatically in each new project.

```
Private Sub Application_Start (Nav As NavigationController)
   NavControl = Nav
   Page1.Initialize("Page1")
   Page1.Title = "Page 1"
   Page1.RootPanel.Color = Colors.White
   NavControl.ShowPage(Page1)
End Sub
```

`NavControl = Nav` > Sets `NavControl` as the `NavigationController`
`Page1.Initialize("Page1")` > Initializes `Page1`, `"Page1"` is the generic EventName of Page1.
`Page1.Title = "Page 1"` > Sets the Page Title
`Page1.RootPanel.Color = Colors.White` > Sets the background color to white.
`NavControl.ShowPage(Page1)` > Shows Page1 on the device.

First, we need our program to load the layout file we defined in the previous pages.
The file must be loaded onto the RootPanel of Page1, we load it just before
`NavControl.ShowPage(Page1)`
We take advantage of the autocomplete and in-line help features of B4i.

Enter P in a new line 29.

```
28  Page1.RootPanel.Color = Colors.White
29  P
30  Application_Start                    (Page1)
31  cPI
32  CreateMap
33  DipToCurrent                         ize(Width As Int,
34  GetType
35  LastException
36  LoadBitmap
37  Loop                                 on_Background
38  Page1
39
40                                       Page1 As Page
```

A drop-down list appears with all available keywords, views, routines etc.

The first item beginning with the letter 'P' is highlighted.

Click on Return to validate.

```
28  Page1.RootPanel.Color = Colors.White
29  Page1
30  NavControl.ShowPage(Page1)
```

P is completed to Page1.
Now enter a dot "."

```
28  Page1.RootPanel.Color = Colors.White
29  Page1.
30  HideBackButton
31  Initialize                           Page1)
32  IsInitialized
33  Prompt
34  ResignFocus                          ze(Width As Int,
35  RootPanel
36  TabBarItem
37  Tag
38  Title                                _Background
39
40
```

The drop-down list contains all properties of a Page view.

With the Down key go down to RootPanel.

```
28  Page1.RootPanel.Color = Colors.White
29  Page1.
30  HideBackButton
31  Initialize                           Page1)
32  IsInitialized
33  Prompt
34  ResignFocus                          ze(Width As Int, Height As Int)
35  RootPanel
36  TabBarItem                           RootPanel As Panel [read only]
37  Tag                                  Gets a reference to the main panel that holds the other views.
38  Title
39
40
```

We see RootPanel highlighted, and besides the list the in-line help with the syntax for the property and an explanation.

```
28    Page1.RootPanel.Color = Colors.White
29    Page1.RootPanel
30    NavControl.ShowPage(Page1)
```

Click on Return to validate.

Enter a dot “.”

```
28    Page1.RootPanel.Color = Colors.White
29    Page1.RootPanel.
30    NavControl.Sh
31  End Sub
32
33  Private Sub Pag
34
35  End Sub
36
37  Private Sub App
38
39  End Sub
40
```

ACTION_DOWN
ACTION_MOVE
ACTION_UP
AddView
Alpha
BringToFront
Color
GetAllViewsRecursive
GetView

Again we get a drop-down list with the properties of a Panel.

With the Down key go down to LoadLayout.

Again we see the syntax and the explanation.

```
28    Page1.RootPanel.Color = Colors.White
29    Page1.RootPanel.
30    NavControl.Sh
31  End Sub
32
33  Private Sub Pag                                  , Height As Int)
34
35  End Sub
36
37  Private Sub App
38
39  End Sub
40
```

Height
Initialize
IsFocused
IsInitialized
Left
LoadLayout
NumberOfViews
RemoveAllViews
RemoveViewAt

LoadLayout (LayoutFile As String) As LayoutValues
Loads a layout file to the panel.

```
28    Page1.RootPanel.Color = Colors.White
29    Page1.RootPanel.LoadLayout
30    NavControl.ShowPage(Page1)
```

Click on Return to validate.

Enter “(”.

```
28    Page1.RootPanel.Color = Colors.White
29    Page1.RootPanel.LoadLayout(
30    NavControl.ShowPage(Page1)
31  End Sub
32
```

LoadLayout (**LayoutFile** As String) As LayoutValues
Loads a layout file to the panel.

The in-line help shows what to do and the explanation.

```
28    Page1.RootPanel.Color = Colors.White
29    Page1.RootPanel.LoadLayout("Main")
30    NavControl.ShowPage(Page1)
```

Complete the line with the layout file name.
The file extension is not needed.
The file name `"Main"` is between quotes because it is a String.

The yellow line in the left border shows that a modification was made in the code.
As soon as you save the code the yellow line will be changed to a green line.

We want to generate a new problem as soon as the program starts. Therefore, we add a call to the New subroutine in Application_Start.

```
Private Sub Application_Start (Nav As NavigationController)
   NavControl = Nav
   Page1.Initialize("Page1")
   Page1.Title = "Page 1"
   Page1.RootPanel.Color = Colors.White
   NavControl.ShowPage(Page1)

   New
End Sub
```

New is displayed in red because the ‘New’ routine has not yet been defined.
Generating a new problem means generating two new random values between 1 and 9 (inclusive) for Number1 and Number2, then showing the values using the lblNumber1 and lblNumber2 ‘Text’ properties.

To do this we enter following code:
In Sub Process_Globals we add two variables for the two numbers.

```
   Private Number1, Number2 As Int
End Sub
```

And the 'New' Subroutine:

```
Private Sub New
   Number1 = Rnd(1, 10)                ' Generates a random number between 1 and 9
   Number2 = Rnd(1, 10)                ' Generates a random number between 1 and 9
   lblNumber1.Text = Number1           ' Displays Number1 in label lblNumber1
   lblNumber2.Text = Number2           ' Displays Number2 in label lblNumber2
   lblComments.Text = "Enter the result" & CRLF & "and click on OK"
   txfResult.Text = ""                 ' Sets edtResult.Text to empty
End Sub
```

The following line of code generates a random number from '1' (inclusive) to '10' (exclusive) :
`Rnd(1, 10)`

In this line `Number1 = Rnd(1, 10)  ' Generates a random number between 1 and 9`
The text after the quote, `' Generates...`, is considered as a comment.
It is good practice to add comments explaining the purpose of the code.

The following line displays the comment in the lblComment view:
`lblComments.Text = "Enter the result" & CRLF & "and click on OK"`
`CRLF` is the LineFeed character.

Now we add the code for the Button click event.

We have two cases:
- When the Button text is equal to "O K", it means that a new problem is displayed, and the program is waiting for the user to enter a result and press the Button.
- When the Button text is equal to "NEW", it means that the user has entered a correct answer and when the user clicks on the Button a new problem will be generated.

```
Private Sub btnAction_Click
   If btnAction.Text = "O K" Then
      If txfResult.Text="" Then
         Msgbox("No result entered","E R R O R")
      Else
         CheckResult
      End If
   Else
      New
      btnAction.Text = "O K"
   End If
End Sub
```

`If btnAction.Text = "O K" Then` checks if the Button text equals "O K"
If yes then we check if the TextField is empty.
If yes, we display a MessageBox telling the user that there is no result in the TextField view.
If no, we check if the result is correct or if it is wrong.
If no then we generate a new problem, set the Button text to "O K" and clear the TextField view.

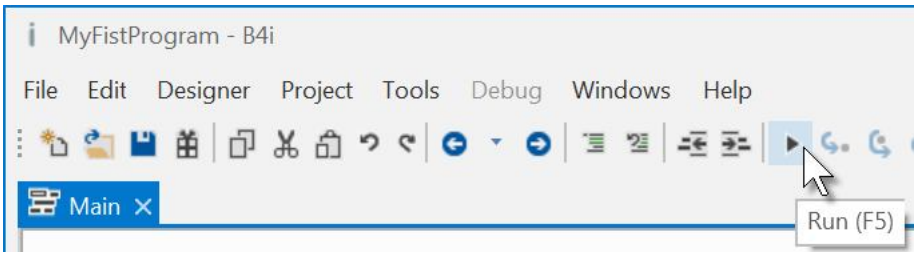
The last routine checks the result.

```
Private Sub CheckResult
   If txfResult.Text = Number1 + Number2 Then
      lblComments.Text = "G O O D  result" & CRLF & "Click on NEW"
      btnAction.Text = "N E W"
   Else
      lblComments.Text = "W R O N G  result" & CRLF & "Enter a new result" & CRLF & "and click OK"
   End If
End Sub
```

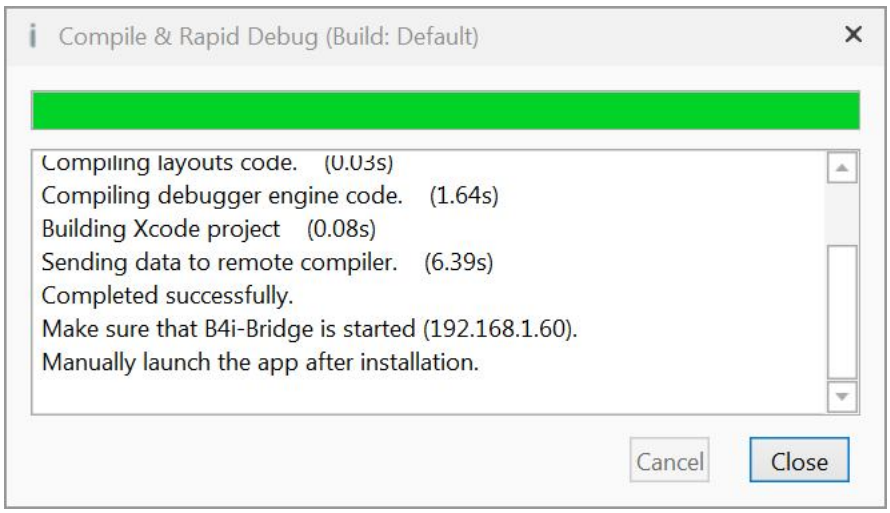
With `If txfResult.Text = Number1 + Number2 Then` we check if the entered result is correct.

If yes, we display in the lblComments label the text below:
'G O O D result'
'Click on NEW'
and we change the Button text to "N E W ".
If no, we display in the lblComments label the text below:
W R O N G result
Enter a new result
and click OK

Let us now compile the program and transfer it to the Device.
In the IDE on top click on  :

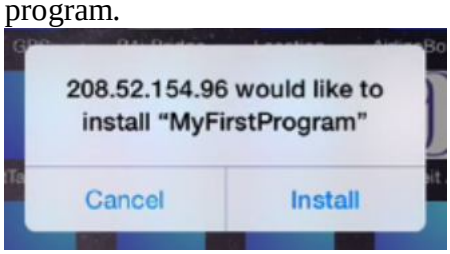


The program is going to be compiled.



When you see `'Completed successfully.'` as in the message box, the compiling and transfer is finished.

Looking at the device, you should see something similar to the image below when you first run the program.



Touch on Install.
Then you will see somewhere on the device the icon of the program  , touch it to run the program.

Then you should see something similar to the image on the left, with different numbers.

Of course, we could make aesthetic improvements in the layout, but this was not the main issue for the first program.

Touch on 

to activate the keyboard

Enter 14

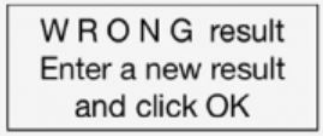
You will see this screen.

Click on 

to confirm the result entry.

If the result is correct you will see the screen on the left.

If the result is wrong the message is:

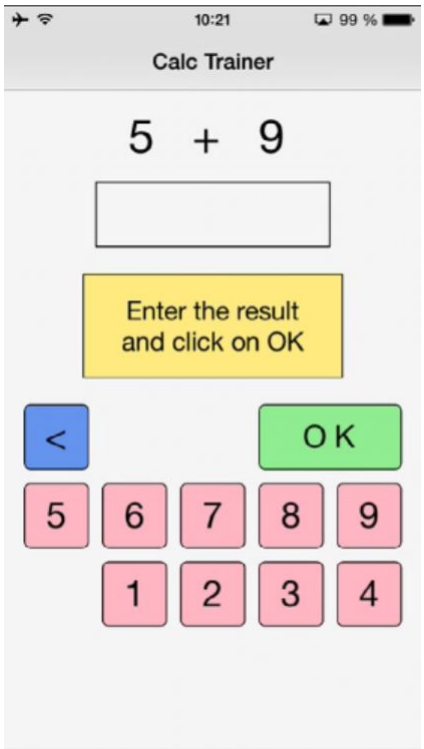


3 Second program

The project is available in the SourceCode folder: SourceCode\SecondProgram\SecondProgram.b4i.

Improvements to “My first program”.
- Independent numeric keyboard to avoid the use of the virtual keyboard.
- Colors in the comment label.

Create a new folder called “SecondProgram”. Copy all the files and folders from MyFirstProgram to the new SecondProgram folder and rename the program file MyFirstProgram.b4i to SecondProgram.b4i and MyFirstProgram.meta to SecondProgram.meta.

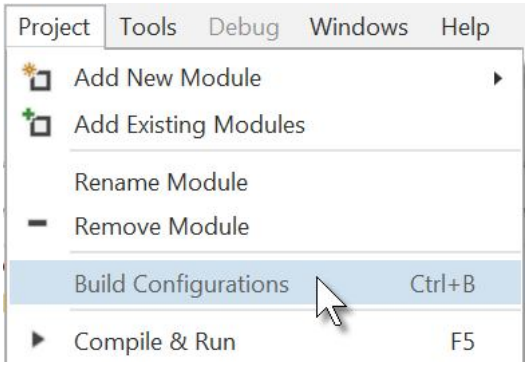


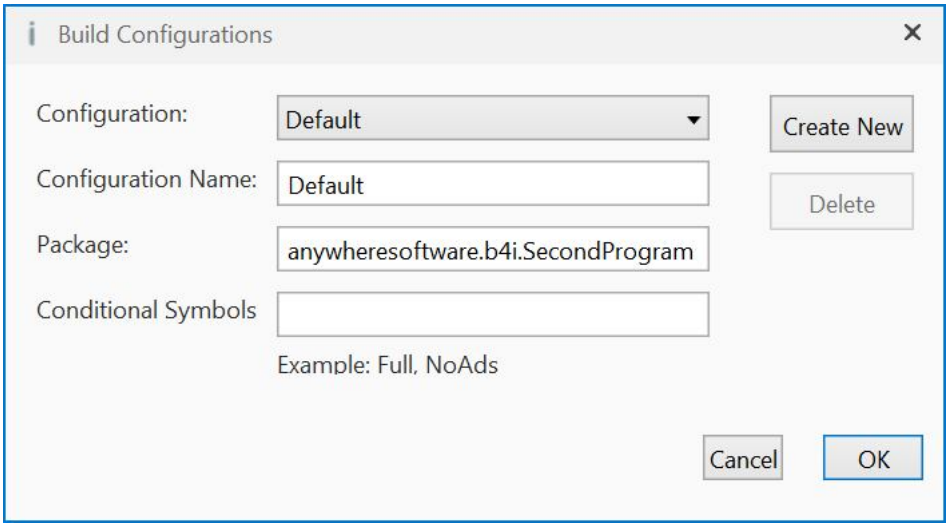
Load this new program in the IDE.

Run the Designer.

We need to change the Package Name.

In the IDE Project menu.
Click on Build Configurations.



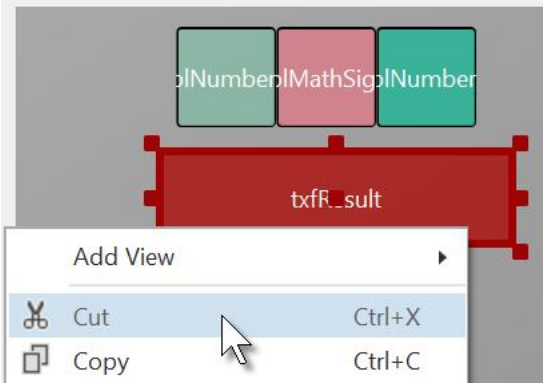


Change the Package name to anywheresoftware.b4i.SecondProgram and click on OK.

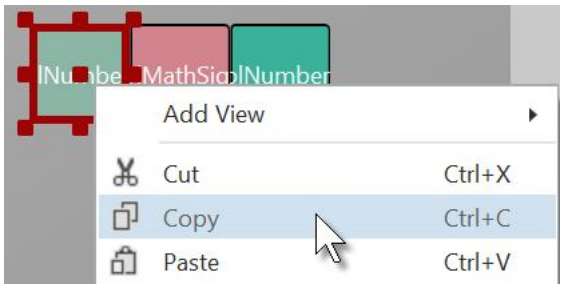
Then we must change the ApplicationLabel on the very top of the code.

```
#Region  Project Attributes
   #ApplicationLabel: SecondProgram
```

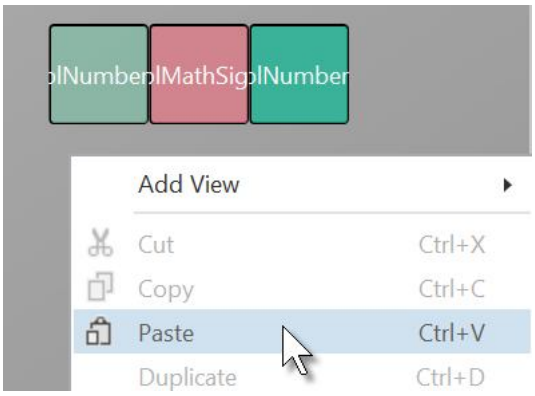
We want to replace the txfResult TextField view by a new Label.
In the Abstract Designer, click on the txfResult view.



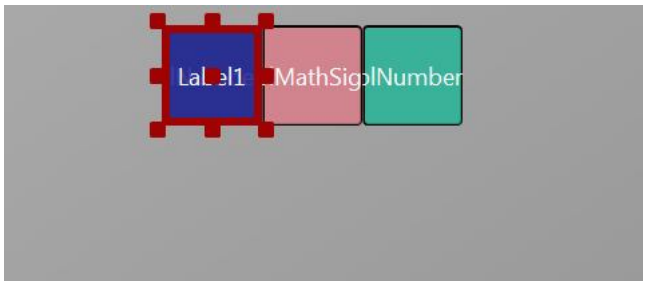
Right click on txfResult and click on Cut.



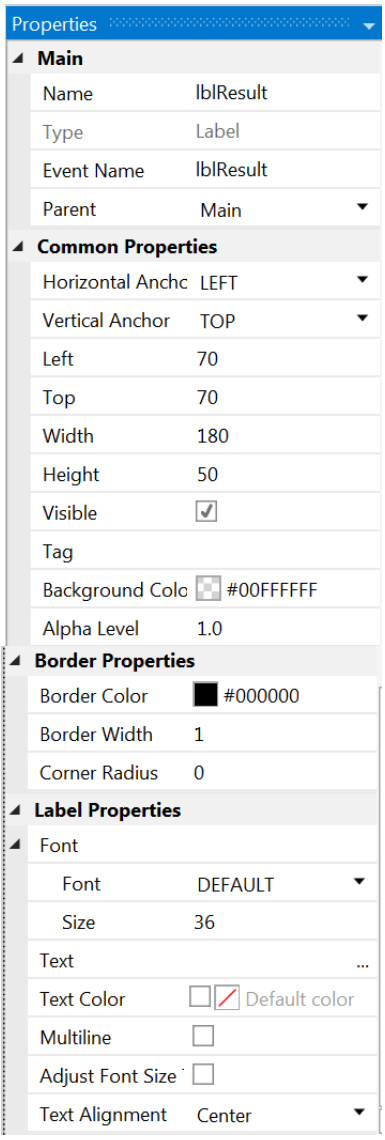
Right click on lblNumber1 and click on Copy.



Right click somewhere else and click on Paste.



The new label covers lblNumber1.



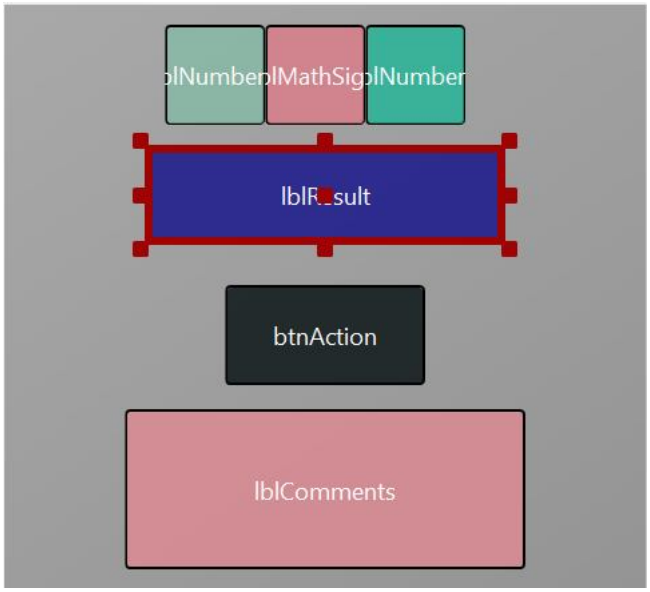
Modify the following properties:

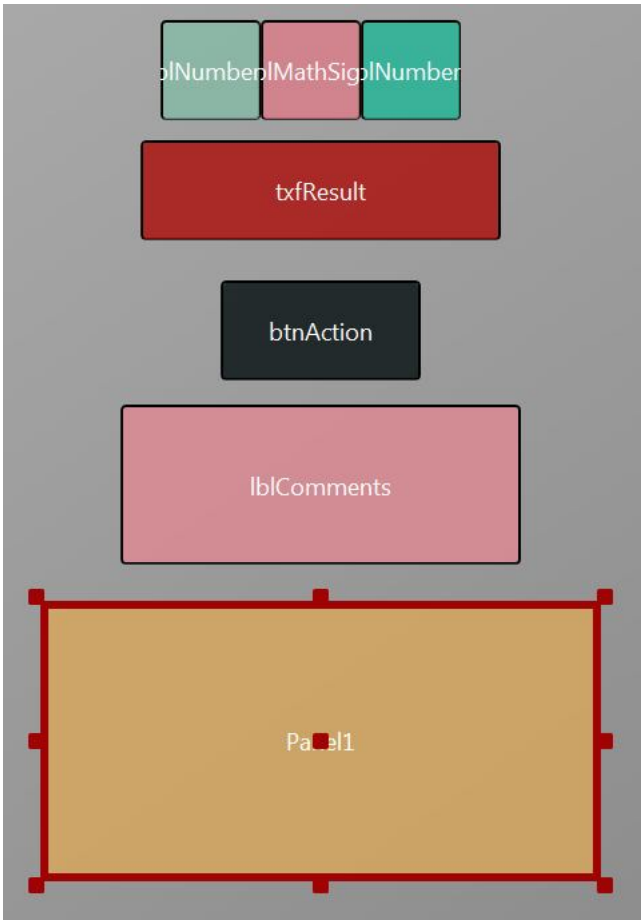
Name to lblResult

Left, Top, Width, Height

Boarder Width to 1

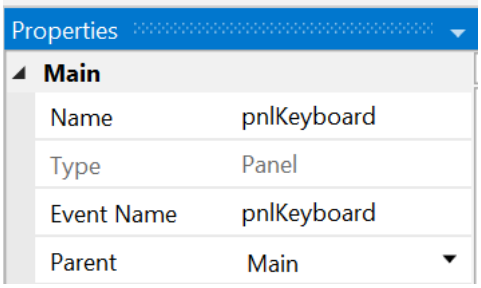
Text to "" no character



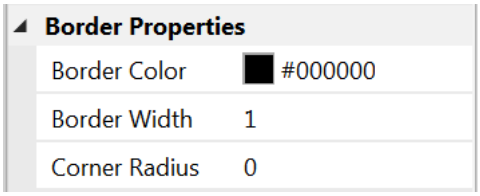


Let us add a Panel for the keyboard buttons.

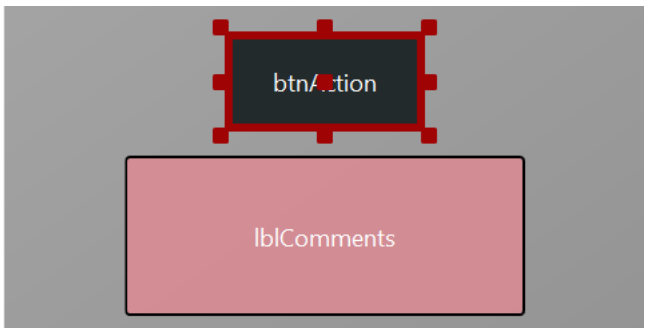
Position and resize it as in the image.



Change its Name to pnlKeyboard
"pnl" for Panel, the view type.

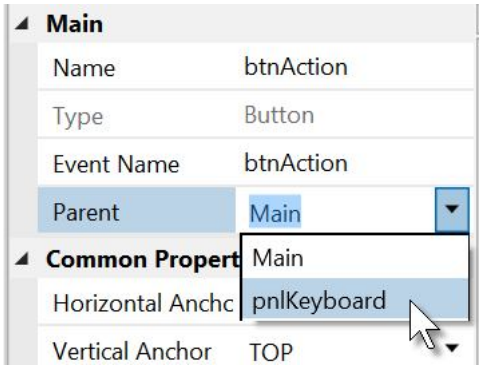


Change
Corner radius to 0

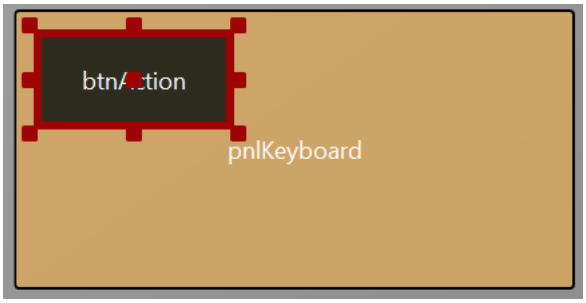


We will move btnAction from the Activity to the pnlKeyboard Panel.

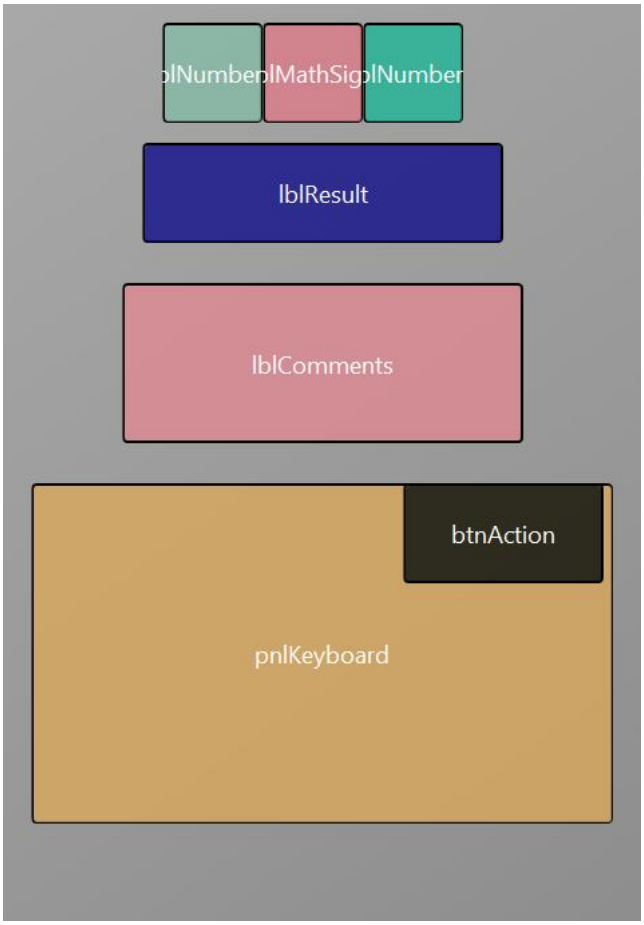
Click on btnAction.



In the Parent list click on pnlKeyboard.



The button now belongs to the Panel.



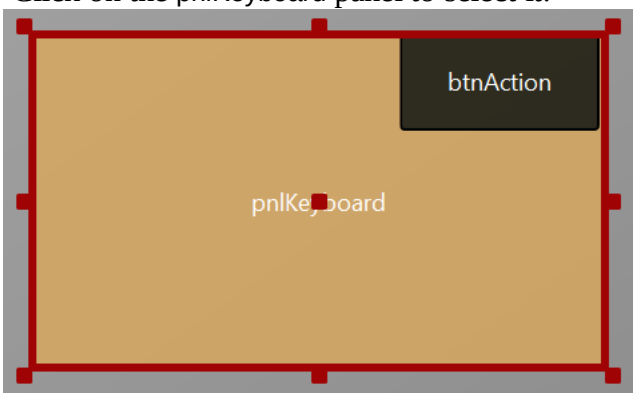
Now we rearrange the views to get some more space for the keyboard.

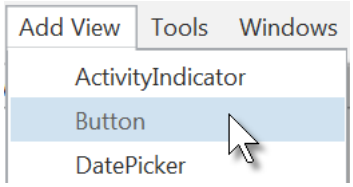
Set the properties below:

lblComments	Top = 140
pnlKeyboard	Left = 15
pnlKeyboard	Top = 240
pnlKeyboard	Width = 290
pnlKeyboard	Height = 170
pnlKeyboard	BorderWidth = 0

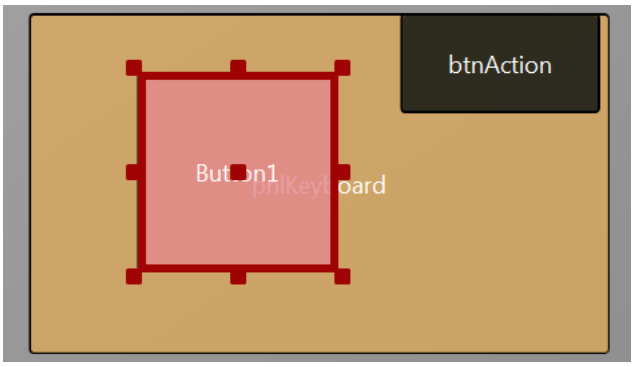
Move btnAction to the upper right corner of pnlKeyboard.

Click on the pnlKeyboard panel to select it.

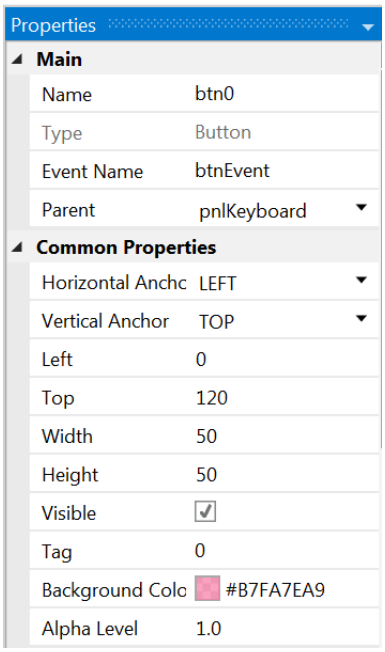


Click on 

to add a new button.



The new button is added.

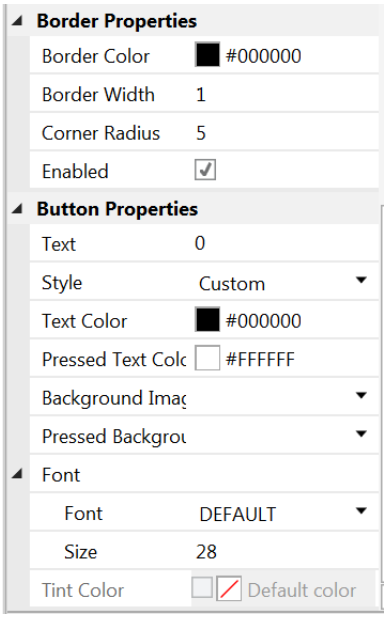


Change following properties:

Name to btn0
Event name to btnEvent

Left to 0
Top to 120
Width to 50
Height to 50

Tag to 0
Background Color to #B7FA7EA9



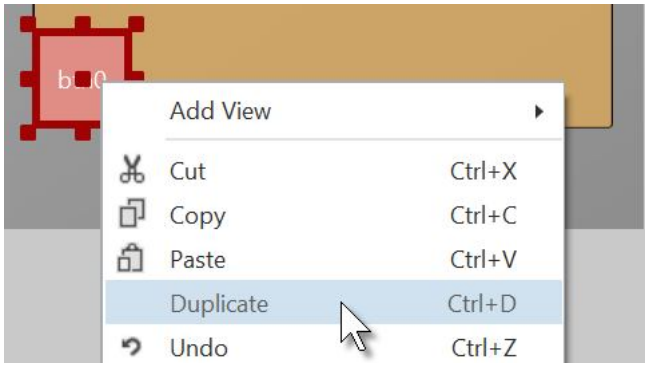
Boarder Width to 1
Corner Radius to 5

Text to 0

Size to 28

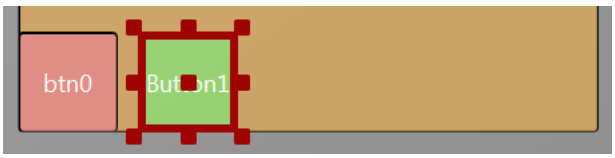
The button looks now like this.

Let us duplicate btn0 and position the new one beside button btn0.

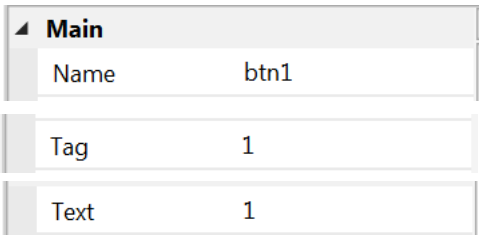


Select the Button btn0.

Right click on btn0
and click on Duplicate.



Move the new Button next to the previous one with a space.



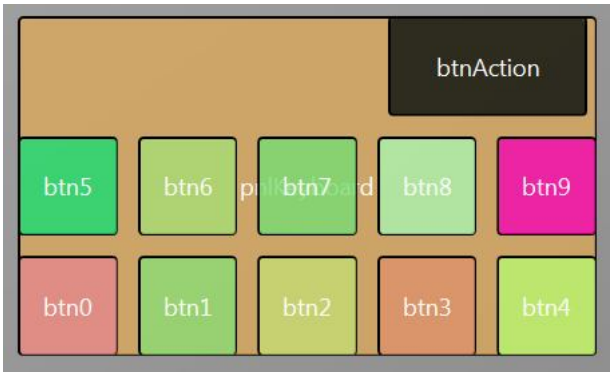
Change the following properties:

Name to btn1

Tag to 1

Text to 1

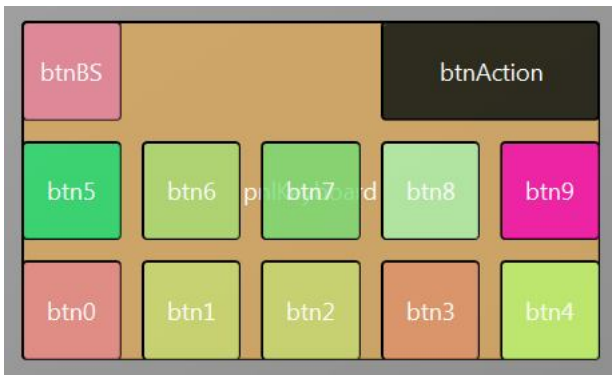
And the result.



Add 8 more Buttons and position them like in the image.

Change following properties:

Name	btn2,	btn3,	btn4	etc.	
Tag	2 ,	3 ,	4	etc.	
Text	2 ,	3 ,	4	etc.	



To create the BackSpace button, duplicate one of the number buttons, and position it in the top left corner.

Resize and position btnAction.

Change their Name, Tag, Text and Color properties as below.

btnBS <

Properties	
Main	
Name	btnBS
Type	Button
Event Name	btnEvent
Parent	pnlKeyboard
Common Properties	
Horizontal Anchc	LEFT
Vertical Anchor	TOP
Left	0
Top	0
Width	50
Height	50
Visible	☑
Tag	BS
Background Colc	#FF7E88FA
Alpha Level	1.0
Border Properties	
Border Color	#000000
Border Width	1
Corner Radius	5
Enabled	☑
Button Properties	
Text	<
Style	Custom
Text Color	#000000
Pressed Text Colc	#FFFFFF
Background Imag	
Pressed Backgrou	
Font	
Font	DEFAULT
Size	28
Tint Color	Default color

btnAction O K

Properties	
Main	
Name	btnAction
Type	Button
Event Name	btnAction
Parent	pnlKeyboard
Common Properties	
Horizontal Anchc	LEFT
Vertical Anchor	TOP
Left	180
Top	0
Width	110
Height	50
Visible	☑
Tag	
Background Colc	#FF03F86D
Alpha Level	1.0
Border Properties	
Border Color	#000000
Border Width	1
Corner Radius	5
Enabled	☑
Button Properties	
Text	O K
Style	Custom
Text Color	#000000
Pressed Text Colc	#FFFFFF
Background Imag	
Pressed Backgrou	
Font	
Font	DEFAULT
Size	24
Tint Color	Default color

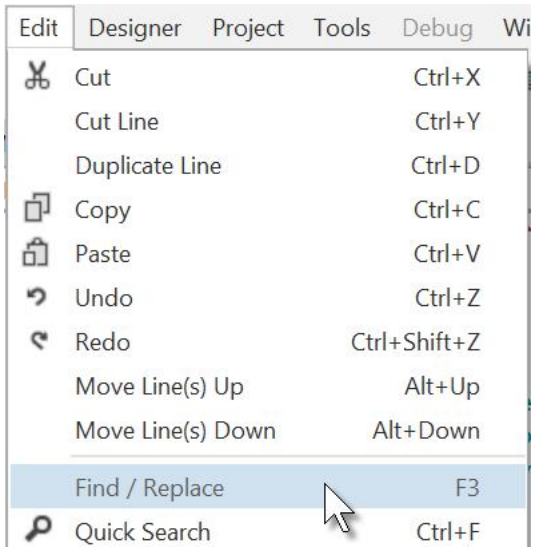
The finished new layout on the device.

If you had connect the device since the beginning you could have followed all the evolutions of the layout on the device.

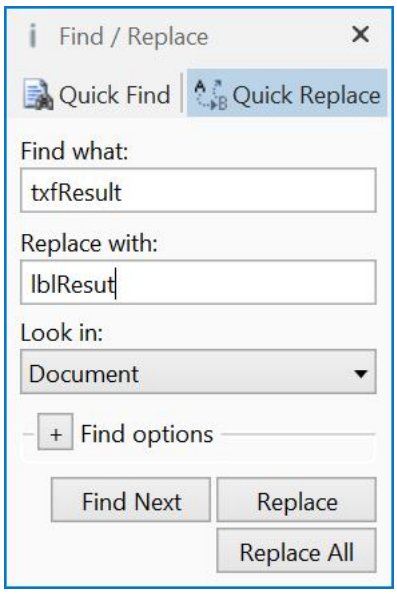
Now we will update the code.
First, we must replace the txfResult by lblResult because we replaced the TextField view by a Label.

```
17  Private btnAction As Button
18  Private lblMathSign As Label
19  Private lblComments As Label
20  Private lblNumber1 As Label
21  Private lblNumber2 As Label
22  Private txfResult As TextField
```

Double click on txfResult to select it.



Click on Find / Replace



The Find / Replace window is displayed.

Click on Replace All and close the window.

We also need to change its view type form TextField to Label.

```
Private lblResult As Label
```

Now we write the routine that handles the Click events of the Buttons.
The Event Name for all buttons, except btnAction, is "btnEvent".
The routine name for the associated click event will be btnEvent_Click.
Enter the following code:

```
Private Sub btnEvent_Click

End Sub
```

We need to know what button raised the event. For this, we use the Sender object which is a special object that holds the object reference of the view that generated the event in the event routine.

```
Private Sub btnEvent_Click
  Dim btnSender As Button

  btnSender = Sender

  Select btnSender.Tag
  Case "BS"
  Case Else
  End Select
End Sub
```

To have access to the properties of the view that raised the event we declare a local variable `Dim btnSender As Button`.
And set `btnSender = Sender`.

Then, to differentiate between the backspace button and the numeric buttons we use a Select / Case / End Select structure and use the Tag property of the buttons.
Remember, when we added the different buttons we set their Tag property to BS, 0, 1, 2 etc.

```
  Select btnSender.Tag
  Case "BS"
  Case Else
```

Select sets the variable to test.
Checks if it is the button with the "BS" tag value.
Handles all the other buttons.

Now we add the code for the numeric buttons.
We want to add the value of the button to the text in the lblResult Label.

```
   Select btnSender.Tag
   Case "BS"
   Case Else
      lblResult.Text = lblResult.Text & btnSender.Text
   End Select
End Sub
```

This is done in this line

```
   lblResult.Text = lblResult.Text & btnSender.Text
```

The "&" character means concatenation, so we just append to the already existing text the value of the Text property of the button that raised the event.

Now we add the code for the BackSpace button.

```
   Select btnSender.Tag
   Case "BS"
      If lblResult.Text.Length > 0 Then
         lblResult.Text = lblResult.Text.SubString2(0, lblResult.Text.Length - 1)
      End If
   Case Else
      lblResult.Text = lblResult.Text & btnSender.Text
   End Select
End Sub
```

When clicking on the BS button we must remove the last character from the existing text in lblResult. However, this is only valid if the length of the text is bigger than 0. This is checked with:

```
If lblResult.Text.Length > 0 Then
```

To remove the last character we use the SubString2 function.

```
lblResult.Text = lblResult.Text.SubString2(0,lblResult.Text.Length - 1)
```

SubString2(BeginIndex, EndIndex) extracts a new string beginning at BeginIndex (inclusive) until EndIndex (exclusive).

Now the whole routine is finished.

```
Private Sub btnEvent_Click
   Private btnSender As Button

   btnSender = Sender
   Select btnSender.Tag
   Case "BS"
      If lblResult.Text.Length >0 Then
         lblResult.Text = lblResult.Text.SubString2(0,lblResult.Text.Length - 1)
      End If
   Case Else
      lblResult.Text = lblResult.Text & btnSender.Text
   End Select
End Sub
```

In `Sub btnAction_Click` we add, at the end, `lblResult.Text = ""` to clear the text.

```
   Else
      New
      btnAction.Text = "O K"
      lblResult.Text = ""
   End If
End Sub
```

We can try to improve the user interface of the program by adding some colors to the lblComments Label.
Let us set:
- Yellow for a new problem
- Light Green for a GOOD answer
- Light Red for a WRONG answer.

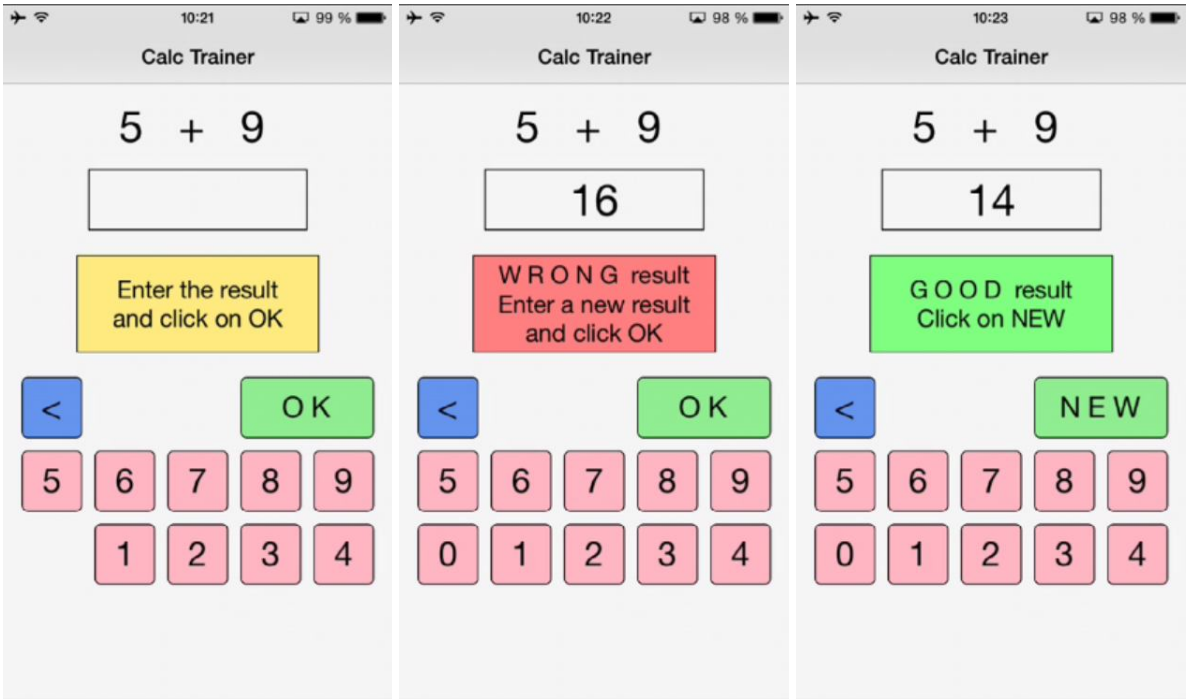
We first modify the New routine, where we add this line
`lblComments.Color = Colors.RGB(255,235,128)`

```
Private Sub New
   Number1 = Rnd(1, 10)          ' Generates a random number between 1 and 9
   Number2 = Rnd(1, 10)          ' Generates a random number between 1 and 9
   lblNumber1.Text = Number1     ' Displays Number1 in label lblNumber1
   lblNumber2.Text = Number2     ' Displays Number2 in label lblNumber2
   lblComments.Text = "Enter the result" & CRLF & "and click on OK"
   lblComments.Color = Colors.RGB(255,235,128)      ' yellow color
   lblResult.Text = ""           ' Sets lblResult.Text to empty
End Sub
```

And in the CheckResult routine we add the two lines with `lblComments.Color =...`

```
Private Sub CheckResult
   If lblResult.Text = Number1 + Number2 Then
      lblComments.Text = "G O O D  result" & CRLF & "Click on NEW"
      lblComments.Color = Colors.RGB(128,255,128)       ' light green color
      btnAction.Text = "N E W"
   Else
      lblComments.Color = Colors.RGB(255,128,128)       ' light red color
      lblComments.Text = "W R O N G  result" & CRLF & "Enter a new result" & CRLF & "and click OK"
   End If
End Sub
```

And we give the program a more meaningful title by adding `Page1.Title = "Calc Trainer"` in Application_Start just before `NavControl.ShowPage(Page1)`.



Another improvement would be to hide the '0' button to avoid entering a leading '0'.
For this, we hide the button in the New subroutine with line `btn0.Visible = False`.

```
Private Sub New
   Number1 = Rnd(1, 10)          ' Generates a random number between 1 and 9
   Number2 = Rnd(1, 10)          ' Generates a random number between 1 and 9
   lblNumber1.Text = Number1     ' Displays Number1 in label lblNumber1
   lblNumber2.Text = Number2     ' Displays Number2 in label lblNumber2
   lblComments.Text = "Enter the result" & CRLF & "and click on OK"
   lblComments.Color = Colors.RGB(255,235,128)      ' yellow color
   lblResult.Text = ""           ' Sets lblResult.Text to empty
   btn0.Visible = False
End Sub
```

We see that `btn0` is in red, this means that this object is not recognized by the IDE.

```
   btn0.Visible = False
```

So we must declare it, by adding `btn0` into line 17:

```
   Private btnAction, btn0 As Button
```

Now `btn0` is no more in red.

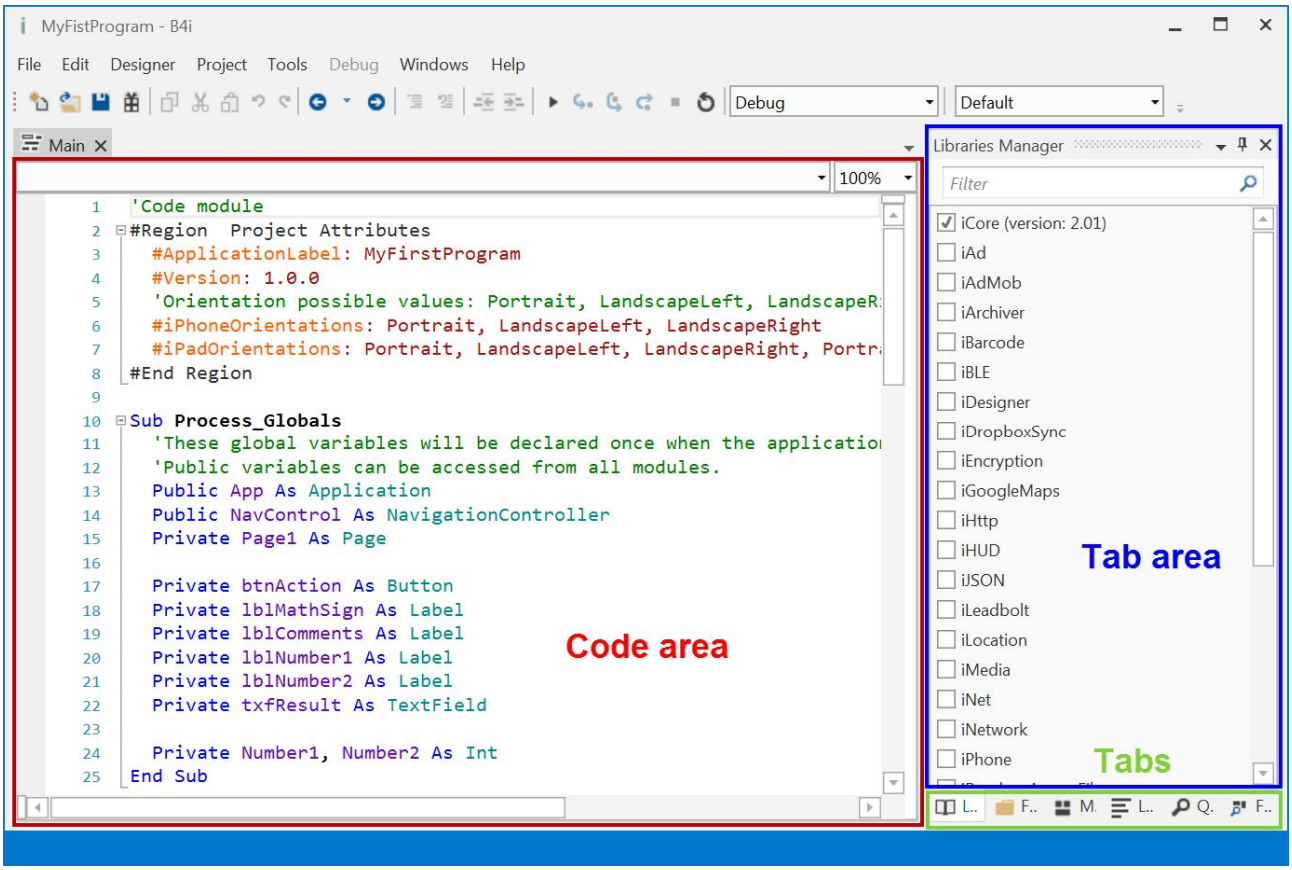
```
btn0.Visible = False
```

In addition, in the btnEvent_Click subroutine, we hide the button if the length of the text in lblResult is equal to zero and show it if the length is greater than zero.

```
Private Sub btnEvent_Click
   Dim btnSender As Button

   btnSender = Sender

   Select btnSender.Tag
   Case "BS"
      If lblResult.Text.Length >0 Then
         lblResult.Text = lblResult.Text.SubString2(0,lblResult.Text.Length - 1)
      End If
   Case Else
      lblResult.Text = lblResult.Text & Send.Tag
   End Select

   If lblResult.Text.Length = 0 Then
      btn0.Visible = False
   Else
      btn0.Visible = True
   End If
End Sub
```

4 The IDE

The **I**ntegrated **D**evelopment **E**nvironment.
When you run the IDE you will get a form like the image below:



You see 3 main areas:

- Code area The code editor
- Tabs area Window showing different data depending on the selected Tab.
- Tabs Tabs for different settings.

4.1 Menu and Toolbar

4.1.1 Toolbar

- Generates a new empty project.
- Loads a project.
- Saves the current project.
- Export as zip, exports the whole project in a zip file.
- Copies the selected text to the clipboard.
- Cuts the selected text and copies it to the clipboard.
- Pastes the text in the clipboard at the cursor position.
- Undoes the last operation.
- Redoes the previous operation.
- Navigate Backwards.
- Navigation History.
- Navigate Forward.
- Sets the selected lines as comments.
- Uncomments the selected lines.
- Decrease the indentation of the selected lines.
- Increase the indentation of the selected lines.
- Runs the compiler.

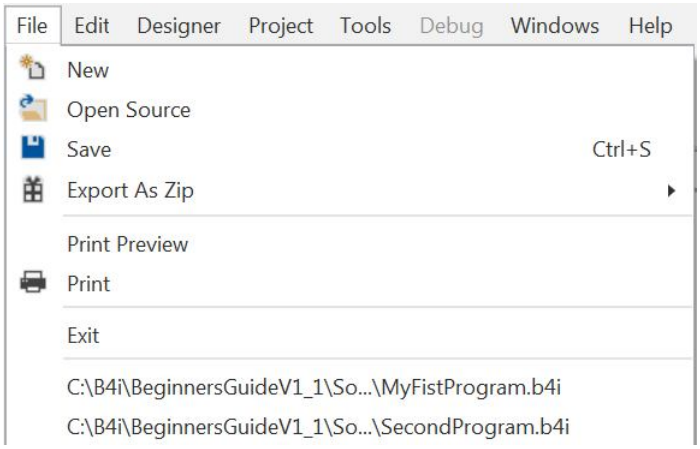
The 5 functions below are active only when the debugger is active.
Details in Debugging.

- Step In [F8].
- Step Over [F9].
- Step Out [F10].
- Stop.
- Restart [F11].

Compiler options list, currently only Debug.

Build Configuration.

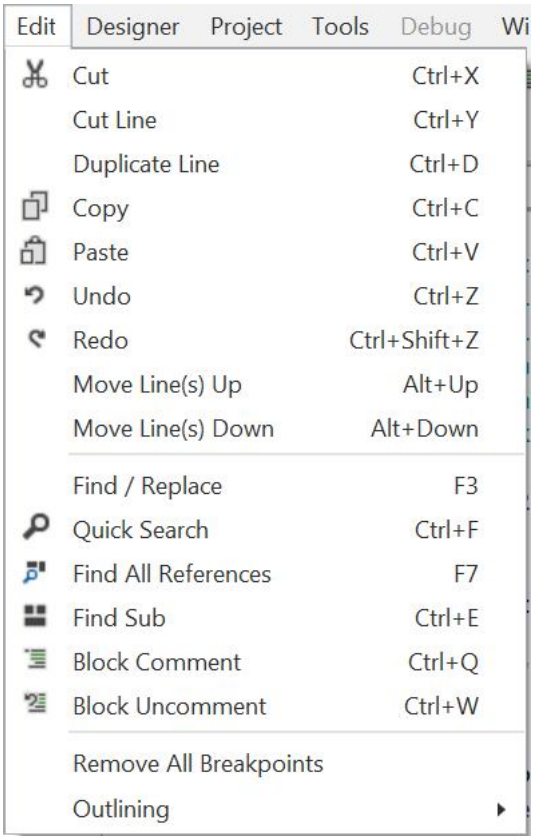
4.1.2 File menu



New
Open Source Loads a project.
Save Saves the current project.
Export As Zip
Exports the whole project in a zip file.
Print Preview Shows a print preview.
Print Prints the code.
Exit Leaves the IDE.

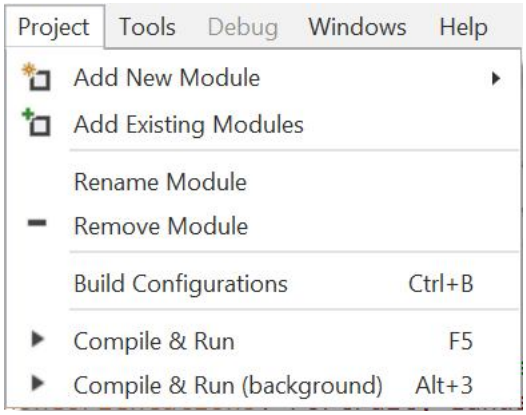
List of last loaded programs.

4.1.3 Edit menu



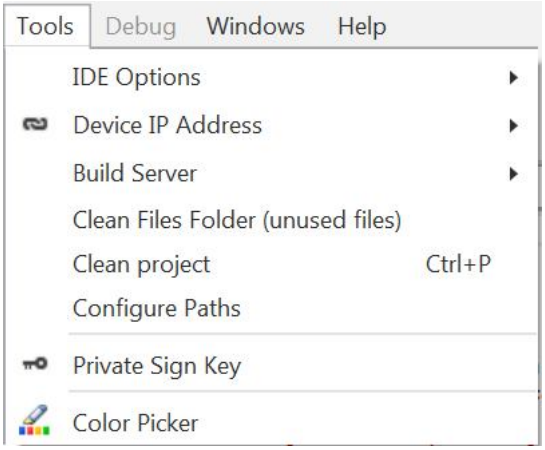
Cut Cuts the selected text and copies it to the clipboard.
Cut Line Cuts the line at the cursor position.
Duplicate Line Duplicates the selected line
Copy Copies the selected text to the clipboard.
Paste Pastes the text in the clipboard at the cursor position.
Undo Undoes the last operation.
Redo Redoes the previous operation.
Move Line(s) Up Moves the selected lines up.
Move Line(s) Down Moves the selected lines down.
Find / Replace Activates the Find and Replace function.
Quick Search Quick search function
Find All References Finds all References of a selected item
Find Sub Finds the selected Sub
Block Comment Sets the selected lines as comments.
Block Uncomment Uncomments the selected lines.
Remove All Breakpoints Breakpoints.
Outlining Collapse the whole code.

4.1.4 Project menu



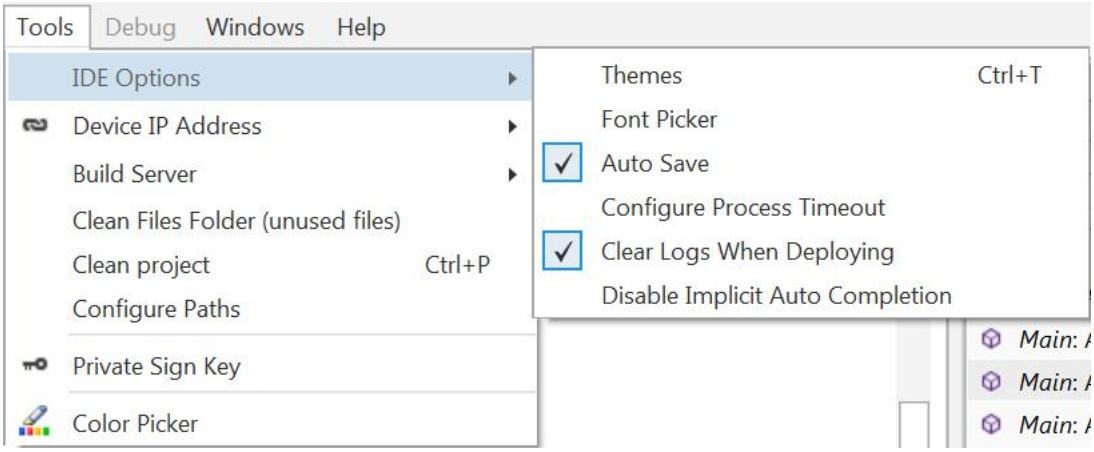
Adds a new module
Adds an existing module
Changes the module name
Removes the current module
Build Configurations Changes the package name.
Compiles and runs the program.
Compile & Run in the background.

4.1.5 Tools menu



IDE Options
B4i Bridge, sets the IP address for connection with Wifi
Build Server
Clean Files Folder (removes unused files)
Clean Project
Configure Paths
Private Sign Key
Color Picker

4.1.5.1 IDE Options



Themes

Font Picker

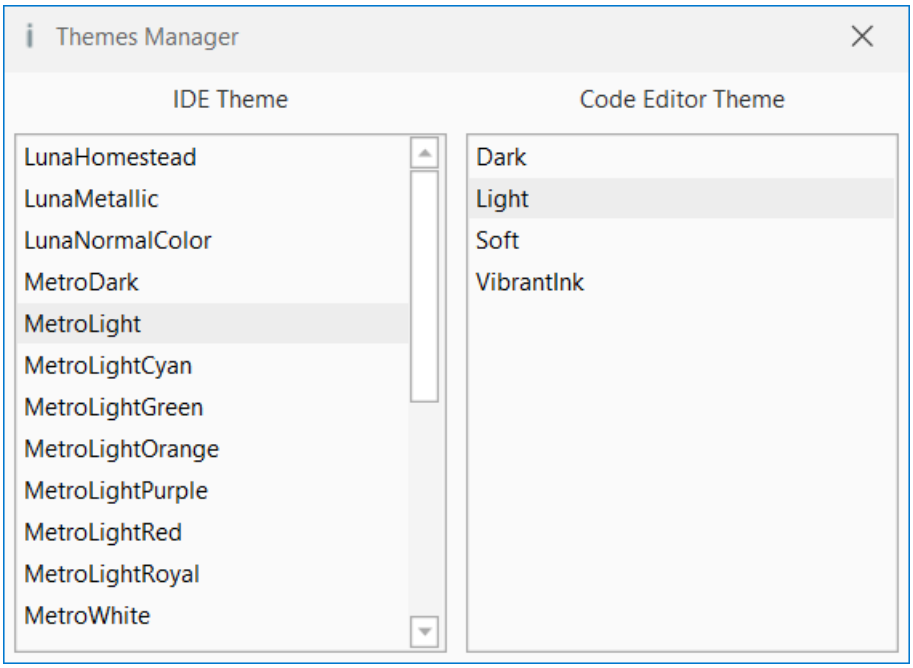
Auto Save — Saves the program every time you run it.

Configure Process Timeout

Clear Logs When Deploying — Removes all Log statements when compiled in Release mode.

Disable Implicit Auto Completion

4.1.5.1.1 Themes

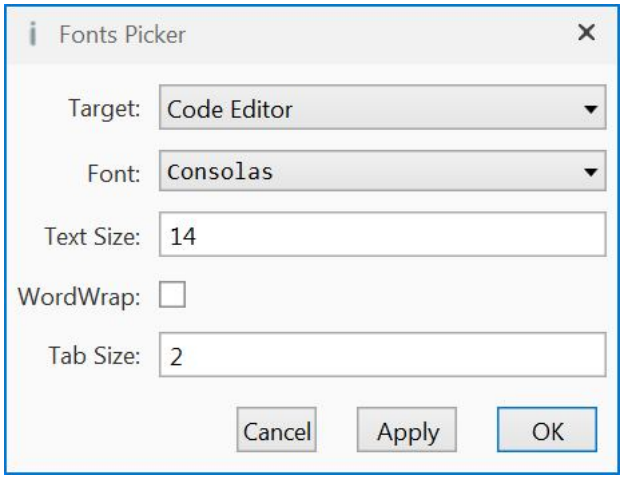


You can select different themes for the IDE.

The default theme is MetroWhite.

When you select one you see directly the new colors.

4.1.5.1.2 Font Picker



You can select a different font and text size.

Code Editor or for the Logs.

Select the font.

Enter the text size.

Select WordWrap

Enter the Tab size.

4.1.5.1.2.1 Word wrap

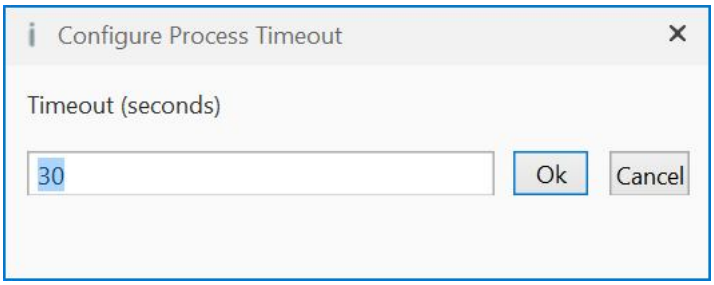
Without word wrap. The end of the line is hidden.

```
66      lblComments.Text = "W R O N G  result" & CRLF & "Enter a new result'
```

With word wrap. The end of the line is wrapped to the next line.

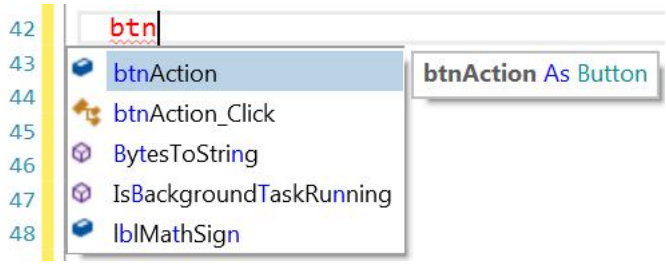
```
66      lblComments.Text = "W R O N G  result" & CRLF & "Enter a new
result" & CRLF & "and click OK"
```

4.1.5.1.3 Configure Process Timeout



Sometimes the compilation needs more time.
If you get a message 'Process timeout' you can increase the time.

4.1.5.1.4 Disable Implicit Auto Completion



If Disable Implicit Auto Completion is unchecked you will see a drop down list with possible words during typing.

If checked ☑ Disable Implicit Auto Completion you won't see the auto completion list.

4.1.5.2 Clean Files Folder (unused files)

Deletes files that are located under the Files folder but are not used by the project (it will not delete any file referenced by any of the project layouts).
A list of unused files will be displayed before deletion (and you may cancel the operation).

4.1.5.3 Clean Project

Deletes all files that are generated during compilation.

4.2 Code area

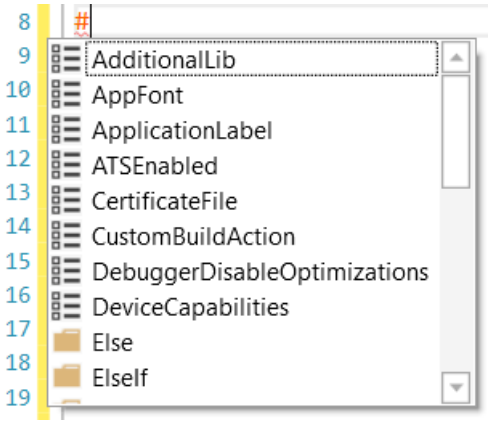
The code of the selected module is displayed in this area and can be edited.
The examples below are based on the code of the SecondProgram.

4.2.1 Code header Project Attributes

On top of the code you find the Project Attributes.

```
'Code module
#Region  Project Attributes
   #ApplicationLabel: SecondProgram
   #Version: 1.0.0
   'Orientation possible values: Portrait, LandscapeLeft, LandscapeRight and
PortraitUpsideDown
   #iPhoneOrientations: Portrait, LandscapeLeft, LandscapeRight
   #iPadOrientations: Portrait, LandscapeLeft, LandscapeRight, PortraitUpsideDown
#End Region
```

When you want to add a new Attribute you can just write # and the inline help shows all possibilities.



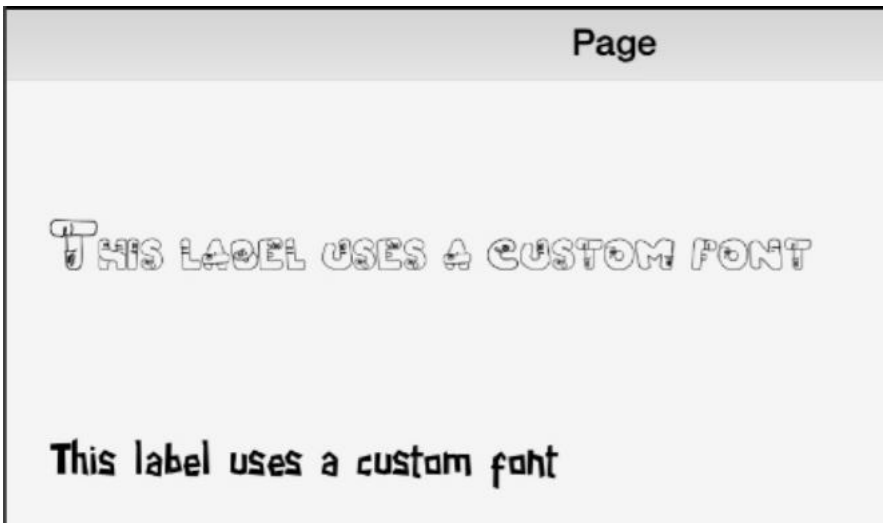
4.2.1.1 #AppFont

In order to add custom font files to your application you need to follow these instructions:
1. Add the font file to the "special" folder: <project>\Files\Special

2. Add the #AppFont attribute for each font file (including the extension):
```
#AppFont: papercuts-2.ttf
#AppFont: vermidirouge 1.0.ttf
```

3. Find the font name. You can double click on the font file:
   C:\Windows\Fonts

4. Create a new font with this font name. The name should not include spaces:
```
Label1.Font = Font.CreateNew2("VermidiRouge", 30)
Label2.Font = Font.CreateNew2("PaperCuts2", 20)
```



4.2.1.2 #ApplicationLabel

Name of the application. This name will be displayed below the application icon on the device.

4.2.1.3 #DeviceCapabilities

You can add device capabilities like:
```
#DeviceCapabilities: location-services
```
Which adds the location capabilities. Example in the Location & GPS Tutorial in the forum.

4.2.1.4 #If / #End If

It is possible to add #If / #End If structures in the code for different compiler options.
Example in the Build Configurations tutorial in the forum. The example is for B4A but the principle is the same.

4.2.1.5 #Region / #End Region

You can define regions in your code and collapse them. Details in Collapse a Region.

4.2.1.6 #IgnoreWarnings

The compiler adds warnings in the Log Tab, you can ignore warnings.
Details in Test Compile / Warnings.

4.2.1.7 #IpadOrientaions / #IPhoneOrientations

Orientations for iPhones and iPads. Possible orientations, by default:

```
#iPhoneOrientations: Portrait, LandscapeLeft, LandscapeRight
#iPadOrientations: Portrait, LandscapeLeft, LandscapeRight, PortraitUpsideDown
```

You can remove values to limit the orientation possibilities.

4.2.1.8 #PlistExtra

List of extra keys.

4.2.1.8.1 Share application files with iTunes

Share your application files with iTunes. A video in the forum shows this feature.

The File.DirDocuments folder can be shared through iTunes.
In order to enable this feature you need to add this attribute:

```
PlistExtra: <key>UIFileSharingEnabled</key><true/>
```

4.2.1.8.2 Prevent the application running in the background

```
#PlistExtra: <key>UIApplicationExitsOnSuspend</key><true/>
```

4.2.1.9 #URLScheme

Allows Dropbox synchronization using the iDropboxSync library.
For more information look at the DropboxSyncTutorial in the forum.

4.2.1.10 #Version

Program version attribute.

4.2.2 Undo – Redo

In the IDE it is possible to undo the previous operations and redo undone operations.
Click on to undo and on to redo.

4.2.3 Collapse a subroutine

In the IDE a subroutine can be collapsed to minimize the number of lines displayed.

```
55 Sub btnAction_Click
56   If btnAction.Text = "O K" Then
57     If txfResult.Text="" Then
58       Msgbox("No result entered","E R R O R")
59     Else
60       CheckResult
61     End If
62   Else
63     New
64     btnAction.Text = "O K"
65   End If
66 End Sub
```

The btnAction_Click routine expanded.

Click on to collapse the subroutine.

The btnAction_Click routine collapsed.

```
54
55 Sub btnAction_Click
67
```

Hovering with the mouse over the collapsed routine name shows its content.

```
54
55 Sub btnAction_Click
67   Sub btnAction_Click
68     If btnAction.Text = "O K" Then
69       If txfResult.Text="" Then
70         Msgbox("No result entered","E R R O R")
71       Else
72         CheckResult
73       End If
74     Else
75       New
76       btnAction.Text = "O K"
77     End If
78   End Sub
```

4.2.4 Collapse a Region

You can define ‘Regions’ in the code, which can be collapsed.

Example:

```
#Region GPS

#End Region
```

`#Region GPS` sets the beginning of a region and `#End Region` the end

```
#Region GPS
Private Sub Routine1
End Sub
Private Sub Routine2
End Sub
Private Sub Routine3
End Sub
#End Region
```

Then you can add the subroutines between the two limits:

```
#Region GPS
Private Sub Routine1
```

Then clicking on ⊟

```
#Region GPS
```

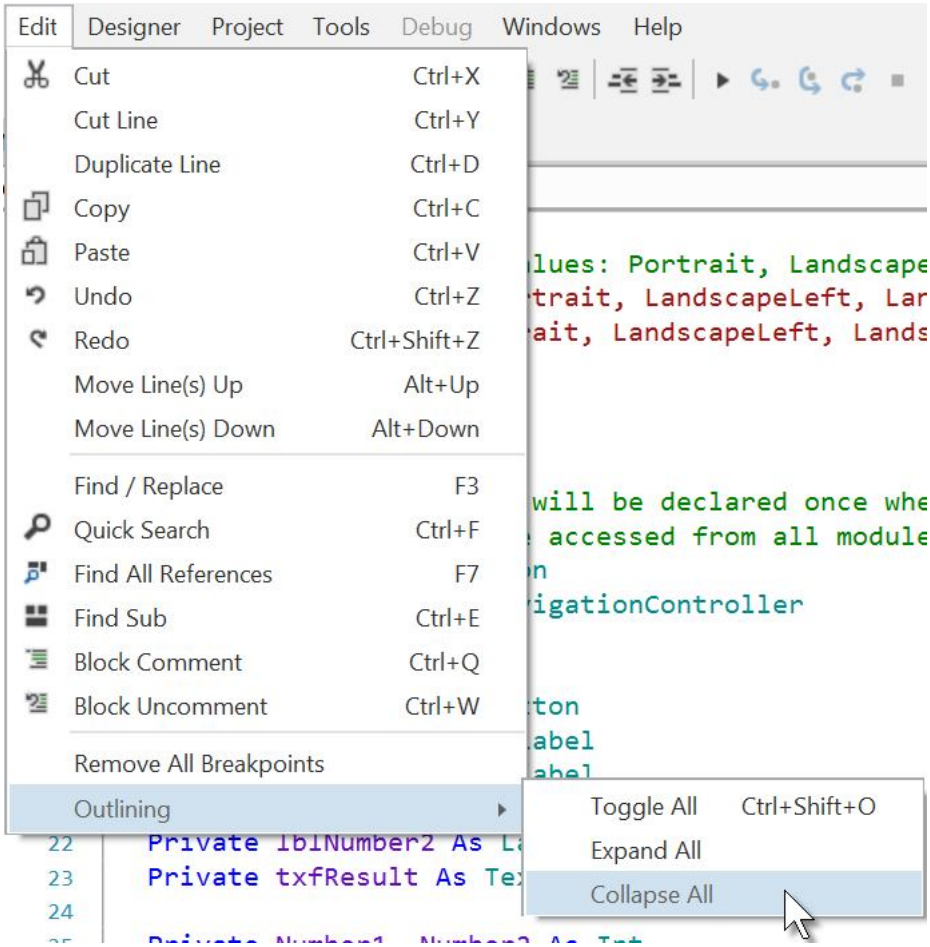
collapses the whole region.

```
#Region GPS
#Region GPS
Private Sub Routine1
End Sub
Private Sub Routine2
End Sub
Private Sub Routine3
End Sub
#End Region
```

Hovering over GPS

shows the beginning of the code, not all the routines in the region.

4.2.5 Collapse the whole code



In the Edit menu there are three functions:

- **Toggle All**
Expands the collapsed routines and collapses the extended routines and regions.

- **Expands All**
Expands the whole code

- **Collapse All**
Collapses the whole code.

Clicking on Collapse All.

The whole code collapsed.

```
1   'Code module
2   #Region  Project Attributes
9
10  Sub Process_Globals
26
27  Private Sub Application_Start (Nav As NavigationController)
37
38  Private Sub Page1_Resize(Width As Int, Height As Int)
41
42  Private Sub Application_Active
45
46  Private Sub Application_Inactive
49
50  Private Sub Application_Background
53
54  Sub btnAction_Click
66
67  Sub New
75
76  Sub CheckResult
84
```

```
67  Sub New
75
76  Sub New
84    Number1 = Rnd(1, 10)        ' Generates a random number between 1 and 9
      Number2 = Rnd(1, 10)        ' Generates a random number between 1 and 9
      lblNumber1.Text = Number1   ' Displays Number1 in label lblNumber1
      lblNumber2.Text = Number2   ' Displays Number2 in label lblNumber2
      lblComments.Text = ...
```

Hovering with the mouse over a subroutine shows the beginning of its content.

4.2.6 Copy a selected bloc of text

It is possible to copy a selected bloc of text to the clipboard.

To select the bloc press Alt and move the mouse cursor.

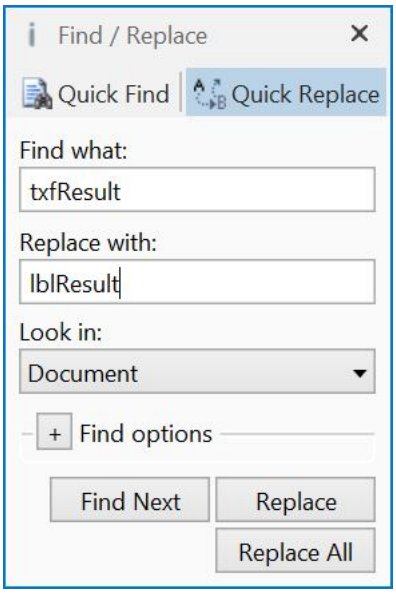
```
15  Private btnAction As Button
16  Private lblMathSign As Label
17  Private lblComments As Label
18  Private lblNumber1 As Label
19  Private lblNumber2 As Label
20  Private txfResult As TextField
```

4.2.7 Find / Replace

```
17  Private btnAction As Button
18  Private lblMathSign As Label
19  Private lblComments As Label
20  Private lblNumber1 As Label
21  Private lblNumber2 As Label
22  Private txfResult As TextField
```

Example:
Click on `txfResult` to select it.

Press F3, or click on Find / Replace in the Edit menu.



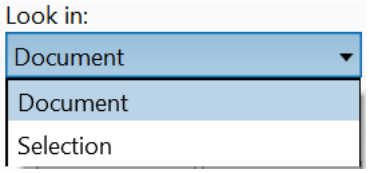
This window will be displayed.

Enter `lblResult` in the 'Replace with' field.

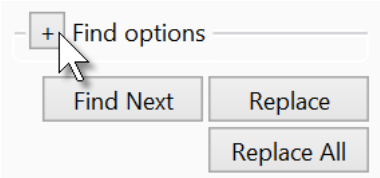
Now, you can either:

- Find Next — Find the next occurrence.
- Replace — Replace the current occurrence and find the next one.
- Replace All — Replace all occurrences.

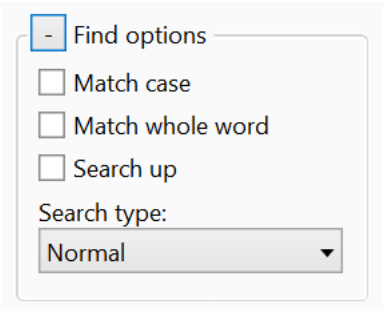
Look in:



You can search either in a Selection or in the Document, which means in the selected module not the whole project.



You can select Find options, click on +.



These options are self-explanatory.

4.2.8 Commenting and uncommenting code

A selected part of the code can be set to comment lines or set to normal.

```
17  Private btnAction As Button
18  Private lblMathSign As Label
19  Private lblComments As Label
20  Private lblNumber1 As Label
21  Private lblNumber2 As Label
22  Private txfResult As TextField
```

Original code

```
17  Private btnAction As Button
18  Private lblMathSign As Label
19  Private lblComments As Label
20  Private lblNumber1 As Label
21  Private lblNumber2 As Label
22  Private txfResult As TextField
```

Select the code.

Click on .

```
17  ' Private btnAction, btn0 As Button
18  ' Private lblMathSign As Label
19  ' Private lblComments As Label
20  ' Private lblNumber1 As Label
21  ' Private lblNumber2 As Label
22  ' Private lblResult As Label
```

All lines are set as comments.

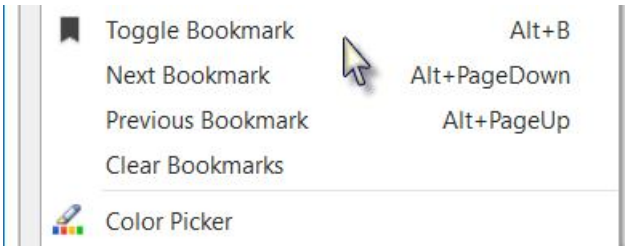
To set the lines to normal, select the lines and click on .

4.2.9 Bookmarks

You can set 'bookmarks' anywhere in the code and jump forward and backwards between these bookmarks.

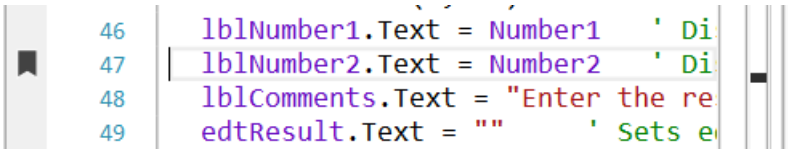
To set or clear a bookmark, select the line and press Alt + B.

Or right click on the line where you want to set a bookmark.



You will get a pop up menu, click on Toggle Bookmark to activate or deactivate a bookmark.

You will see this mark on the left of the line and a small black line in the right slider:



To jump to the next bookmark press Alt + PageDown
or right click and click on Next Bookmark Alt+PageDown

To jump to the previous bookmark press on Alt + PageUp
or right click and click on Previous Bookmark Alt+PageUp

To clear all bookmarks right click and click on Clear Bookmarks

4.2.10 Indentation

A good practice is to use the indentation of code parts.
For example for subroutines, loops, structures etc.

```
Sub btnAction_Click
If btnAction.Text = "O K" Then
If txfResult.Text="" Then
Msgbox("No result entered","E R R O R")
Else
CheckResult
End If
Else
New
btnAction.Text = "O K"
End If
End Sub
```

This code is difficult to read because the structure of the code is not obvious.

```
Sub btnAction_Click
  If btnAction.Text = "O K" Then
    If txfResult.Text="" Then
      Msgbox("No result entered","E R R O R")
    Else
      CheckResult
    End If
  Else
    New
    btnAction.Text = "O K"
  End If
End Sub
```

This code is much easier to read, the structure of the code is in evidence.

A tabulation value of 2 for the indentation is a good value.

```
Sub btnAction_Click
    If btnAction.Text = "O K" Then
        If txfResult.Text="" Then
            Msgbox("No result entered","E R R O R")
        Else
            CheckResult
        End If
    Else
        New
        btnAction.Text = "O K"
    End If
End Sub
```

Example with an indentation of 4

Personally,
I prefer a value of 2.

Whole blocks of code can be indented forth and back at once.

```
17  Private btnAction As Button
18  Private lblMathSign As Label
19  Private lblComments As Label
20  Private lblNumber1 As Label
21  Private lblNumber2 As Label
22  Private txfResult As TextField
```

Original code

```
17  Private btnAction As Button
18  Private lblMathSign As Label
19  Private lblComments As Label
20  Private lblNumber1 As Label
21  Private lblNumber2 As Label
22  Private txfResult As TextField
```

Select the code block.

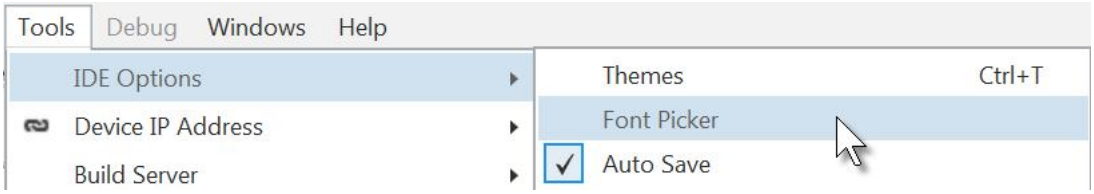
Click on .

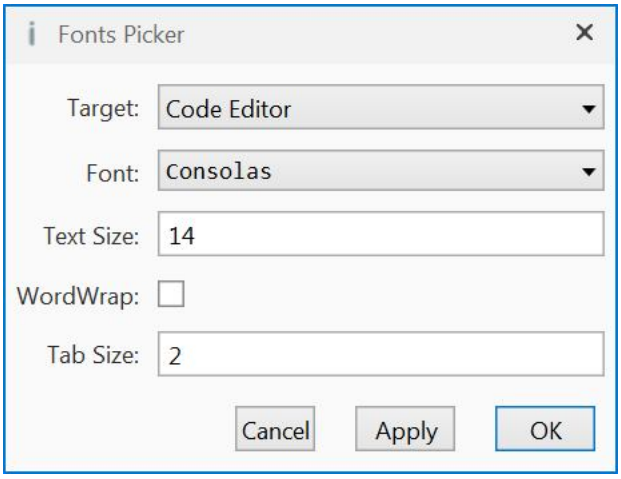
```
17    Private btnAction As Button
18    Private lblMathSign As Label
19    Private lblComments As Label
20    Private lblNumber1 As Label
21    Private lblNumber2 As Label
22    Private txfResult As TextField
```

The whole block has moved one tabulation to the right.

To move a block to the left.
Select the code and click on .

The indentation value can be changed in the Tools menu in the IDE Options Font Picker.





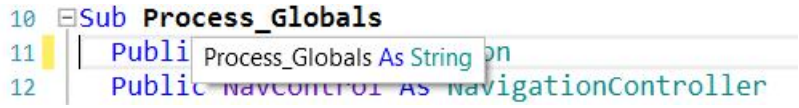
Enter the value and click on OK.

4.2.11 Documentation tool tips when hovering over code elements

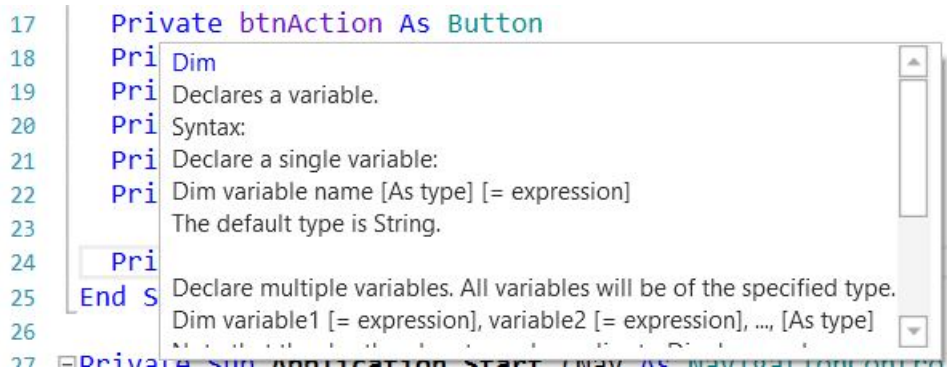
When you hover over code elements the on line help is displayed.

Examples:

Hovering over Globals:

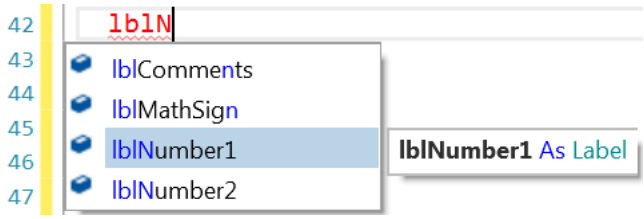


Hovering over Private:



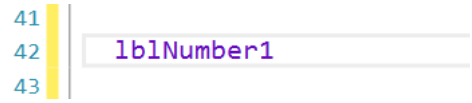
4.2.12 Auto Completion

A very useful tool is the autocomplete function.
Example:



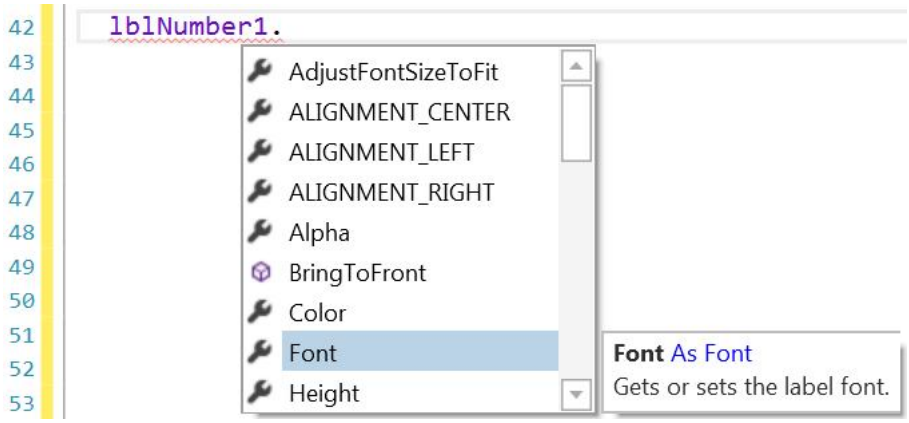
Let us write lblN
All variables, views and property names containing the letters already written are shown in a popup menu with the online help for the highlighted variable, view or property name.

To choose lblNumber1 press Return.



The selected name is completed.

To choose lblNumber2 press the down arrow and press Return.



When selecting an item, the internal help is displayed.

Pressing on the up / down arrows selects the previous or next item with its help.

Pressing a character updates the list and shows the parameter beginning with that character.

The best way to learn it is to 'play' with it.

A second Autocomplete function allows you to create event subroutines.

Enter the Sub word plus a blank character.

```
87
88  Sub
89
90
```

Press Tab to insert event declaration.

Press Tab.

```
87
88  Sub   Select type and press enter
89
90
91
92
93
94
95
96
97
98
99
100
```

ActionSheet
ActivityIndicator
Application
Button
DatePicker
ImageView
Label
MediaPlayer
NativeObject

Select the type, Button in our example.

```
87
88  Sub   Select type and press enter  Button  >
89
90
91
92
```

Click
LongClick

Select the type, Click in our example.

```
87
88  Sub EventName_Click
89
90  End Sub
```

The subroutine frame is generated.

Modify 'EventName' to the EventName of the button, 'btnOK' in our case and press Return.

```
87
88  Sub btnOK_Click
89
90  End Sub
```

```
87
88  Sub btnOK_Click
89
90  End Sub
```

The routine is ready.

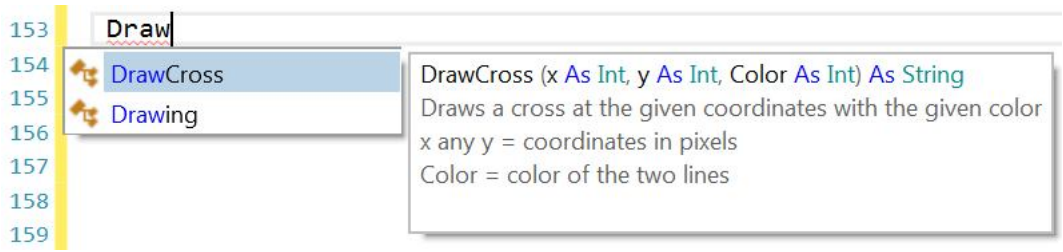
4.2.13 Built in documentation

Another useful function is the built-in documentation.

Comments above subs, such as:

```
'Draws a cross at the given coordinates with the given color
'x any y = coordinates in pixels
'Color = color of the two lines
Sub DrawCross(x As Int, y As Int, Color As Int)
      Private d = 3dip As Int

      cvsLayer(2).DrawLine(x - d, y, x + d, y, Color, 1)
      cvsLayer(2).DrawLine(x, y - d, x, y + d, Color, 1)
End Sub
```

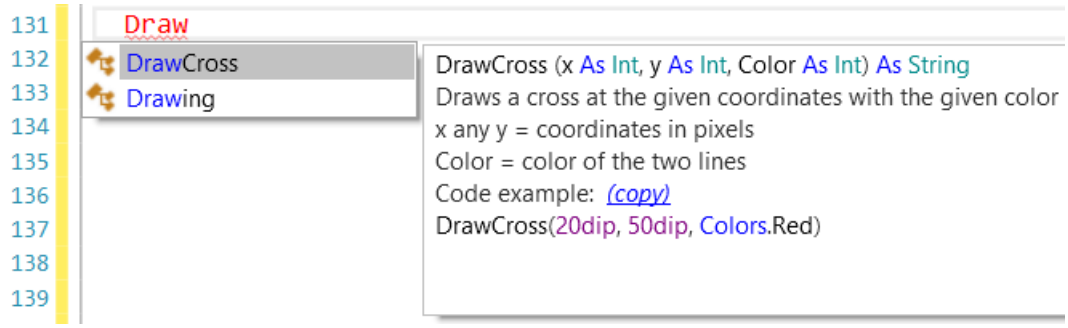
Will automatically appear in the auto complete pop-up window:



If you want to add a code example you can use <code> </code> tags:

```
'Draws a cross at the given coordinates with the given color
'x any y = coordinates in pixels
'Color = color of the two lines
'Code example: <code>
'DarwCross(20dip, 50dip, Colors.Red)
'</code>
Sub DrawCross(x As Int, y As Int, Color As Int)
   Private d = 3dip As Int

   cvsLayer(2).DrawLine(x - d, y, x + d, y, Color, 1)
   cvsLayer(2).DrawLine(x, y - d, x, y + d, Color, 1)
End Sub
```

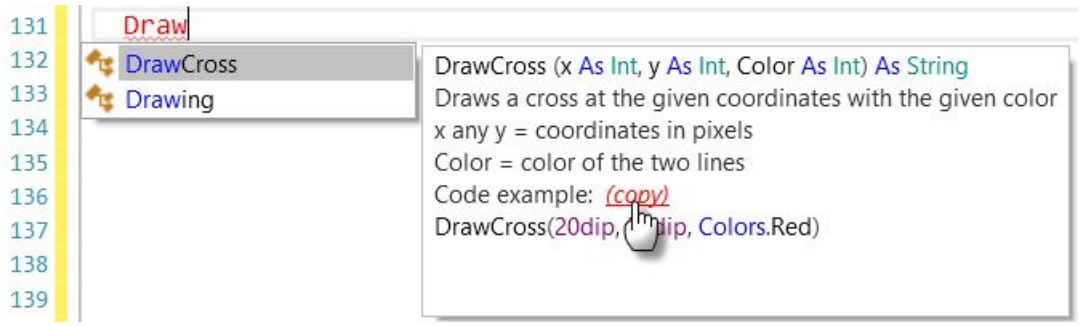
The code will be syntax highlighted:



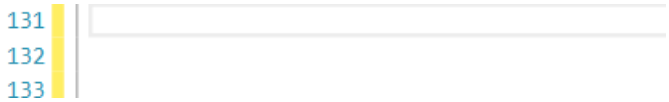
4.2.13.1 Copy code examples

You can copy the code example in your code.

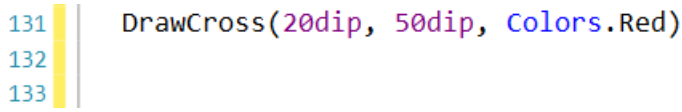
When hovering over (copy) you can copy the code example to the clipboard.



Remove `Draw`



And copy.



4.2.14 Jump to an identifier

Sometimes it is useful to jump from a subroutine call to the subroutine definition or to go from a view or a variable to its definition.

The easiest way is to press Ctrl and click on the desired text.

```
35
36    New
37  End Sub
```

The cursor changes when you hover over the text.

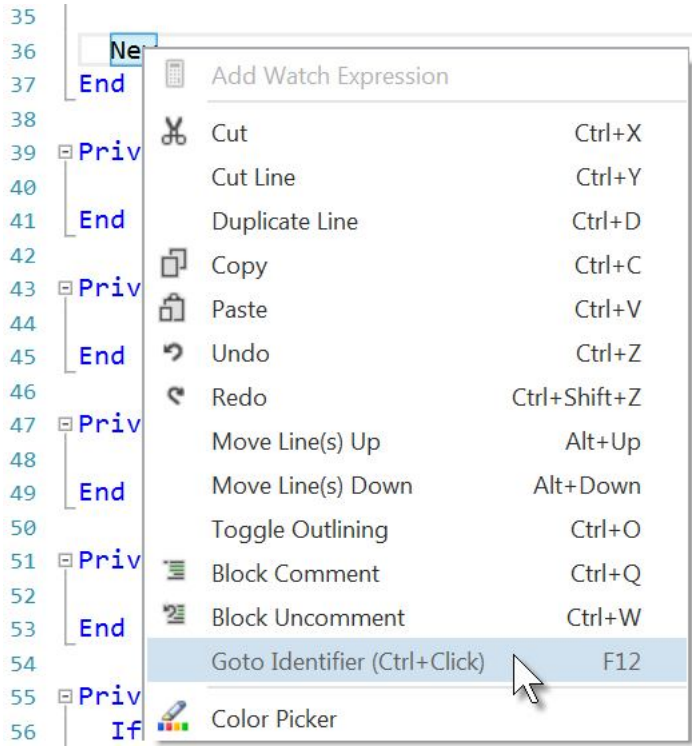
Another way:

```
35
36    New
37  End Sub
38
```

Double click on the text of the subroutine call, the variable or the view to select it.

It is highlighted in dark blue and all the other occurrences are highlighted in light blue.

Press F12, or like below.



Right click on the selected text.

Click on Goto Identifier (Ctrl+Click).

```
69  Private Sub New
70    Number1 = Rnd(1, 10)          ' Genera
71    Number2 = Rnd(1, 10)          ' Genera
72    lblNumber1.Text = Number1     ' Displa
73    lblNumber2.Text = Number2     ' Displa
74    lblComments.Text = "Enter the result
75    lblComments.Color = Colors.RGB(255,2
76    lblResult.Text = ""           ' Sets l
77    btn0.Visible = False
78  End Sub
```

And you are there.

4.2.15 Highlighting occurrences of words

When you select a single word, it is highlighted in dark blue and all the other occurrences in the code are highlighted in light blue and in the scroll view on the right side.
With the slider you can move up or down the code to go to the other occurrences.

```
152 Sub ShowTable
153   Dim i As Int
154   Dim Query As String
155
156   Query = "SELECT "
157   For i = 0 To ColNumber - 1
158     If i < ColNumber - 1 Then
159       Query = Query & ColNames(i) & " As [" & ColAliasNames(i) & "], "
160     Else
161       Query = Query & ColNames(i) & " As [" & ColAliasNames(i) & "] "
162     End If
163   Next
164   Query = Query & " FROM " & SQLTabelName
165
166   'depending if the filter is active or not we add the filter query at the end of
167   'the filter query is defined in the Filter Activity
168   If Filter.flagFilterActive = False Then
169   Else
170     Query = Query & Filter.Query
171   End If
172   'displays the database in a table
173   wbvTable.LoadHtml(ExecuteHtml(SQL1, Query, Null, 0, True))
```

4.2.16 Debug

The Debug mode is activated by default on top of the IDE.

Look at chapter Debugging for debug features.

4.2.17 Breakpoints

Clicking on the left side in a line sets a breakpoint.

```
68  Private Sub New
69    Number1 = Rnd(1, 10)          ' Generates a random number be
70    Number2 = Rnd(1, 10)          ' Generates a random number be
71    lblNumber1.Text = Number1     ' Displays Number1 in label lb
72    lblNumber2.Text = Number2     ' Displays Number2 in label lb
73    lblComments.Text = "Enter the result" & CRLF & "and click
74    lblComments.Color = Colors.RGB(255,235,128) ' yellow color
```

When the program runs it stops at the first encountered breakpoint.

```
68  Private Sub New
69    Number1 = Rnd(1, 10)          ' Generates a random number be
70    Number2 = Rnd(1, 10)          ' Generates a random number be
71    lblNumber1.Text = Number1     ' Displays Number1 in label lb
72    lblNumber2.Text = Number2     ' Displays Number2 in label lb
73    lblComments.Text = "Enter the result" & CRLF & "and click
74    lblComments.Color = Colors.RGB(255,235,128) ' yellow color
```

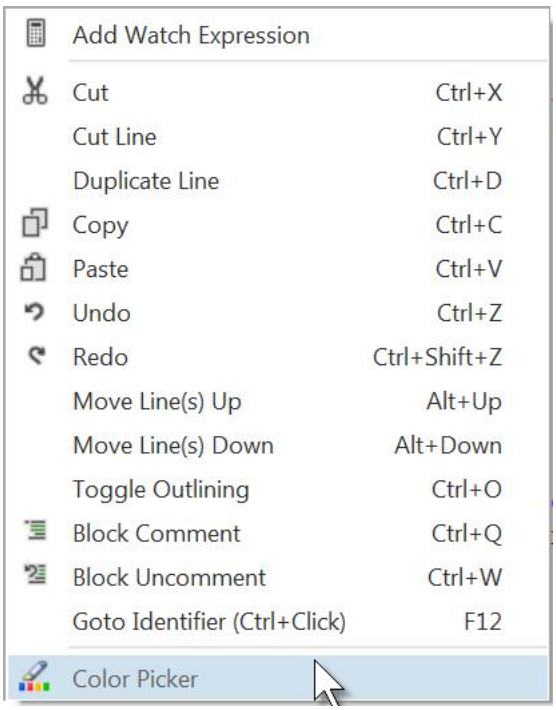
You can remove all breakpoints in the Edit menu with Remove All Breakpoints.

The use of breakpoints is explained in detail in the Debugging chapter.

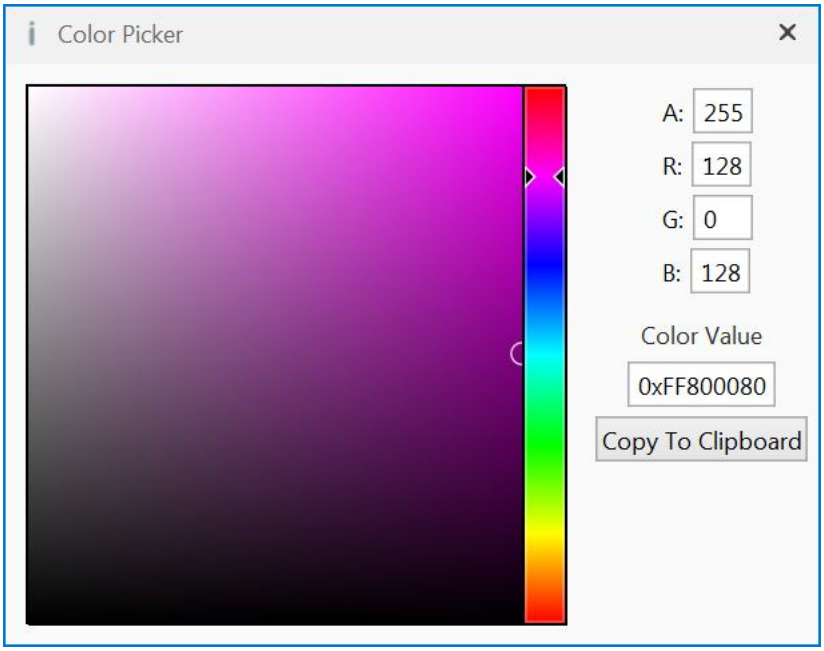
4.2.18 Color Picker

In the Tools click on Color Picker.

Or, in the code, right click to show the pop up menu below and click on menu Color Picker.



The Color Picker will be displayed.



To select a color you can:

- Move the small circle.
- Move the vertical cursor.
- Enter the ARGB values.

Click on Copy To Clipboard to copy the value to the Clipboard.

You may then paste the value into your code.

4.2.19 Colors in the left side

Sometimes, you will see yellow or green vertical lines in the left side od the IDE.

As soon as you modify a line it will be marked with a yellow vertical line on the right of the line number meaning that this line was modified.

```
74  Sub CheckResult
75    If txfResult.Text =
76      lblComments.Text
77      btnAction.Text =
78    Else
79      lblComments.Text
80    End If
81  End Sub
```

If we click on  to save the project the yellow lines become green showing a modified code but already saved. You can also press Ctrl + S to save the project.


```
74  Sub CheckResult
75    If txfResult.Text =
76      lblComments.Text =
77      btnAction.Text = "
78    Else
79      lblComments.Text =
80    End If
81  End Sub
```

If we leave the IDE and load the project again the green lines disappear.

4.2.20 URLs in comments and strings are ctrl-clickable

URLs in comments and strings are ctrl-clickable.

In a comment:

```
162 'https://www.b4x.com
```

If the cursor is on the line and you press Ctrl the url is highlighted in blue and if you click on it the url it is executed. Hovering over the line with Ctrl pressed does also highlight the url.

```
162 'https://www.b4x.com
163
164
```

In a String:

```
165 Private url As String
166
167 url = "https://www.b4x.com"
```

The cursor must be over the String variable and not over text.

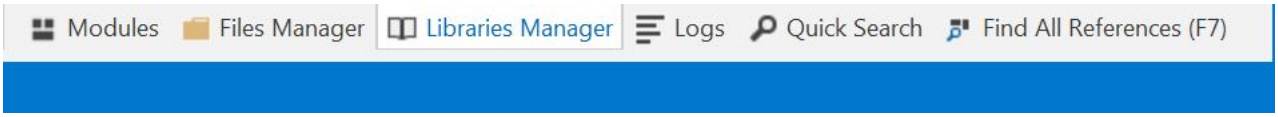
```
165 Private url As String
166
167 url = "https://www.b4x.com"
168     As String
169     (local variable)
170
```

4.3 Tabs

There are 6 tabs at the bottom right corner of the IDE that display different windows.

The short version.

The wide version.

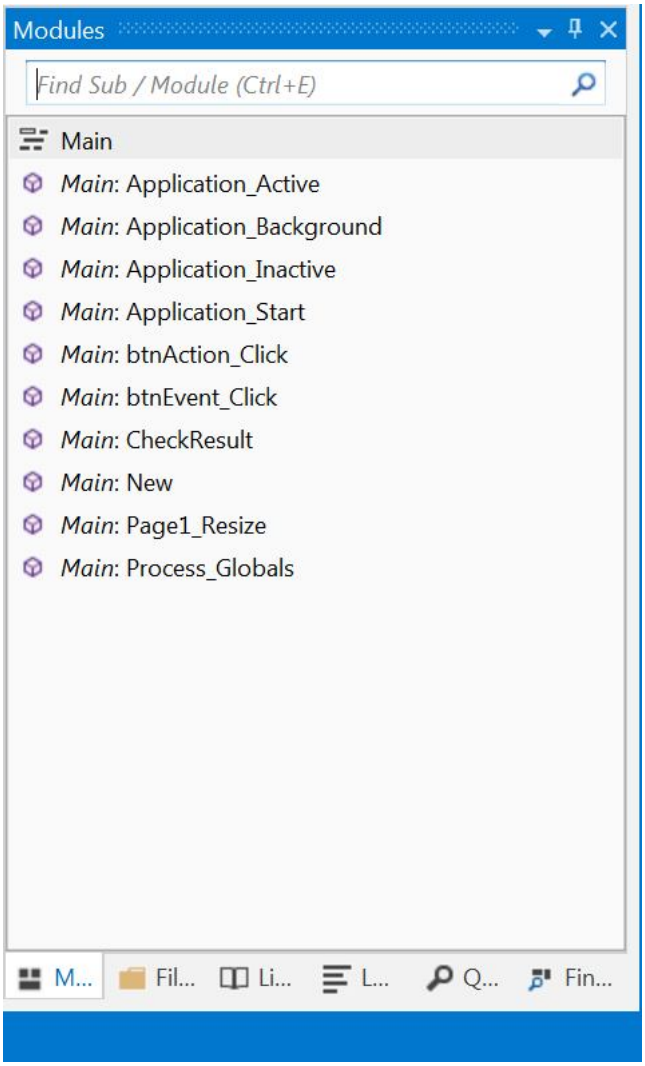


The 6 Tabs are:

- Modules
- Files Manager
- Libraries Manager
- Logs
- Quick Search
- Find All References

4.3.1 Floating Tab windows

When you start the default IDE all Tab windows are docked in the Tab area.

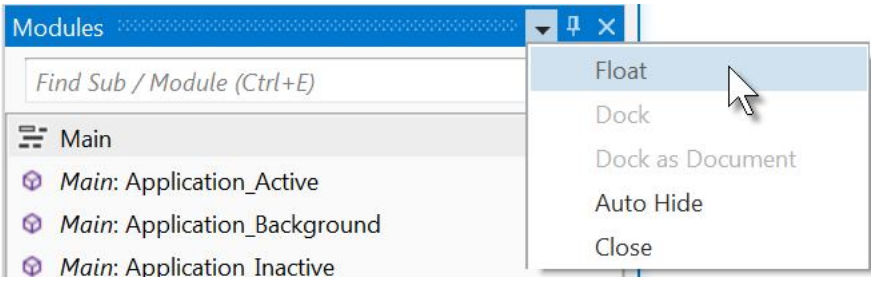


You can set each Tab window as a separate floating window.

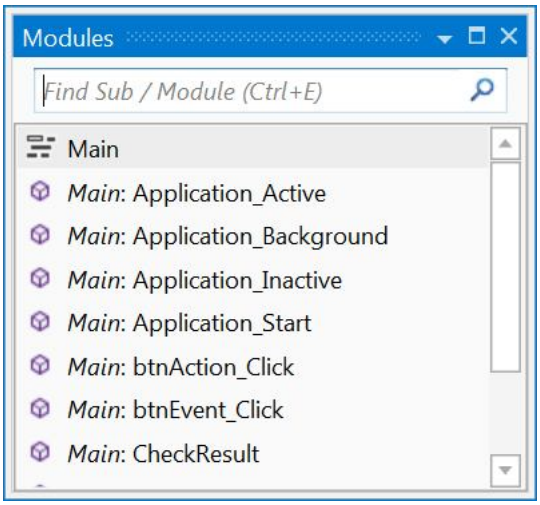
4.3.2 Float

To set the Modules Tab window to floating click in the title on .

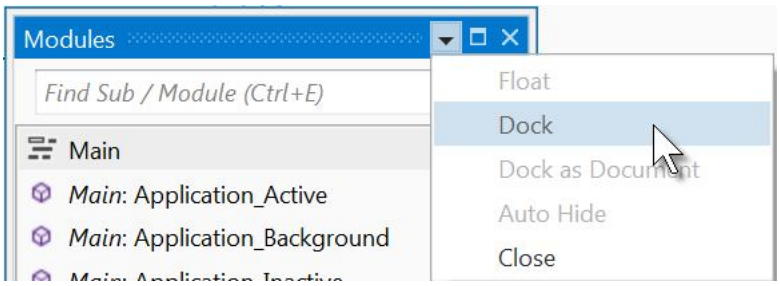
Click on Float.



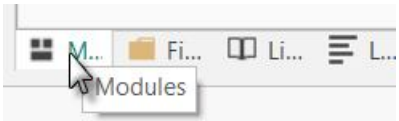
The Modules Tab Window is now floating, you can place it where you want on the screen even on a second monitor.



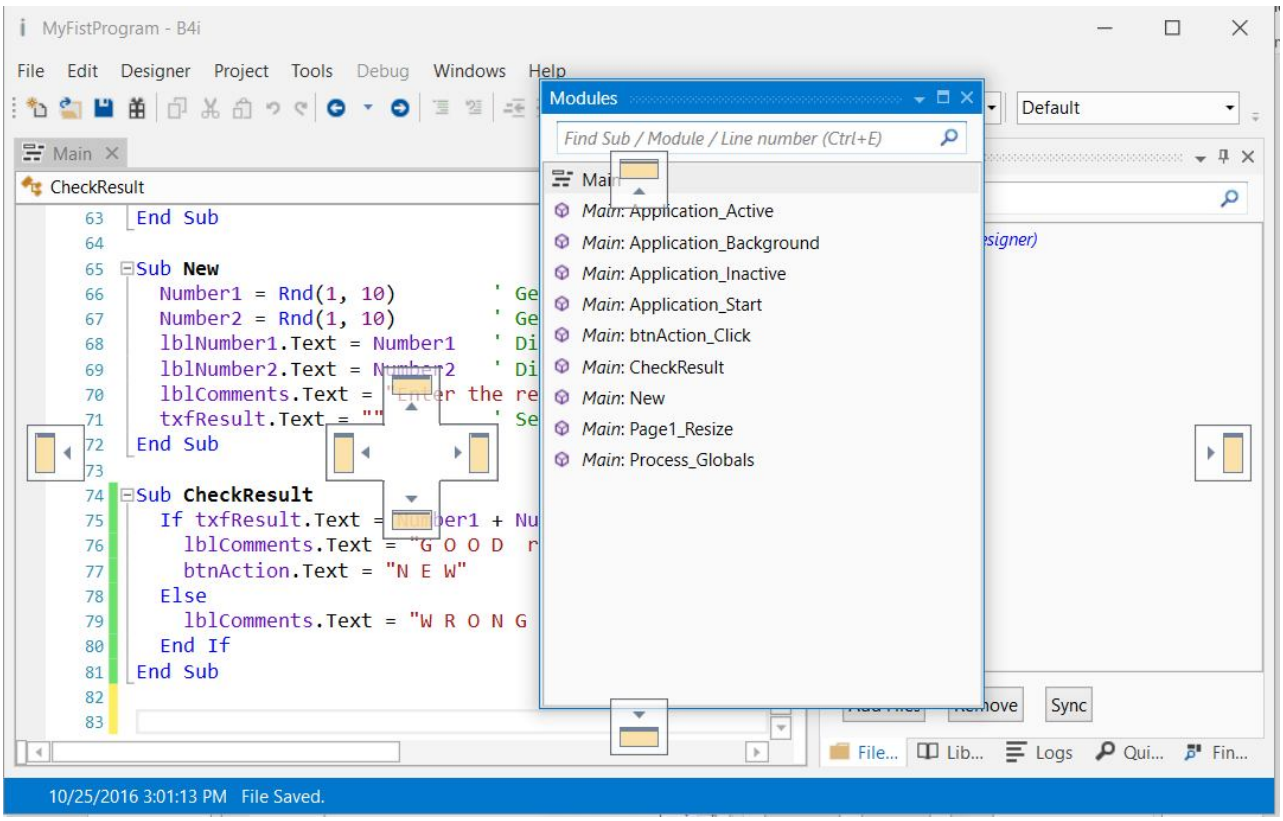
To dock it back to the Tab area click on Dock.



You can also click on a Tab and while maintaining the mouse down, move the Tab.



This will show you all the possible 'docking' areas.



Docking areas:

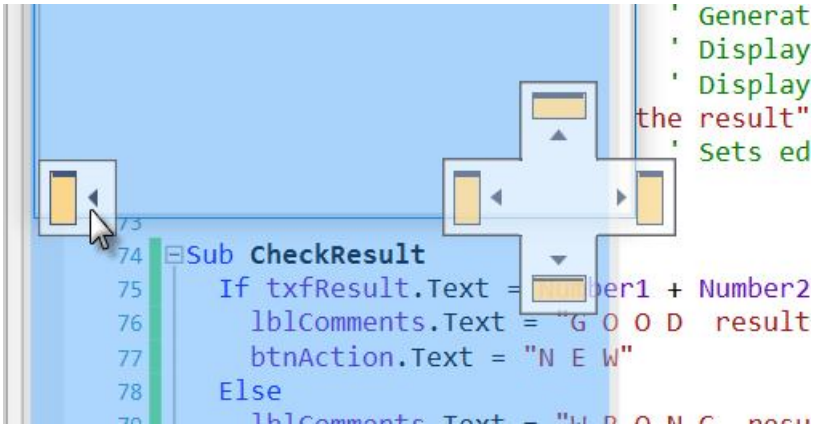

Top


Left

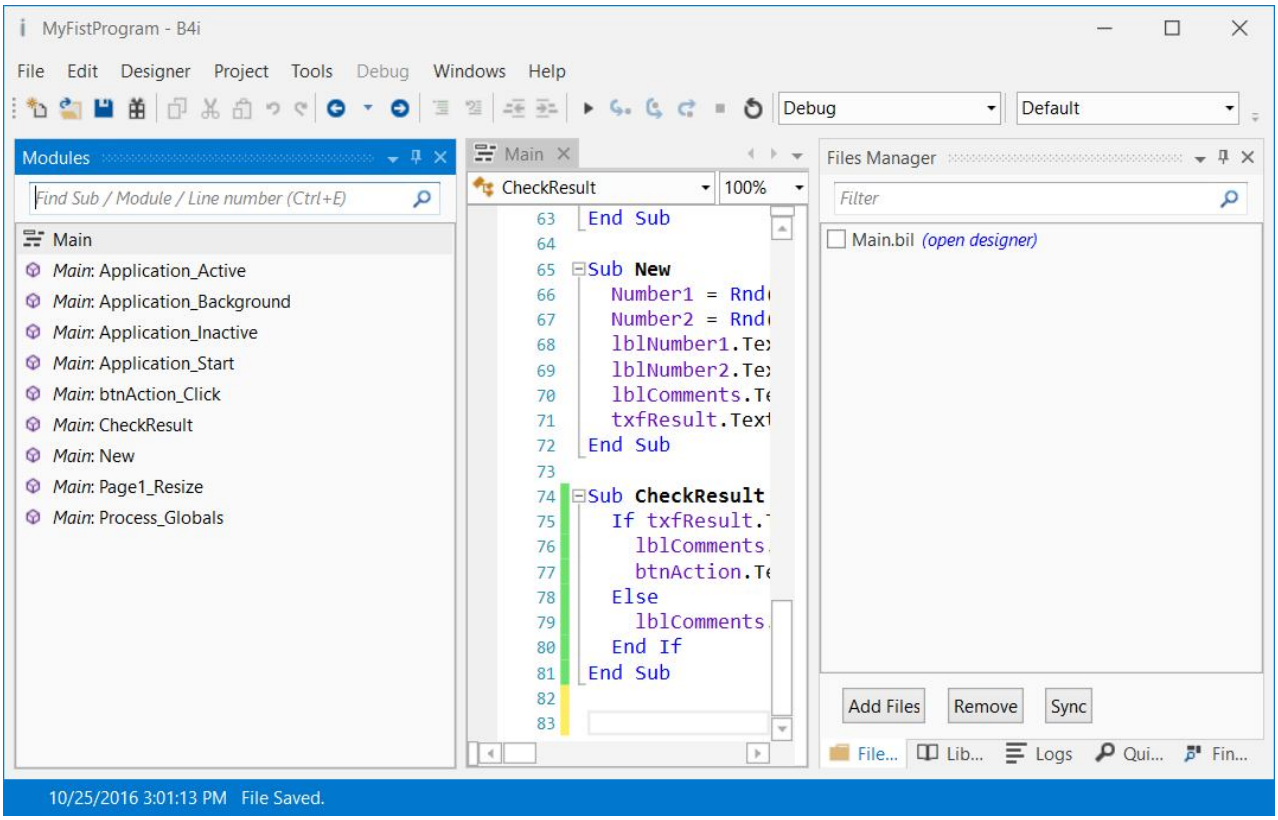

Right


Bottom

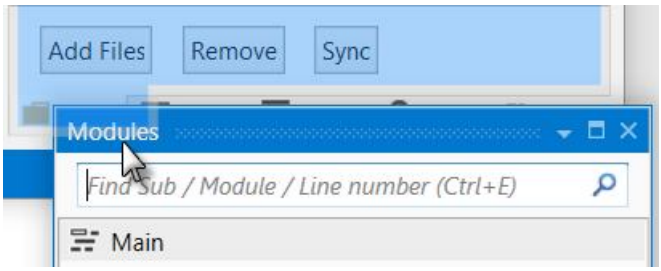
If you mouve the mouse onto one of the docking area symbol, the Tab window will be either on top, on the left, the right or on the bottom.



And the result.

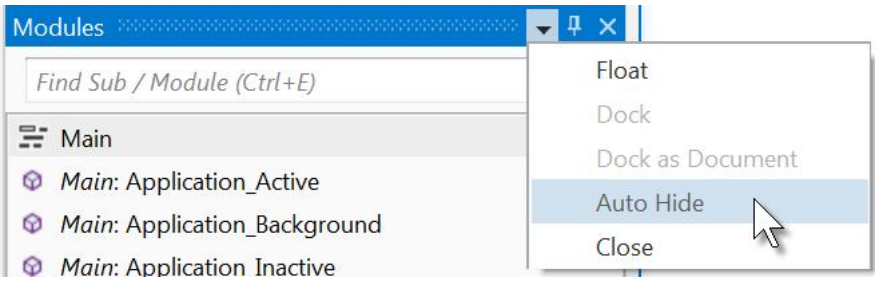


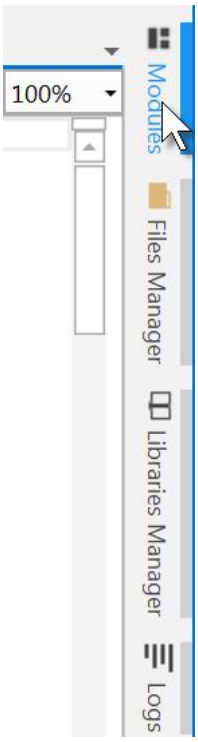
To bring it back to the Tabs, click on the window title and move it back to the Tabs.



4.3.3 Auto Hide

Click on in the title or click on Auto Hide in the Options.

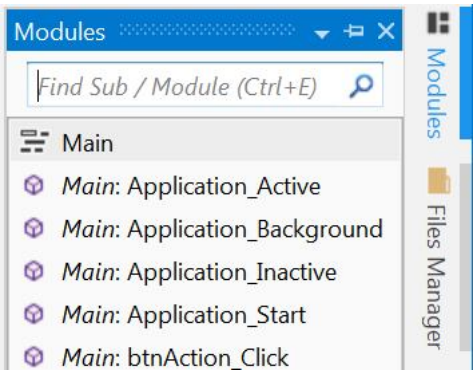




The Tabs move from the bottom of the screen vertically to the right side of the screen and the Tab window is hidden.

Hovering over a Tab highlights it in blue.

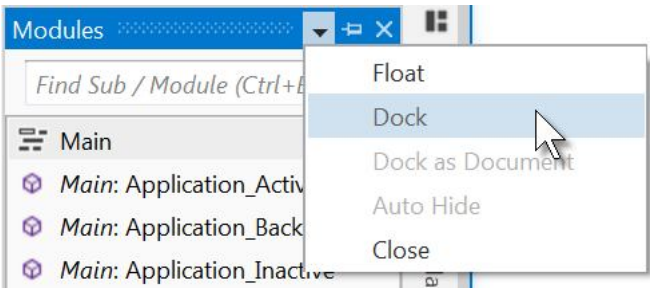
Click on a Tab to show it.



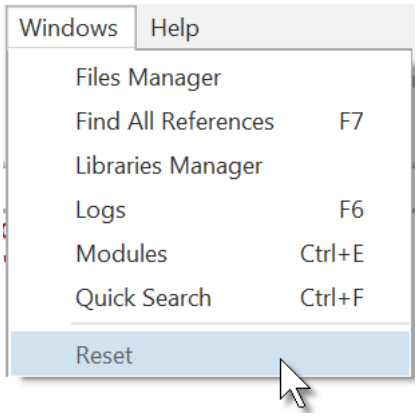
As soon as you click somewhere else in the IDE the Tab is hidden again.

To move the Tabs back to the lower right corner:

Click on Dock in the Options.



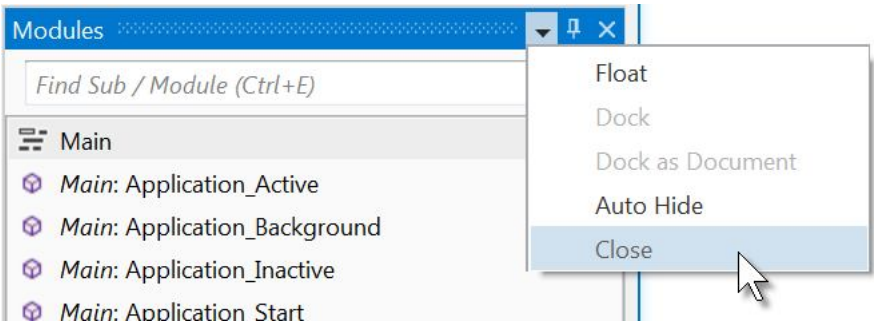
Or click on Reset in the IDE Windows menu.

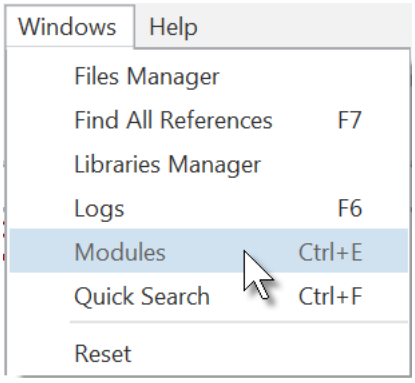


4.3.4 Close

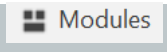
You can close a window, hide it.

Click on ✕ in the title or on Close in the Options.

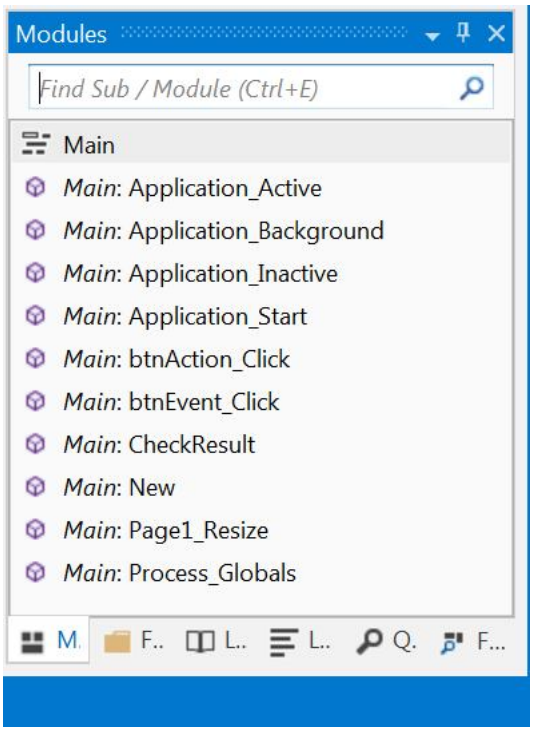


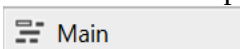


To show it again, in the Windows menu click on the module name you want to show, Modules in our example.

4.3.5 Modules and subroutines list 

All the modules of the project and all subroutines of the selected module are listed in the Modules window. The picture below has been reduced in height.



Module list on top. Only one module in the example.


Clicking on a module shows its code in the code area.

Subroutine list of the selected module.
Clicking on a subroutine shows its code in the middle of the code area.

Note the different icons:  module and  routine.

To show a hidden module, click on the module name in the module list.

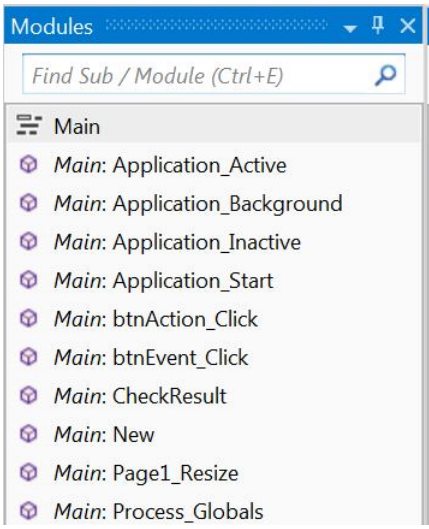
4.3.5.1 Find Sub Tool (Ctrl + E)

The *Find Sub / Module* function is a search engine, on the Top of the Modules Tab, to find subroutines or Modules with a given name or with a given part of the name.

You can press Ctrl + E in the code to select the Modules Tab with the *Find Sub / Module* function.

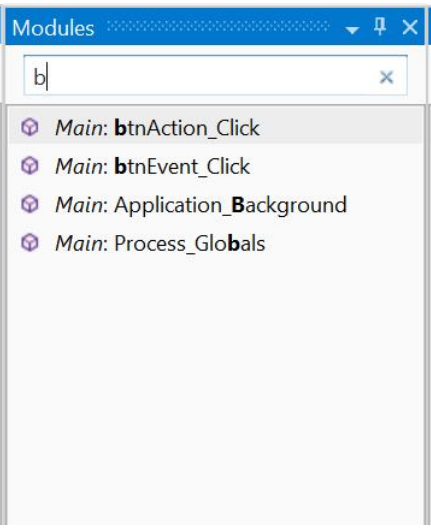
Example with the code of the SecondProgram example.

No text



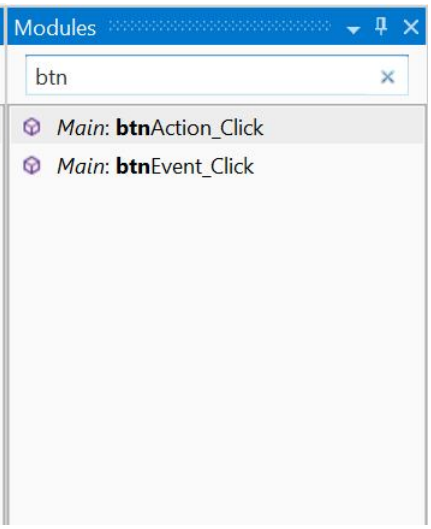
Shows all modules and all routines of the selected Module.

only the character ‘b’



Shows all modules and routines containing ‘b’.

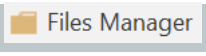
text ‘btn’



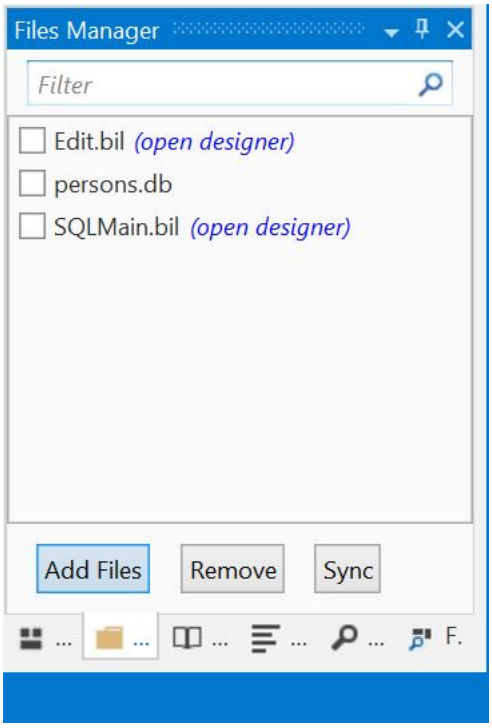
Shows all modules and routines containing ‘btn’.

Clicking on one item shows the code of the selected module or routine, even if it's in another module than the current one.

4.3.6 Files Manager

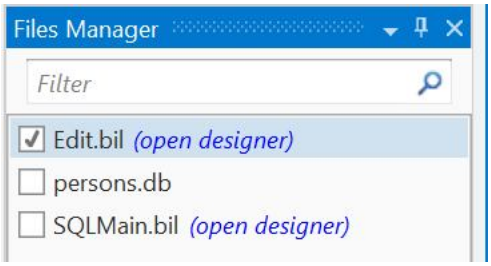


This window lists all the files that have been added to the project.
These files are saved in the 'Files' subfolder under your main project folder.
The files can be of any kind: layouts, images, texts, etc.



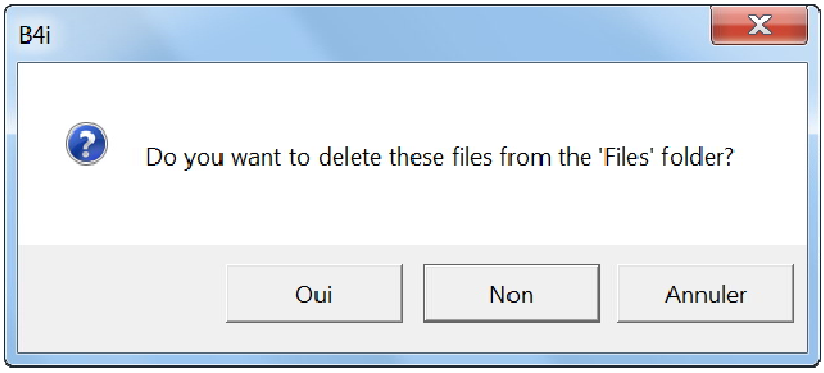
Click on Add Files to add files to the list.
The files in that subfolder can be accessed from your program by using the reference File.DirAssets.

Or click on Sync to add all the files from the projects Files folder into the File Tab.



Checking one or more files enables the Remove button.

Clicking on this button removes the selected files from the list and, if you want, from the Files folder of the project.



You are asked if you want to delete the files from the 'Files' folder.
Oui = Yes
Non = No
Annuler = Cancel

When you answer Yes make sure to have a copy of the files you remove, because they are removed from the Files folder, but not transferred to the Recycle Bin, which means that they are definitely lost if you don't make a copy.

See chapter Files for file handling.

On top of the Files Manager window you can filter the files list.

Enter ‘.bil’ to filter all layout files.

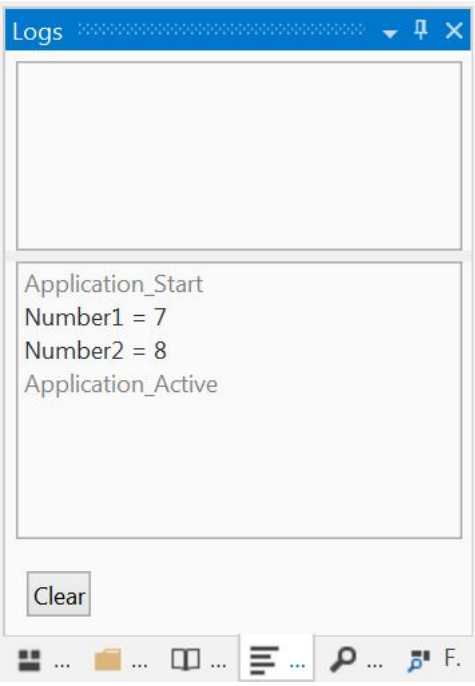
4.3.7 Logs

Display of Log comments generated by the program when it is running.

We add the two lines 51 and 53 in the program 'SecondProgram' in the 'New' routine.
The number of the lines may be different from yours.

```
68  Private Sub New
69    Number1 = Rnd(1, 10)          '
70    Log("Number1 = " & Number1)
71    Number2 = Rnd(1, 10)          '
72    Log("Number2 = " & Number2)
73    lblNumber1.Text = Number1     '
74    lblNumber2.Text = Number2     '
75    lblComments.Text = "Enter the
76    lblComments.Color = Colors.RG
77    lblResult.Text = ""           '
78    btn0.Visible = False
79  End Sub
```

Run the program.



In the Logs window we see the flow of the program.

The top area of the window shows Compile Warnings see next page.

In the lower area of the window we see the flow of the program.

Application_Start
Number1 = 7 First log message
Number2 = 8 Second log message
Application_Active

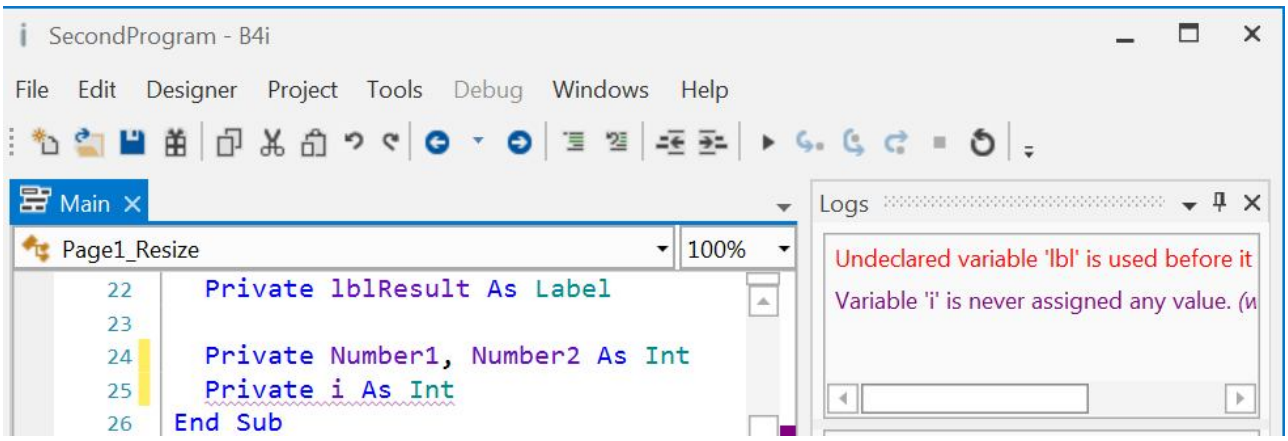
Click Clear to clear the Logs window.

4.3.7.1 Test Compile / Warnings

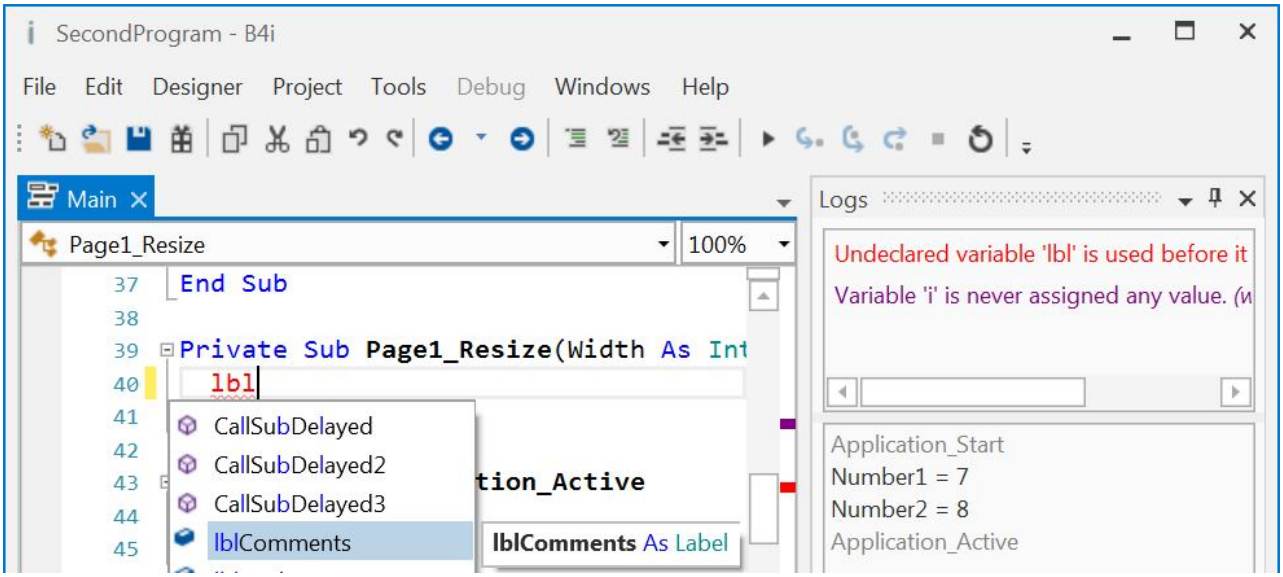
B4A includes a warning engine. The purpose of the warning engine is to find potential programming mistakes as soon as possible. The examples are from the Warnings project.

The compile-time warnings appear above the logs and in the code itself when hovering with the cursor above the code line.
The code lines which cause a warning are underlined like this `Private i As Int`.

Clicking on the warning in the list will take you to the relevant code.



The warning engine runs as soon as you type.



Typing for example 'lbl' at the beginning of a line shows immediately:

- lbl in red, because lbl was not yet declared.
- a warning Undeclared variable 'lbl' is used before it was assigned any value.
- the auto complete pop up window with suggestion containing the written characters.

4.3.7.1.1 Ignoring warnings

You, as the developer, can choose to ignore any warning. Adding an "ignore" comment will disable all the warnings for that specific line:

```
114 Private Sub Test
115
116 End Sub
```

```
114 Private Sub Test 'ignore
115
116 End Sub
```

You can also disable warnings from a specific type in the module by adding the #IgnoreWarning attribute in the Project Attributes regions.

For example, to disable warnings #10 and #12:

```
#Region  Project Attributes
   #ApplicationLabel: SecondProgram
   #Version: 1.0.0
   'Orientation possible values: Portrait, LandscapeLeft,
   #iPhoneOrientations: Portrait, LandscapeLeft, LandscapeRight
   #iPadOrientations: Portrait, LandscapeLeft, LandscapeRight
   #IgnoreWarnings: 10, 12
#End Region
```

You find the warning numbers at the end of each warning line.

4.3.7.1.2 List of warnings

List of warnings
1: Unreachable code detected.
2: Not all code paths return a value.
3: Return type (in Sub signature) should be set explicitly.
4: Return value is missing. Default value will be used instead.
5: Variable declaration type is missing. String type will be used.
6:
7: Object converted to String. This is probably a programming mistake.
8: Undeclared variable '{1}'.
9: Unused variable '{1}'.
10: Variable '{1}' is never assigned any value.
11: Variable '{1}' was not initialized.
12: Sub '{1}' is not used.
13: Variable '{1}' should be declared in Sub Process_Globals. ???
14: File '{1}' in Files folder was not added to the Files tab.\nYou should either delete it or add it to the project.\nYou can choose Tools - Clean unused files.
15: File '{1}' is not used.
16: Layout file '{1}' is not used. Are you missing a call to Page.RootPage.LoadLayout?
17: File '{1}' is missing from the Files tab.
18: TextSize value should not be scaled as it is scaled internally.
19: Empty Catch block. You should at least add Log(LastException.Message).
20: View '{1}' was added with the designer. You should not initialize it.
21: Cannot access view's dimension before it is added to its parent.
22: Types do not match.
23:
24: Accessing fields from other modules in Sub Process_Globals can be dangerous as the initialization order is not deterministic.
25: Sub '{1}' not found.
26:
27:
28:
29: This sub should only be used for variables declaration or assignments of primitive values.
30: Variable name is the same as a module name. This can cause problems during debugging.
32: Library ‘xxxx’ is not used.

1: Unreachable code detected.

There is some code which will never be executed.
This can happen if you have some code in a Sub after a Return statement.

2: Not all code paths return a value.

```
Sub Calc(Val1 As Double, Val2 As Double, Operation As String) As Double
   Select Operation
   Case "Add"
      Return (Val1 + Val2)
   Case "Sub"
      Return (Val1 - Val2)
   Case "Mult"
      Return (Val1 * Val2)
   Case "Div"

   End Select
End Sub
```

In the `Case "Div"` path no value is returned!

3: Return type (in Sub signature) should be set explicitly.

Wrong code
```
Sub Calc(Val1 As Double, Val2 As Double, Operation As String)
```

Correct code
```
Sub Calc(Val1 As Double, Val2 As Double, Operation As String) As Double
```

The return type must be declared!

4: Return value is missing. Default value will be used instead.

Wrong code
```
Private Sub CalcSum(Val1 As Double, Val2 As Double) As Double
   Dim Sum As Double

   Sum = Val1 + Val2
   Return
End Sub
```

Correct code
```
Private Sub CalcSum(Val1 As Double, Val2 As Double) As Double
   Dim Sum As Double

   Sum = Val1 + Val2
   Return Sum
End Sub
```

5: Variable declaration type is missing. String type will be used.

Wrong code
```
Private Sub Calc(Val1, Val2 As Double, Operation As String) As Double
```

Correct code
```
Private Sub Calc(Val1 As Double, Val2 As Double, Operation As String) As Double
```

In sub declarations each variable needs its own type declaration.
But in Dim declarations it's allowed, in the line below both variables are Doubles:
```
Dim Val1, Val2 As Double
```

6: The following value misses screen units ('dip' or %x / %y): {1}.
Not used in B4i. Is used in B4A.

7: Object converted to String. This is probably a programming mistake.

8: Undeclared variable '{1}'.

Wrong code
```
Private Sub SetHeight
   h = 10
End Sub
```

Correct code
```
Private Sub SetHeight
   Dim h As Int
   h = 10
End Sub
```

The variable h was not declared. You see it also with the red color.

9: Unused variable '{1}'.

```
Private Sub SetHeight
   Dim h As Int
   h = 10
End Sub
```

This warning tells that the variable h is not used.
It is declared and assigned a value, but it is not used !

This code gives no warning because variable h is used:
```
Private Sub SetHeight
   Dim h As Int
   h = 10
   lblTest.Height = h
End Sub
```

10: Variable '{1}' is never assigned any value.

```
S Private Sub Test
   Dim h As Int

End Sub
```
This warning shows that the variable h is declared but not assigned any value.
Correct code see above.

11: Variable '{1}' was not initialized.

Wrong code
```
Dim lst As List
lst.Add("Test1")
```

Correct code
```
Dim lst As List
lst.Initialize
lst.Add("Test1")
```

Variables (objects) like List or Map must be initialized before they can be used.
Views added by code must also be initialized before they can be added to a parent view.

12: Sub '{1}' is not used.

This warning is displayed if a Sub routine is never used.

13: Variable '{1}' should be declared in Sub Process_Globals.
Not used in B4i, no Global routine!

Certain objects like Timers and GPS must be declared in Process_Globals.

```
Private Sub Process_Globals
   Dim Timer1 As Timer
   Dim GPS1 As GPS
```

14: File '{1}' in Files folder was not added to the Files tab.

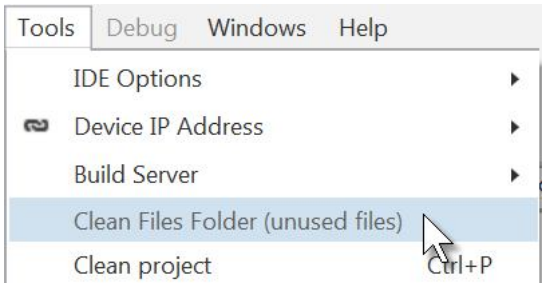
You are using a file which is in the Files folder, but was not added to the Files tab.
You should:
- Make a backup copy.
- Delete it from the Files subfolder.
- Add it to the project in the Files tab.
- Use Clean Files Folder (unused files) in the Tools menu.

15: File '{1}' is not used.

You have files in the Files folder that are not used.
You should remove them from the Files folder.
Or you can clean the Files folder from within the Tools menu (see above).

16: Layout file '{1}' is not used. Are you missing a call to Page.RootPanel.LoadLayout?

You have a layout file in the Files folder that is not used.
You should add LoadLayout or you can remove the layout file from the Files folder.



Or you can clean the Files folder in the Tools menu.

17: File '{1}' is missing from the Files tab.

The given file is in the Files tab but is missing in the Files folder. You should add it.
See chapter Files.

18: TextSize value should not be scaled as it is scaled internally.

Not used. Is used in B4A.

19: Empty Catch block. You should at least add Log(LastException.Message).

Wrong code
```
Try
   imvImage.Bitmap = LoadBitmap(File.DirAssets, "image.jpg")
Catch

End Try
```

Correct code
```
Try
   imvImage.Bitmap = LoadBitmap(File.DirAssets, "image.jpg")
Catch
   Log(LastException.Message)
End Try
```

It is recommended to add at least `Log(LastException.Message)` in the Catch block instead of leaving it empty.

20: View '{1}' was added with the designer. You should not initialize it.

A View defined with the Designer in a layout file must not be initialized!
Only views added by code need to be initialized.

21: Cannot access view's dimension before it is added to its parent.

You must add a view to a parent view before you can access its dimensions.
When you add a view by code its dimensions are defined when you add it with AddView.

22: Types do not match.

23: Modal dialogs are not allowed in Sub Activity_Pause. It will be ignored.

Not used. Is used in B4A.

24: Accessing fields from other modules in Sub Process_Globals can be dangerous as the initialization order is not deterministic.

25: Sub '{1}' not found.

26: Not used.
27: Not used.
28: Not used.

29: This sub should only be used for variables declaration or assignments of primitive values.

30: Variable name is the same as a module name. This can cause problems during debugging.

32: Library ‘xxxx’ is not used.

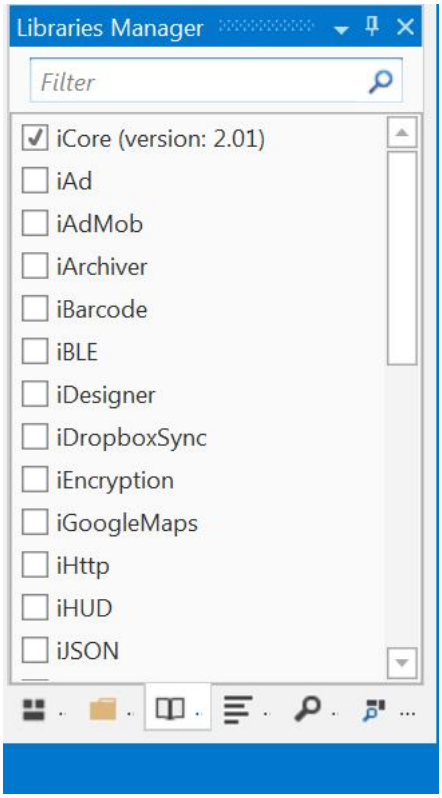
Remove the unused library.

4.3.8 Libraries Manager

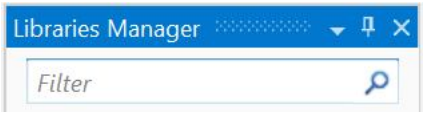
The ‘Libraries Manager’ tab contains a list of the available libraries that can be used in the project.

All B4i libraries have an “i” prefix, like iCore!

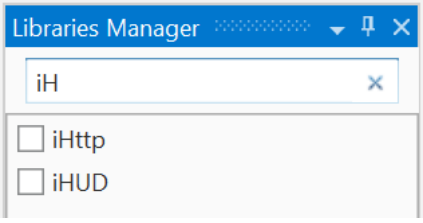
Check the libraries you need for your project.
Make sure that you have the latest version of the libraries.



In the IDE, in the bottom right corner click on Libs.



On the top of the tab you find a field to filter the libraries.



Enter ‘iH’ in the field and you get all libraries beginning with iH.
Note that in B4i all libraries have the prefix ‘i’ ?

The list of all additional libraries can be found here: Additional Libraries.

The documentation for libraries can be found here:
B4i Documentation

Look also at chapter Libraries.

4.3.9 Quick Search

Quick Search allows to search for any text occurrences in the code of the whole project.
Examples with the SecondProgram code.

Several possibilities to select the Quick Search function:

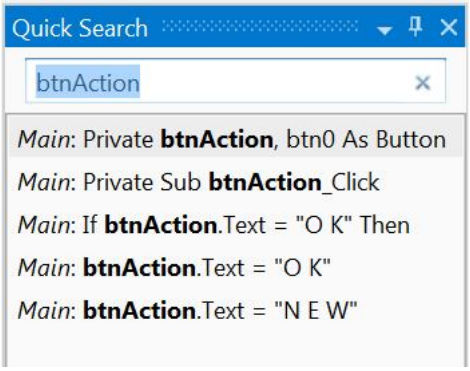
- Ctrl + F, the easiest and most efficient way.
- Click on the Quick Search Tab in the lower right corner of the IDE.
- Click on Quick Search Ctrl+F in the Edit menu.

Example:

```
Private btnAction, btn0 As Button
Private lblMathSign As Label
Private lblComments As Label
```

In the code double click on `btnAction` to select it and press Ctrl + F.

You get the window below in the Tab area.

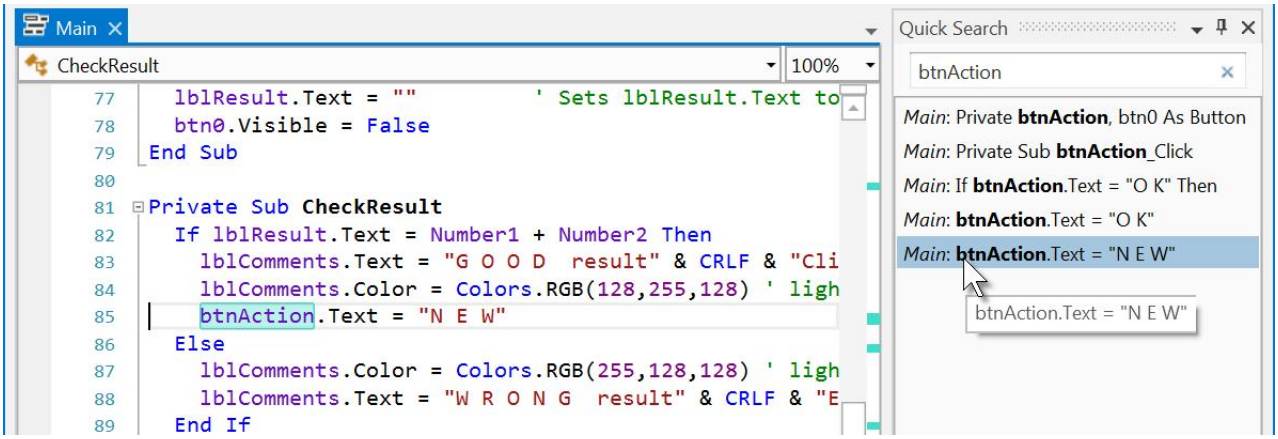


The list shows the occurrences in all Modules.

In each line you find the Module name and the line content.

Like in *Main*: If **btnAction**.Text = "O K" Then

Clicking on a line in the list moves the cursor directly to the selected occurrence in the code.

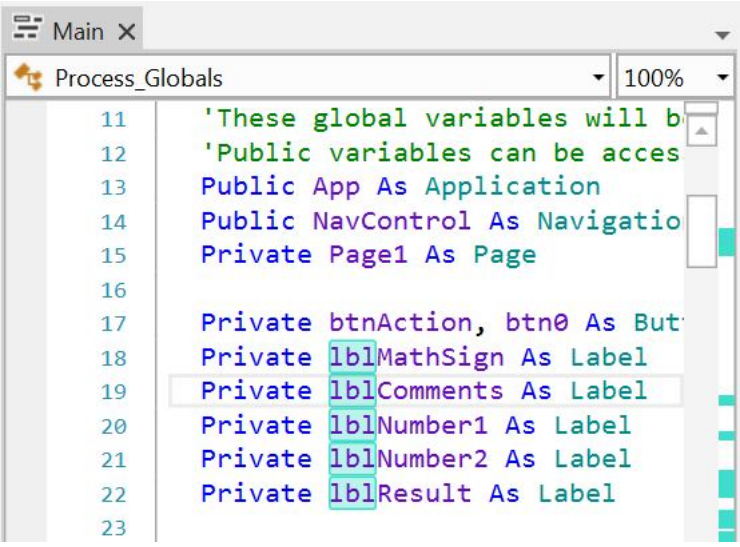


To remove the selection click on × on the top right corner of the Quick Search window.

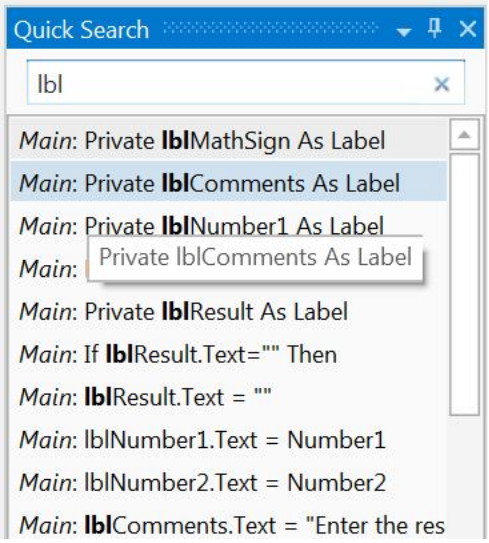
You can also enter any text in the search field:

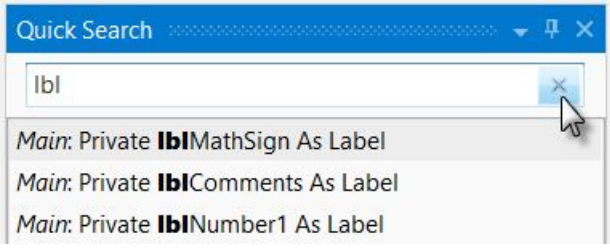
A as an example, enter *lbl* in the Search field and you get the window below where you find all lines containing the text you entered, *lbl* in this example.

The search text is highlighted in all code lines containing this text.

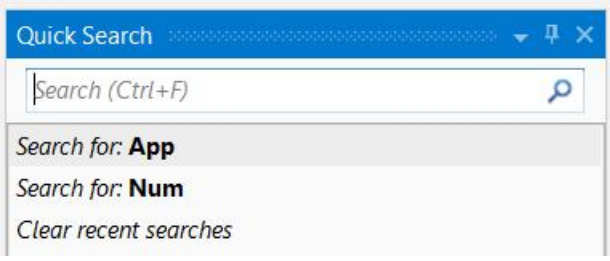


Clicking on one of the lines in the list jumps directly to this line in the IDE.

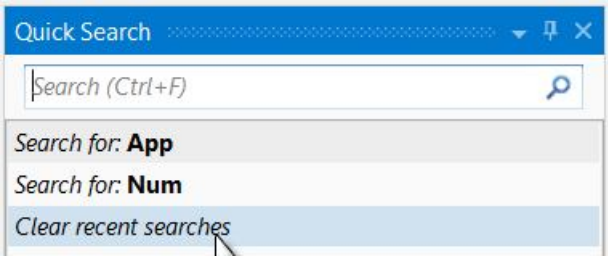




Click on  to remove a search.



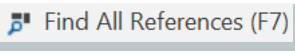
You will see a list of the last searches.



Click on *Clear recent searches* to remove all recent searches.

Items are added to the recent items when:
1. You select one of the results or click enter which selects the first result.
2. You select text in your code and click on Ctrl + F to search for it.

4.3.10 Find All References



This is a search engine to find all references for a given object (view, variable).

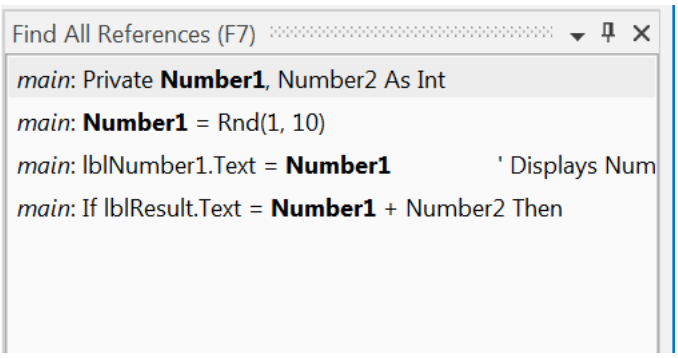
Click on the Find All References (F7) Tab or press F7 to get the screen below showing a list of all code lines with the selected reference or the first object in the current line.

Example with the code of SecondProgram.

Select in the code in line 69 `Number1`.

```
68  Private Sub New
69      Number1 = Rnd(1, 10)          ' Generates a random number
70      Number2 = Rnd(1, 10)          ' Generates a random number
71      lblNumber1.Text = Number1     ' Displays Number1 in label
72      lblNumber2.Text = Number2     ' Displays Number2 in label
```

Click on Find All References (F7) or press F7 and you get the list below with all code lines containing the selected object.



Clicking on a line in the list shows that line in the middle of the IDE code area.

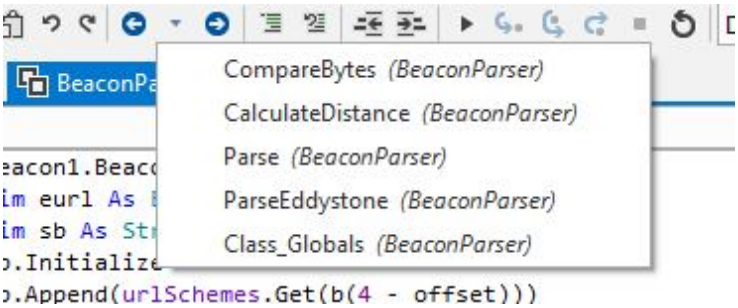
4.4 Navigating in the IDE

4.4.1 Alt + Left / Alt + Right Move backwards and forwards

Moves backwards and forwards based on the navigation stack. This is useful to jump back and forth between the last recent subs.

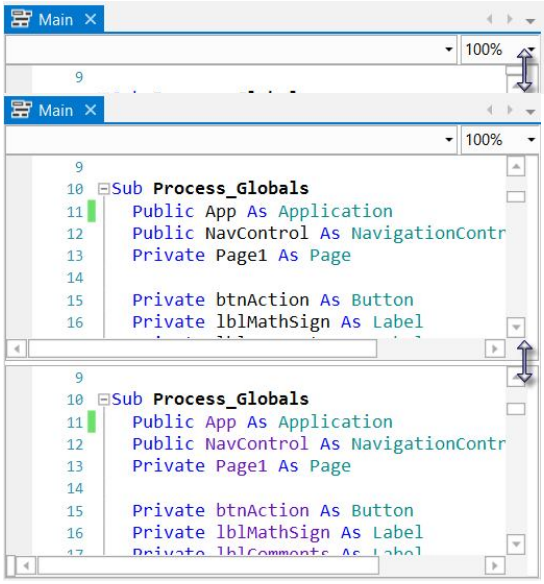
4.4.2 Alt + N Navigation stack menu

Opens the navigation stack menu. You can then choose the location with the up and down keys.

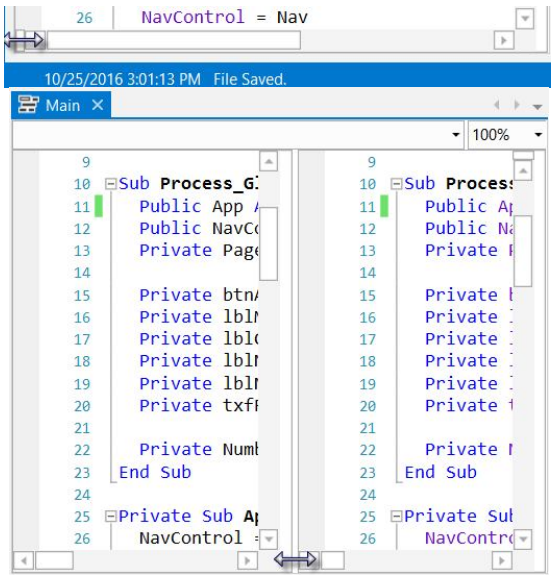


4.4.3 Split the screen

If you are working on two locations in the same module then you can split the code editor (it can be split again vertically):

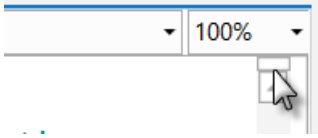


Horizontally



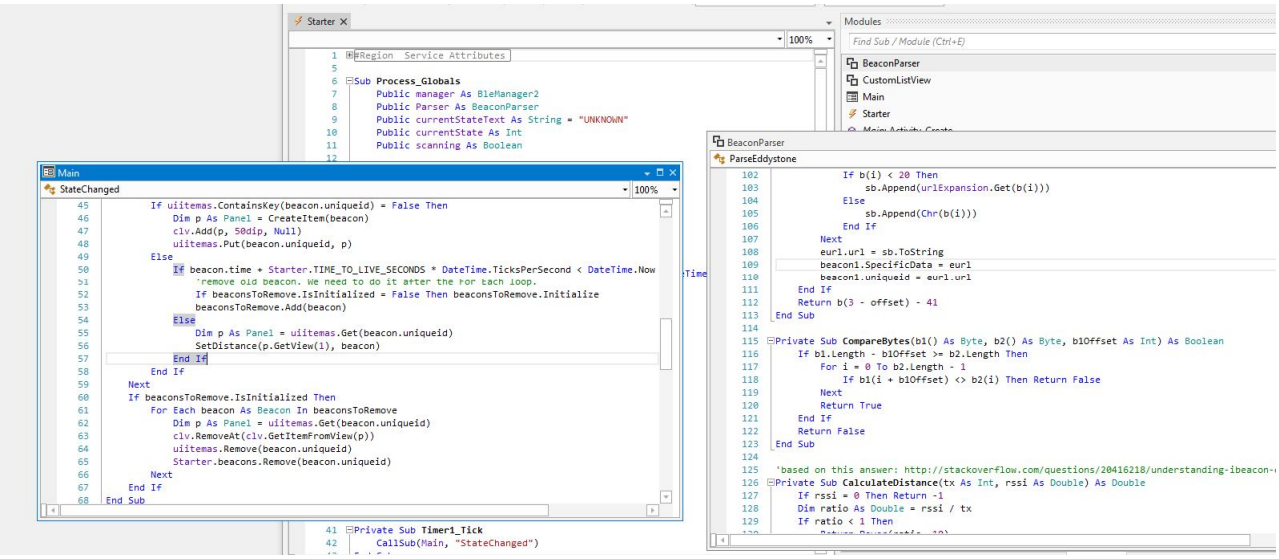
Vertically

You can also double click on the small rectangles to split the screen.



4.4.4 Multiple windows

If you are working with multiple modules you can move the modules out of the main IDE as separate windows.



4.4.5 Ctrl + E Search for sub or module

Ctrl + E - searches for sub or module. Very useful when working with large projects.

4.4.6 Ctrl + Click on any sub or variable

Ctrl + Click on any sub or variable to jump to the declaration location.

4.4.7 F7 - Find all references

Not exactly related to navigation but is also useful when working with large projects.
Details in Find all references.

4.4.8 Ctrl + F Quick Search

Ctrl + F - Index based quick search. Details in Quick Search.

5 Screen sizes and resolutions

5.1 Coordinates

Coordinate values are different in B4i from those in B4A.

Extract from the iOS documentation:

Points Versus Pixels

In iOS there is a distinction between the coordinates you specify in your drawing code and the pixels of the underlying device. When using native drawing technologies such as Quartz, UIKit, and Core Animation, the drawing coordinate space and the view's coordinate space are both **logical coordinate spaces**, with distances measured in **points**. These logical coordinate systems are decoupled from the device coordinate space used by the system frameworks to manage the pixels onscreen.

The system automatically maps points in the view's coordinate space to pixels in the device coordinate space, but this mapping is not always one-to-one. This behavior leads to an important fact that you should always remember:

One point does not necessarily correspond to one physical pixel.

The purpose of using points (and the logical coordinate system) is to provide a consistent size of output that is device independent. For most purposes, the actual size of a point is irrelevant. The goal of points is to provide a relatively consistent scale that you can use in your code to specify the size and position of views and rendered content. How points are actually mapped to pixels is a detail that is handled by the system frameworks. For example, on a device with a high-resolution screen, a line that is one point wide may actually result in a line that is two physical pixels wide. The result is that if you draw the same content on two similar devices, with only one of them having a high-resolution screen, the content appears to be about the same size on both devices.

B4i **B4A**
1 Point = **1 dip** (**d**evice **i**ndependent **p**ixel)

	B4i	**B4A**
Coordinates	Point	Pixel
View dimensions	Point	Pixel
Drawing coordinates	Point	Pixel
Adding a view	`AddView(10, 10, 150, 50)`	`AddView(10dip, 10dip, 150dip, 50dip)`
Standard dpi (dots per inch)	160	16'
`GetDeviceLayoutValues` Example 320 dpi screen		
Scale	Always 1	2
NonnormalizedScale	2	---
Width	Point	Pixel
Height	Point	Pixel
1 Pixel	1 Point / NonnormalizedScale	---
1 Point	---	1 Pixel * Scale

5.2 Screen sizes

5.2.1 Screens

The current screen sizes are, only devices supported by iOS 7+ are shown:

Models	Screen size pixels	Scale	Screen size points	Width /Height ratio
iPhone 6+	1920 x 1080	3	640 x 360	16 / 9
iPhone 6	1334 x 750	2	667 x 375	16 / 9
iPhone 5	1136 x 640	2	568 x 320	16 / 9
iPhone 4	960 x 640	2	480 x 320	3 / 2
iPad / iPad 2 / iPad Mini	1024 x 768	1	1024 x 768	4 / 3
iPad Air / iPad Mini Retina	2048 x 1536	2	2048 x 1536	4 / 3

5.2.2 Sizes of different objects

Below you find the sizes of different objects and icons.

Different sizes	Points
StatusBar height	20
NavigationBar / ToolBar height	44
NavBar / ToolBar icon	22 x 22
TabBar height	49
TabBar icon	25 x 25

5.3 Icon sizes Table taken from the Apple documentation.

Link to the table.

Table 39-1 Size (in pixels) of custom icons and images

Asset	iPhone 6 Plus (@3x)	iPhone 6 and iPhone 5 (@2x)	iPhone 4s (@2x)	iPad and iPad mini (@2x)	iPad 2 and iPad mini (@1x)
App icon (**required** for all apps)	180 x 180	120 x 120	120 x 120	152 x 152	76 x 76
App icon for the App Store (**required** for all apps)	1024 x 1024	1024 x 1024	1024 x 1024	1024 x 1024	1024 x 1024
Launch file or image (**required** for all apps)	Use a launch file (see Launch Images)	For iPhone 6, use a launch file (see Launch Images) For iPhone 5, 640 x 1136	640 x 960	1536 x 2048 (portrait) 2048 x 1536 (landscape)	768 x 1024 (portrait) 1024 x 768 (landscape)
Spotlight search results icon (recommended)	120 x 120	80 x 80	80 x 80	80 x 80	40 x 40
Settings icon (recommended)	87 x 87	58 x 58	58 x 58	58 x 58	29 x 29
Toolbar and navigation bar icon (optional)	About 66 x 66	About 44 x 44	About 44 x 44	About 44 x 44	About 22 x 22
Tab bar icon (optional)	About 75 x 75 (maximum: 144 x 96)	About 50 x 50 (maximum: 96 x 64)	About 50 x 50 (maximum: 96 x 64)	About 50 x 50 (maximum: 96 x 64)	About 25 x 25 (maximum: 48 x 32)
Default Newsstand cover icon for the App Store (**required** for Newsstand apps)	At least 1024 pixels on the longest edge	At least 1024 pixels on the longest edge	At least 1024 pixels on the longest edge	At least 1024 pixels on the longest edge	At least 512 pixels on the longest edge
Web clip icon (recommended for web apps and websites)	180 x 180	120 x 120	120 x 120	152 x 152	76 x 76

5.3.1 File names for icons

You should provide different icon files for the different screen scales.

File names for the different scales.

- myicon.png generic name
- myicon@2x.png file name for scale 2 images.
- myicon@3x.png file name for scale 3 images.

In an Initialize method you must use the generic file name. iOS selects automatically the correct file according to the device scale.

Example for a TabBar item.
You must use only `btnCH0.png` and not `btnCH0@2x.png` nor `btnCH0@3x.png`.

```
' Define a button with custom bitmaps
Dim tbi As TabBarItem          ' define a new TabBarItem
' define a custom TabBarItem with custom icons
tbi.Initialize("Countries", LoadBitmap(File.DirAssets, "btnCH0.png"),
LoadBitmap(File.DirAssets, "btnCH1.png"))
```

5.3.2 Application icons

To cover all screen sizes 5 images like below must be provided.
The black boarders are not part of the images they were added to better delimit the images.
Don't add rounded corners, iOS does it automatically.

					
Pixels	40 x 40	60 x 60	76 x 76	120 x 120	152 x 152
Filename	icon-40.png	icon-60.png	icon-76.png	icon-60@2x.png	icon-76@2x.png

And the image on a device (iPhone 6): 

The images above are those from Erels' Coordinates application.

6 The Visual Designer

Designing layouts is a major concern for developers.
A well organized and nice looking user interface makes a program being accepted immediately by the users or not, and this on different devices with different screen sizes.
Most users, when they look at a new application, decide in the first minutes if they will go further or not! Me too, when I download an application and there are several with the same purpose, the first impression is crucial. If I don't like the layout I don't keep it.
You should have a look at Apples' guidelines about how to design UI Design Basics.

For the navigation between different pages you should use the standard iOS objects instead of reinventing the wheel. Users are used to them and feel directly 'at home'!
These are explained in the User Interfaces chapter.

It's up to you to define what layout you want, what you want to display at the same time and how you want to navigate through the different displays.
In some cases it might be better to have one or two layouts (portrait and / or landscape) for iPhones and one or two layouts for iPads. Use as little layout variants as needed and use the different tool to adapt them to fit the different screen sizes.
As iPads have bigger screens than phones it could be interesting to display more information on iPads than on iPhones.
Depending on the application, it could be interesting to display one panel on pages in portrait on iPhones and display two panels side by side on a same page on iPads.

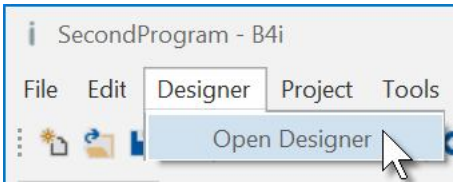
There are no general rules nor templates for user interfaces. They depend on the kind of application, the kind of information to display on what screen size, the number of different pages depending on the screen size and the information, etc.

Several tools are at your disposal to design the layouts, these are explained in the following chapters.

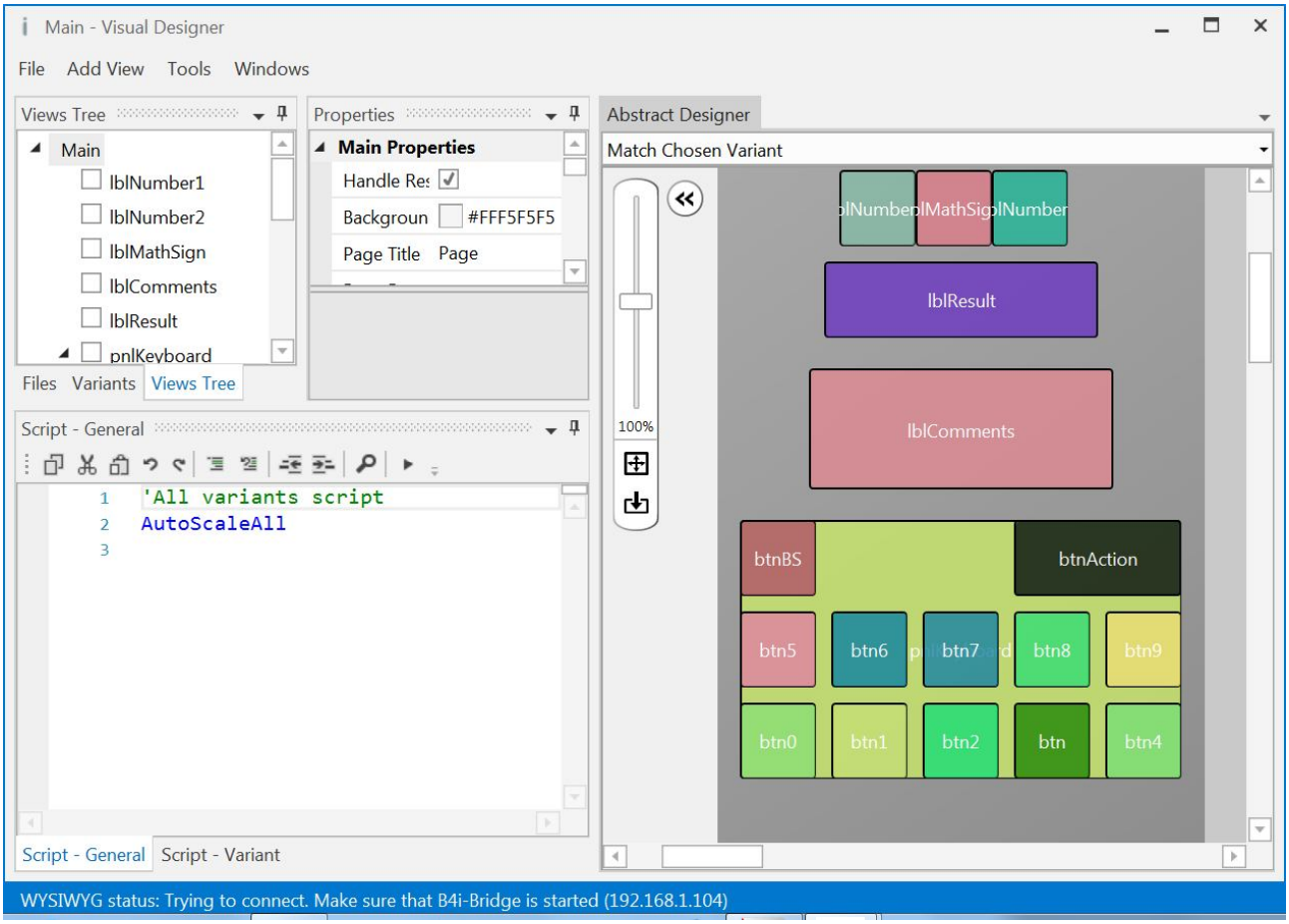
- The Visual Designer.
- The Abstract Designer.
- Anchors.
- Designer Scripts.
- AutoScale.

The Designer allows generating layouts with the Abstract Designer and / or with a real device.

Launch the Designer in the IDE Menu Designer.

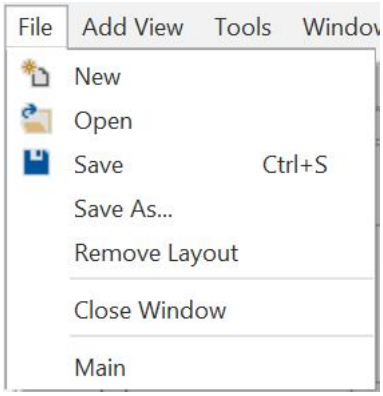


The default Visual Designer looks like this, the layout in the Abstract Designer is from the SecondProgram project.



6.1 The menu

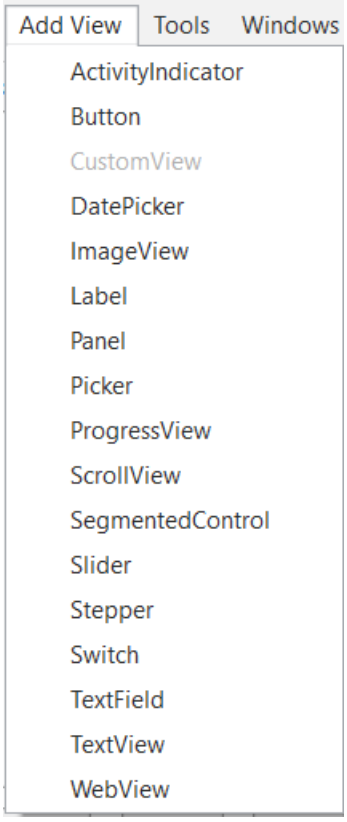
6.1.1 File menu



New	Opens a new empty layout.
Open	Opens an existing layout.
Save	Saves the current layout.
Save As…	Saves the current layout with a new name.
Remove Layout	Removes the layout from the Files directory.
Close Window	Closes the Visual Designer
Main	Layout file list, in this case only one file, 'Main'.

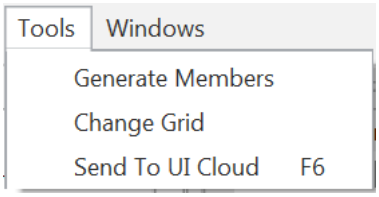
6.1.2 AddView menu

This menu allows you to select the view you want to add to the current layout.



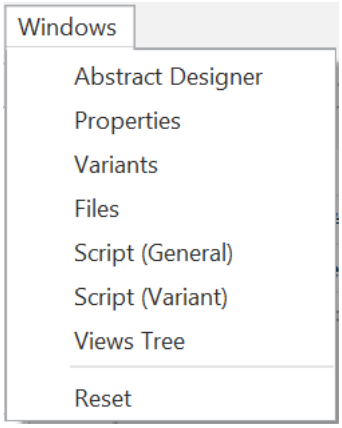
ActivityIndicator	adds an ActivityIndicator
Button	adds a Button
CustomView	adds a CustomView if there are any.
ImageView	adds an ImageView
Label	adds a Label
Panel	adds a Panel
Picker	adds a Picker
ProgressView	adds a ProgressView
ScrollView	adds a ScrollView
SegmentedControl	adds a SegmentedControl
Slider	adds a Slider
Stepper	adds a Stepper
Switch	adds a Switch
TextField	adds a TextField
TextView	adds a TextView
WebView	adds a WebView

6.1.3 The Tools menu



Generate Members	Members generator
Change Grid	Allows to change the grid size
Send To UI Cloud	

6.1.4 Windows menu

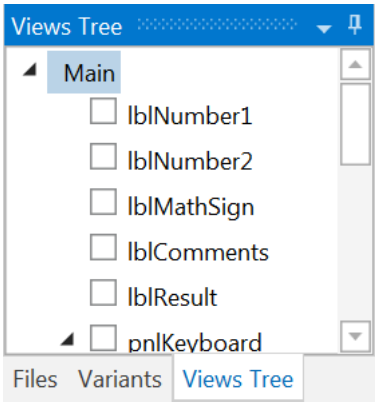


Abstract Designer	Shows the Abstract Designer window.
Properties	Shows the Properties window.
Variants	Shows the Variants window.
Files	Shows the Files window.
Script (General)	Shows the Script (General) window.
Script (Variant)	Shows the Script (Variant) window.
Views Tree	Shows the Views window.
Reset	Resets the Visual Designer layout to the default layout.

6.2 Visual Designer Windows

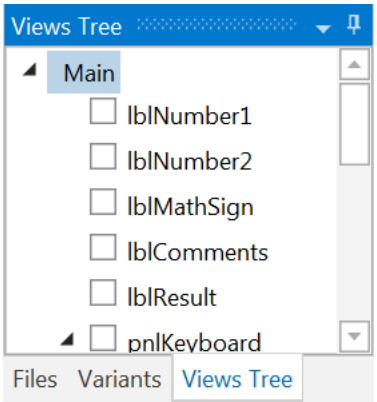
The Visual Designer is composed of different windows.

6.2.1 Views windows Views Tree / Files / Variants



In this Window three windows are combined:
Files, Variants and Views Tree.

6.2.1.1 Views Tree window



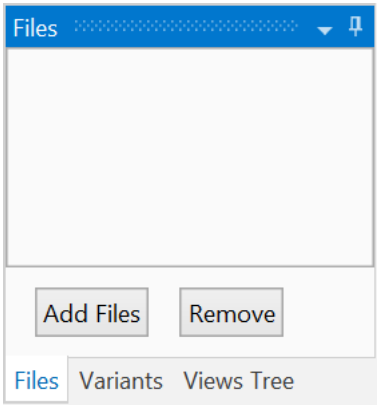
Shows all views of the selected layout in a tree.

When you select a view in the list, all the properties of the selected view are displayed in the Properties window.

You can select several Views at the same time and change common properties.

The selected views are highlighted in the Abstract Designer.

6.2.1.2 Files Window

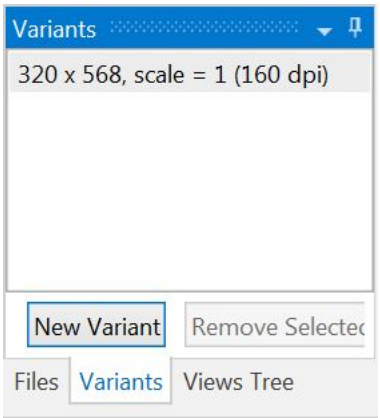


Used to add or remove files to the Visual Designer, mainly image files.

File handling is explained in the Image Files chapter.

The files you add are copied to the Files folder of the project and can be accessed in the code in the File.DirAssets folder.

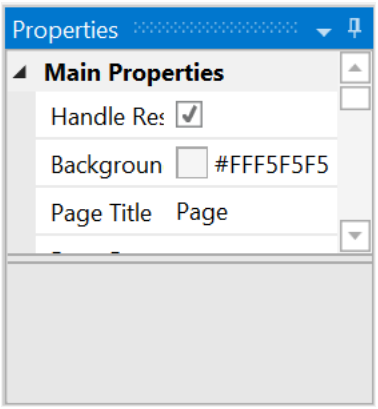
6.2.1.3 Variants window



Used to add and remove layout variants.

Layout variants are explained in the Layout variants chapter.

6.2.2 Properties window



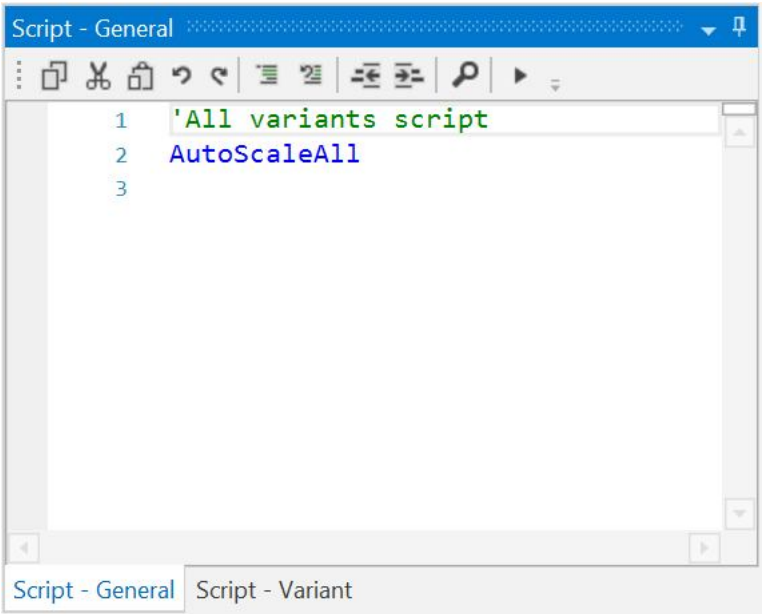
The Properties window shows all properties of the selected View.

The Properties are explained in the Properties list chapter.

6.2.3 Script (General) / (Variant) windows

In the Scrip windows you can add code to position and resize Views.
Two windows are available:

- **Script - General** Code valid for all layout variants.
- **Script - Variant** Specific code for the selected variant.

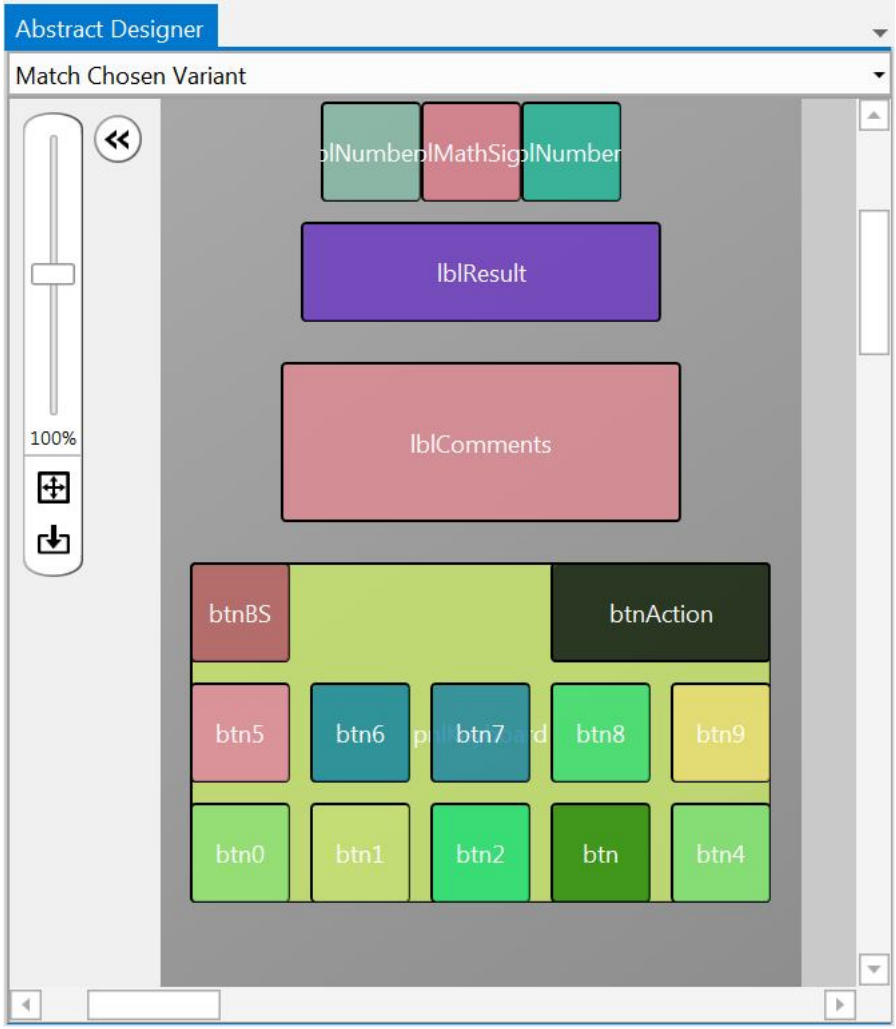


Script code is explained in the Designer Scripts chapter.

6.2.4 Abstract Designer window

The Abstract Designer allows to select, position and resize Views.
It is not a WYSIWYG Designer, for this you need to connect a real device or an Emulator.

The displayed layout in the picture below is from the SecondProgram project.



The Abstract Designer is explained in detail in the Abstract Designer chapter.

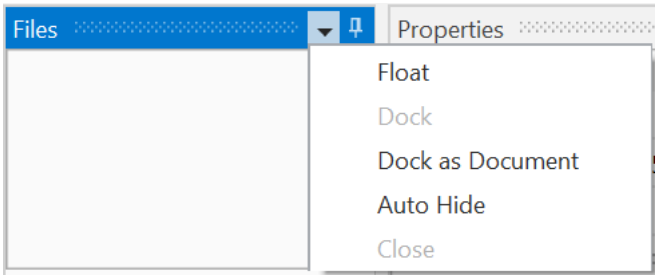
6.3 Floating windows

You can define your own Visual Designer layout, rearrange the windows in size and position, docked or floating.

On the top right of each window two icons allow you to manage the behavior of this window.

Options.

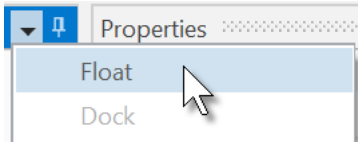
Example with the Files window:



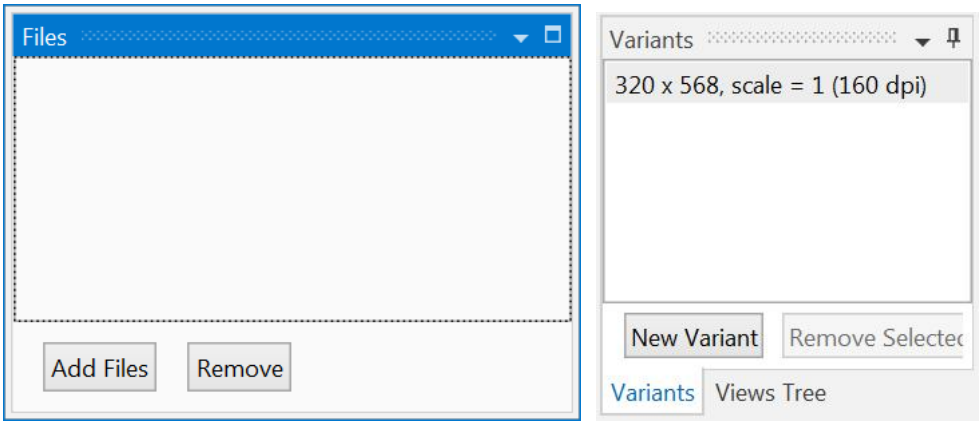
Float sets the window to Float, independent of the Visual Designer window.

Dock as Document.
Auto Hide.

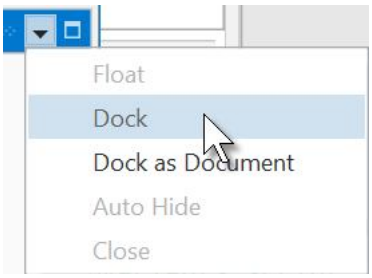
6.3.1 Float

Click on 

The Files windows is independent from the Visual Designer and is removed from the Views window. You can move it where ever you want on the screen.

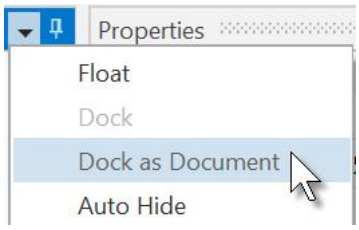


6.3.2 Dock

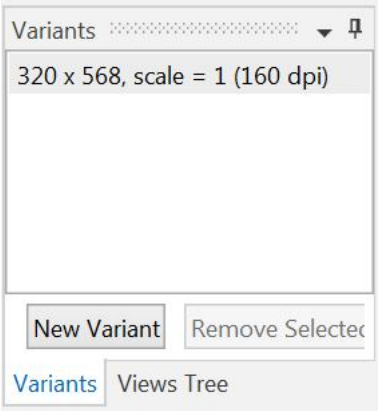
In a floating window click on 

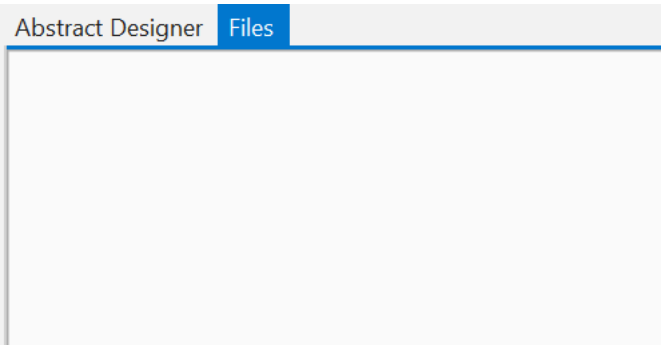
The window is moved back to the Views window.

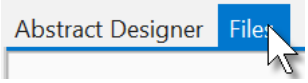
6.3.3 Dock as Document

Click on 

The window is removed from its parent window and added to the Abstract Designer window.

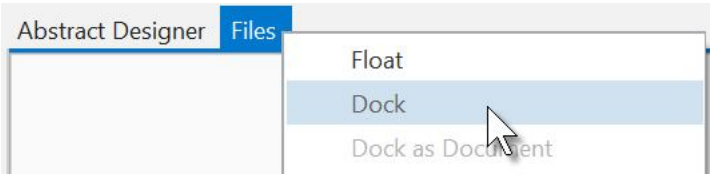




To move it back to its parent window right click on 

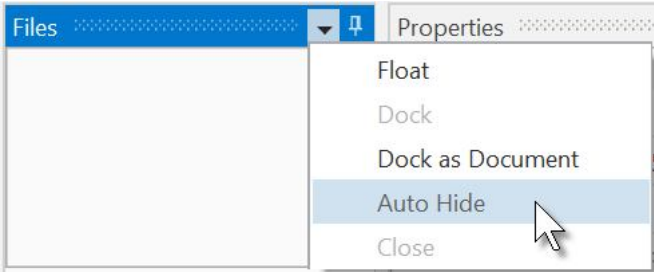
and on 

.

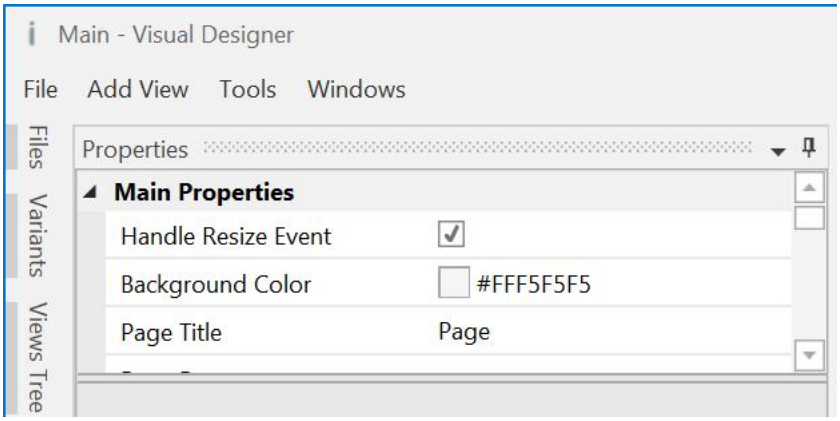


6.3.4 Auto Hide

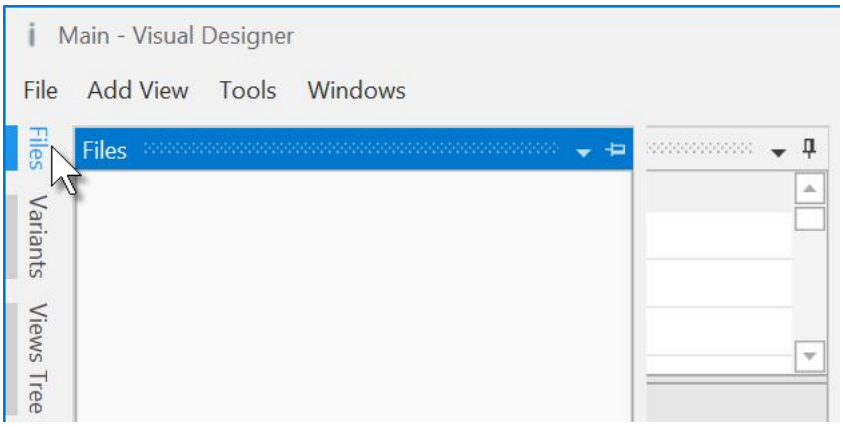
Click either on  or on Auto Hide .



The three windows: Files, Variants and Views Tree are moved as Tabs to the left border of the Visual Designer. The Properties window width is increased.

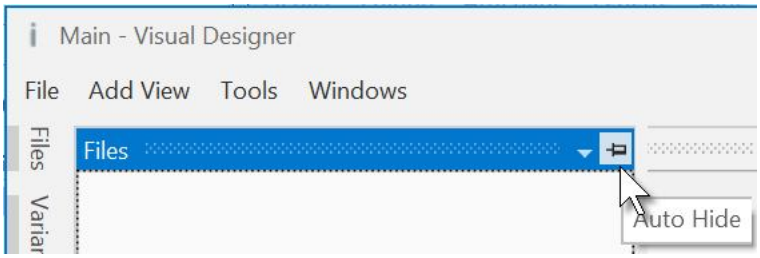


Click on a Tab to show the window.



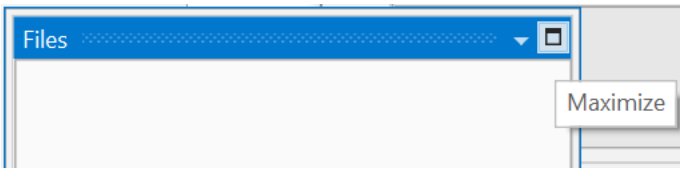
When you click somewhere else, outsides the selected window, hides it automatically.

Click on  in the title to move the windows back to their previous position.

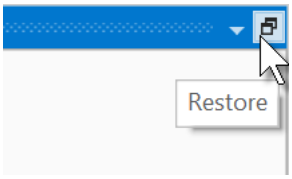


6.3.5 Maximize

Floating windows can be maximized.

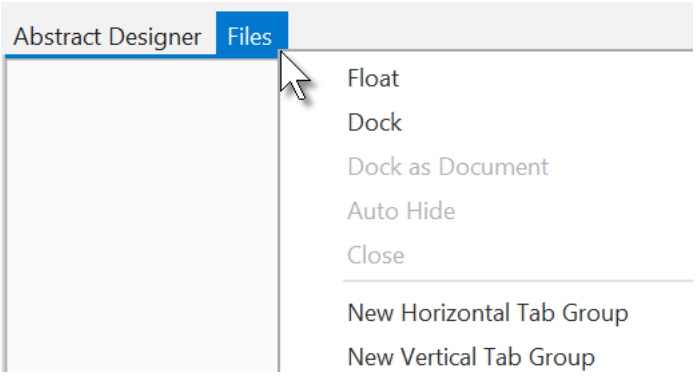


To set it back to its previous size, click on in the top right corner.

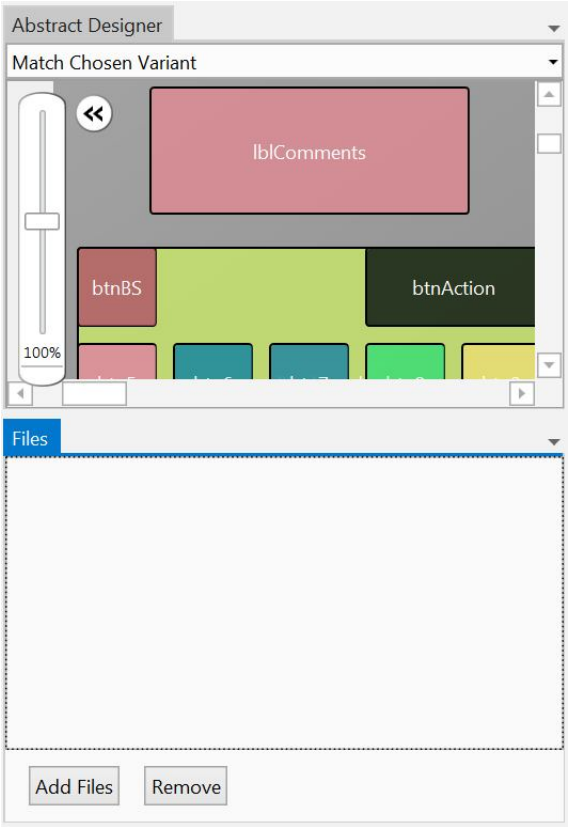


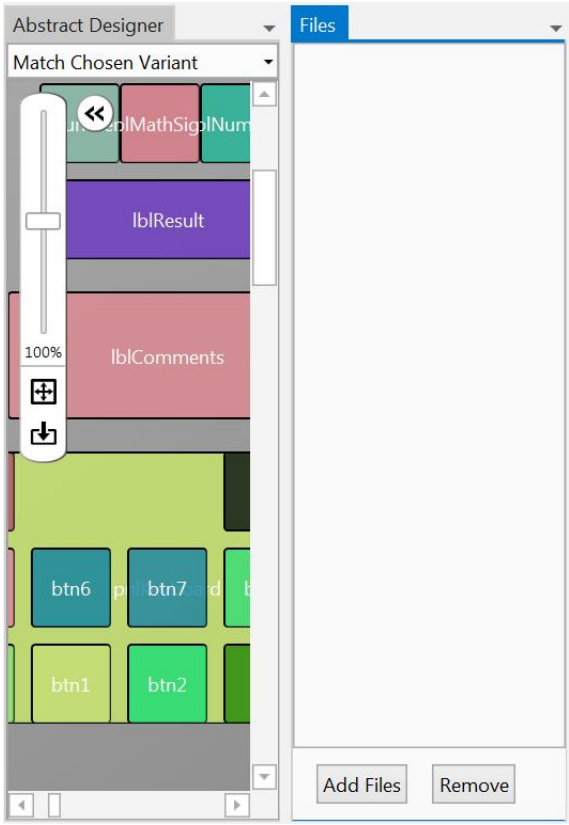
6.3.6 New Horizontal / Vertical Tab Group

When a window is set as *Dock as Document* two other options are available.



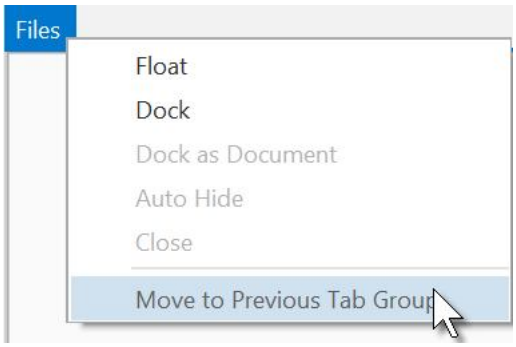
New Horizontal Tab Group
New Vertical Tab Group



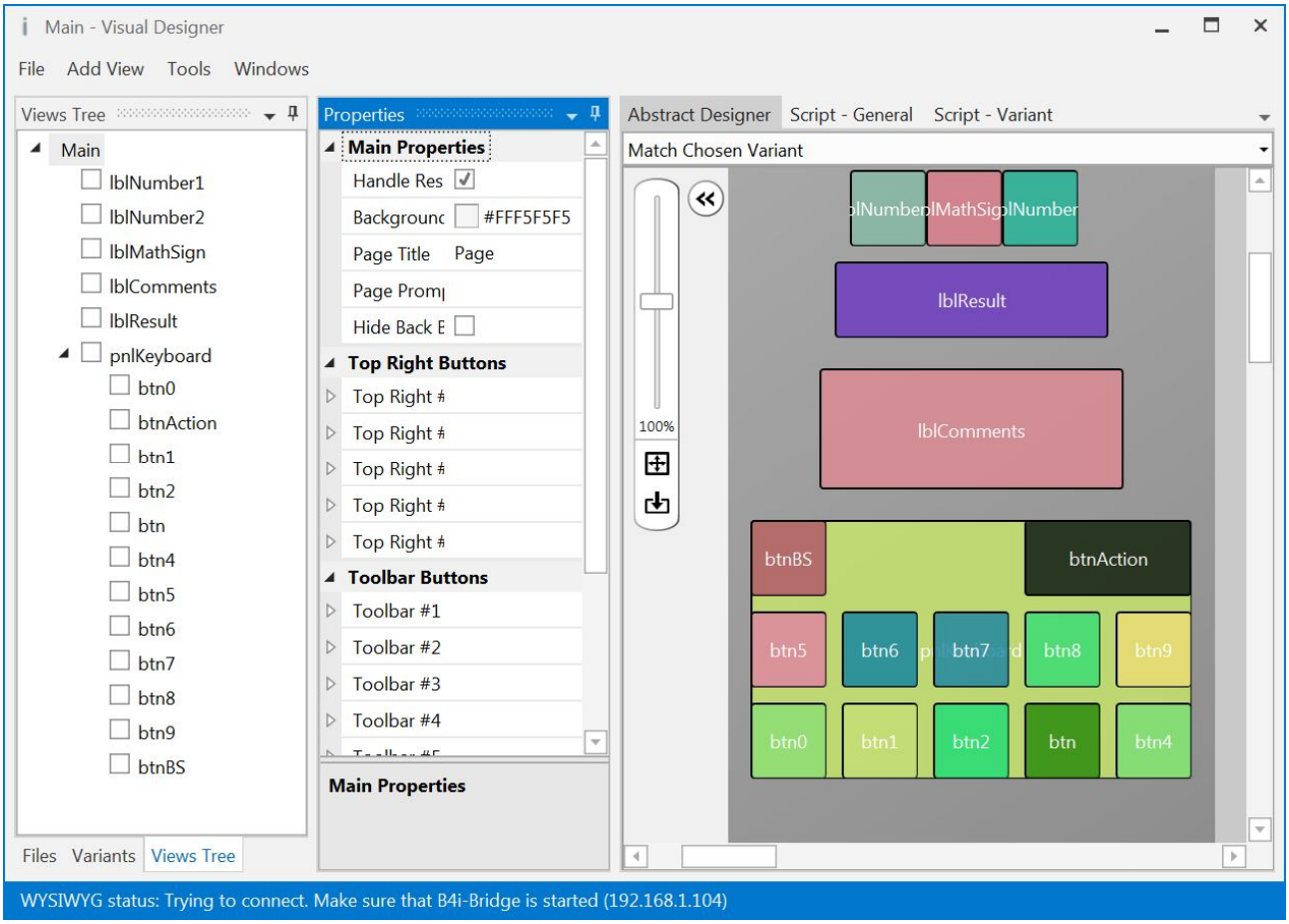


New Horizontal Tab Group New Vertical Tab Group

To remove Tab Group right click on Files and click on Move to Previous Tab Group.



My preferred Designer layout:

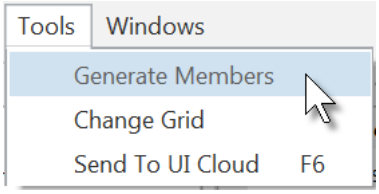


I moved the two Script widows as *Dock as Document* onto the Abstract Designer window.
That way the Views and Properties windows is much higher.

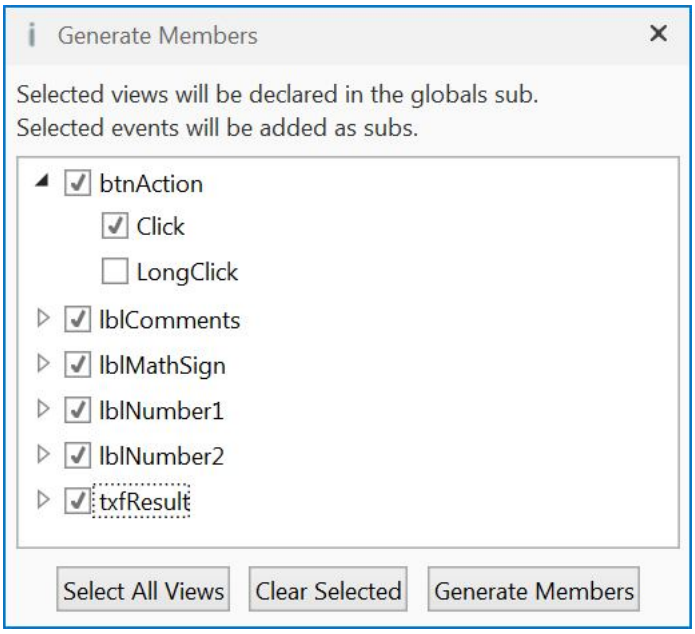
6.4 Tools

6.4.1 Generate Members

Allows generating Dim statements and subroutine frames.
The example is based on the project MyFirstProgram.



In the Tools menu click on Generate Members to open the generator.



Here we find all the views added to the current layout.
We check all views and check the Click event for the btnAction Button.

Checking a view lblNumber1 generates its reference in the Process_Globals Sub in the code.
This is needed to make the view being recognized and allow the autocomplete function.

```
Private btnAction As Button
Private txfResult As TextField
Private lblComments As Label
Private lblMathSign As Label
Private lblNumber1 As Label
Private lblNumber2 As Label
```

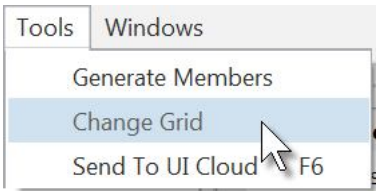
You can open a list for the available events of a view btnAction:
Clicking on an event Click generates the Sub frame for this event.

```
Private Sub btnAction_Click

End Sub
```

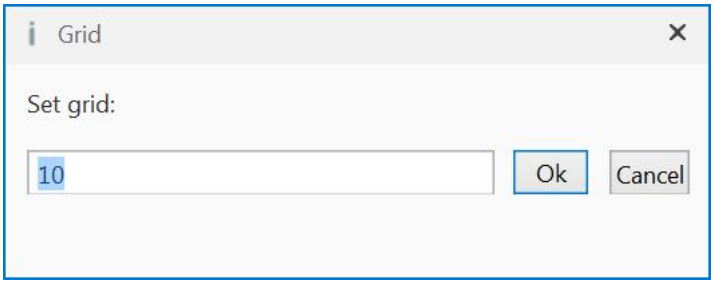
Click on Generate Members to generate the references and sub frames and close the window.
Click on Select All Views to select all views in the list,
Click on Clear Selected to clear the current selections.

6.4.2 Change grid

The grid is an invisible grid with a given size. The default grid size is 10 pixels. That means that all positions and dimensions of a view will be set to values in steps corresponding to the grid size. Moving a view will be done in steps equal to the grid size.



Click on Change Grid in the Tools menu.

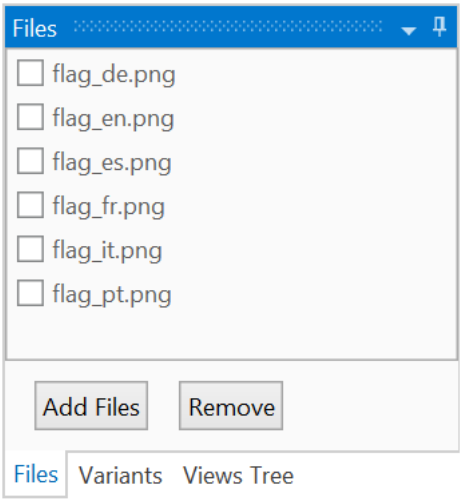


You can change the grid size to the value you want.

The value is saved in the layout file, you will get the same value when you reload this layout.

The default value when you start a new project is 10.

6.5 Image files



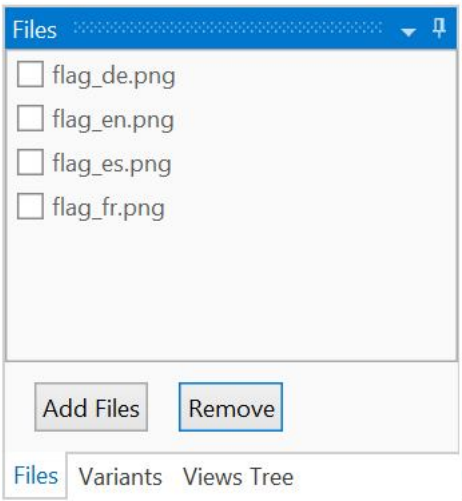
You can add image files to the layout.

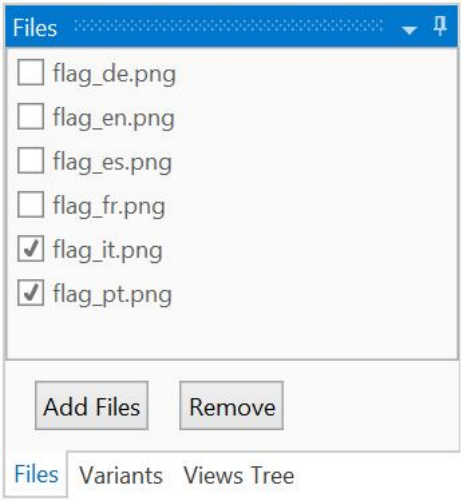
Click on Add Files to select the files(s) to add.

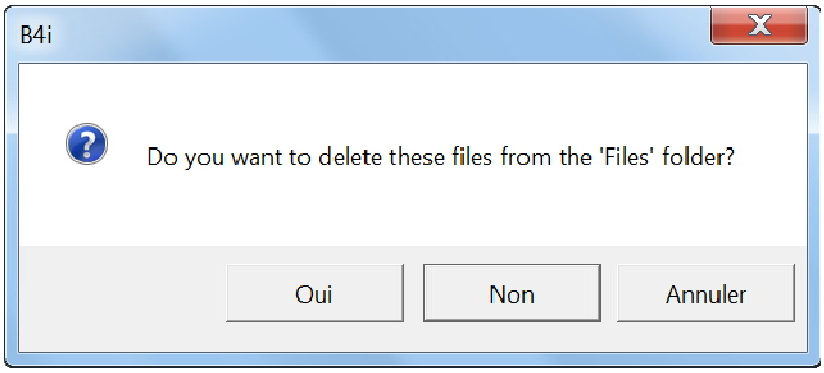
These files will be listed in the Image Files list.

These files are saved to the Files folder of the project and can be accessed in the code in the Files.DirAssets folder.

To remove files, check the files to remove and click on Remove.







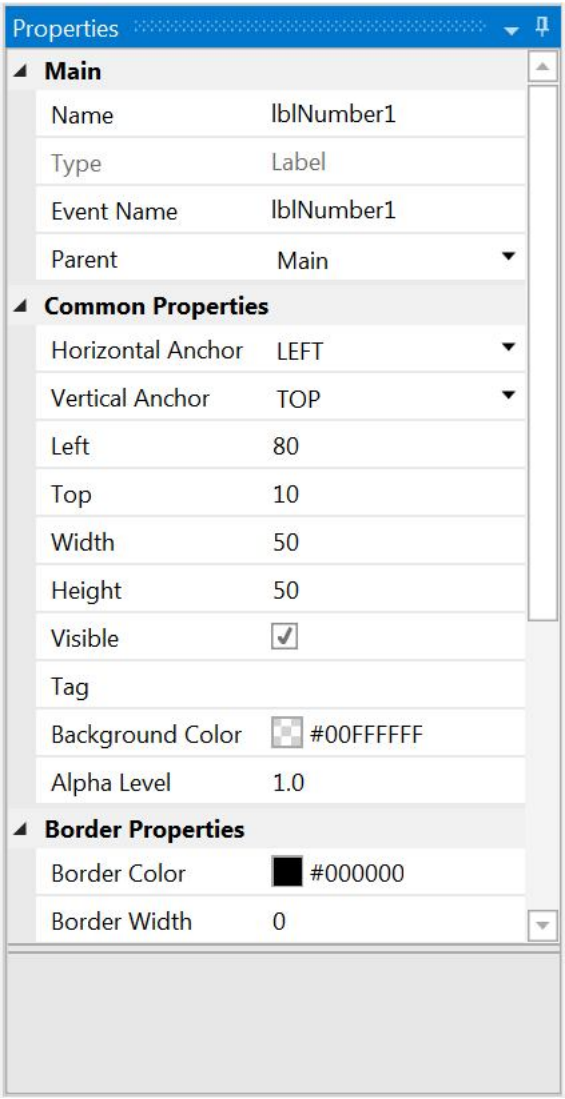
You are asked if you want to delete the files from the 'Files' folder.

Oui = Yes
Non = No
Annuler = Cancel

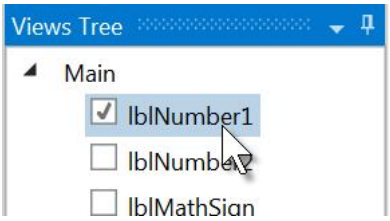
When you answer Yes make sure to have a copy of the files you remove, because they are removed from the Files folder, but not transferred to the Recycle Bin, which means that they are definitely lost if you don't make a copy.

6.6 Properties list

The example is based on the SecondProgram code.



Select for example lblNumber1 in the Views Tree list.



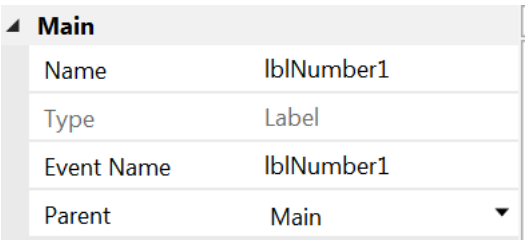
All the properties of lblNumber1 are displayed. These are organized in groups.

All properties can be modified directly in the list.

All properties in the Main group and some of the properties in the other groups are common to all view types.

Explanation of some general properties for all types of Views in the next chapters.

6.6.1 Main properties



Name Name of the view. It is good practice to give meaningful names. Common usage is to give a 3 character prefix and add the purpose of the view. In the example, the view is of type Label and its purpose is to show a result. So we give it the name "lblResult", "lbl" for Label and "Result" for the purpose. This does not take much time during the design of the layout but saves a lot of time during coding, debugging and maintenance of the program.

Type Type of the view, not editable. It is not possible to change the type of a view. If you need to, you must remove the view and add a new one.

Event Name Generic name for the subroutines that manages the view's events. By default, the Event Name is the same as the view's name like in the example. The Events of several Views can be redirected to a same subroutine. In that case you must enter the name of that routine. Look at the SecondProgram example for the Click event management for the buttons of the keyboard in the btnEvent_Click routine.

Tag This is a place holder which can be used to store additional data. Tag can simply be text but can also be any other kind of object.
Tag is used in the SecondProgram example for the numeric buttons click events management in the btnEvent_Click routine.

Parent Name of the parent view. Main, in the example. The parent view can be changed in selecting the new one in the list.

6.6.2 Common properties

Common Properties	
Horizontal Anchor	LEFT
Vertical Anchor	TOP
Left	80
Top	10
Width	50
Height	50
Visible	☑
Tag	
Background Color	#00FFFFFF
Alpha Level	1.0

HorizontalAnchor Horizontal Anchor function. Possible values LEFT, RIGHT or BOTH

VerticalAnchor Vertical Anchor function. Possible values TOP, BOTTOM or BOTH

Left X coordinate of the left edge of the View from the left edge of its parent View, in points.

Top Y coordinate of the upper edge of the View from the upper edge of its parent View, in points.

Width Width of the View in points.

Height Height of the View in points.

Visible Determines if the View is visible to the user or not.

Background color Color of the views background.

Alpha level Sets the transparency, 1 = opaque 0 = transparent.

6.6.3 Border properties

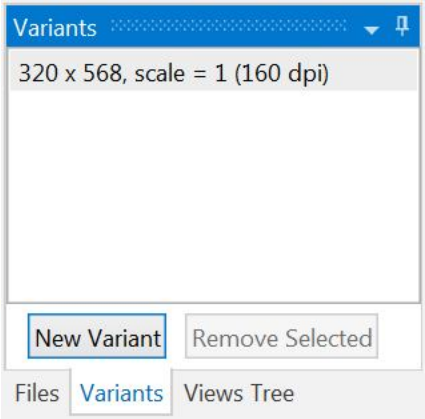
Border Properties	
Border Color	#000000
Border Width	0
Corner Radius	0

Border color Color o

Border Width Width o

Corner Radius Sets the

6.7 Layout variants

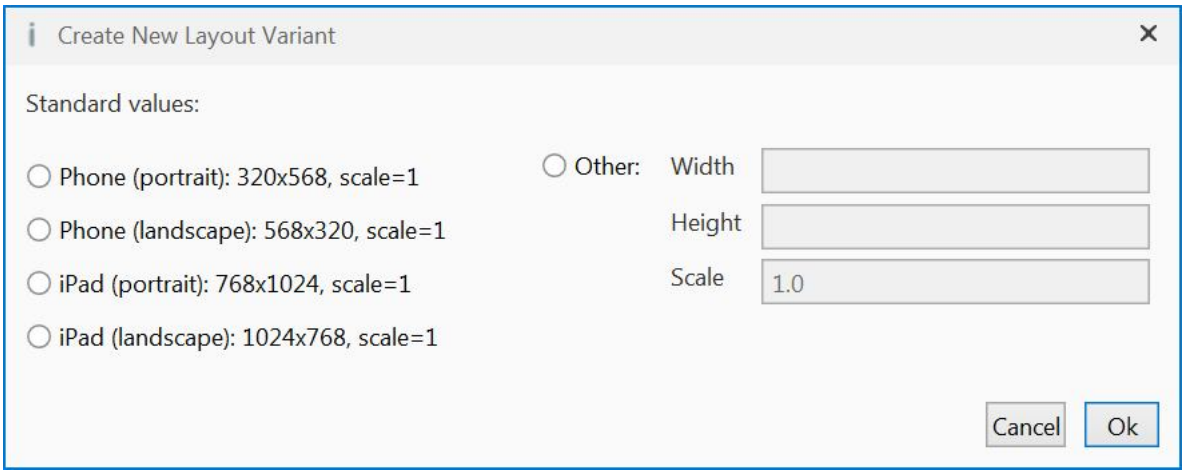


Different layout variants can be managed in a same layout file.

If a device is connected, its details are shown in the bottom left corner of the Visual Designer.

In the example it's an iPhone 6.
We see that the screen is bigger than the layout variant screen, 375x667 instead of 320x568.

Click on New Variant to show the available layout variants.

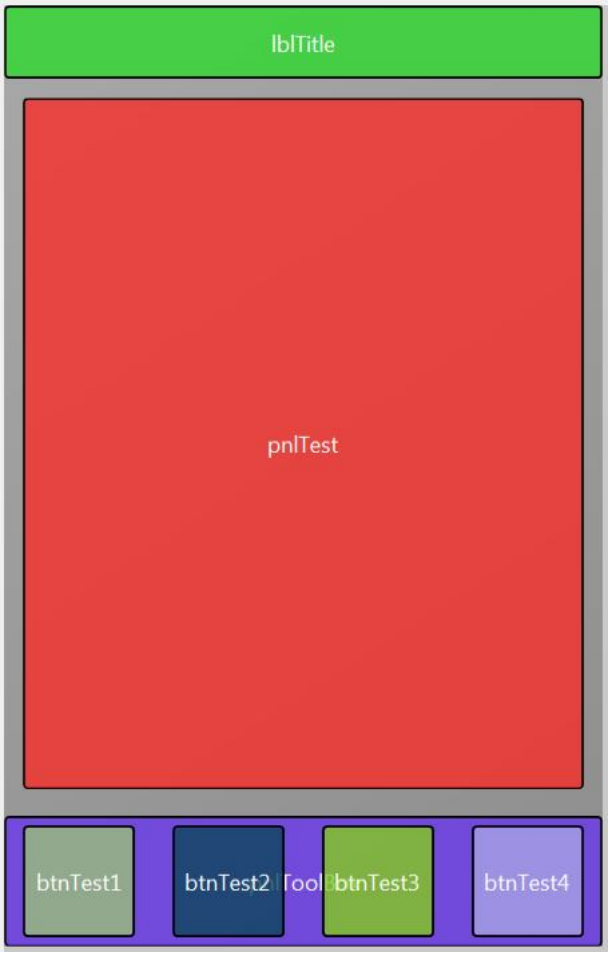


Four layout variants are available:

- Phone, this layout variant corresponds to the iPhone 5 screen size.
  - Phone (portrait): 320x568, scale = 1
  - Phone (landscape): 568x320, scale = 1
- iPad, this layout variant corresponds to the iPad / iPad2 / iPad mini screen size.
  - iPad (portrait): 768x1024, scale = 1
  - iPad (landscape): 1024x768, scale = 1

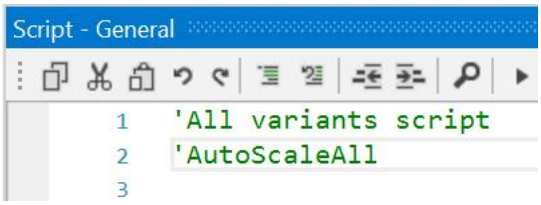
You should only use these layout variants and not specific layouts, even though it is possible, and fine-tune them with Anchors and with Designer Scripts.

Let us make an example project: LayoutVariants
The source code is saved in the Guide\SourceCode\LayoutVariants\LayoutOriginal directory.
- Run the IDE.
- Run the Designer.



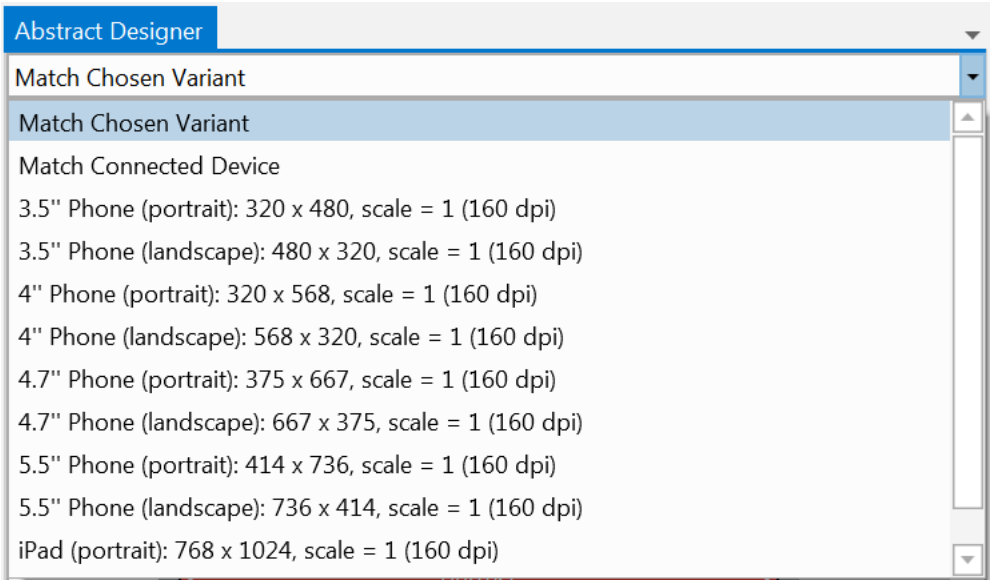
The AbstractDesigner looks like this.

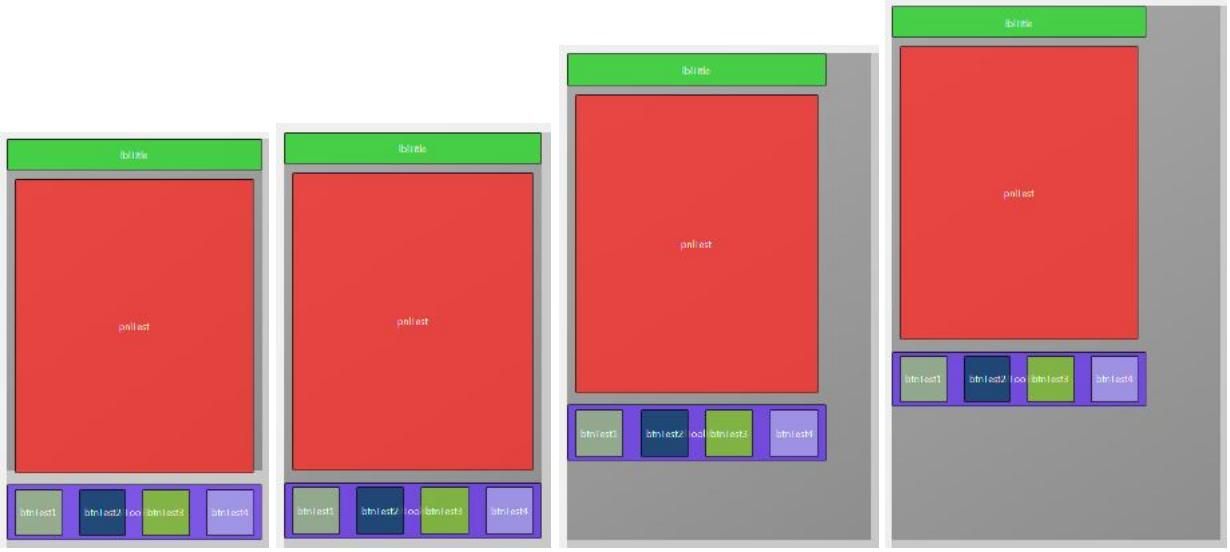
- One Label on top lblTitle.
- One Panel pnlTest covering the center of the screen.
- One Panel pnlToolBox at the bottom of the screen.
- 4 Buttons btnTest1 to btnTest4.
- In the Script General window AutoScaleAll should commented!!!



In the Abstract Designer we can check the look of the layout on different screens.

In the Layout menu you can select the screen size to check.





3.5’’ screen iPhone 4 | 4’’ screen iPhone 5 | 4.7’’ screen iPhone 6 | 5.5’’ screen iPhone 6+

We see that:

- On the 3.5’’ screen the bottom views are outsides the screen.
- On the 4.7’’ and the 5.5’’ screen the views don’t fill the screen.

What can we do to change this? Use Anchors and DesignerScripts.

Let's see what we can do with Anchors. In the Designer we set:

- lblTitleHorizontal Anchor property to BOTH.
- pnlTest Horizontal Anchor and Vertical Anchor to BOTH.
- pnlToolBox Horizontal Anchor to BOTH and Vertical Anchor to BOTTOM.

3.5'' screen iPhone 4 | 4'' screen iPhone 5 | 4.7'' screen iPhone 6 | 5.5'' screen iPhone 6+

We see that:

- lblTitle and pnlToolBox fill the whole width, pnlTest fills the screen, only the 4 buttons are not moved.

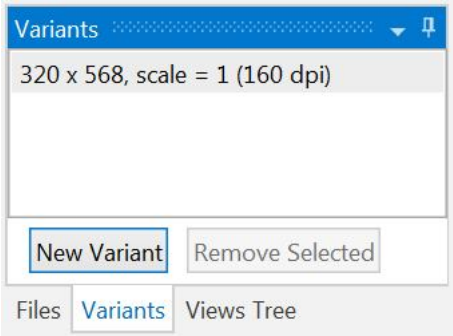
For this we can add the code below in DesignerScript to center the 4 buttons.

```
'All variants script
'AutoScaleAll
btnTest2.Right = 50%x - 10
btnTest1.Right = btnTest2.Left - 20
btnTest3.Left = 50%x + 10
btnTest4.Left = btnTest3.Right + 20
```

3.5'' screen iPhone 4 | 4'' screen iPhone 5 | 4.7'' screen iPhone 6 | 5.5'' screen iPhone 6+

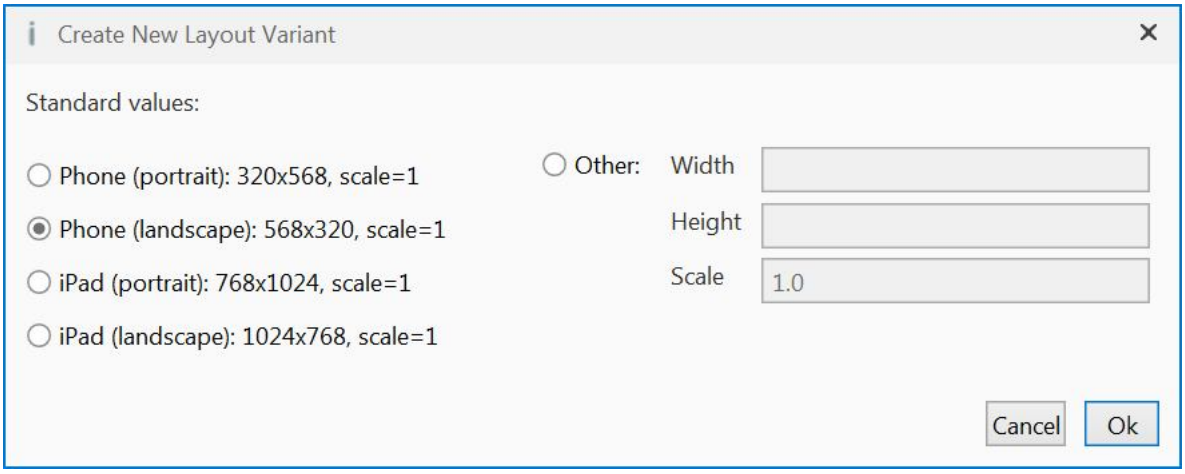
We have not used AutoScale.

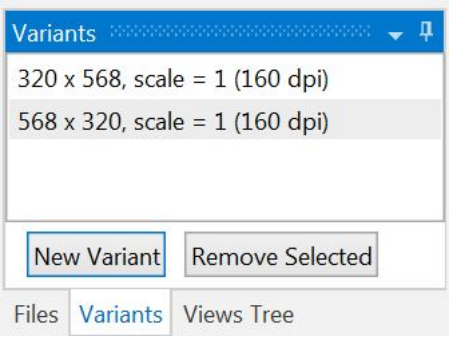
The text sizes in the Label and in the Buttons have not changed, but not really necessary, the text sizes in the iOS Navigation and Toolbar buttons aren't changed either.
Nevertheless, it could be done with AutoScale.



Now we add a landscape layout.

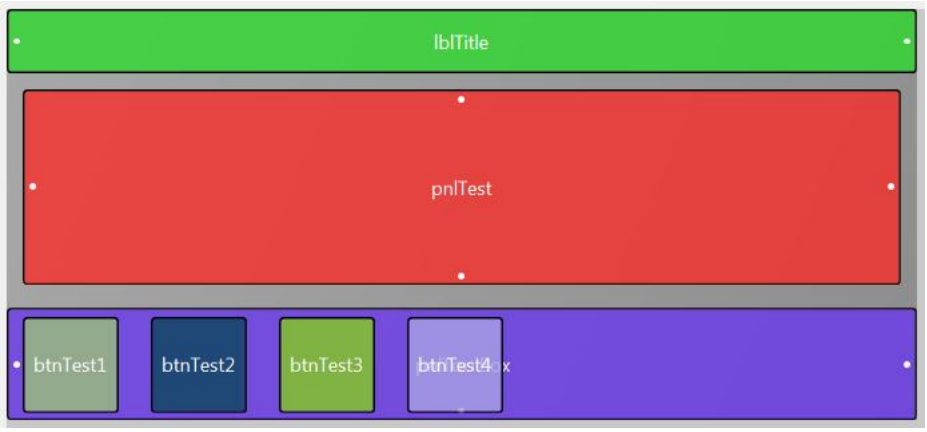
Select the Variants window and click on New Variant to show the available layout variants.



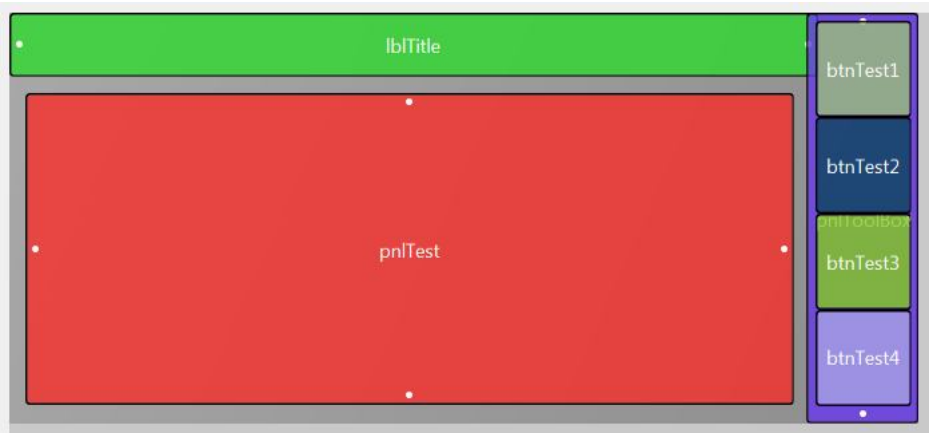


Select 568 x 320, scale = 1 (160 dpi) .

The new layout is added in the list.



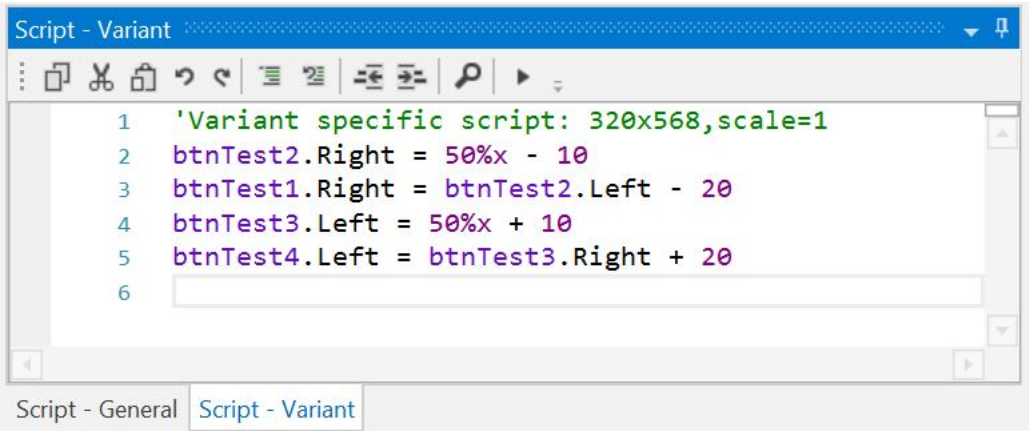
And the layout looks like the image.



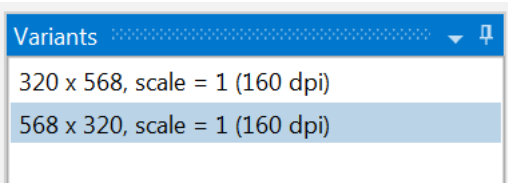
Now we want to have the buttons on the right side instead on the bottom. We need to rearrange the layout.

We need to change the Anchors of pnlToolBox.
Horizontal Anchor from BOTH to RIGHT.
Vertical Anchor from BOTTOM to BOTH.

We must change the code in the Designer Scripts.
Select the portrait layout and copy the code from `All variants script` to `Variant specific script: 320x568,scale=1`.



Then select the landscape variant



and add following code.

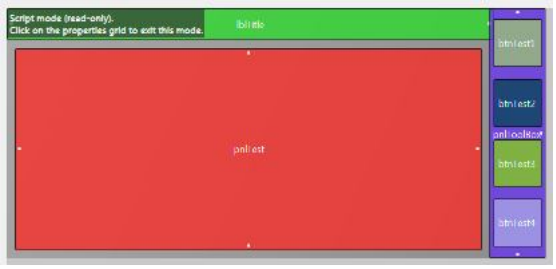
```
Space = (100%y - 4 * btnTest1.Height) / 5
btnTest1.Top = Space
btnTest2.Top = btnTest1.Bottom + Space
btnTest3.Top = btnTest2.Bottom + Space
btnTest4.Top = btnTest3.Bottom + Space
```

We don’t center the buttons like in the portrait layout but move them equally spaced over the height.

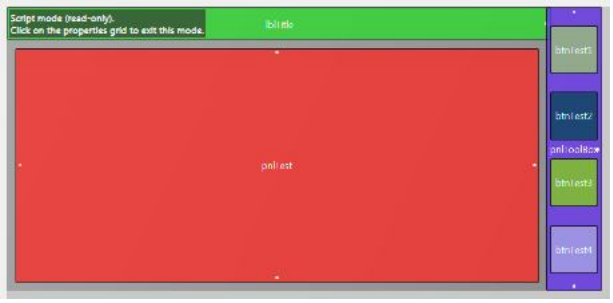
Let’s see what we get in the different screens:

3.5’’ screen iPhone 4

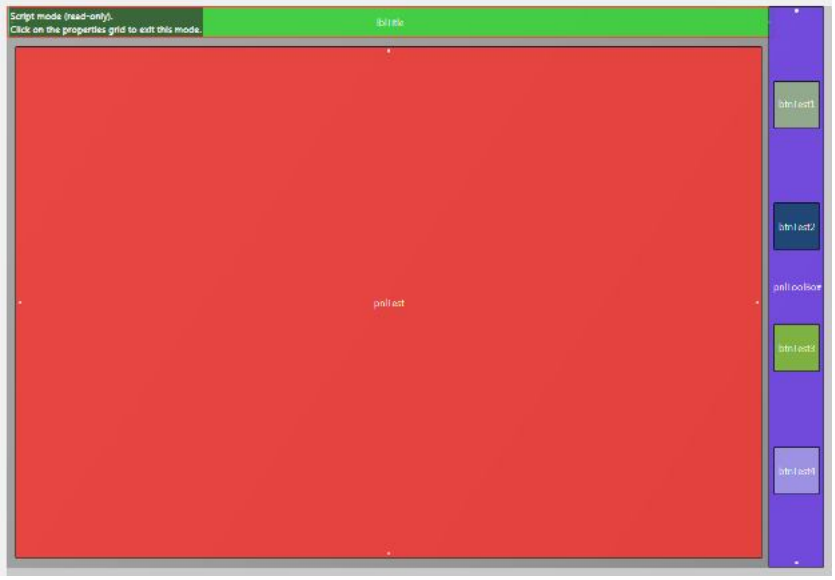
4’’ screen iPhone 5



4.7’’ screen iPhone 6



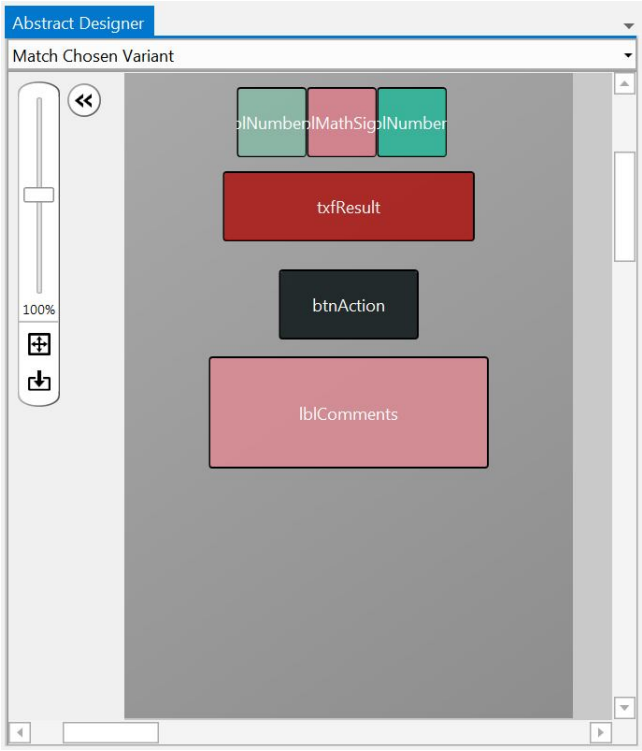
5.5’’ screen iPhone 6+



iPad screen

The layout looks OK on all screens.

6.8 The Abstract Designer

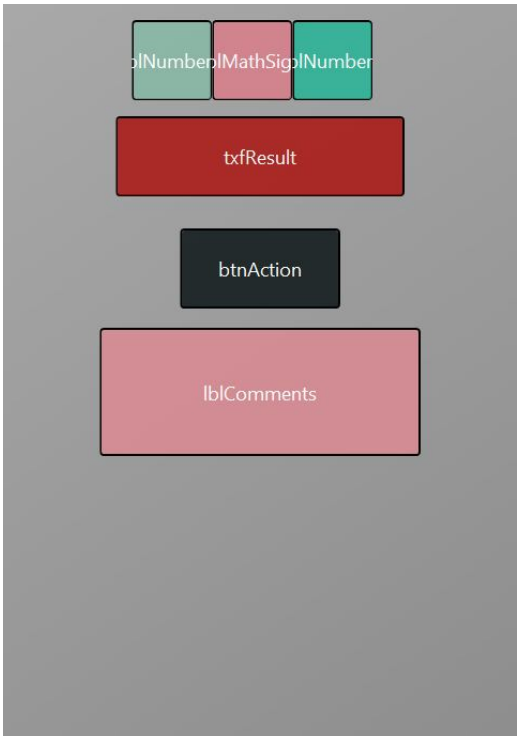


The Abstract Designer is a tool that shows the layout in a separate window and is part of the Visual Designer.
Its main purpose is to create different layout variants.

The different views are not shown with their exact shape but only as colored rectangles. Clicking on a view shows its properties in the Properties window.

To see the exact shape of the current layout you need to connect the Designer to a device.

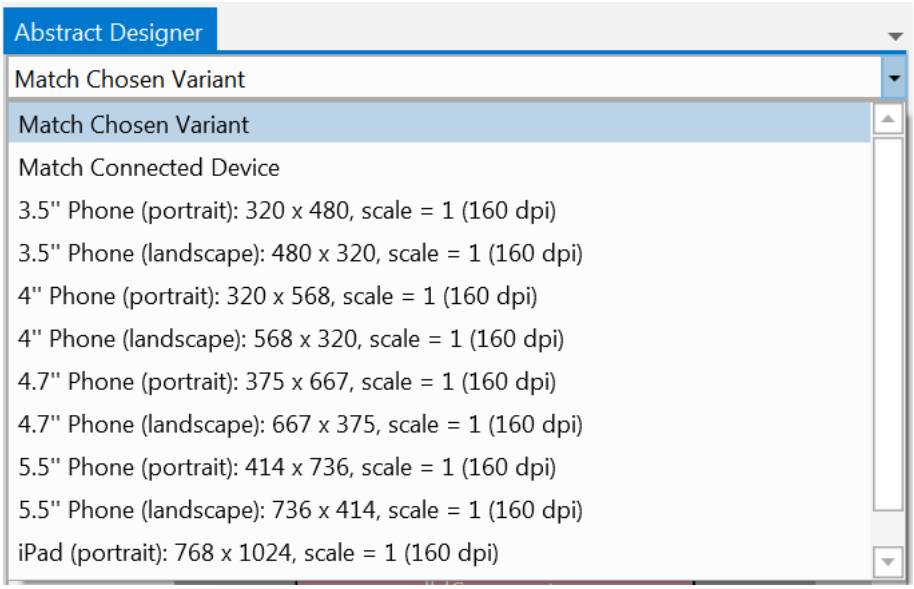
Abstract Designer



Device

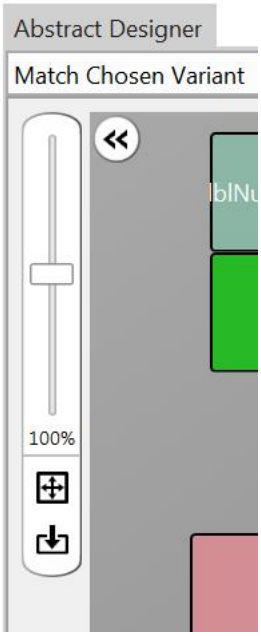
6.8.1 Selection of a screen size

On top you can select different screen sizes:



Match chosen Variant.
Matches the variant selected in the Variant window.
Match Connected Device.
Matches the size of the connected device.
Different 'standard' sizes.
This allows you to see how a layout looks on different screens.

6.8.2 Zoom



With « you can hide the zoom cursor and show it again with ».

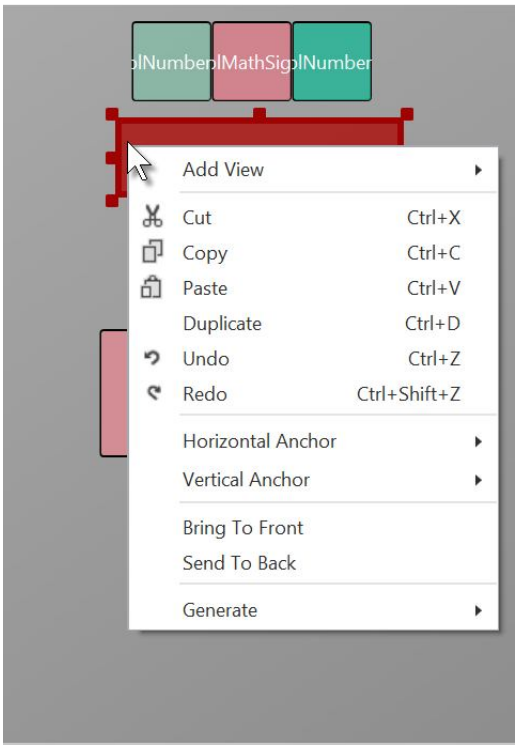
With the cursor you can set the zoom level you want.

With ⊞ you can zoom to fit the selected screen size.
With ⤓ you can reset the zoom back to 100%.

With the bottom and side cursors you can move the layout vertically or horizontally.

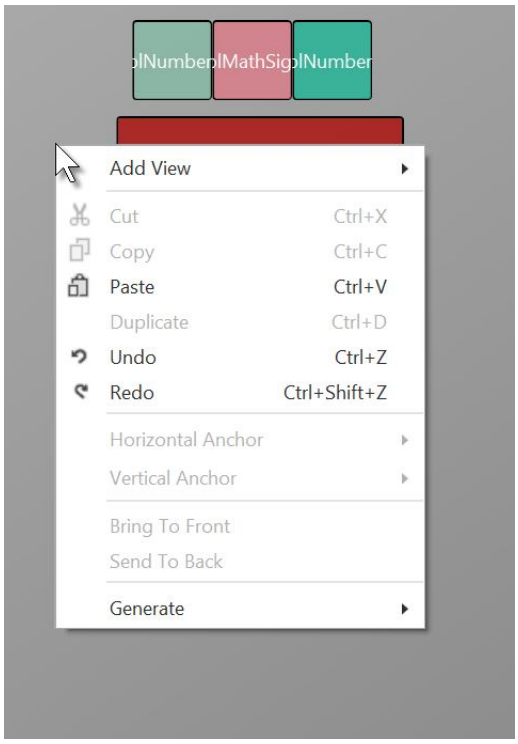
6.8.3 Context menus

Most editing functions can be accessed in a popup menu which is displayed when you right click on a view.



Add View
Cut
Copy
Paste
Duplicate
Undo
Redo
Horizontal Anchor
Vertical Anchor
Bring To Front
Send To Back
Generate

Right clicking somewhere on the Activity area shows the context menu with some functions disabled which are not relevant for a Page.

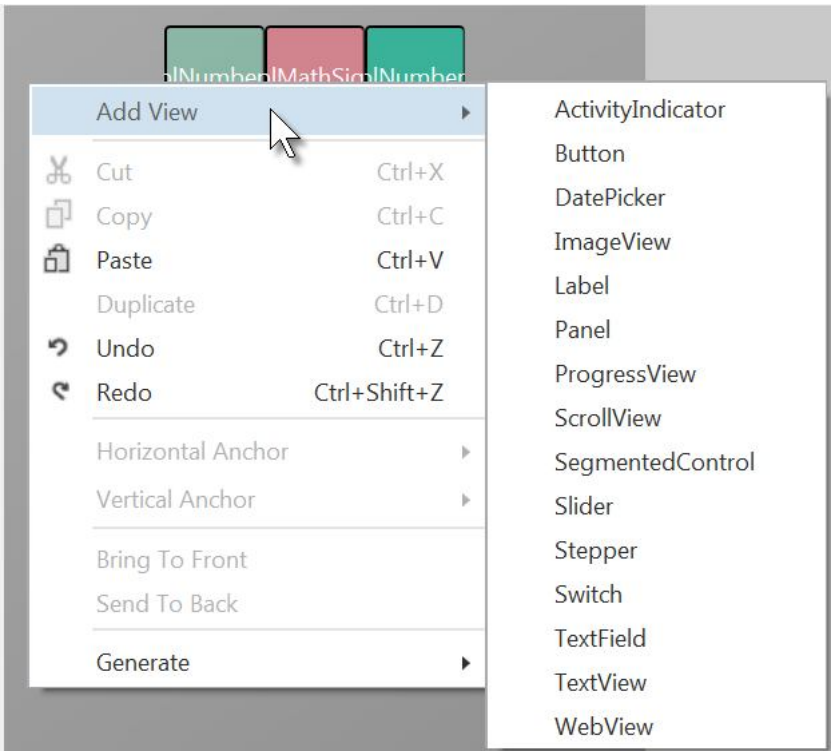


Only Add View, Paste, Undo, Redo and Generate are available for a Page.

6.8.3.1 Add View

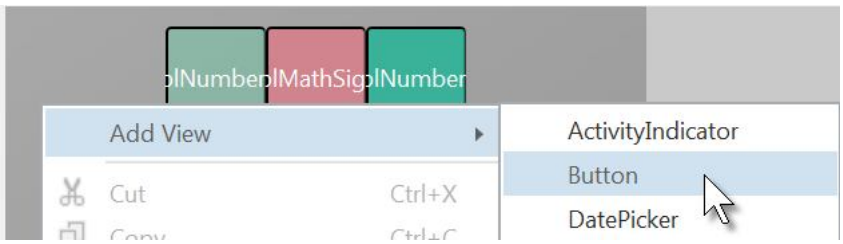
Right click somewhere on the parent view where you want to add a new view and move the cursor onto Add View.

This function is the same as the Add View function in the Visual Designer menu.

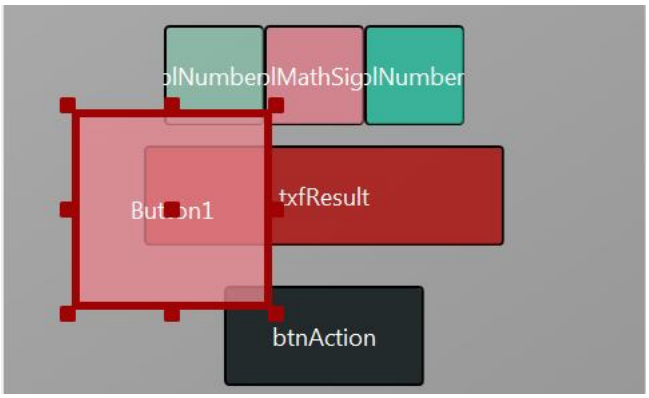


The list of all available views is displayed.

Click on the desired view to add it.

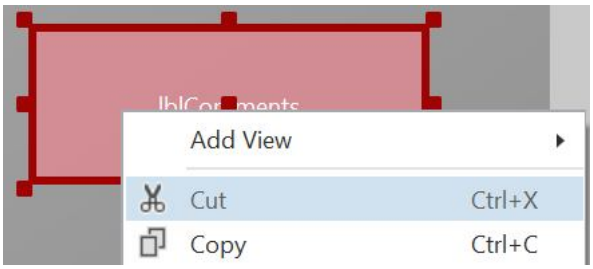


Example for a Button.



The Button is added to the layout.

6.8.3.2 Cut



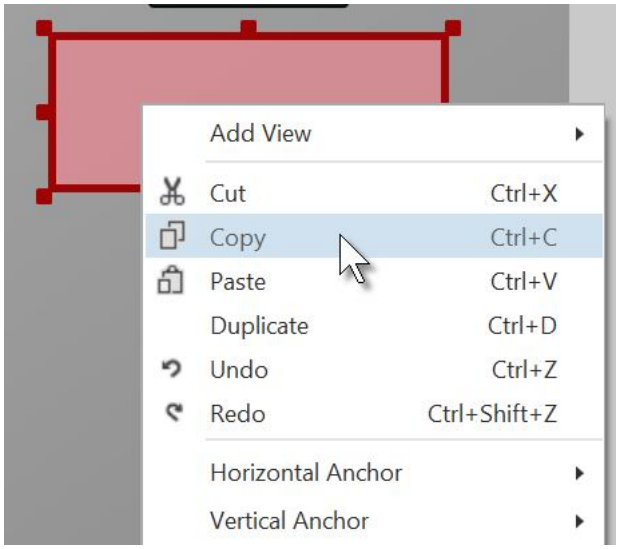
Right click on a view to select it and click on Cut Ctrl+X to cut it.

There is no message for confirmation.

If you selected a Panel, it will be removed with all its child views!

If you cut it by accident click on Undo Ctrl+Z to recover it.

6.8.3.3 Copy / Paste / Duplicate Selected Views

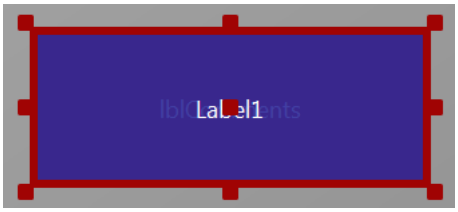


You can copy and paste views onto the same layout or from one layout to another one.

To copy and paste views on the same layout use Duplicate Ctrl+D it's faster, one click to copy and paste.

Right click on a view to select it and click on Copy Ctrl+C to copy it to the clipboard.

You can select several views and copy them.



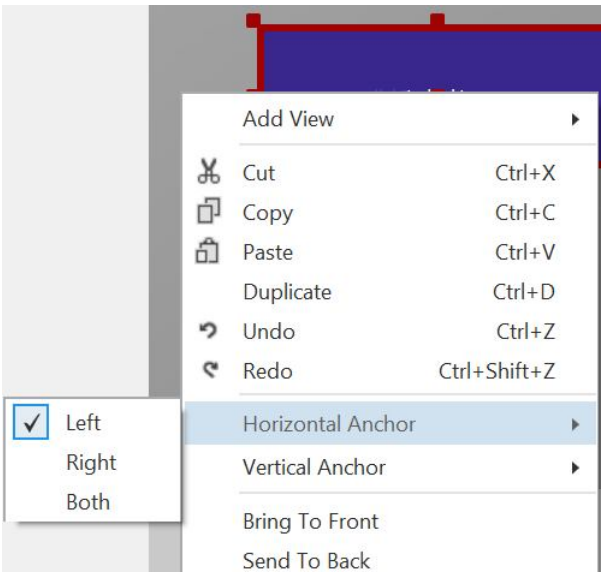
Right click again somewhere else and click on Paste Ctrl+V to paste the view, it will be copied over the original.

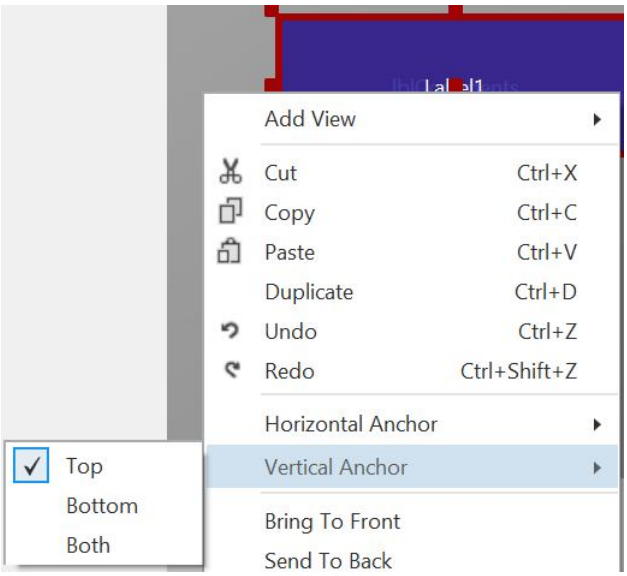
6.8.3.4 Undo / Redo

Undo Ctrl+Z Redo Ctrl+Shift+Z

These two functions allow you to undo or redo the last operations.

6.8.3.5 Anchors horizontal / vertical





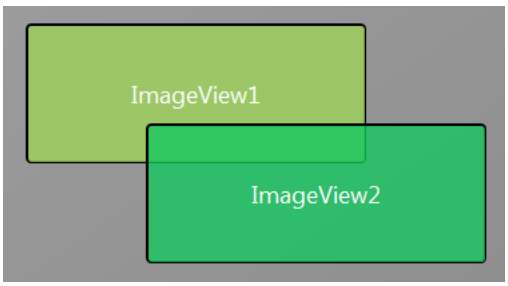
Right click on a view and click on

Horizontal Anchor or Vertical Anchor

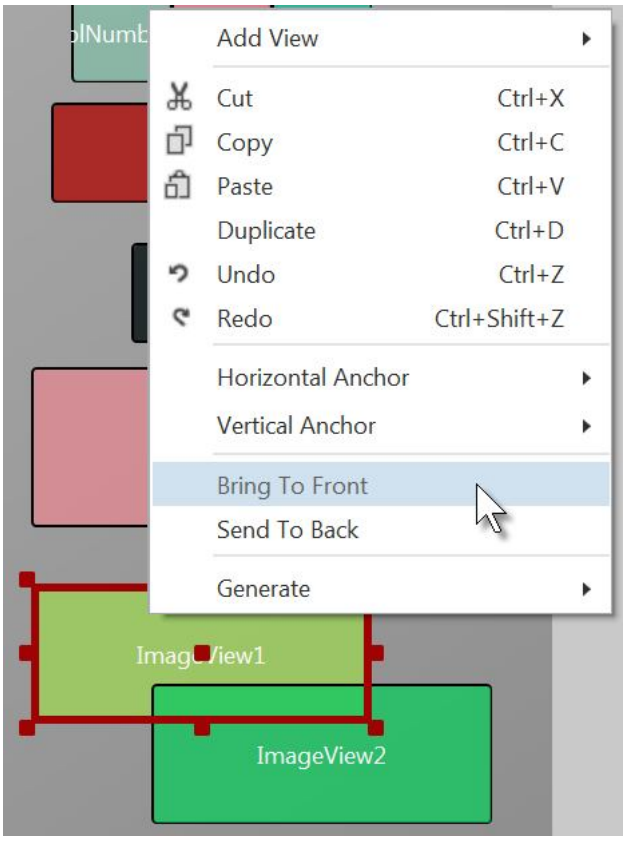
and select the desired anchor.

6.8.3.6 Bring To Front

Bring To Front Moves the selected View on top of the layout.

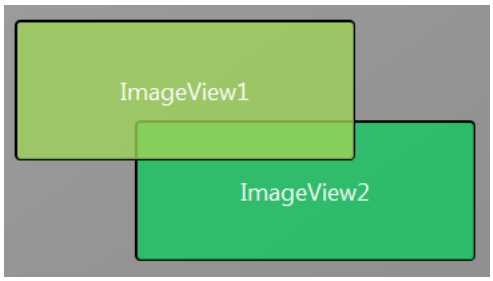


In the picture, ImageView2 is over ImageView1.
You see it with the border color.



Right click on ImageView1 and click on Bring To Front to move ImageView1 to front of all other views.

And the result:



6.8.3.7 Send To Back

Send To Back Is the inverse function of the *Bring To Front* function above.

6.8.3.8 Generate

Generates the declaration statement or an event routine frame for the selected View. It is a shortcut of the Generate Members function in the VisualDesigner Tools menu but only for the selected view.

A popup menu allows you to select what code you want to generate, the possibilities depend on the type of the selected view.

Example with a Button:

Dim btnAction As Button

Generates the declaration statement in the Globals routine.

```
Private btnAction As Button
```

Click

Generates the Click event routine frame.

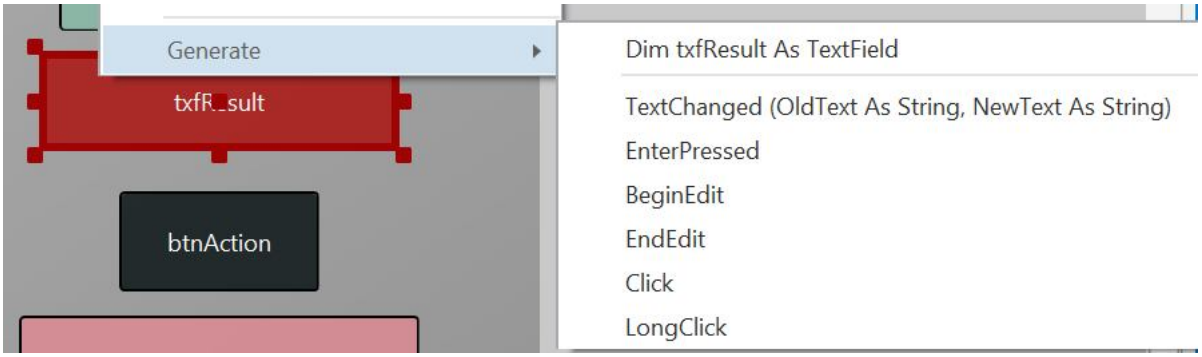
```
Sub btnAction_Click

End Sub
```

LongClick

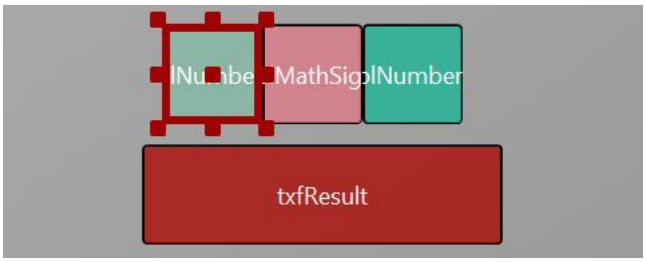
Generates the LongClick event routine frame.

```
Sub btnTest1_LongClick

End Sub
```

Example with a TextField view:

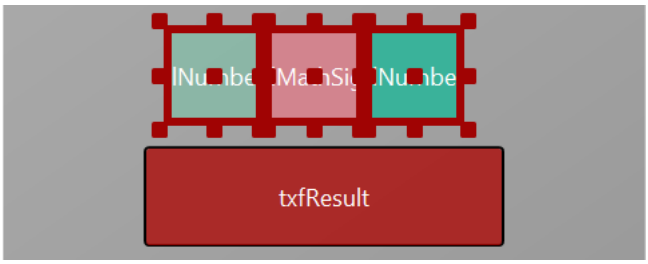


6.8.4 Select views



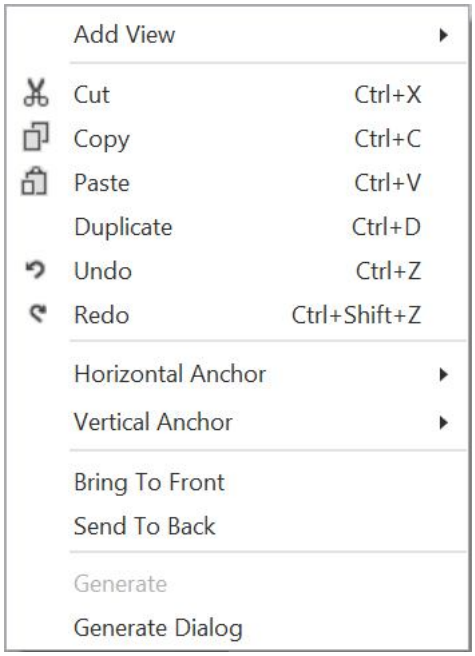
Select a single view:
Click on the view.

The view is highlighted.



Select several views:
Click on the first view.
Press the Ctrl key,
Select the following views.

The selected views are highlighted.

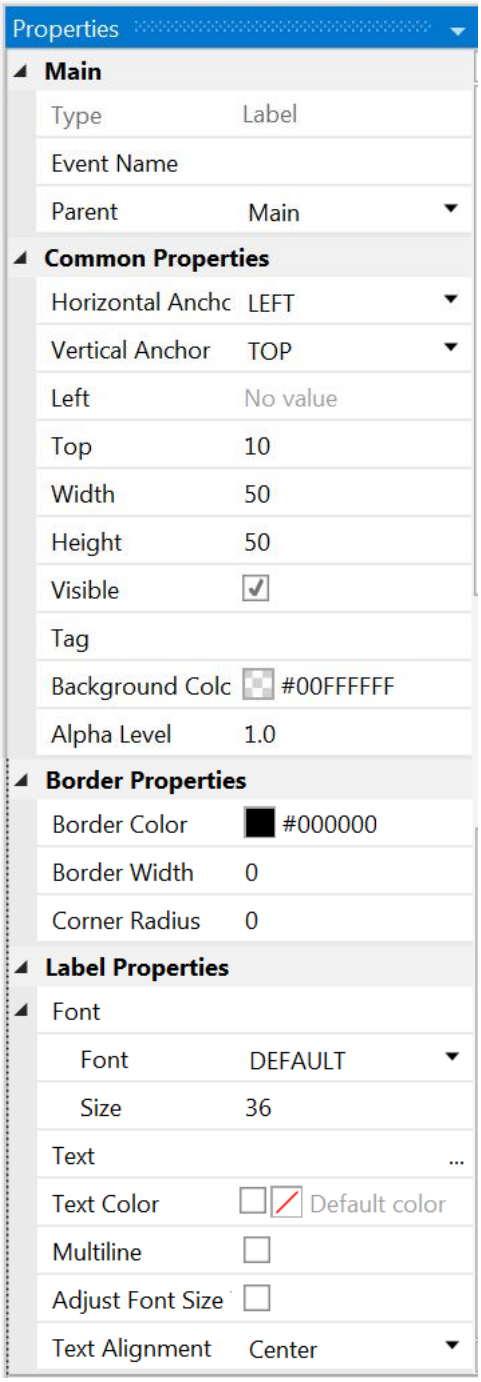


After the selection you can:

- Move the selected views with the mouse or with the arrow keys of the keyboard in the four directions.
- Right click on one of the selected views to show the context menu.

  The functions are the same as for a single view, but a new function, GenerateDialog, is available to Generate Members.
  This is the same function as in the Visual Designer Tools menu.

- In the Properties window you can change all properties common to the selected views.

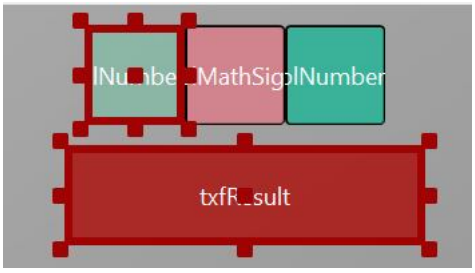


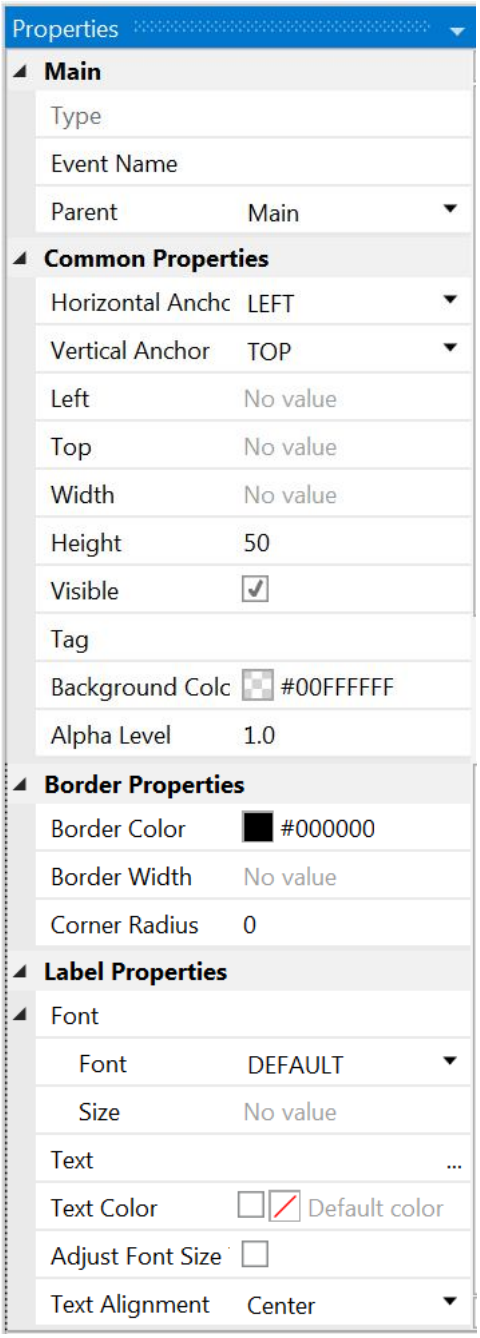
You can change the parent view.

You can change all these properties because they are the same for the four views selected in the example.

Changing, for example, the Height property will change it for all the selected views.

If you select views of different types, only the properties common to the selected views can be changed.



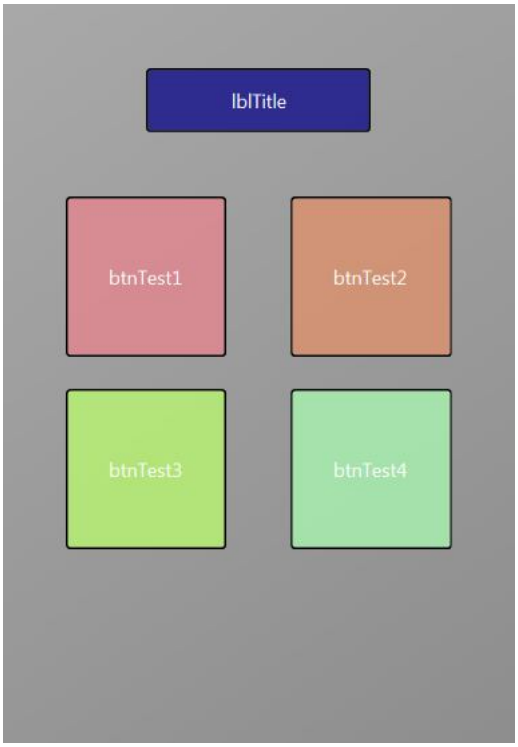


The Left, Top and Width properties cannot be changed because they are different for the selected views.

The Height property can be changed because its value is the same for both views, even for views of different types.

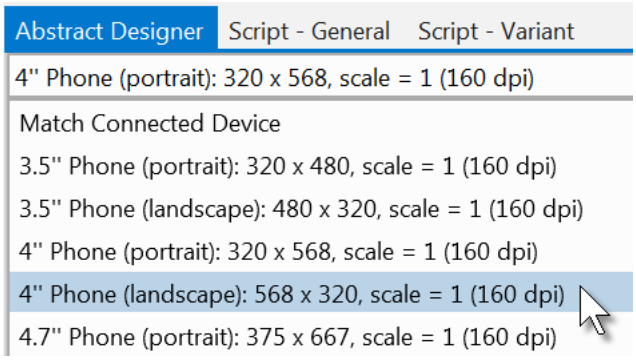
6.8.5 Example

Let us take a simple example with a layout in portrait mode, like the image below.

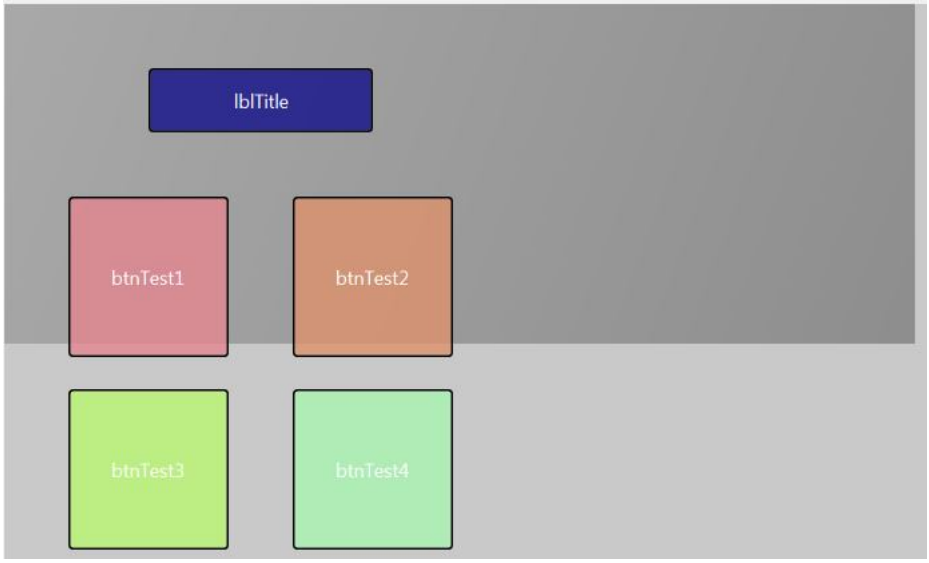


Now we would like to make a landscape variant.

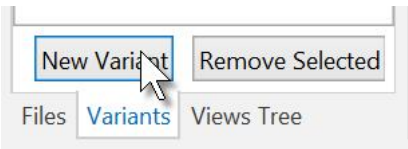
On top of the AbstractDesigner click on 4" Phone (landscape): 568 x 320, scale = 1 (160 dpi)



The Abstract Designer looks like this:

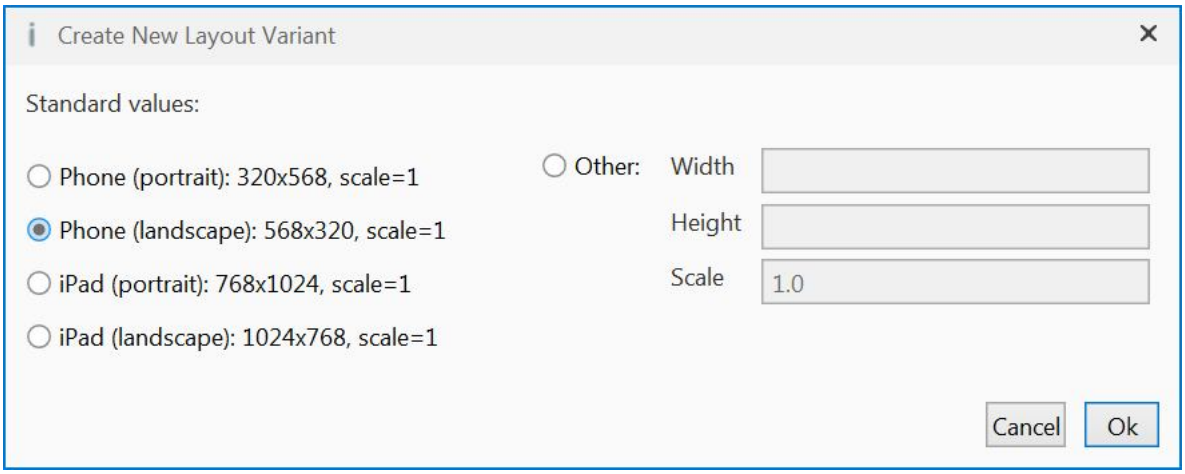


The dark gray rectangle represents the screen.

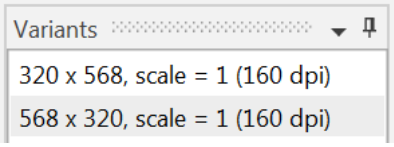


In the Variants window add a new layout.

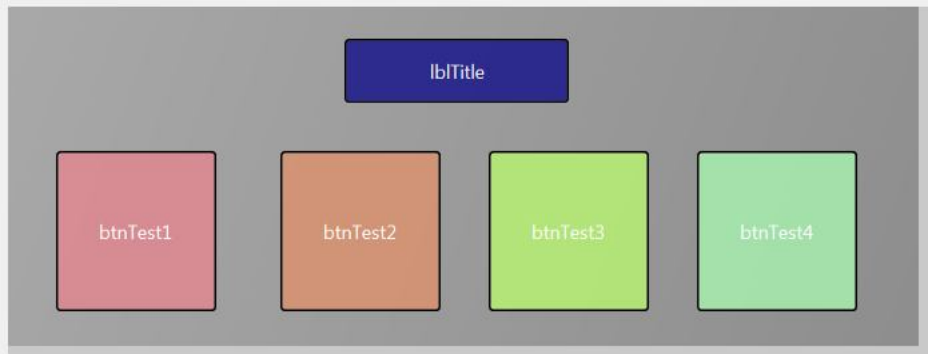
Click on New Variant.



Select Phone (landscape): 568x320, scale=1 and click on Ok.

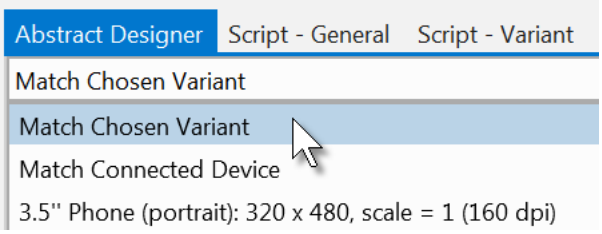


In the Variants window we see the new variant.

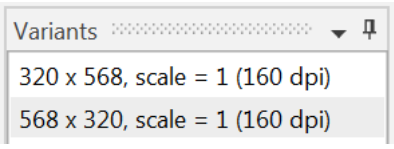


Now we rearrange the views to fit the new orientation.

On top of the Abstract Designer select Match Chosen Variant.



This will show the layout with the correct screen size.



Now you can switch between the two layouts by selecting the layout in the Variants window.

6.9 Adding views by code

It is also possible to add views by code instead of using the Designer with a device or the Abstract Designer.

- Advantage: None.

- Disadvantage: You have to define almost everything and AutoScale doesn't work.

Note that you should avoid adding views in code but use the Designer with Designer Scripts.

The source code is in the source code directory: AddViewsByCode

For the positions and dimensions of the views on the screen two options are available:

- Points, scale independent.
  The default density is 160 dpi **d**ots **p**er **i**nch (pixels per inch).
  All coordinates refer to this density, iOS adapts the values internally to the scale.
  No *dip* values like in Android. More details in Coordinates.

- %x and %y represent distances proportional to the current screen width and height.
  20%x = 0.2 * Page1.RootPanel.Width
  90%y = 0.9 * Page1.RootPanel.Height
  20%x = PerXToCurrent(20) `PerXToCurrent` is a Keyword `%x` is the Shortcut
  90%y = PerYToCurrent(90)

Example:

Let us put a Label on top of the screen and a Panel below it with a Label and a Button on it:

The whole code.

```
Sub Process_Globals
   Public App As Application
   Public NavControl As NavigationController
   Private Page1 As Page

   Private lblTitle, lblPanelTitle As Label
   Private pnlTest As Panel
   Private btnTest As Button
End Sub

Private Sub Application_Start (Nav As NavigationController)
   NavControl = Nav
   Page1.Initialize("Page1")
   Page1.Title = "Page 1"
   Page1.RootPanel.Color = Colors.White
   NavControl.ShowPage(Page1)

   lblTitle.Initialize("")
   lblTitle.Color = Colors.Red
   lblTitle.Font = Font.CreateNew(20)
   lblTitle.TextColor = Colors.Blue
   lblTitle.TextAlignment =lblTitle.ALIGNMENT_CENTER
   lblTitle.Text = "Title"

   pnlTest.Initialize("")
   pnlTest.Color = Colors.LightGray

   btnTest.InitializeCustom("btnTest", Colors.Black, Colors.Blue)
   btnTest.SetBorder(1, Colors.Black, 5)
   btnTest.Text = "Test"

   lblPanelTitle.Initialize("")
   lblPanelTitle.Color = Colors.Red
   lblPanelTitle.Font = Font.CreateNew(16)
   lblPanelTitle.TextColor = Colors.Blue
   lblPanelTitle.TextAlignment = lblPanelTitle.ALIGNMENT_CENTER
   lblPanelTitle.Text = "Panel test"
End Sub

Private Sub Page1_Resize(Width As Int, Height As Int)
   Page1.RootPanel.AddView(lblTitle, 20%x, 10, 60%x, 30)
   Page1.RootPanel.AddView(pnlTest, 10%x, lblTitle.Top + lblTitle.Height + 10, 80%x,
30%y)
   pnlTest.AddView(lblPanelTitle, 20, 10, 100, 30)
   pnlTest.AddView(btnTest, 50, 50, 100, 60)
End Sub
```

Code explanations:

Declaring the views in Process_Globals.

```
Private lblTitle, lblPanelTitle As Label
Private pnlTest As Panel
Private btnTest As Button
```

Initializing the different views in Application_Start:

`lblTitle.Initialize("")`	Initializes the Label, no EventName required.
`lblTitle.Color = Colors.Red`	Sets the Background color to red.
`lblTitle.Font = Font.CreateNew(20)`	Sets the text size to 20.
`lblTitle.TextColor = Colors.Blue`	Sets the text color to blue.
`lblPanelTitle.TextAlignment = lblPanelTitle.ALIGNMENT_CENTER`	Sets the label text alignment to ‘CENTER’.
`lblTitle.Text = "Title"`	Sets the label text to 'Title'.

If the Label had been added in the Designer, all the above code wouldn't have been necessary because the properties would already have been defined in the Designer.

`pnlTest.Initialize("")`	Initializes the Panel, no EventName required.
`pnlTest.Color = Colors.LightGray`	Sets the Background color to light gray.

`btnTest.InitializeCustom("btnTest", Colors.Black, Colors.Blue)`
Initializes the Button, EventName = btnTest, TextColor, PressedTextColor.

`btnTest.SetBorder(1, Colors.Black, 5)`	Sets a Border with the given Width, Color and CornerRadius.
`btnTest.Text = "Test"`	Sets the button text to "Test".

```
lblPanelTitle.Initialize("")
lblPanelTitle.Color = Colors.Red
lblPanelTitle.Font = Font.CreateNew(16)
lblPanelTitle.TextColor = Colors.Blue
lblPanelTitle.TextAlignment = lblPanelTitle.ALIGNMENT_CENTER
lblPanelTitle.Text = "Panel test"
```

Similar to the title Label.

Note that we add the views to their parent views in Page1_Resize and not in Application_Start because the real size of Page1.RootPanel is not known before!

```
Private Sub Page1_Resize(Width As Int, Height As Int)
  Page1.RootPanel.AddView(lblTitle, 20%x, 10, 60%x, 30)
```

Adds the view to the Page1.RootPanel.
In the `Page1.RootPanel.AddView` line we set:

- the Left property to 20%x, 20% of Page1.RootPanel.Width,
- the Top property to 10, 10 points independent of the device scale,
- the Width property to 60%x, 60% of Page1.RootPanel.Width,
- the Height property to 30, 30 points independent of the device scale.

```
Page1.RootPanel.AddView(pnlTest, 10%x, lblTitle.Top + lblTitle.Height + 10, 80%x, 30%y)
```

Adds the Panel pnlTest to the Page1.RootPanel.

- the Left property is set to 0
- the Top property is set to 10 points below the title Label
- the Width property is set to 100%x, the total Page1.RootPanel.Width
- the Height property is set to 30%y, 30% of the Page1.RootPanel.Height

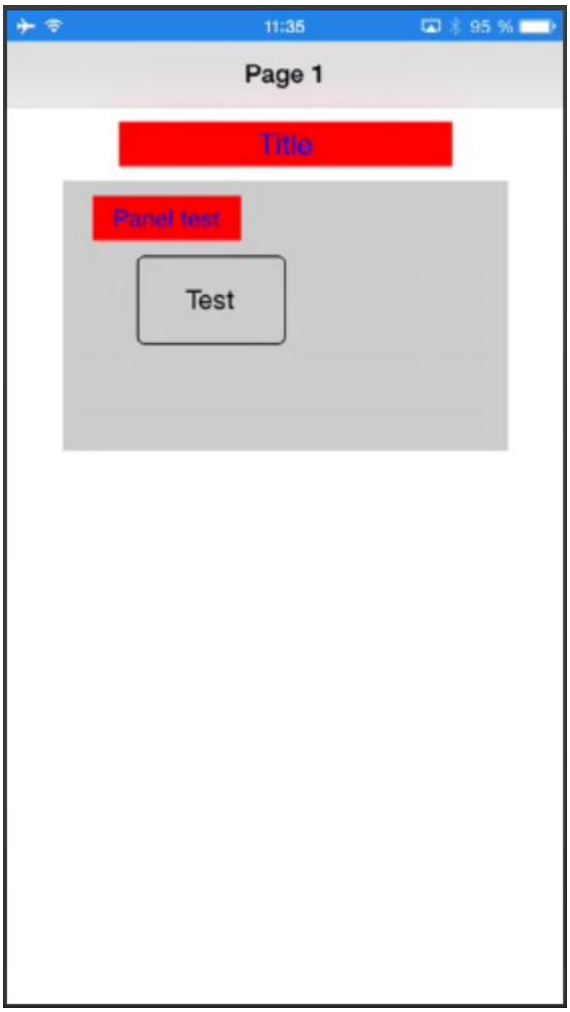
```
pnlTest.AddView(lblPanelTitle, 20, 10, 100, 30)
```

Adds Label lblPanelTitle to Panel pnlTest at the given position and with the given dimensions in points.

```
pnlTest.AddView(btnTest, 50, 50, 100, 60)
```

Adds Button btnTest to Panel pnlTest at the given position and with the given dimensions in points.

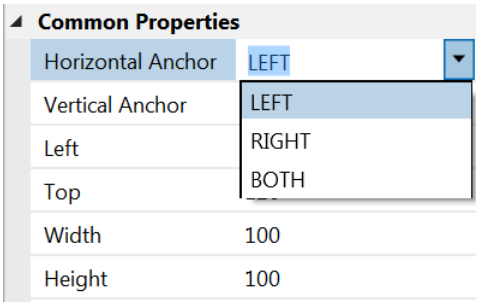
And the result:



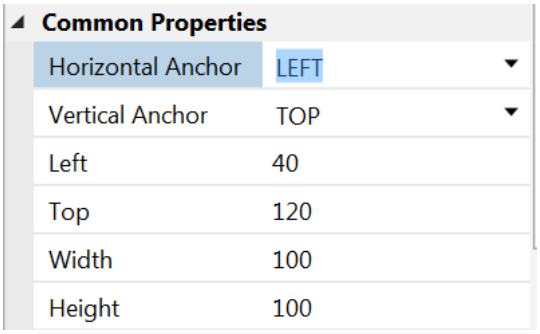
6.10 Anchors

The Designer has two 'special' features to size views, the Horizontal Anchor and the Vertical Anchor.

Horizontal Anchor



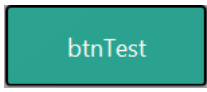
The horizontal anchor property can take three values:

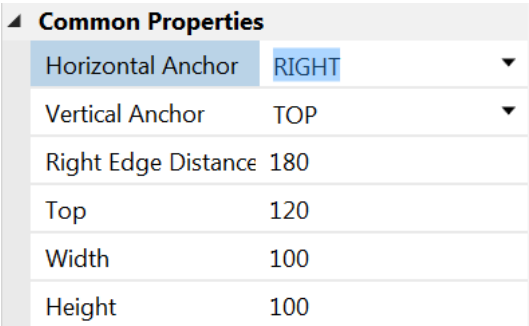


- LEFT

  LEFT is the default value.
  The left edge is anchored to the left edge of the parent view with the distance given in the Left property.

  No anchor symbol is shown.

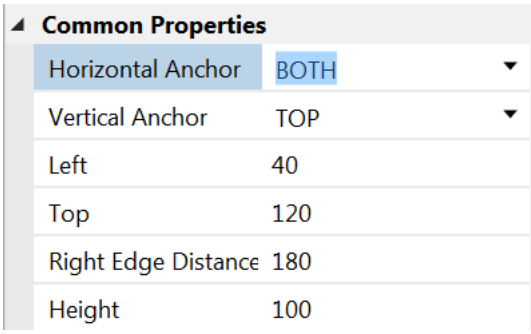




- RIGHT

  The right edge is anchored to the right edge of the parent view with the distance given in the
  Right Edge Distance property.
  The Left property is no more available because it is defined by the width and the right anchor!

  The dot on the right edge shows the anchor.



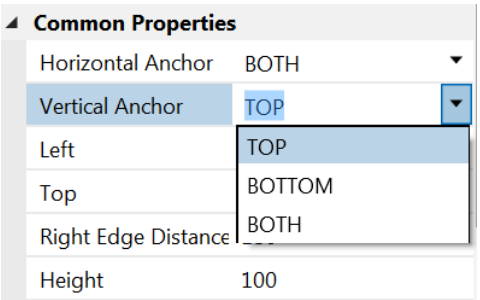
- BOTH

  Both edges are anchored to the parent view with distances defined in the Left and Right Edge Distance properties.
  The Width property is no more available because it is defined by the anchors!

The dots on the two edges show the anchors.

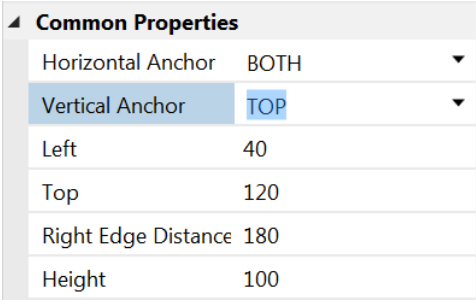
Setting the Horizontal Anchor property to BOTH is similar to the SetLeftAndRight function in the Designer Scripts.

Vertical Anchor

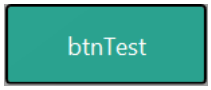


The vertical anchor property can take three values:

- TOP

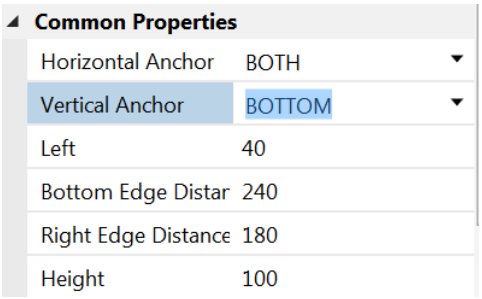


TOP is the default value.
The top edge is anchored to the top edge of the parent view with the distance given in the Top property.
No anchor symbol is shown.



- BOTTOM

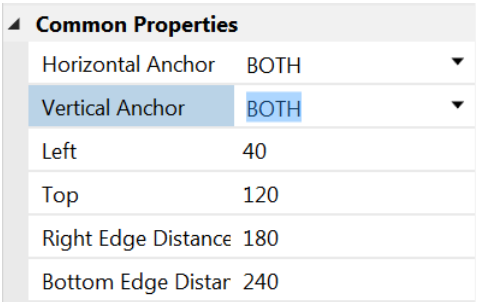
The bottom edge is anchored to the bottom edge of the parent view with the distance given in the Bottom Edge Distance property.



The Top property is no more available because it is defined by the Height and the bottom anchor!
The dot on the bottom edge shows the anchor.

- BOTH

Both edges are anchored to the parent view with distances defined in the Top and Bottom Edge Distance properties.

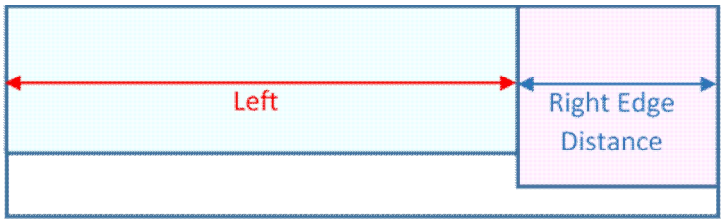


The Height property is no more available because it is defined by the anchors!
The dots on the two edges show the anchors.

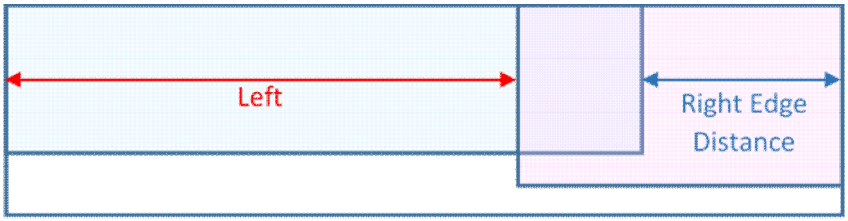
Setting the Vertical Anchor property to BOTH is similar to the SetTopAndBottom function in the Designer Scripts.

What happens when we set the horizontal anchor of the two views below to BOTH and change the parent view width?

The left views right edge is anchored to the right edge of the parent view with the Right Edge Distance.
The right views left edge is anchored to the left edge of the parent view with the Left distance.



If we increase the width of the parent view we get the layout below.



The left views right edge is still at the Right Edge Distance from the parent views right edge.
The right views left edge is still at the Left distance from the parent views left edge.
The result is an overlapping of both views.

In this case you must adjust the views in the Designer Scripts with the `SetLeftAndRight` method!

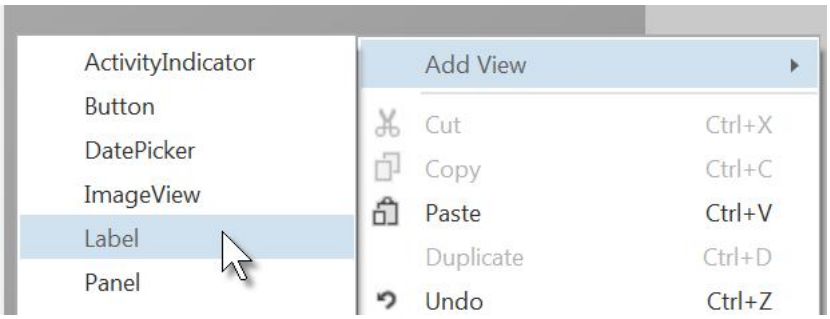
For example:

```
LeftView.SetLeftAndRight(0, 67%x)
RightView.SetLeftAndRight(33%x, 100%x)
```

6.10.1 First example

The examples shown in this chapter are based on the *Anchors* project in the *Designer* folder.

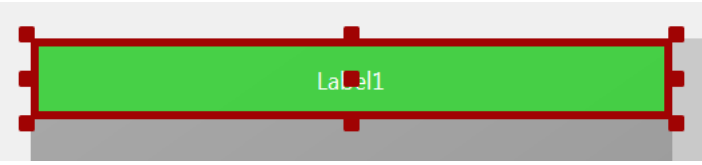
First we add a label on top of the screen which should cover the whole width and stay on top.



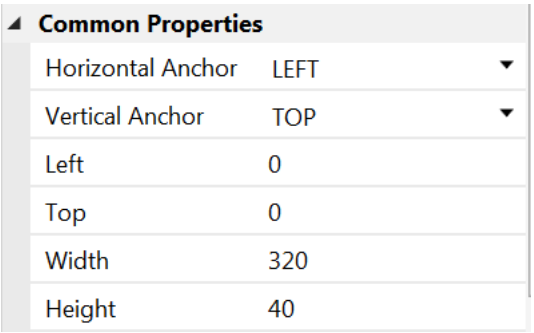
In the AbstractDesigner right click somewhere on the screen, the menu on the left will be displayed:

Click on Add View.

Click on Label.

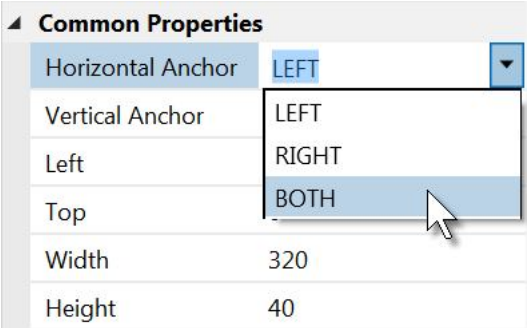


Move the labels upper left corner to the upper left corner of the screen and stretch it to fill the whole width of the screen.



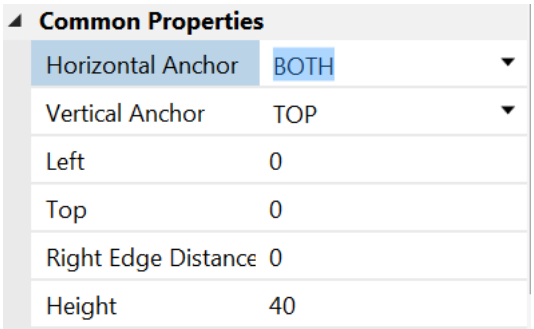
We see these properties:

Left = 0
Top = 0
Width = 320 full layout width
Height = 40



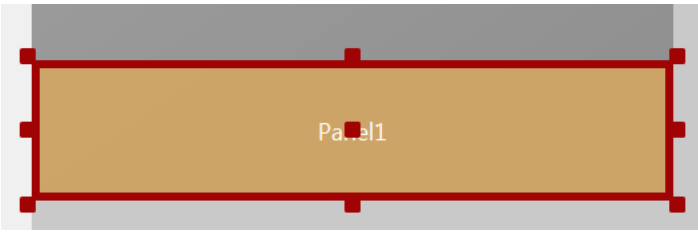
Now we change le 'Horizontal Anchor' property:

Click on BOTH.



We see that the properties changed:
Left, Top and Height are still the same.
But Width has disappeared and is replaced by
Right Edge Distance = 0
Its value = 0 because the right edge is on the right edge of the screen.

Now we see the anchor dots on the left and right edges.



Now, we add a Panel at the bottom of the screen covering also the whole screen width.

Common Properties	
Horizontal Anchor	LEFT
Vertical Anchor	TOP
Left	0
Top	390
Width	320
Height	70

The properties should look like in the picture.

Common Properties	
Horizontal Anchor	BOTH
Vertical Anchor	TOP
Left	0
Top	390
Right Edge Distance	0
Height	70

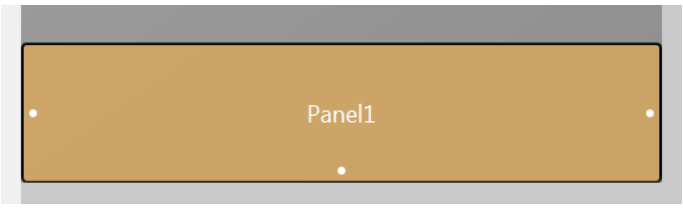
We set the Horizontal Anchor to BOTH.
Same as for Label1.

Common Properties	
Horizontal Anchor	BOTH
Vertical Anchor	TOP
Left	TOP
Top	BOTTOM
Right Edge Distance	BOTH
Height	70

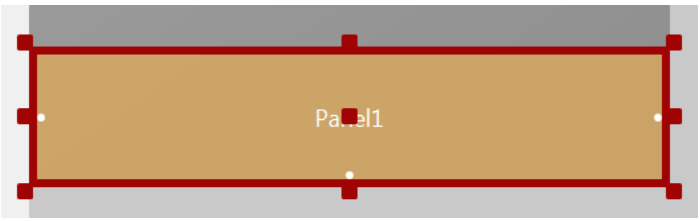
We set the Vertical Anchor to BOTTOM.

Common Properties	
Horizontal Anchor	BOTH
Vertical Anchor	BOTTOM
Left	0
Bottom Edge Distar	0
Right Edge Distance	0
Height	70

The Top property is replaced by the:
Bottom Edge Distance = 0 property.
Its value = 0 because we anchor the bottom edge of Panel1 to the screens bottom edge.

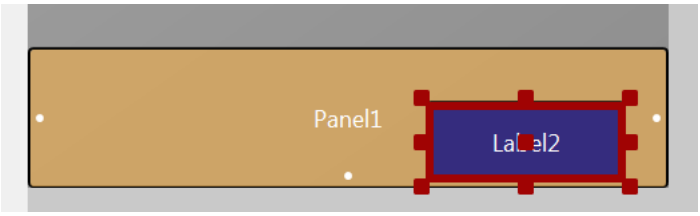


We see the anchor dots on the left, right and bottom edges.

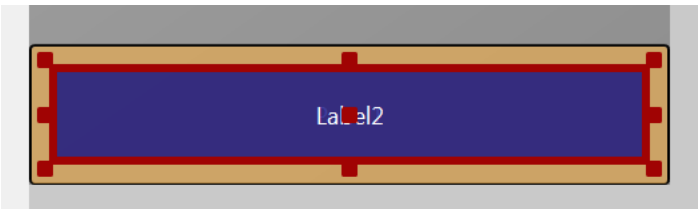


Now we add a second label onto Panel1.

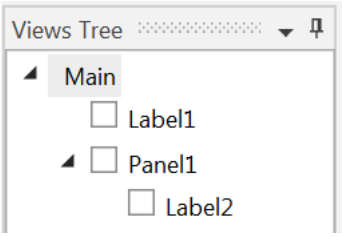
Click on Panel1 to select it.



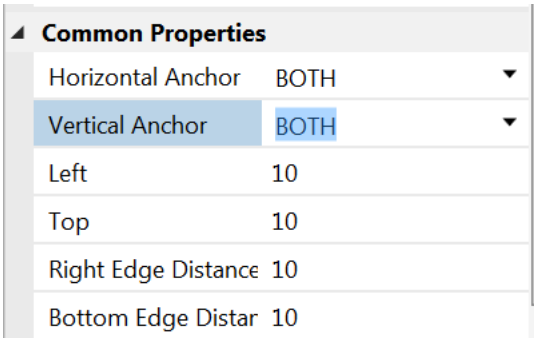
Add the label.



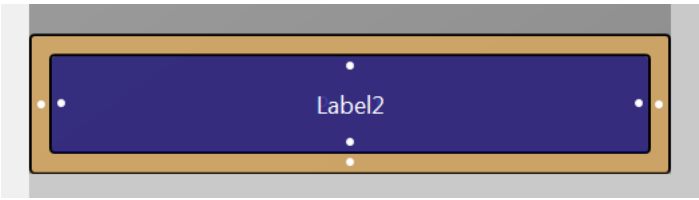
Move and size the label like in the picture with the Left, Top, Width and Height properties like in the list below.



In the Views Tree window we see that Label2 is shifted to the right because its parent view is Panel1 and not Main like for Label1 and Panel1!

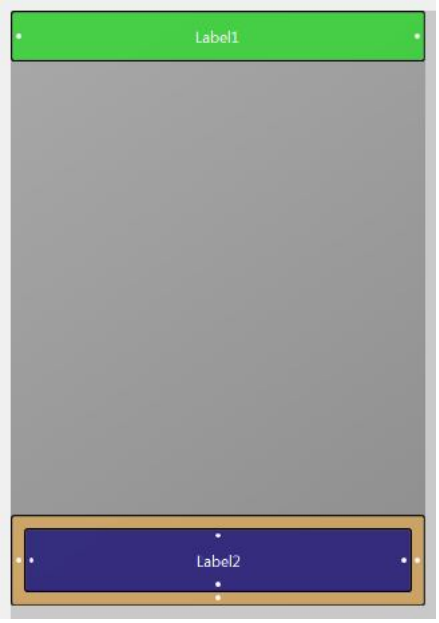


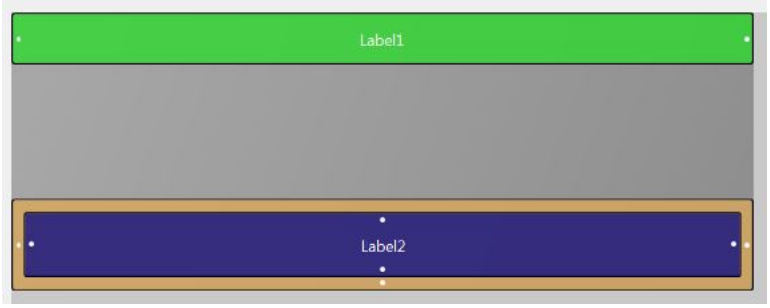
We set the two Anchors to BOTH.
The properties
Left = 10 and
Top = 10 remain the same.
Right Edge Distance = 10 and
Bottom Edge Distance = 10
The two values are equal to 10 because we want a 'frame' around Label2.



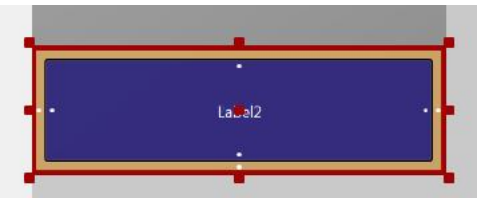
We see the anchor dots on all four edges.

And the result looks like the pictures below in portrait and landscape screen orientations.



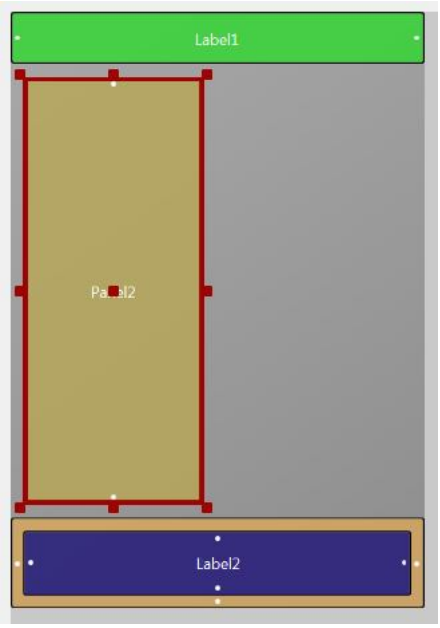


To demonstrate the anchor feature we move, in the Abstract Designer, the top edge of Panel1 upwards.

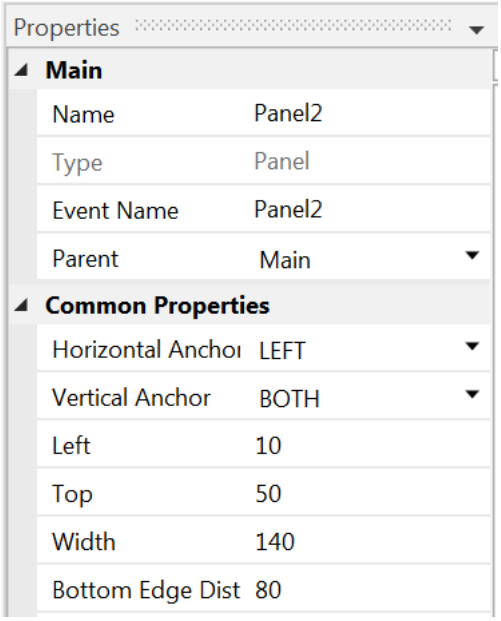


We see that the top edge of Label2 moves with the top edge of Panel1 and the bottom edge remains at its place!

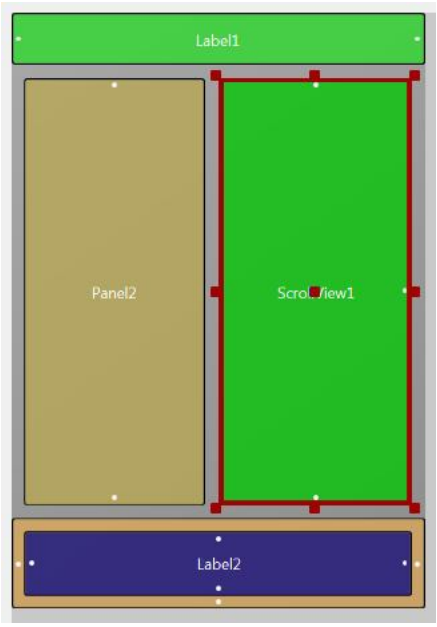
Then move it back to a height of 70.



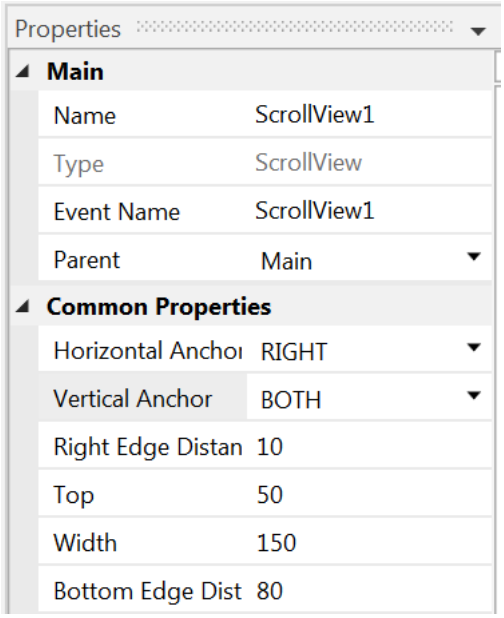
Now, we add another Panel onto the left half of the screen and vertically positioned between Label1 and Panel1 leaving a small space.



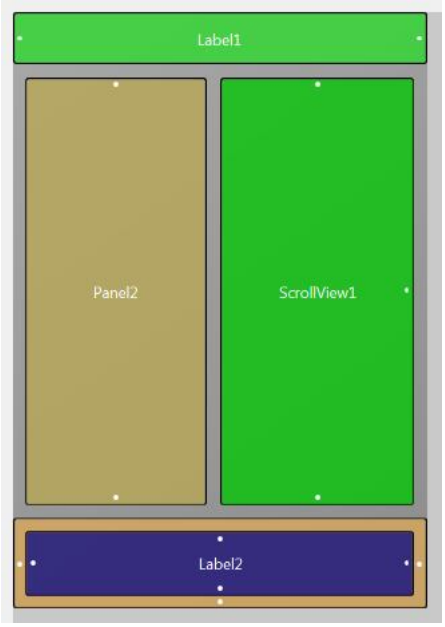
We set the vertical anchor to BOTH.



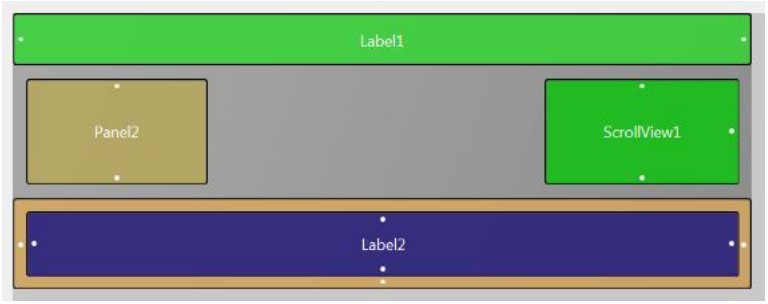
Now, we add a ScrollView on the right half of the screen also positioned between Label1 and Panel1 leaving a small space.



We set the horizontal anchor to RIGHT.
We set the vertical anchor to BOTH.



And the result:
In portrait and landscape screen orientations.



We see that the anchors work fine.
But, we see that there is a big gap
between Panel2 and the ScrollView.

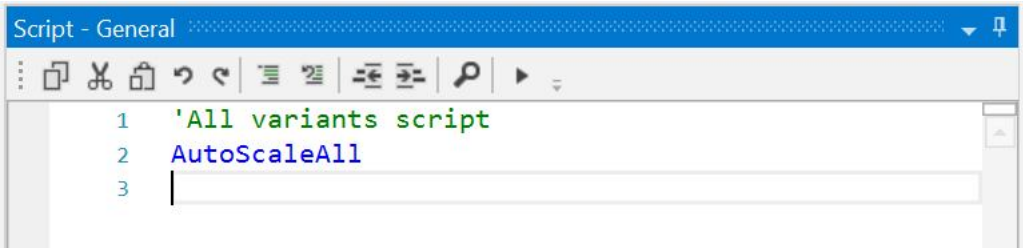
Why do we have this gap?

Because we set the Horizontal Anchor of the ListView to LEFT
and the Horizontal Anchor of the ScrollView to RIGHT.

But the Width property remains the same and that’s why we get the gap between the two views when the screen width is wider than the layout screen width.

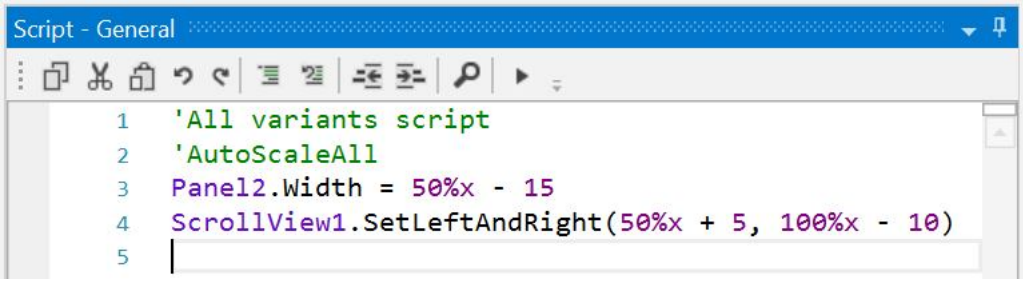
To adjust the width we add two lines in the DesignerScripts.

Click on Designer Scripts to show the DesignerScripts window.



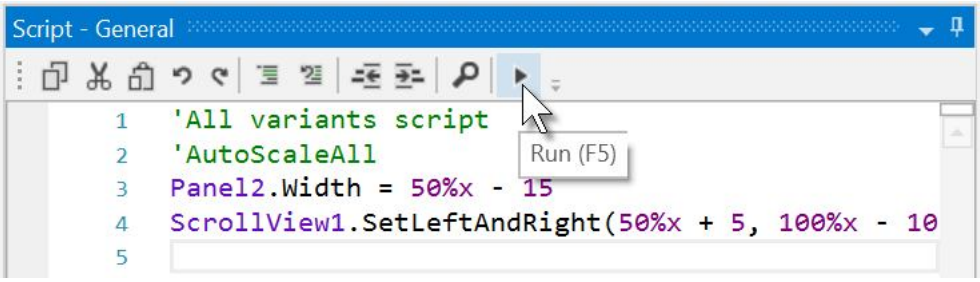
Here we comment AutoScaleAll and add the following two lines:

```
'AutoScaleAll
Panel2.Width = 50%x - 15
ScrollView1.SetLeftAndRight(50%x + 5, 100%x - 10)
```



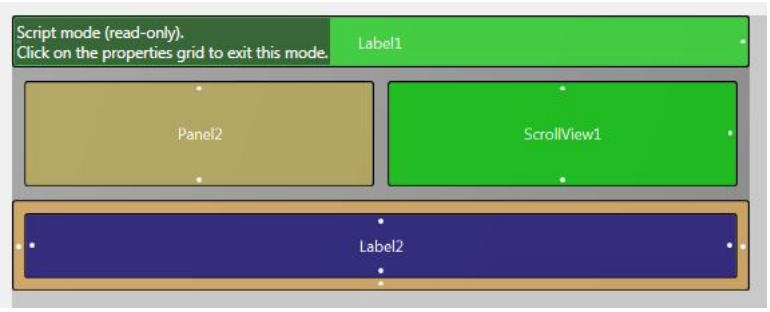
The anchors are valid in the AbstractDesigner but not in Designer Scripts.

For Panel1 it's enough to set its Width property.
But for ScrollView1 we need to define both properties Left and Right which is done with `SetLeftAndRight` because the RIGHT anchor is lost.



In the Script General window click on ▶ to refresh the Abstract Designer.

And the result.



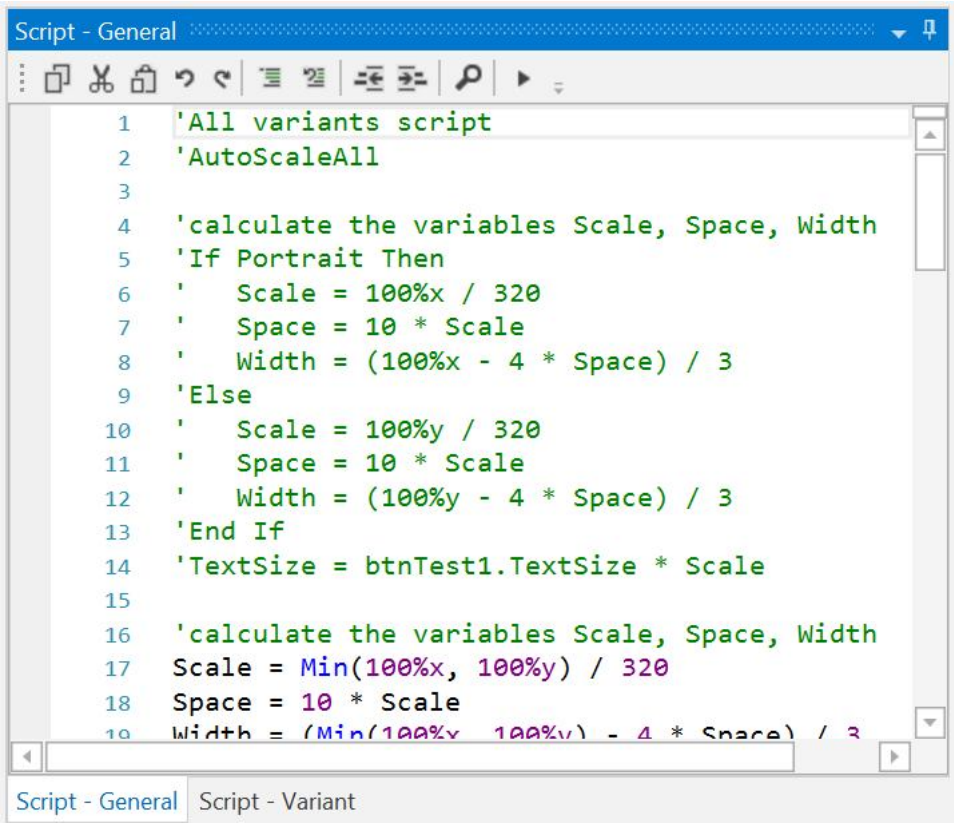
6.11 Designer Scripts

The "Designer Scripts" tool will help you fine tune your layout and easily adjust it to different screens.

It is not recommended to create many layout variants but use AutoScale and the Designer.

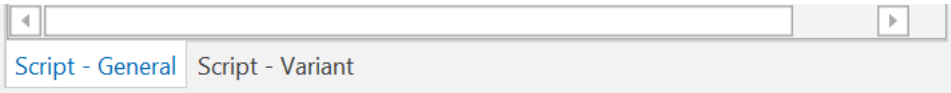
The idea is to combine the usefulness of the visual designer with the flexibility and power of programming code.

You can write a simple script to adjust the layout based on the dimensions of the current device and immediately see the results. You can immediately see the results on the Abstract Designer. This allows you to test your layout on different screen sizes.



6.11.1 General

Every layout file can include script code. The script is written inside the Visual Designer in the Script windows:



There are two types of scripts:
- Script – General, the general script that will be applied to all variants.
- Script – Variant, specific code can be written for each variant.

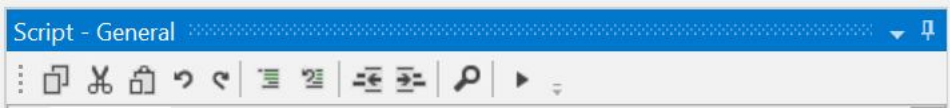
Once you press, in the Script window menu, on the Run Script button (or F5), the script is executed and the connected device / emulator and abstract designer will show the updated layout.

The same thing happens when you run your compiled program. The (now compiled) script is executed after the layout is loaded.

The general script is first executed followed by the variant specific script.

The script language is very simple and is optimized for managing the layout.

6.11.2 The menu



	Ctrl + C	Copy
	Ctrl + X	Cut
	Ctrl + V	Paste
	Ctrl + Z	Undo
	Ctrl + Shift + Z	Redo
	Ctrl + Q	Block Comment
	Ctrl + W	Block Uncomment
		Outdent
		Indent
	F3	Find / Replace
	F5	Run

6.11.3 Supported Properties

The following properties are supported:
- **`Left / Right / Top / Bottom / HorizontalCenter / VerticalCenter`**
  Gets or sets the view's position. The view's width or height will not be changed.

- **`Width / Height`** - Gets or Sets the view's width or height.

- **`TextSize`** - Gets or sets the text size.

- **`Text`** - Gets or sets the view's text. TextSize and Text properties are only available to views that show text.

- **`Visible`** - Gets or sets the view's visible property.

6.11.4 Supported Methods

- **`SetLeftAndRight`** (Left, Right) - Sets the view's left and right properties. This method changes the width of the view based on the two values.

- **`SetTopAndBottom`** (Top, Bottom) - Sets the view's top and bottom properties. This method changes the height of the view based on the two values.

6.11.5 Supported Keywords

- **`And / Or`** - Same as the standard And / Or keywords.

- **`False / True`** - Same as the standard False / True keywords.

- **`Min / Max`** - Same as the standard Min / Max keywords.

- **`Landscape / Portrait`** - Detects if the layout is in landscape or portrait.
  Can be used with If / Then.

- **`AutoScaleAll`** - Autoscales all layout views.

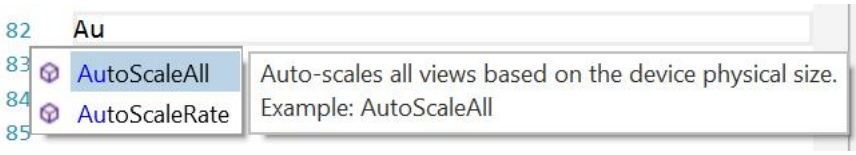
- **`AutoScaleRate`** - Sets the scaling rate, a value between 0 and 1. The default value is 0.3
  Example: `AutoScaleRate(0.5)`

- **`ActivitySize`** - Returns the approximate screen size measured in inches.

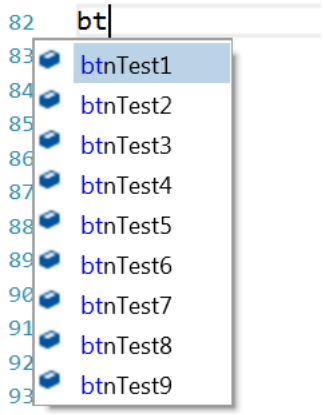
- **`If . Else If . Else .`** condition blocks - Both single line and multiline statements are supported. The syntax is the same as the regular If blocks.

6.11.6 Autocomplete

When you begin writing the Autocomplete function shows all possible keywords or view names containing the written text with the help of the selected keyword.
Example: Au, shows all AutoScale methods.



Example: bt, shows all buttons.



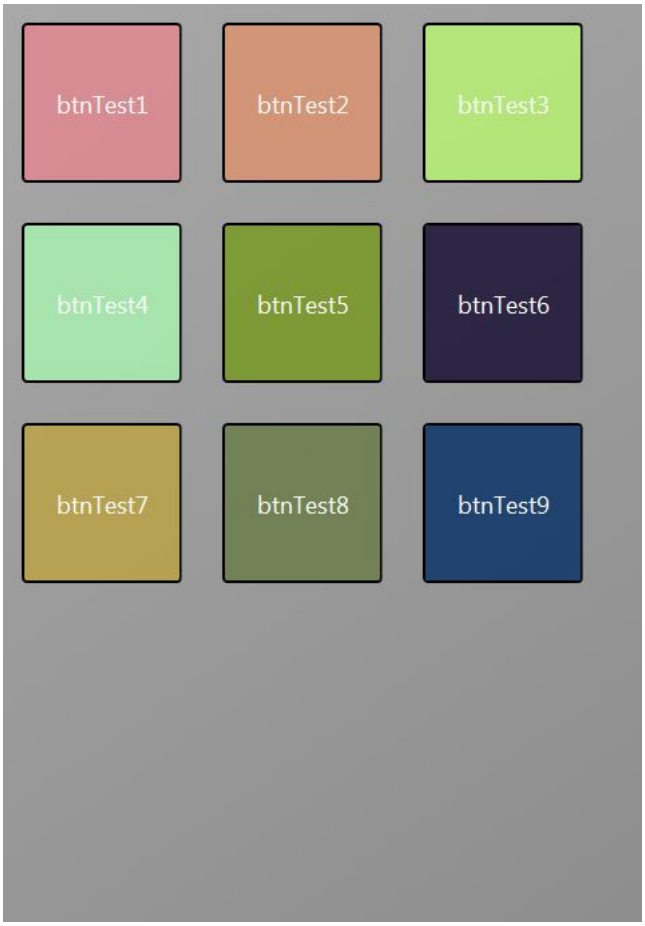
6.11.7 Notes and tips

- %x and %y values are relative to the view that loads the layout.
  Usually it will be the Page. However if you use Panel.LoadLayout then the values are relative to the size of this panel.

- Variables - You can use variables in the script. You do not need to declare the variables before using them (there is no Dim keyword in the script).

6.11.8 Example

In this example we build the following layout with 9 buttons:
The source code is in the *Designer\Scripts* folder.



btnTest5 should be centered in the middle of the screen.
A space of 10 points should be between the screen edges of the smallest screen side and between the buttons. The space and the button width should be increased with the screen size.

The first step is to add the views and position them with the visual designer (you do not need to be accurate).
Now we will select the designer scripts tab and add the code.
Note that the views are locked when the designer scripts tab is selected.

And the code in:

We comment out AutoScaleAll

```
'All variants script
'AutoScaleAll
```

We calculate three variables: Scale, Space and Width.
The value of 320 is the size of the small screen side of the reference variant (width or height).

```
'calculate the variables Scale, Space, Width
If Portrait Then
     Scale = 100%x / 320
     Space = 10 * Scale
     Width = (100%x - 4 * Space) / 3
Else
     Scale = 100%y / 320
     Space = 10 * Scale
     Width = (100%y - 4 * Space) / 3
End If
TextSize = btnTest1.TextSize * Scale
```

This could also have been calculated this way:

```
'calculate the variables Scale, Space, Width
Scale = Min(100%x, 100%y) / 320
Space = 10 * Scale
Width = (Min(100%x, 100%y) - 4 * Space) / 3
TextSize = btnTest1.TextSize * Scale
```

Set the Width and Height properties of the buttons:

```
'set width and height
btnTest1.Width = Width
btnTest1.Height = Width
btnTest2.Width = Width
btnTest2.Height = Width
btnTest3.Width = Width
btnTest3.Height = Width
btnTest4.Width = Width
btnTest4.Height = Width
btnTest5.Width = Width
btnTest5.Height = Width
btnTest6.Width = Width
btnTest6.Height = Width
btnTest7.Width = Width
btnTest7.Height = Width
btnTest8.Width = Width
btnTest8.Height = Width
btnTest9.Width = Width
btnTest9.Height = Width
```

Set the TextSize properties of the buttons:

```
'set the TextSize property
btnTest1.TextSize = TextSize
btnTest2.TextSize = TextSize
btnTest3.TextSize = TextSize
btnTest4.TextSize = TextSize
btnTest5.TextSize = TextSize
btnTest6.TextSize = TextSize
btnTest7.TextSize = TextSize
btnTest8.TextSize = TextSize
btnTest9.TextSize = TextSize
```

We center button btnTest5

```
'position btnTest5 in the screen middle
btnTest5.HorizontalCenter = 50%x
btnTest5.VerticalCenter = 50%y
```

We position the other buttons:

```
'position the other buttons according to btnTest5
btnTest1.Right = btnTest5.Left - Space
btnTest1.Bottom = btnTest5.Top - Space

btnTest2.Left = btnTest5.Left
btnTest2.Bottom = btnTest5.Top - Space

btnTest3.Left = btnTest5.Right + Space
btnTest3.Bottom = btnTest5.Top - Space

btnTest4.Right = btnTest5.Left - Space
btnTest4.Top = btnTest5.Top

btnTest6.Left = btnTest5.Right + Space
btnTest6.Top = btnTest5.Top

btnTest7.Right = btnTest5.Left - Space
btnTest7.Top = btnTest5.Bottom + Space

btnTest8.Left = btnTest5.Left
btnTest8.Top = btnTest5.Bottom + Space

btnTest9.Left = btnTest5.Right + Space
btnTest9.Top = btnTest5.Bottom + Space
```

The result:

3.5’’ 4’’ 4.7’’ 5.5’’

3.5’’ 4’’ 4.7’’

5.5’’

iPad


Script mode (read-only).
Click on the properties grid to exit this mode.


Script mode (read-only).
Click on the properties grid to exit this mode.

6.12 AutoScale

Since B4i version 1.2 two new functions have been added:

- AutoScaleRate(rate)
- AutoScaleAll

Larger devices offer a lot more available space. The result is that even if the physical size of a view is the same, it just "feels" smaller.
Some developers use %x and %y to specify the views size. However the result is far from being perfect. The layout will just be stretched.
The solution is to combine the "dock and fill" strategy with a smart algorithm that increases the views size and text size based on the running device physical size.

The AutoScale function is based on the reference variant you use.

AutoScale calculates a Scale factor depending on the reference layout size and the current device size and multiplies the Left, Top, Width and the Height properties by this Scale factor.
If the view has a Text property, Font Size is also multiplied by this Scale factor.

You can play with the 'rate' value. The rate determines the change amount in relation to the devices physical size.
A value of 0 means no change at all. A value of 1 is almost similar to using %x and %y values. If the physical size is twice the size of the standard phone then the size will be twice the original size.
Values between 0.2 and 0.5 seem to give good results. The default value is 0.3.
Be careful when you ‘downsize’ a layout defined for a big screen to a small screen. The views may become very small.

The abstract designer is useful to quickly test the effect of this value.

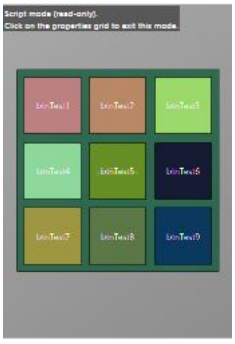
Functions:

- AutoScaleRate(rate) Sets the rate value.
  Example : `AutoScaleRate(0.5)` Sets the rate value to 0.5.

- AutoScaleAll Scales all the views in the selected layout.

6.12.1 Simple example

We use a layout similar to the one used for Designer Scripts.
The source code is in the *Designer / AutoScale* folder.
The main difference is that the 9 buttons are on a panel and not on the screen.



Original layout 4’’ screen.

Result on a 5.5’’ screen with different values of AutoScaleRate:

Rate = 0 0.25 0.5 0.75 1

7 Process life cycle

Each B4i program runs in its own process.

A process starts when the user launches your application.

The process end is less determinant. It will happen sometime after the user has closed the program.

A B4i application is made of one or more Pages.

Pages are somewhat similar to B4A Activities or Windows Forms.

Another delicate point happens when there is a major configuration change in the device. The most common is an orientation change (the user rotates the device). When such a change occurs the current pages are destroyed and then recreated. Now it is possible to create the page according to the new configuration (for example, we now know the new screen dimensions), in the Page_Resize event routine.

7.1 How do we handle it ?

The life cycle of iOS applications is quite simple.

The two most important events are Application_Start and Application_Background.

The standard way to start an application is by clicking on its icon.
This will cause **Application_Start** to run. Application_Start only runs once when the process starts.
It is always the first sub to run.
You will usually load the layout in this sub and optionally restore the state.
Note that at this point the actual page size is not known.

One more important event is **Page_Resize**, this event is raised just after Application_Start or after an orientation change.
In this event the current size is known allowing to adjust view dimensions if needed and initialize Canvases.

The process will continue running until the user closes the application. This happens when the user presses on the home button.

Application_Background will be called when the application is moving to the background. This is the place to save the user data and also to save the state. You should assume that the process will be killed shortly after this sub.

Normal applications do not run in the background. There are no services in iOS.

Application_Active will be raised when the app is active. Which means that the user can interact with it. It will follow Application_Start and Application_Foreground events.

Application_Inactive will be raised when the app is still in the foreground but the user cannot interact with it. It will be raised before Application_Background. It can also be raised when there is an interruption such as a phone call. In this case Application_Inactive will be raised and if the user doesn't answer the call Application_Active will be raised.

Application_Foreground will be called after the app transitions from the background to the foreground. This event will only fire if the process was not killed while the app was in the background.

The last event is **Application_OpenUrl**. This event is fired when another app sends an URL to the system that your app is registered to. This is another way to start applications. This is how for example B4i-Bridge launches the apps during debugging.

When you start the IDE you will see the default template below.

```
'Code module
#Region  Project Attributes
   #ApplicationLabel: B4i Example
   #Version: 1.0.0
   'Orientation possible values: Portrait, LandscapeLeft, LandscapeRight and
PortraitUpsideDown
   #iPhoneOrientations: Portrait, LandscapeLeft, LandscapeRight
   #iPadOrientations: Portrait, LandscapeLeft, LandscapeRight, PortraitUpsideDown
#End Region

Sub Process_Globals
   'These global variables will be declared once when the application starts.
   'Public variables can be accessed from all modules.
   Public App As Application
   Public NavControl As NavigationController
   Private Page1 As Page

End Sub

Private Sub Application_Start (Nav As NavigationController)
   NavControl = Nav
   Page1.Initialize("Page1")
   Page1.Title = "Page 1"
   Page1.RootPanel.Color = Colors.White
   NavControl.ShowPage(Page1)
End Sub

Private Sub Page1_Resize(Width As Int, Height As Int)

End Sub

Private Sub Application_Background

End Sub
```

Variables can be either `Public` or `Private`.
Variables declared in Process_Globals as `Private` accessible only in this module whereas variables declared as `Public` can be accessed from everywhere in the project.
`Private` variables are local to the containing sub. Once the sub ends, these variables no longer exist.
`Public` variables can be accessed from all subs.

7.2 Process global variables

All ‘global’ variables must be declared here, ‘global’ means outsides sub routines.
If you need variables accessible from everywhere in the project you must declare them as `Public`, otherwise as `Private`. `Public` variables live as long as the process lives.
This sub is called once when the process starts (this is true for all modules, not just Main module).
Not all types of objects can be declared as `Public`.
All of the views for example cannot be declared as process global variables.

7.3 Sub Application_Start (Nav As NavigationController)

This sub is called when the application is started.
The activity is created

- when the user first launches the application
- the device configuration has changed (user rotated the device) and the activity was destroyed
- when the activity was in the background and the OS decided to destroy it in order to free memory.

The primary purpose of this sub is to load or create the layout.(among other uses).

7.4 Sub Page1_Resize (Width As Int, Height As Int)

This routine is called each time the page size is changed, mostly after an orientation change.
Generally there are two types of mechanisms that allow you to save the page state.
Information that is only relevant to the current application instance can be stored in one or more process variables.
Other information should be stored in a persistent storage (file or database).
For example, if the user changed some settings you should save the changes to a persistent storage at this point. Otherwise the changes may be lost.

7.5 Application_Background

This routine will be called when the application is moving to the background. This is the place to save the user data and also to save the state. You should assume that the process will be killed shortly after this sub.

8 Variables and objects

A **variable** is a symbolic name given to some known or unknown quantity or information, for the purpose of allowing the name to be used independently of the information it represents. A variable name in computer source code usually associated with a data storage location and thus also its contents, and these may change during the course of program execution (source Wikipedia).

There are two types of variables: primitives and non-primitives types.
Primitives include the numeric types: Byte, Short, Int, Long, Float and Double.
Primitives also include: Boolean and Char.

8.1 Variable Types

List of types with their ranges:

B4i	Type	min value	max value
Boolean	boolean	False	True
Byte	integer 8 bits	-2^{7} -128	$2^{7}-1$ 127
Short	integer 16 bits	-2^{15} - 32768	$2^{15}-1$ 32767
Int	integer 32 bits	-2^{31} -2147483648	$2^{31}-1$ 2147483647
Long	long integer 64 bits	-2^{63} -9223372036854775808	$2^{63}-1$ 9223372036854775807
Float	floating point number 32 bits	-2^{-149} 1.4E-45	$(2-2^{-23}) * 2^{127}$ 3.4028235 E 38
Double	double precision number 64 bits	-2^{-1074} 2.2250738585072014 E -308	$(2-2^{-52}) * 2^{1023}$ 1.7976931348623157 E 308
Char	character		
String	array of characters		

Primitive types are always passed by value to other subs or when assigned to other variables.
For example:

```
Private Sub S1
   Dim A As Int
   A = 12                    The variable A = 12
   S2(A)                     It's passed by value to routine S2
   Log(A)  ' Prints 12       Variable A still equals 12, even though B was changed in routine S2.
End Sub

Private Sub S2(B As Int)     Variable B = 12
   B = 45                    Its value is changed to B = 45
End Sub
```

All other types, including arrays of primitive types and strings are categorized as non-primitive types.
When you pass a non-primitive to a sub or when you assign it to a different variable, a copy of the reference is passed.
This means that the data itself isn't duplicated.
It is slightly different than passing by reference as you cannot change the reference of the original variable.

All types can be treated as Objects.
Collections like lists and maps work with Objects and therefore can store any value.
Here is an example of a common mistake, where the developer tries to add several arrays to a list:

```
Dim arr(3) As Int
Dim List1 As List
List1.Initialize
For i = 1 To 5
   arr(0) = i * 2
   arr(1) = i * 2
   arr(2) = i * 2
   List1.Add(arr)    'Add the whole array as a single item
Next
arr = List1.Get(0) 'get the first item from the list
Log(arr(0))          'What will be printed here???
```

You may expect it to print 2. However it will print 10.
We have created a single array and added 5 references of this array to the list.
The values in the single array are the values set in the last iteration.
To fix this we need to create a new array each iteration.
This is done by calling Dim each iteration:

```
Dim arr(3) As Int 'This call is redundant in this case.
Dim List1 As List
List1.Initialize
For i = 1 To 5
   Dim arr(3) As Int
   arr(0) = i * 2
   arr(1) = i * 2
   arr(2) = i * 2
   List1.Add(arr) 'Add the whole array as a single item
Next
arr = List1.Get(0) 'get the first item from the list
Log(arr(0)) 'Will print 2
```

8.2 Names of variables

It is up to you to give any name to a variable, except reserved words.
A variable name must begin with a letter and must be composed by the following characters A-Z, a-z, 0-9, and underscore "_", no spaces, no brackets etc.
Variable names are case insensitive, that means that Index and index refer to the same variable.

But it is good practice to give them meaningful names.
Example:

Interest = Capital * Rate / 100	is meaningful
n1 = n2 * n3 / 100	not meaningful

For Views it is useful to add to the name a three character prefix that defines its type.
Examples:

`lblCapital`	lbl > Label	Capital > purpose
`txfInterest`	txf > TextField	Interest > purpose
`btnNext`	btn > Button	Next > purpose

8.3 Declaring variables

8.3.1 Simple variables

Variables are declared with the `Private`, `Public` or `Dim` keyword followed by the variable name and the `As` keyword and followed by the variable type.

`Public` Declares variables accessible from everywhere in the program.
Valid only in `Process_Globals` and `Class_Globals` routines !

`Private` Declares variables accessible only in the environment where they are declared.
Variables declared in a Module are accessible only in this Module.
Variables declared in a Sub are accessible only in this Sub.

`Dim` Can be used in Subs.
Any variable declared in a sub is `Private` and accessible only in this Sub.
Declaring a variable as `Public` in a Sub is non sense.

Examples:

```
Public Capital As Double
Public Interest As Double
Public Rate As Double
```

Declares three variables as Double, double precision numbers.

```
Private i As Int
Private j As Int
Private k As Int
```

Declares three variables as Int, integer numbers.

```
Private txfCapital As TextField
Private txfInterest As TextField
Private txfRate As TextField
```

Declares three variables as TextField views.

```
Private btnNext As Button
Private btnPrev As Button
```

Declares two variables as Button views.

Variables of same type can also be declared in a short way.

```
Private Capital, Interest, Rate As Double
Private i, j, k As Int
Private txfCapital, txfInterest, txfRate As TextField
Private btnNext, btnPrev As Button
```

The names of the variables separated by commas and followed by the type declaration.

Following variable declarations are also valid:

```
Private i = 0, j = 2, k = 5 As Int

Private txt = "test" As String, value = 1.05 As Double, flag = False As Boolean
```

View names must be declared if we want to use them in the code.
For example, if we want to change the text in a TextField view in the code, like
`txfCapital.Text = "1200"`,
we need to reference this TextField view by its name `txfCapital`, this is done with the Private declaration.
If we never make any reference to this TextField view anywhere in the code no declaration is needed.
Using an event routine for that view doesn't need a declaration either.

To allocate a value to a variable write its name followed by the equal sign and followed by the value, like:

```
Capital = 1200
LastName = "SMITH"
```

Note that for `Capital` we wrote just `1200` because `Capital` is a number.
But for `LastName` we wrote `"SMITH"` because `LastName` is a string.
Strings must always be written between double quotes.

8.3.2 Array variables

Arrays are collections of data or objects that can be selected by indices. Arrays can have multiple dimensions.
The declaration contains the `Public` or `Private` keyword followed by the variable name `LastName`, the number of items between brackets `(50)`, the keyword `As` and the variable type `String`.

Examples:

`Public LastName(50) As String`	One dimension array of strings, total number of items 50.
`Public Matrix(3, 3) As Double`	Two dimensions array of Doubles, total number of items 9.
`Public Data(3, 5, 10) As Int`	Three dimensions array of integers, total number of items 150.

The first index of each dimension in an array is 0.
`LastName(0), Matrix(0,0), Data(0,0,0)`

The last index is equal to the number of items in each dimension minus 1.
`LastName(49), Matrix(2,2), Data(2,4,9)`

This example shows how to access all items in a three dimensional array.

```
For i = 0 To 2
   For j = 0 To 2
      For k = 0 To 2
         Data(i, j, k) = ...
      Next
   Next
Next
```

A more versatile way to declare arrays is to use:
Variables instead of numbers.

```
Public NbPers = 50 As Int
Public LastName(NbPers) As String
Public FirstName(NbPers) As String
Public Address(NbPers) As String
Public City(NbPers) As String
```

```
Public LastName(50) As String
Public FirstName(50) As String
Public Address(50) As String
Public City(50) As String
```

We declare the variable `NbPers As Int` and set its value to 50, `NbPers = 50`.
Then we declare the arrays with this variable instead of the number 50.
The big advantage is if at some point we need to change the number of items, we change only ONE value.
For the Data array we could use the following code.

```
Public NbX = 2 As Int
Public NbY = 5 As Int
Public NbZ = 10 As Int
Public Data(NbX, NbY, NbZ) As Int
```

And the access routine.

```
For i = 0 To NbX - 1
   For j = 0 To NbY -
      For k = 0 To NbZ - 1
         Data(i, j, k) = ...
      Next
   Next
Next
```

Filling an array with the Array keyword :

```
Public Name() As String
Name = Array As String("Miller", "Smith", "Johnson", "Jordan")
```

8.3.3 Array of views (objects)

Views or objects can also be in an Array. The following code shows an example:
The individual names (b1, b2 etc.) must also be declared.

In the example below the Buttons are added to the Application by code.

```
Sub Process_Globals
   Private Buttons(7) As Button
End Sub

Private Sub Application_Start (Nav As NavigationController)
   Private i As Int

   For i = 0 To 6
      Buttons(i).Initialize("Buttons")
      Activity.AddView(Buttons(i), 10, 10 + i * 60, 150, 50)
      Buttons(i).Tag = i + 1
      Buttons(i).Text = "Test " & (i + 1)
   Next
End Sub

Sub Buttons_Click
   Private btn As Button

   btn = Sender

   Activity.Title = "Button " & btn.Tag & " clicked"
End Sub
```

The Buttons could also have been added in a layout file, in that case they must neither be initialized, nor added to the Activity and the Text and Tag properties should also be set in the Designer.
In that case the code would look like this:

```
Sub Process_Globals
   Private btn1, btn2, btn3, btn4, btn5, btn6, btn7 As Button
   Private Buttons() As Button
End Sub

Private Sub Application_Start (Nav As NavigationController)
   Dim i As Int

   Buttons = Array As Button (btn1, btn2, btn3, btn4, btn5, btn6, btn7)
End Sub

Sub Buttons_Click
   Dim btn As Button

   btn = Sender

   Activity.Title = "Button " & btn.Tag & " clicked"
End Sub
```

8.3.4 Type variables

A Type cannot be private. Once declared it is available everywhere (similar to Class modules).
The best place to declare them is in the Process_Globals routine in the Main module.

Let us reuse the example with the data of a person.
Instead of declaring each parameter separately, we can define a personal type variable with the Type keyword:

```
Public NbUsers = 50 As Int
Type Person(LastName As String, FirstName As String. Address As String, City As String)
Public User(NbUsers) As Person
Public CurrentUser As Person
```

The new personal type is `Person`, then we declare either single variables or arrays of this personal type.

Before you can use a Type variable it must be initialized!

```
User().Initialize
CurrentUser.Initialize
```

To access a particular item use following code.

```
CurrentUser.FirstName
CurrentUser.LastName

User(1).LastName
User(1).FirstName
```

The variable name, followed by a dot and the desired parameter.
If the variable is an array then the name is followed by the desired index between brackets.

It is possible to assign a typed variable to another variable of the same type, as shown below.

```
CurrentUser = User(1)
```

8.4 Casting

B4i casts types automatically as needed. It also converts numbers to strings and vice versa automatically.
In many cases you need to explicitly cast an Object to a specific type.
This can be done by assigning the Object to a variable of the required type.
For example, Sender keyword references an Object which is the object that raised the event.
The following code changes the color of the pressed button.
Note that there are multiple buttons that share the same event sub.

```
Sub Process_Globals
   Dim Btn1, Btn2, Btn3 As Button
End Sub

Private Sub Application_Start (Nav As NavigationController)
   Btn1.Initialize("Btn")    ' Note the same EventName for all three buttons
   Btn2.Initialize("Btn")
   Btn3.Initialize("Btn")
   Page1.RootPanel.AddView(Btn1, 10, 10, 200, 50)
   Page1.RootPanel.AddView(Btn2, 10, 70, 200, 50)
   Page1.RootPanel.AddView(Btn3, 10, 130, 200, 50)
End Sub

Private Sub Btn_Click
   Dim btn As Button
   btn = Sender      ' Cast the Object to Button
   btn.Color = Colors.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255))
End Sub
```

The above code could also be written more elegantly:

```
Sub Process_Globals

End Sub

Private Sub Application_Start (Nav As NavigationController)
   Dim i As Int
   For i = 0 To 9   ' create 10 Buttons
      Dim Btn As Button
      Btn.Initialize("Btn")
      Page1.RootPanel.AddView(Btn, 10dip, 10dip + 60dip * i, 200dip, 50dip)
   Next
End Sub

Sub Btn_Click
   Dim btn As Button
   btn = Sender      ' Cast the Object to Button
   btn.Color = Colors.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255))
End Sub
```

8.5 Scope

8.5.1 Process variables

Variables must be declared in `Process_Globals` or `Class_Globals` routines.
They can be declared with either `Public` or `Private:`
`Public` variables can be accessed from any other module.
`Private` variables can be accessed only from the module they are declared in.

Variables can be declared with the `Dim` keyword:
```
Dim MyVar As String
```
In this case the variable is public same as Public. `Dim` remains for compatibility.
Using `Dim` in `Process_Globals` or `Class_Globals` routines is not recommended!

These subs are called once when the process starts.

Not all types of objects can be declared as `Public` variables.
All of the views for example cannot be declared as process variables.

To access `Public` variables in other modules than the module where they were declared their names must have the module name they were declared as a prefix.
Example:
Variable defined in a module with the name : *MyModule*
```
Sub Process_Globals
   Public MyVar As String
End Sub
```

Accessing the variable in *MyModule* module:
```
   MyVar = "Text"
```

Accessing the variable in any other module:
```
   MyModule.MyVar = "Text"
```

8.5.2 Local variables

Variables declared in a subroutine are local to this subroutine even if they are declared as `Public` which is none sense.
They are `Private` and can only be accessed from within the subroutine where they were declared.
All objects types can be declared as local variables.
At each call of the subroutine the local variables are initialized to their default value or to any other value you have defined in the code and are 'destroyed' when the subroutine is left.

8.6 Tips

A view can be assigned to a `Private` variable so you can easily change the common properties of the view.

For example, the following code disables all Buttons a Page:

```
For Each v As View In Page1.RootPanel.GetAllViewsRecursive
   If v Is Button Then
      Private b As Button = v
      b.Enabled = False
   End If
Next
```

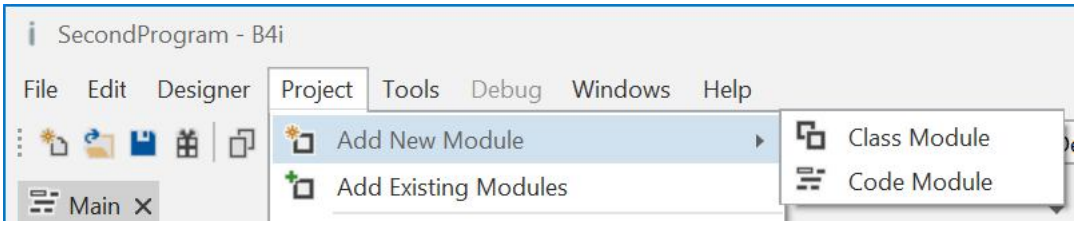
9 Modules

At least one module exists, the main module.
Its name is always **Main** and cannot be changed.

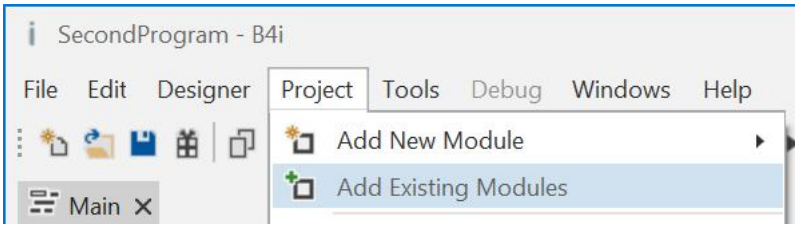
There do exist two types of modules:

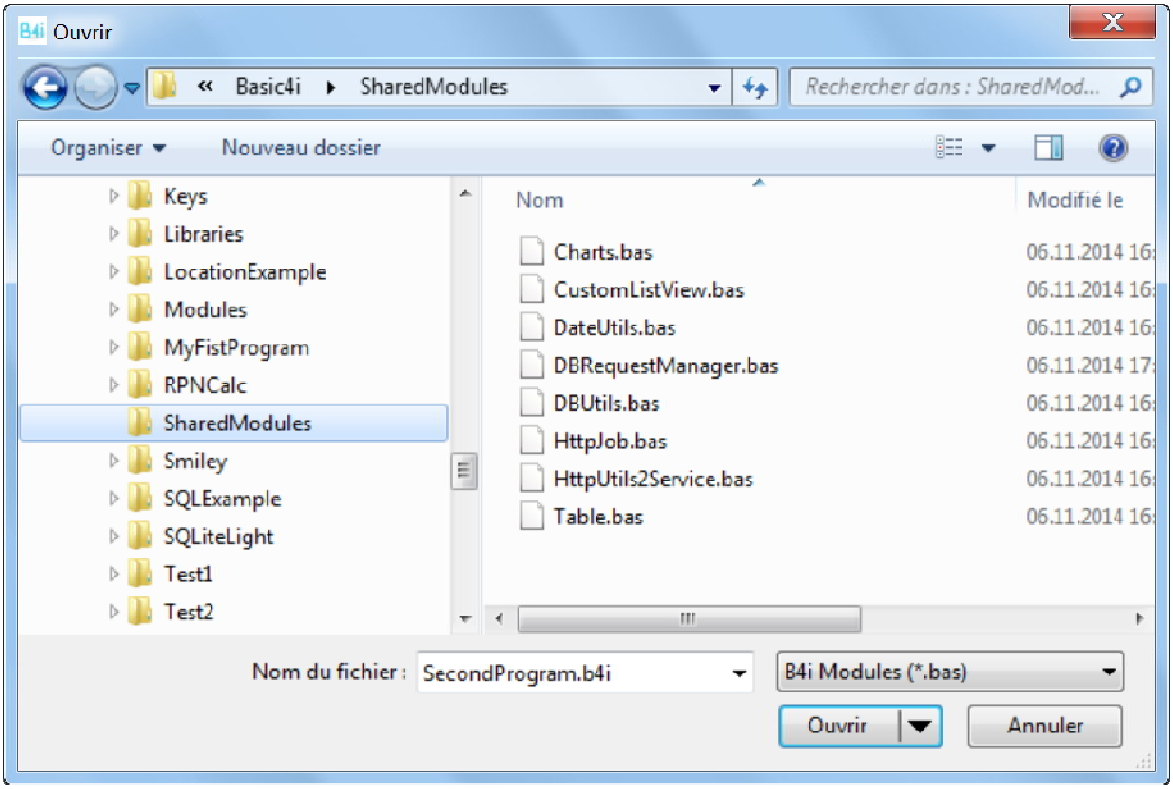
- Class modules
- Code modules

To add a new module click on either Class Module or Code Module in the IDE menu Project / Add New Module.



To add an existing module click on Add Existing Module in the IDE menu Project.





Select the file to add, it will be copied to the project folder.

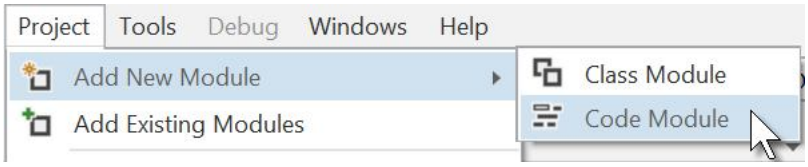
9.1 Code modules

Code modules contain code only.

The purpose and advantage of code modules is sharing same code in different programs, mainly for calculations or other general management.

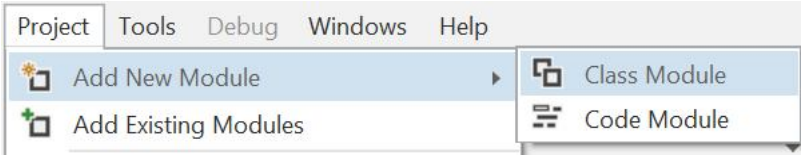
Some code modules, called utilities, are already published by Erel in the forum:

- DBUtils, Database management utilities.
- DateUtils, Date calculations.
- ChartsFrameWork, Charts drawing module.



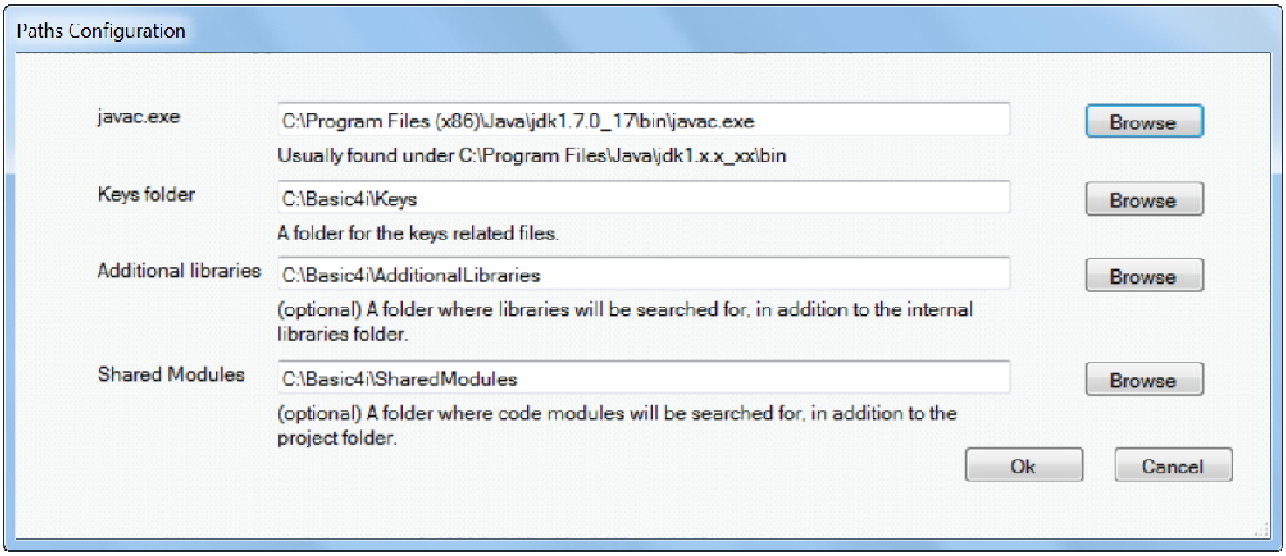
9.2 Class modules

Class modules are out of the scope of this guide.

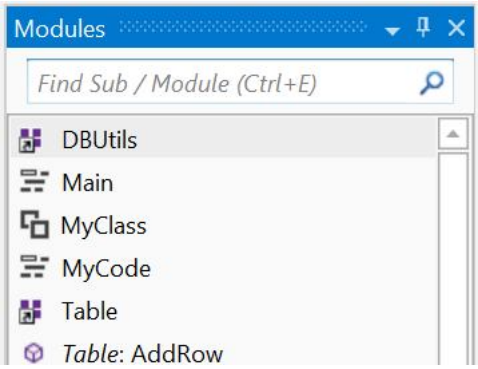


9.3 Shared modules

Modules are in principle saved it the folder of each project using it.
Modules useful in several project can be shared without loading them in each projects folder, that's the purpose of *shared modules*.
Shared module files must be stored in a specific 'Shared Modules' folder which must be defined in the IDE menu *Tools - Configure Paths*.



You can see that a module was loaded from the shared folder in the list of modules:



Main and MyCode are code modules.

MyClass Class module.

DBUtils Shared module.

Adding a shared module to a project is done in the same way as adding a non-shared module.
You choose Project -> Add Existing Module. If the module file is in the 'SharedModules' folder then the module will be loaded as a shared module and will not be copied to the project folder.

If you want to convert a non-shared module to a shared module then you need to manually move the module file to the shared modules folder and reload the project

10 Basic language

In computer programming, BASIC (an acronym which stands for **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode) is a family of high-level programming languages designed to be easy to use. The original Dartmouth BASIC was designed in 1964 by John George Kemeny and Thomas Eugene Kurtz at Dartmouth College in New Hampshire, USA to provide computer access to non-science students. At the time, nearly all use of computers required writing custom software, which was something only scientists and mathematicians tended to do. The language and its variants became widespread on microcomputers in the late 1970s and 1980s.
BASIC remains popular to this day in a handful of highly modified dialects and new languages influenced by BASIC such as Microsoft Visual Basic (source Wikipedia).

10.1 Program flow

This is a summary of the more detailed explanations in Process and Activity life cycle.

```
Sub Process_Globals
   'These global variables will be declared once when the application starts.
   'Public variables can be accessed from all modules.
   Public App As Application
   Public NavControl As NavigationController
   Private Page1 As Page
End Sub

Private Sub Application_Start (Nav As NavigationController)
   NavControl = Nav
   Page1.Initialize("Page1")
   Page1.Title = "Page 1"
   Page1.RootPanel.Color = Colors.White
   NavControl.ShowPage(Page1)
End Sub

Private Sub Page1_Resize(Width As Int, Height As Int)

End Sub

Private Sub Application_Background

End Sub
```

The program goes through following routines when starting from top to down:

10.1.1 Process_Globals routine

Dedicated to the declaration of variables and objects.
Variables declared with:

- `Private` are valid only in the module where they are declared.
- `Public` are valid during the whole life time of the process and accessible from everywhere in the program.

10.1.2 Application_Start routine

Area to initialize, add views and to define view properties, if necessary.
No dimensions! Dimension can only be defined in Page_Resize.
When you run the IDE a default template is included.

```
Private Sub Application_Start (Nav As NavigationController)
   NavControl = Nav
   Page1.Initialize("Page1")
   Page1.Title = "Page 1"
   Page1.RootPanel.Color = Colors.White
   NavControl.ShowPage(Page1)
End Sub
```

10.1.3 Page1_Resize routine

This routine is called every time a page is resized, at program start or screen orientation change.
Here you can set any setup parameters depending on the Page size.
Canvases must be initialized here!

10.1.4 Application_Background routine

When the program is closed this routine is called.
Here you need to save the current parameters of the program you want get back after a screen orientation change or when you start the program again.

10.1.5 Other Application event routines

10.1.5.1 Application_Foreground

This event is raised is called when the Application is in the Foreground, visible on the screen.

10.1.5.2 Application_Active

This event is raised is called when the Application is Active.

10.1.5.3 Application_Inactive

This event is raised is called when the Application gets Inactive.

10.2 Expressions

An expression in a programming language is a combination of explicit values, constants, variables, operators, and functions that are interpreted according to the particular rules of precedence and of association for a particular programming language, which computes and then produces (returns) another value. This process, like for mathematical expressions, is called evaluation. The value can be of various types, such as numerical, string, and logical (source Wikipedia).

For example, 2 + 3 is an arithmetic and programming expression which evaluates to 5. A variable is an expression because it is a pointer to a value in memory, so y + 6 is an expression. An example of a relational expression is 4 = 4 which evaluates to True (source Wikipedia).

10.2.1 Mathematical expressions

Operator	Example	Precedence level	Operation
`+`	`x + y`	3	Addition
`-`	`x - y`	3	Subtraction
`*`	`x * y`	2	Multiplication
`/`	`x / y`	2	Division
`Mod`	`x Mod y`	2	Modulo
`Power`	`Power(x,y)` x^y	1	Power of
`Logarithm`	`Logarithm(x, y)`	1	Logarithm of

Precedence level: In an expression, operations with level 1 are evaluated before operations with level 2, which are evaluated before operations with level 3.

Examples:

4 + 5 * 3 + 2 = 21 > 4 + 15 + 2

(4 + 5) * (3 + 2) = 45 > 9 * 5

$(4 + 5)^2 * (3 + 2) = 405$ > $9^2 * 5$ > 81 * 5
`Power(4+5,2)*(3+2)`

11 Mod 4 = 3 > Mod is the remainder of 10 / 4

23^3 `Power(23,3)` > 23 at the power of 3

$- 2^2 = - 4$
$(-2)^2 = 4$

10.2.2 Relational expressions

In computer science in relational expressions an operator tests some kind of relation between two entities. These include numerical equality (e.g., 5 = 5) and inequalities (e.g., 4 >= 3).
In B4i these operators return **True** or **False**, depending on whether the conditional relationship between the two operands holds or not (source Wikipedia).

Operator	Example	Used to test
=	x = y	the equivalence of two values
<>	x <> y	the negated equivalence of two values
>	x > y	if the value of the left expression is greater than that of the right
<	x < y	if the value of the left expression is less than that of the right
>=	x >= y	if the value of the left expression is greater than or equal to that of the right
<=	x <= y	if the value of the left expression is less than or equal to that of the right

10.2.3 Boolean expressions

In computer science, a Boolean expression is an expression that produces a Boolean value when evaluated, i.e. one of **True** or **False**. A Boolean expression may be composed of a combination of the Boolean constants **True** or **False**, Boolean-typed variables, Boolean-valued operators, and Boolean-valued functions (source Wikipedia).

Boolean operators are used in conditional statements such as IF-Then and Select-Case.

Operator	Comment
Or	Boolean Or Z = X Or Y Z = True if X or Y is equal to True or both are True
And	Boolean And Z = X And Y Z = True if X and Y are both equal to True
Not ( )	Boolean Not X = True Y = Not(X) > Y = False

		Or	And
X	Y	Z	Z
False	False	False	False
True	False	True	False
False	True	True	False
True	True	True	True

10.3 Conditional statements

Different conditional statements are available in Basic.

10.3.1 If – Then – End If

The **If - Then - Else** structure allows to operate conditional tests and execute different code sections according to the test result.
General case:

```
If test1 Then
   ' code1
Else If test2 Then
   ' code2
Else
   ' code3
End If
```

The **If - Then - Else** structure works as follows:

1. When reaching the line with the **`If`** keyword, **`test1`** is executed.
2. If the test result is **True**, then **`code1`** is executed until the line with the **`Else If`** keyword. And jumps to the line following the **`End If`** keyword and continues.
3. If the result is **False**, then **`test2`** is executed.
4. If the test result is **True**, then **`code2`** is executed until the line with the **`Else`** keyword. And jumps to the line following the **`End If`** keyword and continues.
5. If the result is **False**, then **`code3`** is executed and continues at the line following the **`End If`** keyword.

The tests can be any kind of conditional test with two possibilities **True** or **False**.
Some examples:

```
If b = 0 Then
   a = 0
End If
```

The simplest **If - Then** structure.

```
If b = 0 Then a = 0
```

The same but in one line.

```
If b = 0 Then
   a = 0
Else
   a = 1
End If
```

The simplest **If - Then - Else** structure.

```
If b = 0 Then a = 0 Else a = 1
```

The same but in one line.

Personally, I prefer the structure on several lines, better readable.
An old habit from HP Basic some decades ago, this Basic accepted only one instruction per line.

Note. Difference between:

B4i / B4A	VB
`Else If`	`ElseIf`

In B4A there is a blank character between **`Else`** and **`If`**.

Some users try to use this notation:

```
If b = 0 Then a = 0 : c = 1
```

There is a big difference between B4i and VB that gives errors:
The above statements is equivalent to:

B4i / B4A

```
If b = 0 Then
   a = 0
End If
c = 1
```

VB

```
If b = 0 Then
   a = 0
   c = 1
End If
```

The colon character ' : ' in the line above is treated in B4i like a CarriageReturn CR character.

10.3.2 Select – Case

The **Select - Case** structure allows to compare a **TestExpression** with other **Expressions** and to execute different code sections according to the matches between the **TestExpression** and **Expressions**.

General case:

```
Select TestExpression
Case ExpressionList1
   ' code1
Case ExpressionList2
   ' code2
Case Else
   ' code3
End Select
```

TestExpression is the expression to test.

ExpressionList1 is a list of expressions to compare to **TestExpression**

ExpressionList2 is another list of expressions to compare to **TestExpression**

The **Select - Case** structure works as follows:

1. The **TestExpression** is evaluated.
2. If one element in the **ExpressionList1** matches **TestExpression** then executes **code1** and continues at the line following the **End Select** keyword.
3. If one element in the **ExpressionList2** matches **TestExpression** then executes **code2** and continues at the line following the **End Select** keyword.
4. For no expression matches **TestExpression** executes **code3** and continues at the line following the **End Select** keyword.

TestExpression can be any expression or value.
ExpressionList1 is a list of any expressions or values.

Examples:

```
Select Value
Case 1, 2, 3, 4
```

The Value variable is a numeric value.

```
Select a + b
Case 12, 24
```

The **TestExpression** is the sum of a + b

```
Select Txt.CharAt
Case "A", "B", "C"
```

The **TestExpression** is a character at

```
Sub Panel1_Touch (Action As Int, X As Float, Y As Float)
   Select Action
   Case Panel1.ACTION_DOWN

   Case Panel1.ACTION_MOVE

   Case Panel1.ACTION_UP

   End Select
End Sub
```

Note. Differences between:

B4i / B4A	VB
`Select Value`	`Select Case Value`
`Case 1,2,3,4,8,9,10`	`Case 1 To 4 , 8 To 9`

In VB the keyword `Case` is added after the `Select` keyword.
VB accepts `Case 1 To 4` , this is not implemented in B4i.

10.4 Loop structures

Different loop structures are available in Basic.

10.4.1 For – Next

In a **For – Next** loop a same code will be executed a certain number of times.
Example:

```
For i = n1 To n2 Step n3
   ' Specific code
Next
```

i	incremental variable
n1	initial value
n2	final value
n3	step

The **For – Next** loop works as below:

1. At the beginning, the incremental variable **i** is equal to the initial value **n1**.
   i = n1
2. The specific code between the **For** and **Next** keywords is executed.
3. When reaching **Next**, the incremental variable **i** is incremented by the step value **n3**.
   i = i + n3.
4. The program jumps back to **For**, compares if the incremental variable **i** is lower or equal to the final value **n2**.
   test if i <= n2
5. If **Yes**, the program continues at step 2, the line following the **For** keyword.
6. If **No**, the program continues at the line following the **Next** keyword.

If the step value is equal to '+1' the step keyword is not needed.

```
For i = 0 To 10
Next
```

is the same as

```
For i = 0 To 10 Step 1
Next
```

The step variable can be negative.

```
For i = n3 To 0 Step -1

Next
```

It is possible to exit a For – Next loop with the `Exit` keyword.

```
For i = 0 To 10
   ' code
   If A = 5 Then Exit
   ' code
Next
```

In this example, if the variable a equals 5

Then exit the loop.

Note: Differences between

B4i / B4A	**VB**
`Next`	`Next i`
`Exit`	`Exit For`

In VB:

- The increment variable is added after the **`Next`** Keyword.
- The loop type is specified after the **`Exit`** keyword.

10.4.2 For - Each

It is a variant of the For - Next loop.

Example:

```
For Each n As Type In Array
   ' Specific code
Next
```

n	variable any type or object
Type	type of variable n
Array	Array of values or objects

The **For – Each** loop works as below:

1. At the beginning, **n** gets the value of the first element in the Array.
   n = Array(0)
2. The specific code between the **`For`** and **`Each`** keywords is executed.
3. When reaching **Next**, the program checks if **n** is the last element in the array.
4. If **No**, the variable **n** gets the next value in the Array and continues at step 2, the line following the **`For`** keyword.
   n = Array(next)
5. If **Yes**, the program continues at the line following the **`Each`** keyword.

Example For - Each:

```
Public Numbers() As Int
Public Sum As Int

Numbers = Array As Int(1, 3, 5, 2, 9)

Sum = 0
For Each n As Int In Numbers
   Sum = Sum + n
Next
```

Same example but with a For - Next loop :

```
Public Numbers() As Int
Public Sum As Int
Public i As Int

Numbers = Array As Int(1, 3, 5 , 2, 9)

Sum = 0
For i = 0 To Numbers.Length - 1
   Sum = Sum + Numbers(i)
Next
```

This example shows the power of the For - Each loop:

```
For Each lbl As Label In Page1.RootPanel
   lbl.Font = Font.CreateNew(20)
Next
```

Same example with a For - Next loop :

```
For i = 0 To Page1.RootPanel.NumberOfViews - 1
   Private v As View
   v = Page1.Rootctivity.GetView(i)
   If v Is Label Then
      Private lbl As Label
      lbl = v
      lbl.Font = Font.CreateNew(20)
   End If
Next
```

10.4.3 Do - Loop

Several configurations exist:

```
Do While test
   ' code
Loop
```

test is any expression
Executes the **code** while **test** is **True**

```
Do Until test
   ' code
Loop
```

test is any expression
Executes the **code** until **test** is **True**

The **Do While - Loop** loop works as below:

1. At the beginning, **test** is evaluated.
2. If **True**, then executes **code**
3. If **False** continues at the line following the **Loop** keyword.

The **Do Until - Loop** loop works as below:

1. At the beginning, **test** is evaluated.
2. If **False**, then executes **code**
3. If **True** continues at the line following the **Loop** keyword.

It is possible to exit a Do-Loop structure with the Exit keyword.

```
Do While test
   ' code
   If a = 0 Then Exit
   ' code
Loop
```

If **a = 0** then exit the loop

Examples:

Do Until Loop :

```
Private i, n As Int

i = 0
Do Until i = 10
   ' code
   i = i + 1
Loop
```

Do While Loop:

```
Private i, n As Int

i = 0
Do While i < 10
   ' code
   i = i + 1
Loop
```

Read a text file and fill a List:

```
Private lstText As List
Private line As String
Private tr As TextReader

tr.Initialize(File.OpenInput(File.DirDocuments, "test.txt"))
lstText.Initialize
line = tr.ReadLine
Do While line <> Null
   lstText.Add(line)
   line = tr.ReadLine
Loop

tr.Close
```

Note : Difference between:

B4i / B4A	**VB**
`Exit`	`Exit Loop`

In VB the loop type is specified after the **`Exit`** keyword.

VB accepts also the following loops, which are not supported in B4i.

```
Do                          Do
  ' code                      ' code
Loop While test             Loop Until test
```

10.5 Subs

A Subroutine ("Sub") is a piece of code. It can be any length, has a distinctive name and a defined scope (in the means of variables scope discussed earlier). In Basci4i code, a subroutine is called "Sub", and is equivalent to procedures, functions, methods and subs in other programming languages. The lines of code inside a Sub are executed from first to last.
It is not recommended to have too long Subs, they get less readable.

10.5.1 Declaring

A Sub is declared in the following way:

```
Public Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

It starts with the keyword `Private` or `Public`, depending on the scope, followed by the keyword `Sub`, followed by the Subs name, followed by a parameter list, followed by the return type and ends with the keywords `End Sub`.
Subs are always declared at the top level of the module, you cannot nest two Subs one inside the other.

10.5.2 Calling a Sub

When you want to execute the lines of code in a Sub, you simply write the Sub's name.

For example:

```
   Interest = CalcInterest(1234, 5.2)
```

Interest	Value returned by the Sub.
CalcInterest	Sub name.
1235	Capital value transmitted to the Sub.
5.25	Rate value transmitted to the Sub.

10.5.3 Calling a Sub from another module

A subroutine declared in a code module can be accessed from any other module but the name of the routine must have the name of the module where it was declared as a prefix.

Example: If the *CalcInterest* routine is declared in module *MyModule* then calling the routine must be :

```
   Interest = MyModule.CalcInterest(1234, 5.2)
```

instead of:

```
   Interest = CalcInterest(1234, 5.2)
```

10.5.4 Naming

Basically, you can name a Sub any name that is legal for a variable. It is recommended to name the Sub with a significant name, like **CalcInterest** in the example, so you can tell what it does from reading the code.
There is no limit on the number of Subs you can add to your program, but it is not allowed to have two Subs with the same name in the same module.

```
Public Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

10.5.5 Parameters

Parameters can be transmitted to the Sub. The list follows the sub name. The parameter list is put in brackets.
The parameter types should be declared directly in the list.

```
Public Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

In B4i, the parameters are transmitted by value and not by reference.

It is possible to transmit views or objects to Subs, like:

```
Public Sub MySub(lbl As Label)
```

10.5.6 Returned value

A sub can return a value, this can be any object.
Returning a value is done with the Return keyword.
The type of the return value is added after the parameter list.

```
Public Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

10.6 Events

In Object-oriented programming we have objects which can react on different user actions called events.
The number and the type of events an object can raise depend on the type of the object.
User interface objects are called 'Views' in iOS.

Summary of the events for different views:

	Events													
Views	Click	LongClick	BeginEdit	EndEdit	EnterPressed	TextChanged	Touch	Resize	ScrollChanged	ValueChanged	ItemSelected	IndexChanged	OverrideUrl	PageFinished
Button	■	■												
TextField	■	■	■	■	■	■								
TextView	■	■												
ImageView	■	■												
Label	■	■												
Panel	■	■					■	■						
ScrollView	■	■							■					
Slider	■	■								■				
Picker	■	■									■			
Stepper	■	■								■				
Switch	■	■								■				
SegmentedControl	■	■										■		
Slider	■	■								■				
Stepper	■	■								■				
WebView	■	■											■	■

The most common events are:

- **Click** Event raised when the user clicks on the view.
  Example:
  ```
  Private Sub Button1_Click
    ' Your code
  End Sub
  ```

- **LongClick** Event raised when the user clicks on the view and holds it pressed for a while.
  Example:
  ```
  Private Sub Button1_LongClick
    ' Your code
  End Sub
  ```

- **Touch** `(Action As Int, X As Float, Y As Float)`
  Event raised when the user touches a Panel on the screen.

  Three different actions are handled:
  - Panel.ACTION_DOWN, the user touches the screen.
  - Panel.ACTION_MOVE, the user moves the finger without leaving the screen.
  - Panel.ACTION_UP, the user leaves the screen.

  The X and Y coordinates of the finger positions are given in Points not in Pixels.

  Example:
  ```
  Private Sub Panel_Touch (Action As Int, X As Float, Y As Float)
    Select Action
    Case Panel.ACTION_DOWN
      ' Your code for DOWN action
    Case Panel.ACTION_MOVE
      ' Your code for MOVE action
    Case Panel.ACTION_UP
      ' Your code for UP action
    End Select
  End Sub
  ```

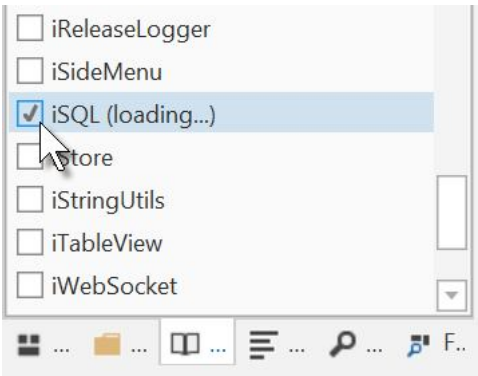
10.7 Libraries

Libraries add more objects and functionalities to B4i.
Some of these libraries are shipped with B4i and are part of the standard development system.
Others, often developed by users, can be downloaded (by registered users only) to add supplementary functionalities to the B4i development environment.

All B4i libraries have “i” as a prefix.

When you need a library, you have to:

- Check in the Lib Tab, if you already have the library.
- For additional libraries, check if it's the latest version.
- If **yes**, then check the library in the list to select it.



- If **no**, download the library, unzip it and copy the <LibraryName>.xml file to the additional libraries folder.
- Right click in the Lib area and click on 

and check the library in the list to select it.

10.7.1 Standard libraries

The standard B4i libraries are saved in the Libraries folder in the B4i program folder.
Normally in: C:\Program Files\Anywhere Software\B4i\Libraries

10.7.2 Additional libraries folder

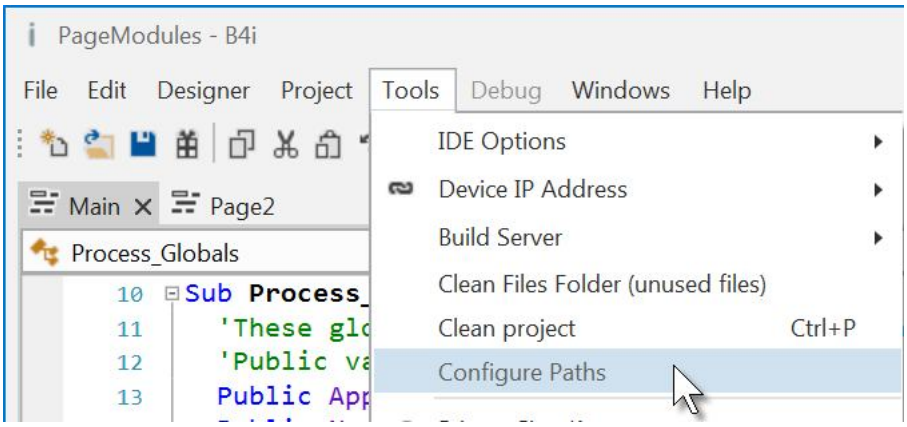
For the additional libraries it is useful to setup a special folder to save them somewhere else.
For example: C:\B4i\AdditionalLibraries

When you install a new version of B4i, all standard libraries are automatically updated, but the additional libraries are not included. The advantage of the special folder is that you don't need to care about them because this folder is not affected when you install the new version of B4i.
The additional libraries are not systematically updated with new version of B4i.

When the IDE starts, it looks first for the available libraries in the Libraries folder of B4i and then in the folder for the additional libraries.

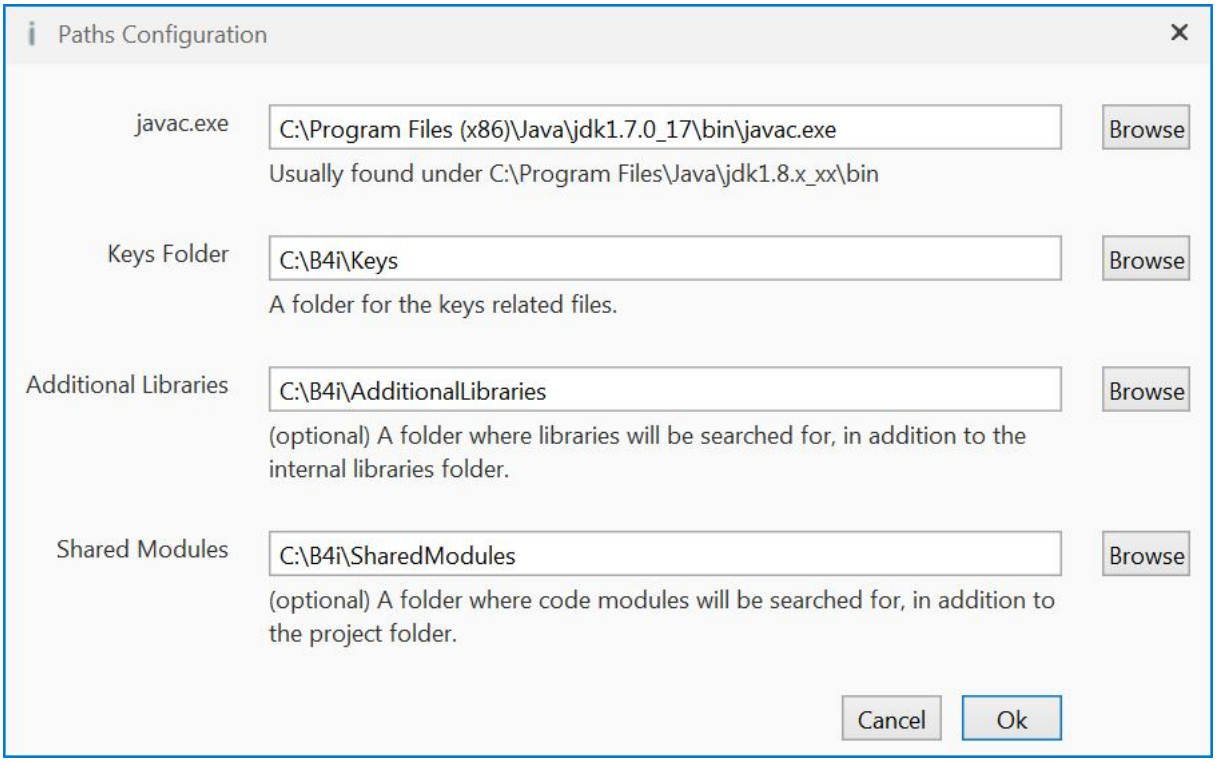
If you setup a special additional libraries folder you must specify it in the IDE.

In the menu Tools / Configure Paths:



Enter the folder name and click on Ok.

10.7.3 Load and update a Library



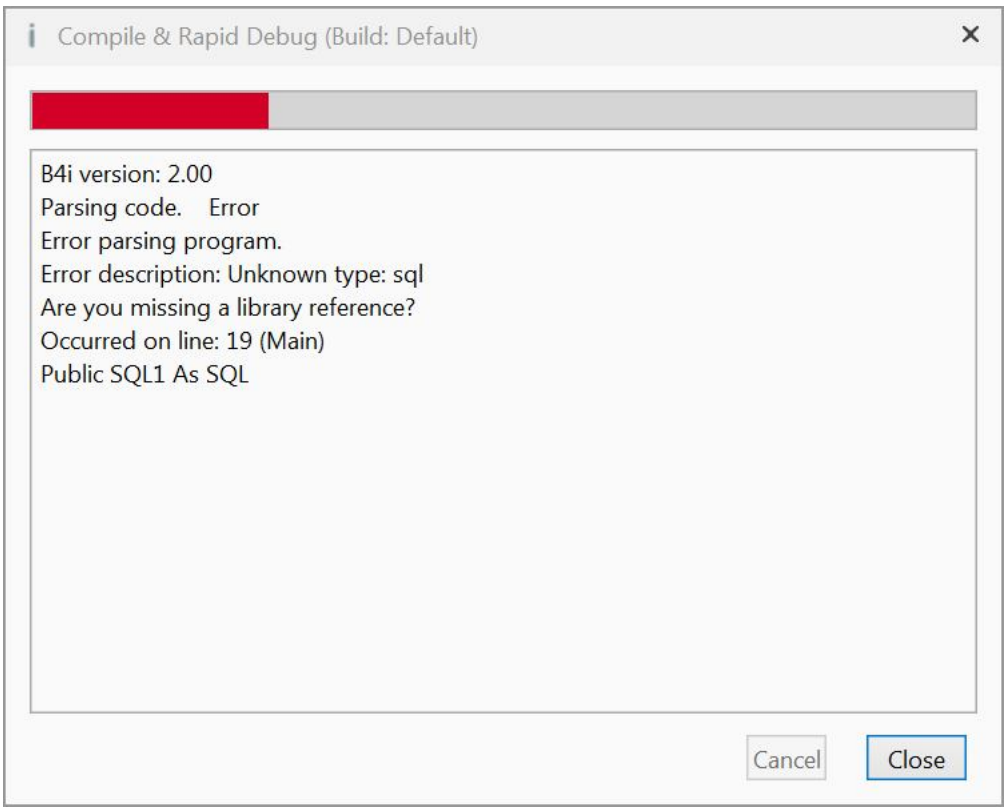
A list of the official and additional libraries with links to the relevant forum threads is shown in B4i Libraries.

To load or update a library follow the steps below:

- Download the library zip file somewhere.
- Unzip it.
- Copy the xxx.xml file to the
  - B4i Library folder for a standard B4i library
  - Additional libraries folder for an additional library.
- Right click in the Lib area and click on Refresh and check the library in the list to select it.

10.7.4 Error message "Are you missing a library reference?"

If you get this message, means that you forgot to check the specified library in the Lib Tab list!



In the program line where the error occurs the unknown object is highlighted in red.

```
18
19    Public SQL1 As SQL
20  End Sub
```

10.8 String manipulation

B4i allows string manipulations like other Basic languages but with some differences.

These manipulations can be done directly on a string.
Example:

```
txt = "123,234,45,23"
txt = txt.Replace(",", ";")
```

Result: `123; 234; 45; 23`

The different functions are:

- **CharAt(Index)** Returns the character at the given index.
- **CompareTo(Other)** Lexicographically compares the string with the Other string.
- **Contains(SearchFor)** Tests whether the string contains the given SearchFor string.
- **EndsWith(Suffix)** Returns True if the string ends with the given Suffix substring.
- **EqualsIgnoreCase(Other)** Returns True if both strings are equal ignoring their case.
- **GetBytes(Charset)** Encodes the Charset string into a new array of bytes.
- **IndexOf(SearchFor)** Returns the index of the first occurrence of SearchFor in the string.
- **IndexOf2(SearchFor, Index)** Returns the index of the first occurrence of SearchFor in the string. Starts searching from the given index.
- **LastIndexOf(SearchFor)** Returns the index of the first occurrence of SearchFor in the string. Starts searching from the end of the string.
- **LastIndexOf2(SearchFor, Index)** Returns the index of the first occurrence of SearchFor in the string. The search starts at the given index and advances to the beginning.
- **Length** Returns the length, number of characters, of the string.
- **MeasureHeight(Font)** Returns the height of this string drawn with the given font.
- **MeasureWidth(Font)** Returns the width of this string drawn with the given font.
- **Replace(Target, Replacement)** Returns a new string resulting from the replacement of all the occurrences of Target with Replacement.
- **StartsWith(Prefix)** Returns True if this string starts with the given Prefix.
- **Substring(BeginIndex)** Returns a new string which is a substring of the original string. The new string will include the character at BeginIndex and will extend to the end of the string.
- **Substring2(BeginIndex,EndIndex)** Returns a new string which is a substring of the original string. The new string will include the character at BeginIndex and will extend to the character at EndIndex, not including the last character.
- **ToLowerCase** Returns a new string which is the result of lower casing this string.
- **ToUpperCase** Returns a new string which is the result of upper casing this string.
- **Trim** Returns a copy of the original string without any leading or trailing white spaces.

Number formatting, display numbers as strings with different formats, there are two keywords:

- **NumberFormat**(Number As Double, MinimumIntegers As Int, MaximumFractions As Int)
  `NumberFormat(12345.6789, 0, 2)` = 12,345.68
  `NumberFormat(1, 3 ,0)` = 001
  `NumberFormat(Value, 3 ,0)` variables can be used.
  `NumberFormat(Value + 10, 3 ,0)` arithmetic operations can be used.
  `NumberFormat((lblScore.Text + 10), 0, 0)` if one variable is a string add parentheses.

- **NumberFormat2**(Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean)
  `NumberFormat2(12345.67, 0, 3, 3, True)` = 12,345.670

10.9 Timers

A Timer object generates ticks events at specified intervals. Using a timer is a good alternative to a long loop, as it allows the UI thread to handle other events and messages.

A timer has:

- Two methods.
  - **Initialize** Initializes the timer with two parameters, the EventName and the interval.
    Timer1.Initialize(EventName As String, Interval As Long)
    Ex: `Timer1.Initialize("Timer1", 1000)`

  - **IsInitialized** Returns True if the Timer is initialized.
    Timer1.IsInitialized
    Ex: `Init = Timer1.IsInitialized`

- Two properties.
  - **Interval** Sets the timer interval in milli-seconds.
    Timer1. Interval = Interval
    Ex: `Timer1.Interval = 1000`, 1 second

  - **Enabled** Enables or disables the timer. **It is False by default.**
    Ex: `Timer1.Enabled = True`

- One Event
  - **Tick** The Tick routine is called every time interval.
    Ex: `Sub Timer1_Tick`

The Timer must be declared in a Process_Global routine.

```
Sub Process_Globals
  Public Timer1 As Timer
```

But it must be initialized in the Application_Start routine in the module where the timer tick event routine is used.

```
Private Sub Application_Start (Nav As NavigationController)
  Timer1.Initialize("Timer1", 1000)
```

And the Timer Tick event routine.
This routine will be called every second (1000 milli-seconds) by the operating system.

```
Private Sub Timer1_Tick
  ' Do something
End Sub
```

10.10 Files

Many applications require access to a persistent storage. The two most common storage types are files and databases.

10.10.1 File keyword

The predefined keyword `File` has a number of functions for working with files.

Files locations - There are several important locations where you can read or write files.

File.DirAssets
Returns a reference to the files added with the file manager in the IDE.
These files are read-only.

File.DirDocuments
The documents folder should only be used to store user generated content. It is possible to make this folder sharable through iTunes.
This folder is backed up by iTunes automatically.

File.DirLibrary
The place for any non-user generated persistent files.
This folder is backed up by iTunes automatically.
You can create a subfolder named Caches, Files under that folder will not be backed up.

File.DirTemp
A temporary folder. Files in this folder are not backed up by iTunes and may be deleted from time to time.

File.Exists (Dir As String, FileName As String)
Returns True if the file exists and False if not.

The File object includes several methods for writing to files and reading from files.

To be able to write to a file or to read from a file, it must be opened.

File.OpenOutput (Dir As String, FileName As String, Append As Boolean)
- Opens an InputStream to the given file. The Append parameter tells whether the text will be added at the end of the existing file or not. If the file doesn't exist it will be created.

File.OpenInput (Dir As String, FileName As String)
- Opens an InputStream to the given file.

File.WriteString (Dir As String, FileName As String, Text As String)
- Writes the string to a new file with UTF-8 encoding.

File.WriteString2 (Dir As String, FileName As String, Text As String, CharSet As String)
- Writes the string to a new file with the specified encoding.

File.ReadString (Dir As String, FileName As String) As String
- Reads a file and returns its content as a string with UTF-8 encoding.

File.ReadString2 (Dir As String, FileName As String) As String
- Reads a file and returns its content as a string with the specified encoding.

File.WriteList (Dir As String, FileName As String, List As List)
- Writes a list of strings or numbers to a text file. Each item is written in a single line.

File.ReadList (Dir As String, FileName As String) As List
- Reads the given text file and returns a list. Each line in the file is converted to a list item.

File.WriteMap (Dir As String, FileName As String, Map As Map)
- Takes a map object which holds pairs of key and value elements and stores it in a text file. The file format is known as Java Properties file: .properties - Wikipedia, the free encyclopedia.
The file format is not too important unless the file is supposed to be edited manually. This format makes it easy to edit it manually.
One common usage of File.WriteMap is to save a map of "settings" to a file.

File.ReadMap (Dir As String, FileName As String) As Map
- Reads a properties file and returns its key/value pairs as a Map object. Note that the order of entries returned might be different than the original order.

Some other useful functions:

File.Copy (DirSource As String, FileSource As String, DirTarget As String, FileTarget As String)
- Copies the source file from the source directory to the target file in the target directory.
Note that it is not possible to copy files to the DirAssets folder.

File.Copy2 (In As InputStream, Out As OutputStream)
- Copies the input stream to the output stream. The input stream is closed at the end.

File.Delete (Dir As String, FileName As String) As Boolean
- Deletes the given file from the given directory. Returns True if the file was deleted.

File.ListFiles (Dir As String) As List
- Returns a List with the files under the specified directory.
Example:
```
Dim List1 As List
List1 = File.ListFiles(File.DirDocuments)
```
List1 can be 'dimed' in Sub Process_Globals

File.Size (Dir As String, FileName As String)
- Returns the size in bytes of the specified file.
This method does not support files in the assets folder.

File.Combine (Dir As String, FileName As String) As String
- Combines the Dir and FileName to a single string.

File.MakeDir (Parent As String, Dir As String)
- Creates a new folder under the Parent folder.

File.GetAttributes (Dir As String, FileName As String) As Map
- Returns a Map with the files attributes (advanced).

10.10.2 Filenames

iOS file names allow following characters:
a to **z**, **A** to **Z**, **0** to **9** dot **.** underscore **_** and even following characters **+ - % &**
Spaces and following characters *** ?** are not allowed.

Example : MyFile.txt

Note that iOS file names are case sensitive!
`MyFile.txt` is different from `myfile.txt`

10.10.3 Subfolders

You can define subfolders in B4i with.

```
File.MakeDir(File.DirDocuments, "Pictures")
```

To access the subfolder you should add the subfolder name to the folder name with "/" in-between.

```
ImageView1.Bitmap = LoadBitmap(File.DirDocuments & "/Pictures", "test1.png")
```

Or add the subfolder name before the filename with "/" in-between.

```
ImageView1.Bitmap = LoadBitmap(File.DirDocuments, "Pictures/test1.png")
```

Both possibilities work.

10.10.4 Text encoding

Text encoding or character encoding consists of a code that pairs each character from a given repertoire with something else. Other terms like character set (charset), and sometimes character map or code page are used almost interchangeably (source Wikipedia).

The default character set in iOS is Unicode UTF-8.

In Windows the most common character sets are ASCII and ANSI.

- ASCII includes definitions for 128 characters, 33 are non-printing control characters (now mostly obsolete) that affect how text and space is processed.
- ANSI, Windows-1252 or CP-1252 is a character encoding of the Latin alphabet, used by default in the legacy components of Microsoft Windows in English and some other Western languages with 256 definitions (one byte). The first 128 characters are the same as in the ASCII encoding.

Many files generated by Windows programs are encoded with the ANSI character-set in western countries. For example: Excel csv files, Notepad files by default.
But with Notepad, files can be saved with *UTF-8* encoding.

iOS can use following character sets:

- UTF-8 default character-set
- UTF -16
- UTF - 16 BE
- UTF - LE
- US-ASCII ASCII character set
- ISO-8859-1 almost equivalent to the ANSI character-set
- Windows-1251 Cyrillic characters
- Windows-1252 Latin alphabet

To read Windows files encoded with ANSI you should use the *Windows-1252* character-set.
If you need to write files for use with Windows you should also use the *Windows-1252* character-set.

Another difference between Windows and iOS (Android also) is the end of line character:

- iOS, only the LF (Line Feed) character Chr(10) is added at the end of a line.
- Windows, two characters CR (Carriage Return Chr(13)) and LF Chr(10) are added at the end of a line. If you need to write files for Windows you must add CR yourself.

The symbol for the end of line is:

- B4i and B4A CRLF Chr(10)
- Windows CRLF Chr(13) & Chr(10)

To read or write files with a different encoding you must use File.ReadString2 or File.WriteString2. Even for reading csv files.

Tip for reading Excel csv files:
You can either:

- On the desktop, load the csv file in a text editor like *NotePad* or *Notepad++*
- Save the file with *UTF-8* encoding
  With *Notepad++* use Encode in UTF-8 without BOM, see below.

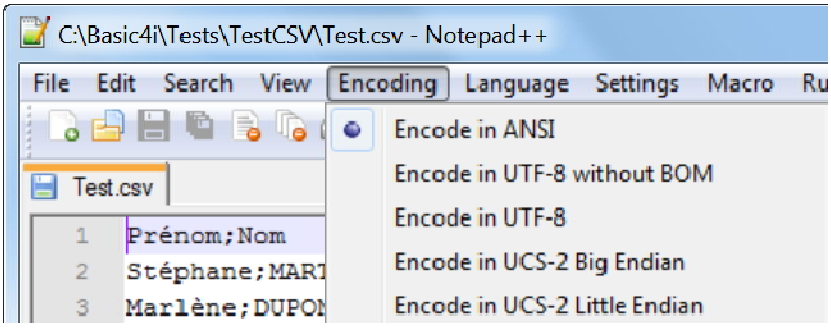
Or

- Read the whole file with File.ReadString2 and "Windows-1252" encoding.
- Save it back with File.WriteString2 with the standard iOS encoding.
- Read the file with LoadCSV or LoadCSV2 from the iStringUtils library.

```
Dim txt As String
txt = File.ReadString2(File.DirAssets, "TestCSV1_W.csv", "Windows-1252")

File.WriteString(txt)

lstTest = StrUtil.LoadCSV2(File.DirDocuments, "TestCSV1_W.csv", ";", lstHead)
```

When you save a file with NotePad three additional bytes are added.
These bytes are called BOM characters (Byte Order Mark).
In *UTF-8* they are represented by this byte sequence: `0xEF,0xBB,0xBF`.
A text editor or web browser interpreting the text as *Windows-1252* will display the characters `ï»¿`.

To avoid this you can use *Notepad++* instead of *NotePad* and use Encode in *UTF-8* without BOM.



Another possibility to change a text from *Windows-1252* to *UTF-8* is to use the code below.

```
Dim var, result As String
var = "Gestió"
Dim arrByte() As Byte
arrByte = var.GetBytes("Windows-1252")
result = BytesToString(arrByte, 0, arrByte.Length, "UTF8")
```

10.11 Lists

Lists are similar to dynamic arrays.

Lists are often used and many examples can be found in code examples:

- StringUtils LoadCSV, SaveCSV
- DBUtils module InsertMaps, UpdateRecord, ExecuteMemoryTable, ExecuteSpinner, ExecuteListView, ExecuteHtml, ExecuteJSON
- Charts module to hold different variables.

A list must be initialized before it can be used.

- **Initialize** Initializes an empty List.
```
Dim List1 As List
List1.Initialize
List1.AddAll(Array As Int(1, 2, 3, 4, 5))
```

- **Initialize2** (SomeArray)
  Initializes a list with the given values. This method should be used to convert arrays to lists. Note that if you pass a list to this method then both objects will share the same list, and if you pass an array the list will be of a fixed size.
  Meaning that you cannot later add or remove items.
  Example 1:
```
Dim List1 As List
List1.Initialize2(Array As Int(1, 2, 3, 4, 5))
```
  Example 2:
```
Dim List1 As List
Dim SomeArray(10) As String
' Fill the array
List1.Initialize2(SomeArray)
```

You can add and remove items from a list and it will change its size accordingly.
With either:

- Add (item As Object).
  Adds a value at the end of the list.
```
List1.Add(Value)
```

- AddAll (Array As String("value1", "value2")).
  Adds all elements of an array at the end of the list.
```
List1.AddAll(List2)
List1.AddAll(Array As Int(1, 2, 3, 4, 5))
```

- AddAllAt (Index As Int, List As List).
  Inserts all elements of an array in the list starting at the given position.
```
List1.AddAll(12, List2)
List1.AddAllAt(12, Array As Int(1, 2, 3, 4, 5))
```

- InsertAt (Index As Int, Item As Object)
  Inserts the specified element in the specified index.
  As a result all items with index larger than the specified index are shifted.
```
List1.InsertAt(12, Value)
```

- RemoveAt (Index As Int)
  Removes the specified element at the given position from the list.
```
List1.RemoveAt(12)
```

A list can hold any type of object.
B4i automatically converts regular arrays to lists. So when a List parameter is expected you can pass an array instead.

Get the size of a List:

- `List1.Size`

Use the Get method to get an item from the list with:

- Get(Index As Int)
  ```
  number = List1.Get(i)
  ```

  You can use a For loop to iterate over all the values:
  ```
  For i = 0 To List1.Size - 1
        Dim number As Int
        number = List1.Get(i)
        ...
  Next
  ```

Lists can be saved and loaded from files with:

- File.WriteList(Dir As String, FileName As String, List As List)
  ```
  File.WriteList(File.DirDocuments, "Test.txt", List1)
  ```
- File.ReadList (Dir As String, FileName As String) As List
  ```
  List1 = File.ReadList(File.DirDocuments, "Test.txt")
  ```

A single item can be changed with:

- List1. Set(Index As Int, Item As Object)
  ```
  List1.Set(12, Value)
  ```

A List can be sorted (the items must all be numbers or strings) with:

- Sort(Ascending As Boolean)
  `List1.Sort(True)` sort ascending
  `List1.Sort(False)` sort descending
- SortCaseInsensitive(Ascending As Boolean)

Clear a List with:

- `List1.Clear`

10.12 Maps

A Map is a collection that holds pairs of keys and values.

The keys are unique. Which means that if you add a key/value pair (entry) and the collection already holds an entry with the same key, the previous entry will be removed from the map.

The key should be a string or a number. The value can be any type of object.

Maps are very useful for storing applications settings.

Maps are used in these example codes:

- DBUtils module
  used for database entries, keys are the column names and values the values.
- Table module used for settings

A list must be initialized before it can be used.

- **Initialize** Initializes an empty Map.
```
Dim Map1 As Map
Map1.Initialize
```

Add a new entry:

- **Put**(Key As Object, Value As Object)
```
Map1.Put("Language", "English")
```

Get an entry:

- **Get**(Key As Object)
```
Language = Map1.Get("Language")
```

Check if a Map contains an entry, tests whether there is an entry with the given key:

- **ContainsKey**(Key As Object)
```
If Map1.ContainsKey("Language") Then
  Msgbox("There is already an entry with this key !", "ATTENTION")
  Return
End If
```

Remove an entry:

- **Remove**(Key As Object)
```
Map1.Remove("Language")
```

Clear an entry, clears all items from the map:

- **Clear**
```
Map1.Clear
```

Maps can be saved and loaded with:

- **File.WriteMap**(Dir As String, FileName As String, Map As Map)
```
File.WriteMap(File.DirDocuments, "settings.txt", mapSettings)
```

- **File.ReadMap**(Dir As String, FileName As String)
  Reads the file and parses each line as a key-value pair (of strings).
  Note that the order of items in the map may not be the same as the order in the file.
```
mapSettings = File.ReadMap(File.DirDocuments, "settings.txt")
```

11 Differences B4i <> B4A

Even though B4i and B4A are similar, there are differences because of the different operating systems.

Only differences are explained in this chapter.

Some of the differences were reported by sorex in the forum.

11.1 Screens Page <> Activity

B4i
The different screens are managed with Pages in the same Main module with the NavigationController.

B4A
The different screens are manly managed with Activities in separate modules, or on Panels managed in the Main Activity or a mix of both.

11.2 Panel

B4i
Panels don’t have a background bitmap.
But you can draw onto a Panel with a Canvas.

If a Panel, without event routines, covers other views the events are NOT submitted to the underlying views. In B4A they are!
If you want to submit them, you must set the UserInteractionEnabled property to False:

```
Panel1.UserInteractionEnabled = False
```

Transparent panel: Background: Color.Transparent Alpha = 1

B4A
Panels have a background bitmap.
It is also possible to draw onto a Panel with a Canvas.
If a Panel, without event routines, covers other views the events ARE submitted to the underlying views. In B4i they are NOT!
To avoid this, one solution is to add an empty event routine.

Transparent panel: Background: Color.Transparent Alpha = 0

11.3 Canvas

When you use a Canvas, you need to refresh the drawing to make it visible.

Difference in DrawBitmap method, no SourceRectangle in B4i.

B4i
You refresh the Canvas with Canvas.Refresh.

Canvas.DrawBitmap(Bitmap1 As B4iBitmap, DestRect As B4iRect)

A workaround was proposed by Erel with the two routines below:

```
Sub DrawBitmap(canvas1 As Canvas, Bitmap1 As Bitmap, SrcRect As Rect, DestRect As Rect)
   If SrcRect = Null Then
     Dim SrcRect As Rect
     SrcRect.Initialize(0, 0, Bitmap1.Width, Bitmap1.Height)
   End If
   Dim p1 As Path
   p1.InitializeRect(DestRect, 0)
   canvas1.ClipPath(p1)
   Dim sx, sy As Float
   sx = (DestRect.Right - DestRect.Left) / (SrcRect.Right - SrcRect.Left)
   sy = (DestRect.Bottom - DestRect.Top) / (SrcRect.Bottom - SrcRect.Top)
   Dim x, y, width, height As Int
   x = DestRect.Left - sx * SrcRect.Left
   y = DestRect.Top - sy * SrcRect.Top
   width = Bitmap1.Width * sx
   height = Bitmap1.Height * sy
   Dim d2 As Rect
   d2.Initialize(x, y, x + width, y + height)
   canvas1.DrawBitmap(Bitmap1, d2)
   canvas1.RemoveClip
End Sub

Sub DrawBitmapRotated(canvas1 As Canvas, Bitmap1 As Bitmap, SrcRect As Rect, DestRect As Rect, De
grees As Float)
   Dim no As NativeObject = canvas1
   no.RunMethod("rotate:::", Array(Degrees, DestRect.CenterX, DestRect.CenterY))
   DrawBitmap(canvas1, Bitmap1, SrcRect, DestRect)
   no.RunMethod("rotate:::", Array(-Degrees, DestRect.CenterX, DestRect.CenterY))
End Sub
```

B4A
You refresh the Canvas view like Panel1.Invalidate.
In B4A you can also refresh only a part of the Canvas view, limited by a rectangle Rect, with Invalidate2(Rect).

Canvas1.DrawBitmap(Bitmap1 As Bitmap, SrcRect As Rect, DestRect As Rect)

11.4 Text views TextField / TextView <> EditText

B4i
B4i has two views to enter text:

TextField	Single line.
TextView	Multiline.

B4A
B4A has only one view to enter text, single line or multiline:

EditText	Can be single line or multiline.

11.5 ScrollViews

B4i
ScrollView Scrolls in both directions.
It's equivalent to ScrollView2D in B4A.
Can optionally scroll the whole width or the whole height with one swipe.
Change the internal panel size.

```
ScrollView1.ContentHeight
ScrollView1.ContentWidth
```

B4A

ScrollView	Scrolls only in vertical direction.
HorizontalScrollView	Scrolls only in horizontal direction.
ScrollView2D	Scrolls in both directions.

Change the internal panel size.

```
ScrollView1.Panel.Height
HorizontalScrollView1.Panel.Width
ScrollView2D1.Panel.Height
ScrollView2D1.Panel.Widht
```

11.6 Picker <> Spinner

B4i
Picker Wheel selecting view with several columns.

B4A
Spinner One line visible and extended with a click onto the Spinner.

11.7 ListView

B4i
Doesn't exist.
You can use the CustomListView class instead.

B4A
Shows a list.

11.8 RadioButton

B4i
Doesn't exist.

B4A
Exist

11.9 Switch <> CheckBox

B4i

Switch Shows a Switch with two states.

B4A

CheckBox Shows a check box with two states.

11.10 SegmentedControl

B4i

Shows several buttons side by side for selection.

B4A

Doesn’t exist

11.11 Stepper

B4i

Allows up and down counting.

B4A

Doesn’t exist

11.12 Slider / SeekBar

Similar views, only the name changes.

11.13 View Background

B4i

Only color with border.

B4A

The Background is a Drawable.

- ColorDrawable same as Color in B4i.
- GradientDrawable doesn’t exist in B4i
- BitmapDrawable doesn’t exist in B4i
- StateListDrawable doesn’t exist in B4i

11.14MsgBox / MsgBox2

B4i

Non modal object!

The program stops at the line where MsgBox is called and continues.

MsgBox2 needs an event

```
Msgbox(Message As String, Title As String)
Msgbox2(EventName As String, Title As String, Message As String, Buttons As List)
```

Example:

```
Msgbox2("MsgDelete", "Delete entry", "Do you really want to delete the entry ?",
Array As String("Yes", "No"))

Private Sub MsgDelete_Click(ButtonText As String)
   ' your code
End Sub
```

B4A

Modal object.

The program stops at the line where MsgBox is called and waits for the user input.

```
Msgbox(Message As String, Title As String)
Answer = Msgbox2(Message As String, Title As String, Positive As String, Cancel As
String, Negative As String, Icon As Bitmap)
```

Example:

```
Answer = Msgbox2("MsgDelete", "Do you really want to delete the entry ?", "Delete
entry", "Yes", "", "No", Null))
```

11.15SQLite ResultSet <> Cursor

B4i

The returned object is called ResultSet.
Code to go through the results

```
Private rs As ResultSet
rs = SQL1.ExecQuery2("SELECT * FROM table1 WHERE col1 = ?", Array(100))
Do While rs.NextRow
   ' your code
Loop
```

B4A

In B4A you can use the same code as in B4i.

In B4A you can use another object called Cursor, the Cursor object doesn't exist in B4i.
It is possible to position the Cursor at a given item with Cursor.Position = i
Code to go through the results

```
Private Curs As Cursor
Private i As Int
For i = 0 To Curs.RowCount - 1
   Curs.Position = i
   ' your code
Next
```

11.16Font <> TextSize

B4i

Font is an object, to change the text size you need to create a new Font object

```
Label1.Font = Font.CreateNew(20)
```

B4A

TextSize is a property which can be changed directly.

```
lblLabel1.TextSize = 20
```

11.17File object and Folders

B4i
Predefined folders:

- **DirAssets**
  Same as B4A
- **DirDocuments**
  The documents folder should only be used to store user generated content. It is possible to make this folder sharable through iTunes.
  This folder is backed up by iTunes automatically.
- **DirLibrary**
  The place for any non-user generated persistent files. This folder is backed up by iTunes automatically.
  You can create a subfolder named Caches. Files under that folder will not be backed up.
- **DirTemp**
  A temporary folder. Files in this folder are not backed up by iTunes and may be deleted from time to time.

B4A
Predefined folders:

- **DirAssets**
  Same as B4i
- **DirInternal**
  Returns the folder in the device internal storage that is used to save application private data.
- **DirInternalCache**
  Returns the folder in the device internal storage that is used to save application cache data. This data will be deleted automatically when the device runs low on storage.
- **DirDefaultExternal**
  Returns the application default external folder which is based on the package name.
  The folder is created if needed.
- **DirRootExternal**
  Returns the root folder of the external storage media.
  If the device has an intenal sdcard, then DirRooiExternal points to this one and not to an external sdcard.

11.18Map

Example to go through the keys:

B4i
No GetKeyAt method.

```
'B4i method
For Each k As String In myMap.Keys
   kd = myMap.Get(k)
```

B4A
GetKeyAt exists.

```
'B4A method
For k = myMap.Size - 1 To 0 Step -1
   kd = myMap.GetKeyAt(k)
```

11.19Regex

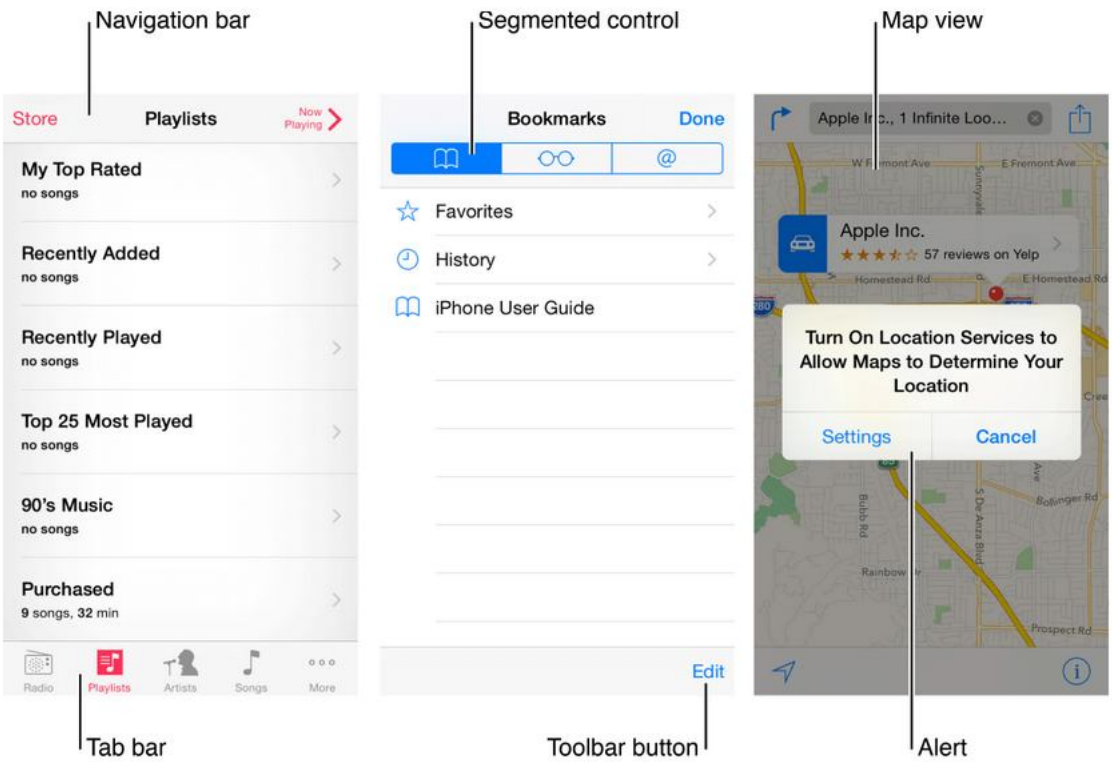
B4i
Regex.Split doesn’t allow “”(empty string) as the split string

B4A
Regex.Split allows “” (empty string) as the split string

12 User Interfaces

In this chapter several example programs are explained with code examples.
All projects are saved in the SourceCode folder shipped with the guide.

Almost all iOS apps use at least some of the UI components defined by the UIKit framework. Knowing the names, roles, and capabilities of these basic components helps you make informed decisions as you design the UI of your app.



Source : Apple Documentation
https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/Anatomy.html#//apple_ref/doc/uid/TP40006556-CH24-SW1

The site in the link above is worth a reading!

12.1 Bars

There are three bars in iOS:

- StatusBar on top of the screen.
- NavigationBar on top just below the StatusBar.
- ToolBar on the bottom.

The NavigationController manages the NavigationBar and the ToolBar.

12.1.1 Status Bar

On top of the screen the **status bar** displays important information about the device and the current environment (shown below on iPhone).

12.1.2 NavigationController

The NavigationController manages two bars:

- The NavigationBar on the top of the screen, just below the StatusBar.
- The ToolBar on the bottom of the screen.

Each Page has its own NavigationBar and ToolBar.

12.1.2.1 NavigationBar

The NavigationBar is on the top of the screen just below the StatusBar.

The NavigationBar is visible by default, its `Page.NavigationBarVisible` property is `True`.
If you want to hide it you need to set the property to `Page.NavigationBarVisible = False`.

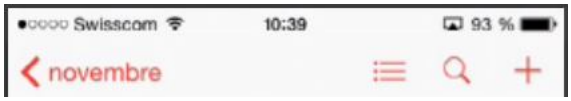
It can contain the following optional parts:

- TopLeftButtons.
  The left side is manly used for the '< back button' to go back to the previous Page.
  But can be used for other buttons.

- Title.
  It is good practice to add a title, which is displayed in the middle of the NavigationBar to inform the user of the purpose of the selected page.

- TopRightButtons.
  On the right side you can add buttons for different functions.

The buttons in the NavigationBar are BarButtons.

Example from the Agenda program. 

No title, the back button 

and other buttons  .

12.1.2.1.1 TopRightButtons

In the NavigationBar you can add buttons on the right side of the bar.

You can either add them in the Designer or in the code.

12.1.2.1.2 TopLeftButtons

Normally the upper left button is used as a back button allowing to move back to the previous page.

You can also use it for other buttons, but be careful because most users expect the back button there.

To hide the back button you must set the HideBackButton property to:

- in the Designer

Hide Back Button	True

- in the code `Page.HideBackButton = True`.

12.1.2.2 ToolBar

The optional ToolBar is displayed at the bottom of the screen.
The default value of the `Page.ToolBarVisible` property is `False`, so you need to set it to `True` to display it `Page.ToolBarVisible = True`.

The buttons in the ToolBar are BarButtons.

Example from the Agenda program: 

12.1.2.3 BarButtons

The Buttons on the NavigationBar and on the ToolBar are BarButtons.

There are four bar button types:

- Text buttons they show only text.
- Bitmap buttons they show a bitmap.
- System buttons they show either predefined texts or predefined bitmaps.
- Custom items they show any view.

BarButtons have two properties:

- The reference to its type.
  - Text the text to display.
  - Bitmapthe bitmap to display.
  - System the index to the system text or bitmap.
  - Custom the view to display, the Custom button has no Tag property.

- Tag this property is used in the `Page_BarButtonClick` event routine to know what button raised the event.

You can add BarButtons either in the Designer or in the code.

Adding BarButtons in the Designer and in the code is different:

- Designer
  - BarButtons added in the Designer are positioned automatically
    First
    First □ Second
    First □ Second □ Third
    □ represents a FLEXIBLE_SPACE button

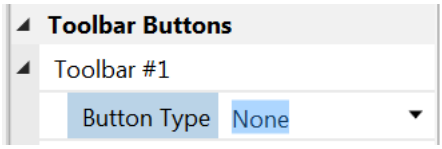
  - Be aware that the number of BarButtons includes the FLEXIBLE_SPACE buttons and the indexes in the Page.ToolbarButtons list take this into account.
    For example in the third example above:
    First has index 0
    Second has index 2
    Third has index 4

  - If you add two buttons they are positioned one on the left side and the other on the right side.
    The Designer adds automatically a FLEXIBLE_SPACE button in between.

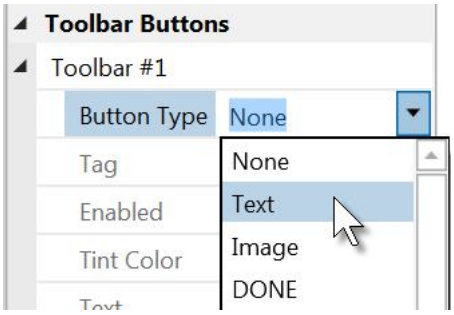
  - If you want two buttons, one in the middle and the second one on the right side you must add a FLEXIBLE_SPACE button as the first button.

  - If you add three buttons one on the left, the second in the middle and the third on the right, the Designer adds automatically two FLEXIBLE_SPACE buttons in between.

- Example:

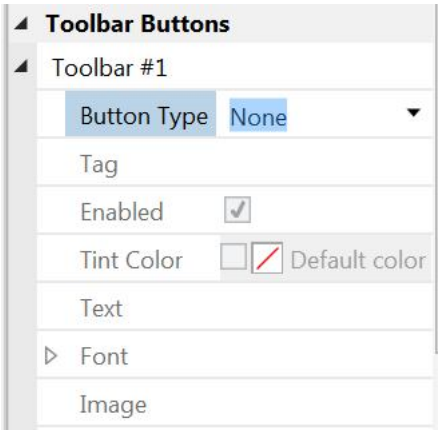


In the Designer select Main and open the first Toolbar Button.



You must select the Button Type in the drop-down list, for example Text.

Then you must enter the Tag and Text properties.



- Code
  - BarButtons added in the code are placed side by side starting on the left side.
    First
    First Second
    First Second Third
  - For two buttons one on the left and the other on the right side you must add a FLEXIBLE_SPACE button between the two others.
  - Same for three BarButtons Left, Middle, Right you must add two FLEXIBLE_SPACE button between the others.
    First
    First □ Second
    First □ Second □ Third
  - Example with two buttons one in the middle and one on the right side:

```
' Dim the buttons
Dim bbt1, bbt2, bbt3 As BarButton
' Initialize the buttons
bbt1.InitializeSystem(bbt2.ITEM_FLEXIBLE_SPACE, "")
bbt2.InitializeText("Page 2", "Page2")
bbt3.InitializeText("Page 3", "Page3)
' Add the buttons onto Page1
Page1.ToolbarButtons.AddAll(Array As Object(bbt1, bbt2, bbt3))
```

12.1.2.3.1 Text BarButtons

To define a Text BarButton in the code you need to Dim and Initialize it:

BarButton.InitializeText (Text As String, Tag As String)

- Text text to display
- Tag used in the `Page_BarButtonClick` event routine to know what button raised the event.

Example:

```
Dim BarButton1 As BarButton
BarButton1.InitializeText("Page 1", "Page1")
```

12.1.2.3.2 Bitmap BarButtons

To define a Bitmap BarButton in the code you need to Dim and Initialize it:

BarButton.InitializeBitmap (bmp As Bitmap, Tag As String)

- bmp bitmap to display.
- Tag used in the `Page_BarButtonClick` event routine to know what button raised the event.

Example:

```
Dim BarButton1 As BarButton
BarButton1.InitializeBitmap(LoadBitmap(File.DirAssets, "image.png"), "Image1")
```

One bitmap is used, the color changes automatically when the button is touched.

You may provide several image files for the different screen resolutions (scales):

Pixel size	Image file name
60 x 60	image.png
120 x 120	image @2x.png
180 x 180	image @3x.png

In the initialization, don't add @2x to the files, iOS loads the correct file with the generic file name according to the device scale.

12.1.2.3.3 System BarButtons

To define a System BarButton in the code you need to Dim and Initialize it:

BarButton.InitializeSystem (Type As Int, Tag As String)

- Type index of the system button to display (see below).
- Tag used in the `Page_BarButtonClick` event routine to know what button raised the event.

Example:

```
Dim BarButton1 As BarButton
BarButton1.InitializeSystem(BarButton1.ITEM_BOOKMARKS, "BookMark")
```

List of system BarButtons with icons (Source Apple documentation).

Button	Name	Meaning
	Action	Open a modal view that lists system-provided and app-provided actions that can work with the current content.
	Camera	Open an action sheet that displays a photo picker in camera mode.
	Compose	Open a new message view in edit mode.
	Bookmarks	Show app-specific bookmarks.
	Search	Display a search field.
	Add	Create a new item.
	Trash	Delete current item.
	Organize	Move or route an item to a destination within the app, such as a folder.
	Reply	Send or route an item to another location.
	Refresh	Refresh contents (use only when necessary; otherwise, refresh automatically).
	Play	Begin media playback or slides.
	Fast Forward	Fast forward through media playback or slides.
	Pause	Pause media playback or slides (note that this implies context preservation).
	Rewind	Move backwards through media playback or slides.

List of system BarButtons with text (Source Apple documentation).

- Cancel
- Done
- Edit
- Redo
- Save
- Stop
- Undo

There is one special system BarButton:

- FLEXIBLE_SPACE This button is used as a placeholder.

12.1.2.3.4 Custom BarButtons

To define a Custom BarButton in the code you need to Dim and Initialize it:

BarButton.InitializeCustom (View1 As View)

- View view to display

Example:

```
Dim BarButton1 As BarButton
BarButton1.InitializeCustom("Page 1", "Page1") ??????
```

12.1.2.4 Tips

Be careful with the space taken by the title and the buttons in the NavigationBar.
Don’t use more than 5 BarButtons in the ToolBar.
Leave enough place between Text BarButton.

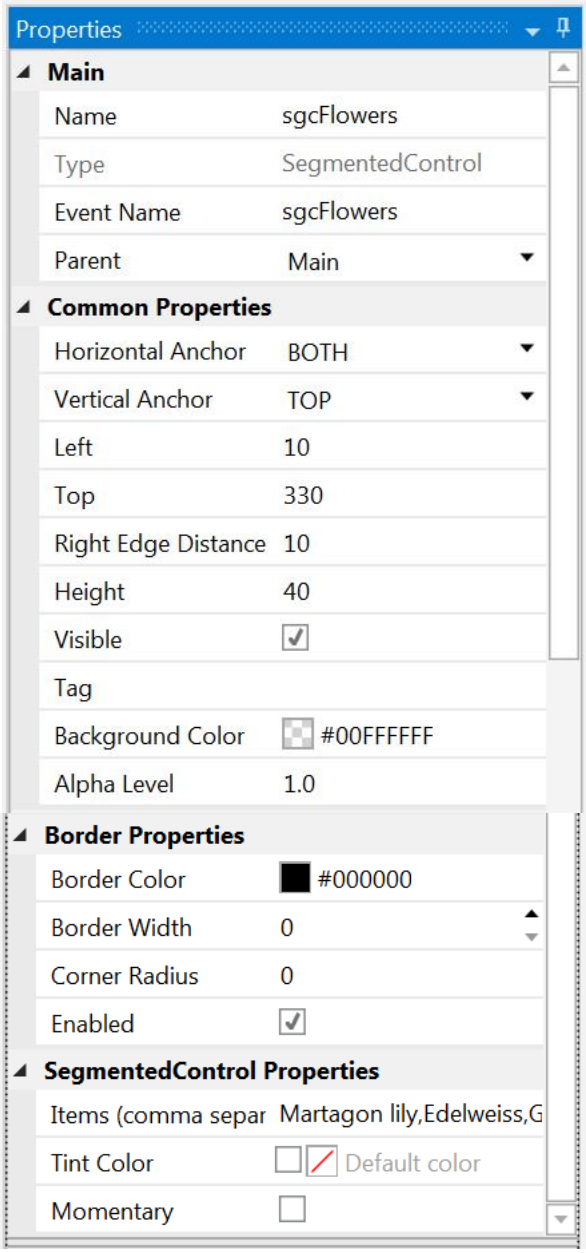
12.2 SegmentedControl

A **segmented control** is a horizontal control made of multiple segments, each segment functioning as a discrete button.

A SegmentedControl can be added either in the Designer or in the code.

The source code is in the UINavBar project in the SourceCode\UsersInterfaces folder.
It is shown in the program on Page 3.

12.2.1 Added in the Designer



The simplest way to add a SegmentedControl is:
(it will look like the image on top of the page)

- Add in on the layout, position and size it.
- Add the Items (comma separated), like in the example.
  These texts will be displayed in the buttons.
- Leaving all the other properties with their default values.

You can, of course, change the other properties to your convenience.
Changing the BorderWidth will override the default appearance (width, color and corner radius).

Momentary property: True (default value) highlights the selected button.

12.2.2 Added in the code

You must:

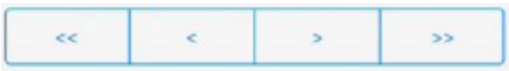
- Dim it in Process_Globals
```
Private sgcButtons  As SegmentedControl
```

- Initialize it in Application_Start
```
sgcButtons.Initialize("sgcButtons")
' Add the button items, these texts will be displayed in the buttons.
sgcButtons.SetItems(Array As String("<<", "<", ">", ">>"))
' We set Momentary = True to not show the selected button
sgcButtons.Momentary = True
```

- Add it to the page in Page1_Resize, to ensure that the current size is known.
```
' sgcButtons is added here because Widht and Height
' are known only from here
Private Sub Page1_Resize(Width As Int, Height As Int)
   Page3.RootPanel.AddView(sgcButtons, 10, sgcFlowers.Top + sgcFlowers.Height +
10, 100%x - 20, 40)
End Sub
```

- And the event routine.
```
Private Sub sgcButtons_IndexChanged (Index As Int)
  Msgbox("Button " & sgcButtons.GetItems.Get(Index) & " clicked", "Buttons")
  Select Index
  Case 0
    'code
  Case 1
    'code
  Case 2
    'code
  Case 3
    'code
  End Select
End Sub
```

The result:



12.3 ActionSheet

The content of this page is an extract from the Apple documentation.

Action sheets display a set of buttons representing several alternative choices to complete a task initiated by the user.

An action sheet:

- Appears as the result of a user action
- Displays two or more buttons

In a horizontally compact environment, an action sheet emerges from the bottom of the screen

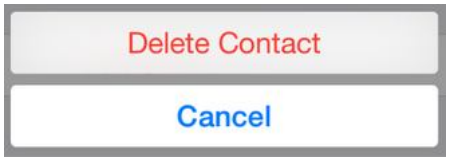
In a horizontally regular environment, an action sheet is always displayed in a popover

Use an action sheet to:

- Provide alternative ways to complete a task. An action sheet lets you to provide a range of choices that make sense in the context of the current task, without giving these choices a permanent place in the UI.
- Get confirmation before completing a potentially dangerous task. An action sheet prompts users to think about the potentially dangerous effects of the step they're about to take and gives them some alternatives.

In a horizontally compact environment, include a Cancel button so that users can easily and safely abandon the task. Place the Cancel button at the bottom of the action sheet to encourage users to read through all the alternatives before making a choice.

In all environments, use red for the button that performs a potentially destructive action. Display a red button at the top of the action sheet, because the closer to the top of the action sheet a button is, the more eye-catching it is.



Avoid making users scroll an action sheet. If you include too many buttons in an action sheet, users must scroll to see all their choices. This is a disconcerting experience for users, because they must spend extra time to distinguish the choices. Also, it can be very difficult for users to scroll without inadvertently tapping a button.

Example code in the UINavBar project in the SourceCode\UserInterfaces\UINavBar folder.

Two examples, they are shown on page 2.

Action sheets must be added in the code and need:

- to be dimed in Process_Globals.
- to be initialized in Application_Start.
- an Event routine.
- to be displayed with the Show method.

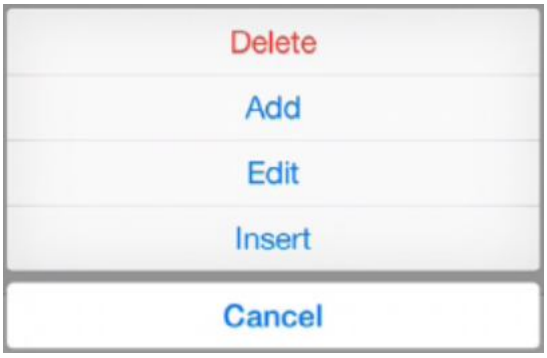
Action sheet initialization parameters:

Although the first parameter of the initialization method enables you to provide a title for an action sheet, iOS human interface guidelines recommend that you do not use a title.

Initialize(EventName As String, Title As String, CancelItem As String, DestructiveItem As String, OtherItems As List)

- EventName Generic event name
- Title Title of the ActionSheet. Appel advices NOT to use a title.
- CancelItem This is the button at the bottom of the list.
  Apple advices to always use Cancel Button
- DestructiveItem Shows an item in red, like Delete in the example below.
- OtherItems Other buttons in a List, the first button in the list is on top.

```
ashTest1.Initialize("ashTest2", "Edition", "Cancel", "Delete", Array As String("Add",
"Edit", "Insert"))
```



Delete, DestructiveItem

Add, Edit, Insert additional buttons

Cancel button

Source code:
Dim: in Sub Process_Globals
```
Private ashTest1 As ActionSheet
```

Initialize: in Sub Application_Start
```
ashTest1.Initialize("ashTest2", "", "Cancel", "Delete", Array As String("Add", "Edit",
"Insert"))
```

Show the ActionSheet: in Sub Page2_BarButtonClick
```
ashTest1.Show(Page2.RootPanel)
```

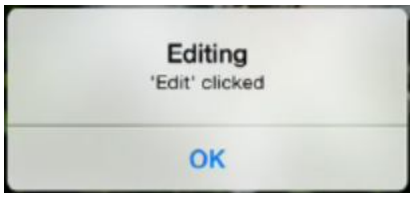
Event routine:
```
Private Sub ashTest1_Click(Item As String)
   Select Item
   Case "Cancel"
      Msgbox("'Cancel' clicked", "Editing")
   Case "Add"
      Msgbox("'Add' clicked", "Editing")
   Case "Edit"
      Msgbox("'Edit' clicked", "Editing")
   Case "Insert"
      Msgbox("'Insert' clicked", "Editing")
   Case "Delete"
      Msgbox2("msgDelete", "Dou you really want to delete the entry ?", "Editing", Array
("Yes", "No"))
   End Select
End Sub
```

12.4 MessageBoxes MsgBox / Alerts

Message boxes, called Alerts in iOS, are called MsgBox in B4i like in B4A.
They are not exactly the same, the major difference is that B4i MsgBoxes are not modal views. This means that the code continues after the calling of the MsgBox and not waiting on the user answer.
The answer must be handled in an event routine.

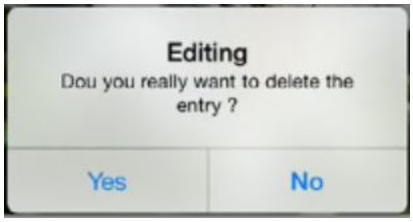
The two message box callings.

- MsgBox (Message As String, Title As String)
  Displays the title, the message and an OK button, no event routine because only one button.

  Example:
  ```
  Msgbox("'Insert' clicked", "Editing")
  ```

- Msgbox2(EventName As String, Message As String, Title As String, Buttons As List)
  Displays the Title, the message and the buttons of the list.
  The user answer must be handled in an event routine.

  Example:
  ```
  Msgbox2("msgDelete", "Dou you really want to delete the entry ?", "Editing",
  Array ("Yes", "No"))
  ```

  And the event routine:
  ```
  ' Message box event routine
  Private Sub msgDelete_Click(ButtonText As String)
     Select ButtonText
     Case "Yes"
        ' your code
     Case "No"
        ' your code
     End Select
  End Sub
  ```

The examples above are from the UINavBar project on Page 2, Edit ToolBar.

12.5 Example program

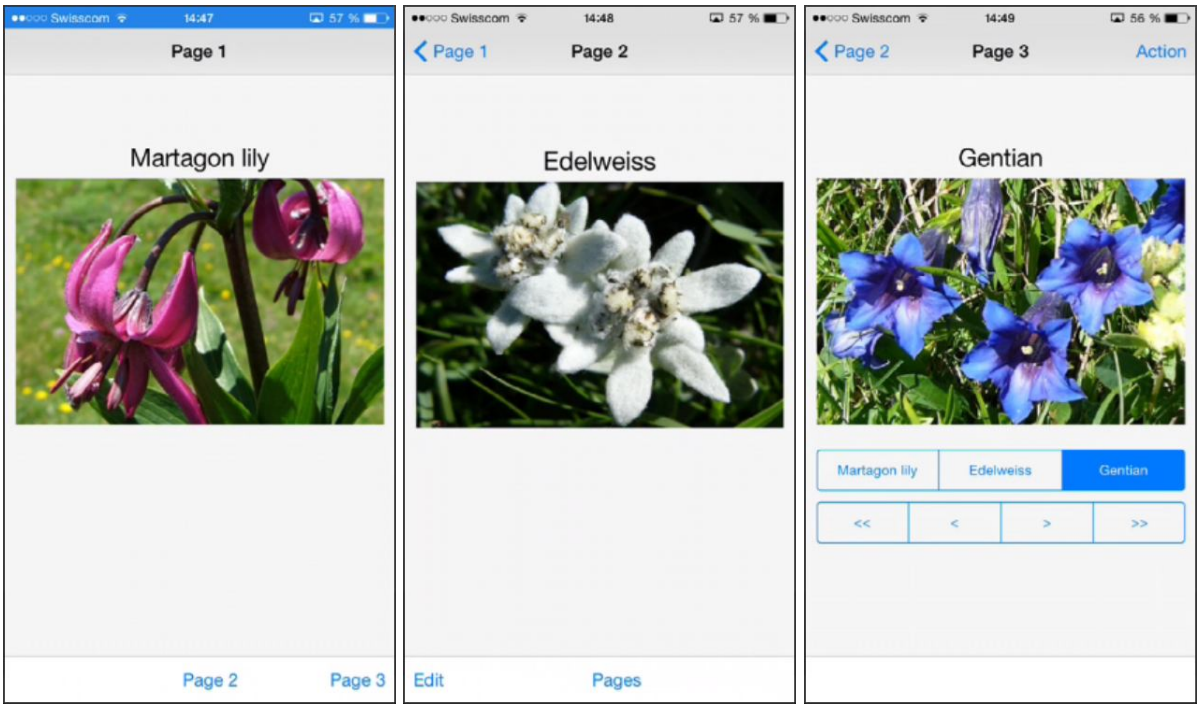
The source code is in the UINavBar project in the SourceCode\UserInterfaces\UINavBar folder.

The program has no other function than to show different user interface possibilities.

- NavigationBar
- ToolBar
- SegmentedControl
- ActionSheet
- Alerts MsgBox / MsgBox2

The user interface in this project is not coherent because of the use of too many different user interface objects, but the purpose is to show different possibilities in one project not showing best practice.

The program has three pages, with different user interface examples.
The pictures are just there to show the page changes (pictures taken during my mountain hikes).



Page 1

- NavigationBar Only the title.
- ToolBar Two BarButtons to navigate.

Page 2

- NavigationBar Title and back button.
- ToolBar Two BarButtons activating the ActionSheets.
- Other Two ActionSheets, Pages to navigate.

Page 3

- NavigationBar Title, back button and a top right button.
- ToolBar No ToolBar.
- Other Two SegmentedControls.

Some explanations:

Page 1:
Nothing special, only the two navigation buttons.

Page 2:
The Pages button shows an ActionSheet to navigate.
The Edit button shows another ActionSheet.

Page 3:
No ToolBar for the navigation, only the return button.
Note that with the return button you return to the calling page, either Page 1 or Page 2.
The Action button only shows the use of a TopRightButton.

ToolBar and TopRightButton event are handled in the same BarButtonClick_Click event routine.

The source code is hopefully enough self-explaining.

12.6 TabBarController

The TabBarController shows buttons at the bottom edge of the screen.

The example project UITabBar is in the SourceCode\UserInterfaces\UITabBar folder.
The images on the pages have no direct meaning, just to make them different.
They show some flower pictures I have taken during my mountain hiking.

Three types of buttons are used:

- 

System button with predefined text and icons.
- 

A custom text only button.
- 

A button with custom text and custom icons.

The code is, I hope, self-explaining with the comments:

```
Sub Process_Globals
   Public App As Application
   Public TabControl As TabBarController
   Private Page1, Page2, Page3 As Page
End Sub
```

Code in Application_Start initializing the TabBarController and the Pages.

```
Private Sub Application_Start (Nav As NavigationController)
   ' Initialize the TabBarController
   TabControl.Initialize("TabControl")

   ' Initialize Page 1 and load a layout file
   Page1.Initialize("Page1")
   Page1.RootPanel.LoadLayout("Page1")

   ' Initialize Page 2 and load a layout file
   Page2.Initialize("Page2")
   Page2.RootPanel.LoadLayout("Page2")

   ' Initialize Page 3 and load a layout file
   Page3.Initialize("Page3")
   Page3.RootPanel.LoadLayout("Page3")

   ' Initialize the TabBarItems
   SetTabButtons

   ' Set the pages to the TabBarController
   TabControl.Pages = Array(Page1, Page2, Page3)

   ' Set the TabBarController to the Application
   App.KeyController = TabControl
End Sub
```

Code of the SetTabButtons.

```
Private Sub SetTabButtons
   ' Define a system button
   Dim tbi As TabBarItem                      ' define a new TabBarItem
   tbi.InitializeSystem(tbi.ITEM_BOOKMARKS) ' initializes Bookmark system button
   Page1.TabBarItem = tbi               ' set this TabBarItem to Page1

   ' Define a button with text only
   Dim tbi As TabBarItem                      ' define a new TabBarItem
   tbi.Initialize("Page 2", Null, Null)       ' initialize without icons
   tbi.SetFont(Font.CreateNew(15))            ' set the font size
   tbi.SetTitleOffset(0, -12)    ' set the text offset to show it in the center
   Page2.TabBarItem = tbi               ' set this TabBarItem to Page1

   ' Define a button with custom bitmaps
   Dim tbi As TabBarItem                      ' define a new TabBarItem
   ' define a custom TabBarItem with custom icons
   tbi.Initialize("Countries", LoadBitmap(File.DirAssets, "btnCH0.png"),
   LoadBitmap(File.DirAssets, "btnCH1.png"))
   Page3.TabBarItem = tbi                 ' set this TabBarItem to Page1
End Sub
```

12.6.1.1 TabBar system item

List of TabBar system items and their icons (Source Apple documentation).

The item type is defined in the initialize declaration `tbi.ITEM_BOOKMARKS`:
`tbi.InitializeSystem(tbi.ITEM_BOOKMARKS)`

Table 35-2 Standard icons for use in the tabs of a tab bar

Icon	Name	Meaning
	Bookmarks	Show app-specific bookmarks.
	Contacts	Show contacts.
	Downloads	Show downloads.
	Favorites	Show user-determined favorites.
	Featured	Show content featured by the app.
	History	Show history of user actions.
	More	Show additional tab bar items.
	Most Recent	Show the most recent item.
	Most Viewed	Show items most popular with all users.
	Recents	Show the items accessed by the user within an app-defined period.
	Search	Enter a search mode.
	Top Rated	Show the highest-rated items, as determined by the user.

12.6.1.2 TabBar text item

The ‘standard’ items contain icons, but it is possible to show text only. For this we must change following properties:

- The text size.
  ```
  tbi.SetFont(Font.CreateNew(15))
  ```
- And the text position.
  ```
  tbi.SetTitleOffset(0, -12)
  ```

12.6.1.3 TabBar custom item

The TabBar custom item needs two bitmaps with a size of about 30 x 30 pixels.

One bitmap for the unselected state and the second one for the selected state.
Example bitmaps, with a size of 30 x 30 pixels for a scale of 1.
The black frames are not part of the images, the white parts are transparent:


Unselected


Selected

The color has no importance, only the Alpha value is considered by the system.
The color of the non-transparent part is defined by the system like this:

You may provide several image files for the different screen resolutions (scales):

Pixel size	Image file name
30 x 30	btnCH0.png
60 x 60	btnCH0@2x.png
90 x 90	btnCH0@3x.png

Initialization:
iOS loads the correct file with the generic file name according to the device scale, you must not add @2x to the files.

```
tbi.Initialize("Countries", LoadBitmap(File.DirAssets, "btnCH0.png"), _
LoadBitmap(File.DirAssets, "btnCH1.png"))
```

12.6.1.4 Tips

Don’t set more than 5 items in the TabBar on phones.
On big devices more items can be added.
Be aware to leave enough place between text items.

12.7 Side menu

The example project UISideMenu is in the SourceCode\UserInterfaces\ UISideMenu folder.

The code is self explanatory.

You can have a SideMenu on each side or on both sides.

```
Sub Process_Globals
   'These global variables will be declared once when the application starts.
   'Public variables can be accessed from all modules.
   Public App As Application
   Public NavControl As NavigationController
   Private Page1 As Page

   Private smc As SideMenuController
End Sub

Private Sub Application_Start (Nav As NavigationController)
   'We need to create a new navigation controller
   Dim nc As NavigationController
   nc.Initialize("nc")
   NavControl = nc

   Page1.Initialize("Page1")
   Page1.Title = "Page 1"
   Page1.RootPanel.Color = Colors.White
   Page1.RootPanel.LoadLayout("Main")

   'Initialize the left page
   Dim LeftPage As Page
   LeftPage.Initialize("LeftPage")
   LeftPage.RootPanel.LoadLayout("LeftPage")

   'Initialize the right page
   Dim RightPage As Page
   RightPage.Initialize("RightPage")
   RightPage.RootPanel.LoadLayout("RightPage")

   'Initialize the SideMenu
   smc.Initialize(LeftPage, NavControl, RightPage)
'  smc.OpenGesturesEnabled = False 'This would disable the gesture opening
   App.KeyController = smc
   NavControl.ShowPage(Page1)

   'Add two buttons on the top left and right
   Page1.TopRightButtons = Array(smc.CreateBarButton("RightPage"))
   Page1.TopLeftButtons = Array(smc.CreateBarButton("LeftPage"))
End Sub

Private Sub Page1_Resize(Width As Int, Height As Int)
   'You can limit the width of the pages
   smc.LeftMenuMaxWidth = 50%x
End Sub
```

13 Debugging

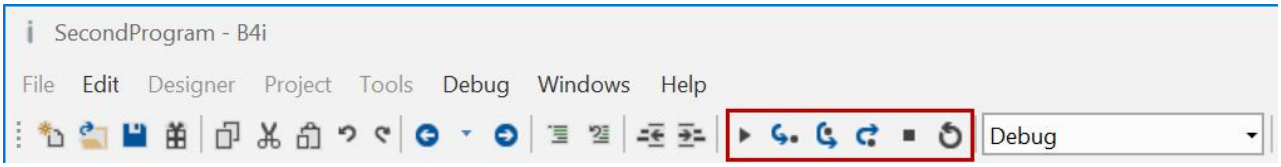
Debugging is an important part when developing.

The two major utilities for debugging are:

Breakpoints - You can mark lines of codes as breakpoints. This is done by pressing on the grey area left of the line.
The program will pause when it reaches a breakpoint and will allow you to inspect the current state.

Logging - The Logs tab at the right pane is very useful. It shows messages related to the components life cycle and it can also show messages that are printed with the Log keyword. You should press on the Connect button to connect to the device logs. Note that there is a Filter checkbox. When it is checked you will only see messages related to your program. When it is unchecked you will see all the messages running in the system. If you are encountering an error and do not see any relevant message in the log, it is worth unchecking the filter option and looking for an error message.
Note that the log is maintained by the device. When you connect to a device you will also see previous messages.

13.1 Debug Toolbar



Debug Toolbar:

	Run the program	F5	Runs the program, no action in Debug (rapid)
	Step In	F8	Executes the next statement.
	Step Over	F9	Executes a routine without jumping in it.
	Step Out	F10	Finishes executing the rest of a routine.
	Stop		Stops the program.
	Restart	F11	Restarts the program.

The examples below use the SecondProgram project.

13.1.1 Run F5

Runs the program,
If the program is stopped at a breakpoint the program runs until the next breakpoint or completes running.

13.1.2 Step In F8

The debugger executes the code step by step.

```
27 Private Sub Application_Start (Nav As
28   NavControl = Nav
29   Page1.Initialize("Page1")
30   Page1.RootPanel.Color = Colors.White
31   Page1.RootPanel.LoadLayout("Main")
32   Page1.Title = "Calc Trainer"
33   NavControl.ShowPage(Page1)
34
35   New
36 End Sub
```

In the SecondProgram project we set a Breakpoint at line 35 `New`.

```
32   Page1.Title = "Calc Trainer"
33   NavControl.ShowPage(Page1)
34
35   New
36 End Sub
```

We run the program, it will stop executing at line 35 `New`.

```
68 Private Sub New
69   Number1 = Rnd(1, 10)          ' Genera
70   Number2 = Rnd(1, 10)          ' Genera
71   lblNumber1.Text = Number1     ' Displa
72   lblNumber2.Text = Number2     ' Displa
73   lblComments.Text = "Enter the result
74   lblComments.Color = Colors.RGB(255,2
75   lblResult.Text = ""           ' Sets l
76   btn0.Visible = False
77 End Sub
```

Click on .
The debugger executes the next line, `Sub New` in this case.

```
68 Private Sub New
69   Number1 = Rnd(1, 10)          ' Genera
70   Number2 = Rnd(1, 10)          ' Genera
71   lblNumber1.Text = Number1     ' Displa
```

Click once more on .
The debugger executes the next line, `Number1 =...`

```
68 Private Sub New
69   Number1 = Rnd(1, 10)          ' Genera
70   Number2 = Rnd(1, 10)          ' Genera
71   lblNumber1.Text = Number1     ' Displa
```

Click once more on .
The debugger executes the next line, `Number2 =...`

13.1.3 Step Over F9

If the current line is a sub calling line the debugger executes the code in this subroutine and jumps to the line after the calling line.

```
27 Private Sub Application_Start (Nav As
28   NavControl = Nav
29   Page1.Initialize("Page1")
30   Page1.RootPanel.Color = Colors.White
31   Page1.RootPanel.LoadLayout("Main")
32   Page1.Title = "Calc Trainer"
33   NavControl.ShowPage(Page1)
34
35   New
36 End Sub
```

In the SecondProgram project we set a Breakpoint at line 35 `New`.

```
32   Page1.Title = "Calc Trainer"
33   NavControl.ShowPage(Page1)
34
35   New
36 End Sub
```

We run the program, it will stop executing at line 35 `New`.

```
32   Page1.Title = "Calc Trainer"
33   NavControl.ShowPage(Page1)
34
35   New
36 End Sub
```

Click on .
The debugger executes the code in `New` and jumps directly to the next line which is `End Sub` of `Application_Start`.

13.1.4 Step Out F10

If the current line is a sub calling line the debugger executes the code in this subroutine and jumps to the line after the calling line.

```
27 Private Sub Application_Start (Nav As
28   NavControl = Nav
29   Page1.Initialize("Page1")
30   Page1.RootPanel.Color = Colors.White
31   Page1.RootPanel.LoadLayout("Main")
32   Page1.Title = "Calc Trainer"
33   NavControl.ShowPage(Page1)
34
35   New
36 End Sub
```

In the SecondProgram project we set a Breakpoint at line 35 `New`.

```
32   Page1.Title = "Calc Trainer"
33   NavControl.ShowPage(Page1)
34
35   New
36 End Sub
```

We run the program, it will stop executing at line 35 `New`.

```
68 Private Sub New
69   Number1 = Rnd(1, 10)          ' Genera
70   Number2 = Rnd(1, 10)          ' Genera
71   lblNumber1.Text = Number1     ' Displa
```

We go step by step with to a line in the subroutine.

```
32   Page1.Title = "Calc Trainer"
33   NavControl.ShowPage(Page1)
34
35   New
36 End Sub
```

Click on .
The debugger executes the rest of the code in the subroutine and jumps to the next line which is `End Sub` of `Application_Start`.

13.1.5 Stop

Stops the program and leaves the Rapid Debugger.

13.1.6 Restart F11

Restarts the program remaining in the Rapid Debugger.
Executes Process_Globals, Globals, Activity_Create and reloads the layout.

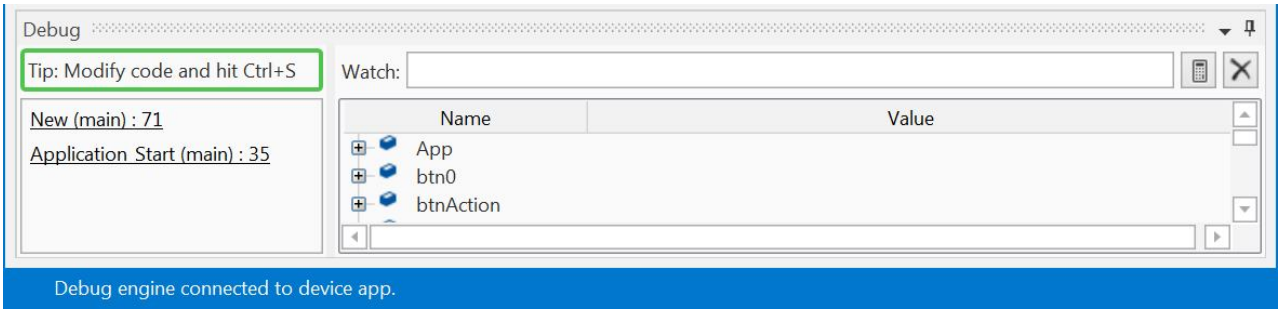
This is useful if you changed a layout file.

It is different from

```
Code changed
Hit Ctrl+S to update.
```

explained in the next chapter.

13.2 Debug window



In the debug window we have (example with the SecondProgram, and a breakpoint in line 72:

13.2.1 The status button

Tip: Modify code and hit Ctrl+S

Shows that the program is running, the button border is green.

When you change the code the button border changes to red.
To update the code click on the button or hit Ctrl + S.

Code changed
Hit Ctrl+S to update.

13.2.2 The breakpoint window

New (main) : 71
Application_Start (main) : 35

The breakpoint window shows where the program has stopped.

```
68  Private Sub New
69    Number1 = Rnd(1, 10)          ' Generates
70    Number2 = Rnd(1, 10)          ' Generates
71    lblNumber1.Text = Number1     ' Displays
72    lblNumber2.Text = Number2     ' Displays
73    lblComments.Text = "Enter the result" &
74    lblComments.Color = Colors.RGB(255,235,
75    lblResult.Text = " "            ' Sets lb
76    btn0.Visible = False
77  End Sub
```

New (main) : 71
The program stopped in line 71, in routine New in the main module.

```
27  Private Sub Application_Start (Nav As Nav
28    NavControl = Nav
29    Page1.Initialize("Page1")
30    Page1.RootPanel.Color = Colors.White
31    Page1.RootPanel.LoadLayout("Main")
32    Page1.Title = "Calc Trainer"
33    NavControl.ShowPage(Page1)
34
35    New
36  End Sub
```

Application_Start (main) : 35
The calling routine is Application_Start, and the calling line 35 New.

When you click on one of the lines the cursor jumps to that line.

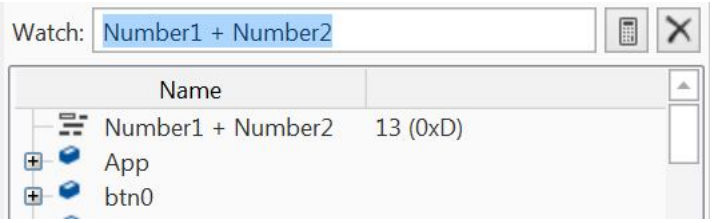
13.2.3 The Watch window

The Watch window allows to check more complex functions for testing and debugging.

```
68  Private Sub New
69      Number1 = Rnd(1, 10)
70      Log(Number1)
71      Number2 = Rnd(1, 10)
72      Log(Number2)
73      lblNumber1.Text = Number1
74      lblNumber2.Text = Number2
```

In the same program, as in the previous subchapter with the two Log line, we set a breakpoint in line 73.

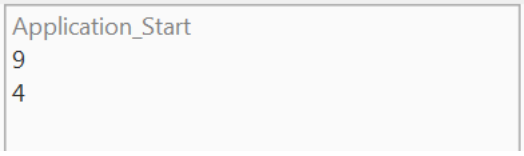
Run the program.



In the Add Watch field enter:

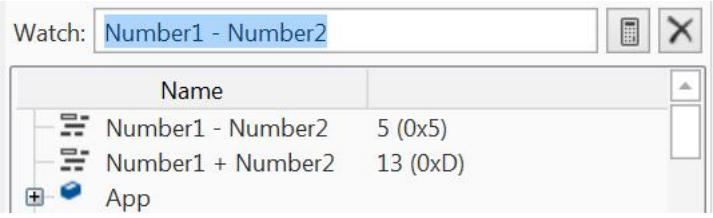
`Number1 + Number2`

Click on  to show the result on top of the list.



As we left the two Log lines in the code we still see the values of Number1 and Number2.

You can enter a new watch line and show it.



Click on  to remove the watch functions. This removes all the functions.

We could, of course, also have done this test with a Log.

13.2.4 The object window

Shows all variables and objects in the list ordered by alphabetical order.

Click on ⊞ to show the details of the object:

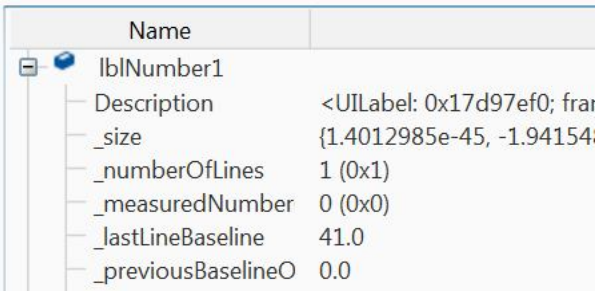
Examples:

- Number1

Shows the current value (8).

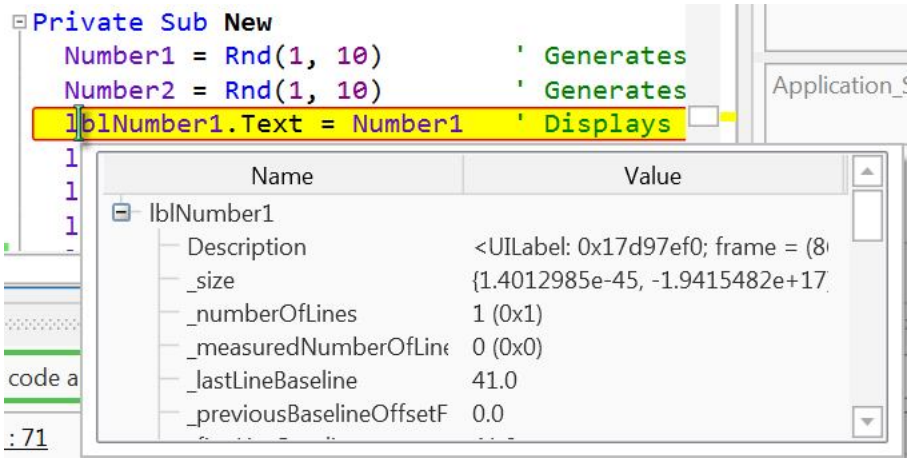
- lblNumber1

Shows all properties of the object, a Label in the example.

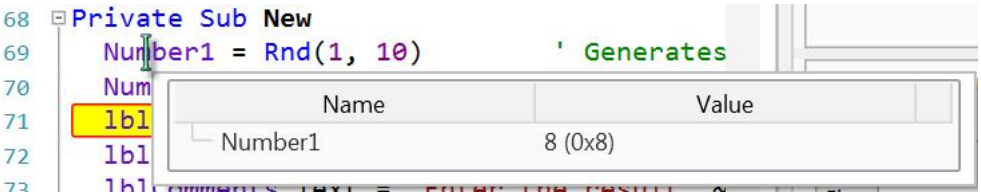


You get the same information when you hover over the object in the code:

lblNumber1



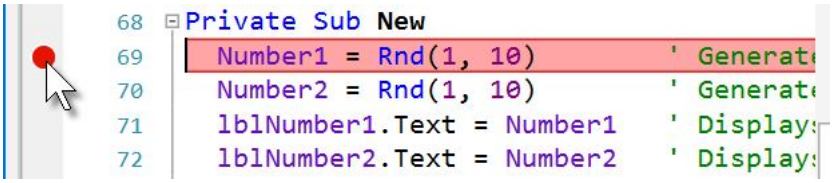
Number1



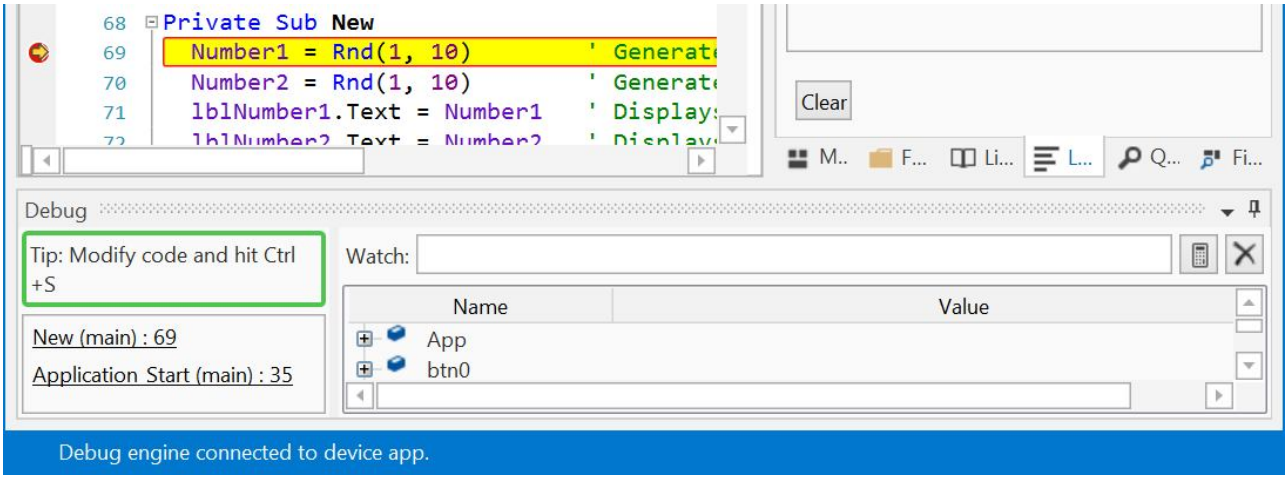
13.3 Breakpoints

One important feature to make debugging easier are breakpoints. You can set breakpoint almost wherever you want in the code.
Breakpoints, in Process_Globals are ignored.

Clicking on a line in the left margin adds a breakpoint. When the program is running it stops at the first encountered breakpoint.



Run the program, the program stops at the breakpoint and the IDE looks like below. The breakpoint line is highlighted in yellow.



On the bottom of the window you see the debug window.

Example with the SecondProgram:

```
68  Private Sub New
69    Number1 = Rnd(1, 10)
70    Number2 = Rnd(1, 10)
71    lblNumber1.Text = Number1
```

Set a breakpoint in line 69 and run the program.

Name	
lblResult	
NavControl	
Number1	0 (0x0)
Number2	0 (0x0)
Page1	

In the variable window look at Number1 and Number2:

The values are 0 for both.
If you see this ⊞ at the left side of Number1 or Number2 click on it to show the details.

```
68  Private Sub New
69    Number1 = Rnd(1, 10)
70    Number2 = Rnd(1, 10)
71    lblNumber1.Text = Number1
```

Click on .
The program jumps to the next line.

Name	
NavControl	
Number1	
Description	6 (0x6)
Number2	0 (0x0)
Page1	

Click on ⊞.
You see that the value of Number1 has changed.

```
68  Private Sub New
69    Number1 = Rnd(1, 10)
70    Number2 = Rnd(1, 10)
71    lblNumber1.Text = Number1
72    lblNumber2.Text = Number2
```

Click on again.
The program jumps to the next line.

Name	
Number1	
Description	6 (0x6)
Number2	
Description	5 (0x5)
Page1	

Click on ⊞.
You see that the value of Number2 has changed.

The best way to learn debugging is testing, testing and testing!

13.4 With Logs

Example with the SecondProgram.

```
68 Private Sub New
69   Number1 = Rnd(1, 10)
70   Log(Number1)
71   Number2 = Rnd(1, 10)
72   Log(Number2)
73   lblNumber1.Text = Number1
```

We add the two lines with the Log keyword to display the two numbers in the Log Tab.
We and add a breakpoint in line 69 to watch what happens.

```
68 Private Sub New
69   Number1 = Rnd(1, 10)
70   Log(Number1)
71   Number2 = Rnd(1, 10)
72   Log(Number2)
73   lblNumber1.Text = Number1
```

Run the program, it stops at line 69.

```
Application_Start
```

Clear

M.. F... L... L... Q.. F...

In the Log Tab we see at the moment only `Application_Start` telling that the program has started.

```
68 Private Sub New
69   Number1 = Rnd(1, 10)
70   Log(Number1)
71   Number2 = Rnd(1, 10)
72   Log(Number2)
73   lblNumber1.Text = Number1
74   lblNumber2.Text = Number2
```

Click four times on till the program reaches line 73.

```
Application_Start
8
3
```

In the Log Tab we see the values of the two variables.

```
Application_Start
8
3
Application_Active
```

Click on to run to the end.

We see that the program has passed the routine `Application_Active`.

```
Application_Start
8
3
Application_Active
8
6
```

When you are using the program the two new values will be shown every time the program runs the New routine.

13.5 Modifying code in the Debugger

It is possible to change the code in the Debugger and see the new behavior without restarting the program.

Still with SecondProgram and the two Logs and the breakpoint in line 73.

```
68 Private Sub New
69     Number1 = Rnd(1, 10)
70     Log(Number1)
71     Number2 = Rnd(1, 10)
72     Log(Number2)
73     lblNumber1.Text = Number1
74     lblNumber2.Text = Number2
```

Run the program till it stops at the breakpoint.

```
68 Private Sub New
69     Number1 = Rnd(1, 20)
70     Log(Number1)
71     Number2 = Rnd(1, 20)
72     Log(Number2)
73     lblNumber1.Text = Number1
74     lblNumber2.Text = Number2
```

We change the two numbers 10 to 20.

```
Code changed
Hit Ctrl+S to update.
```

The status button color has changed confirming a code change. To rerun the program click on Ctrl + S.

Using the program we see now that the numbers can be between 1 and 19.

```
Application_Start
8
1
13
6
```

14 Example programs

In this chapter several example program are explained with code examples.
All projects are saved in the SourceCode folder shipped with the guide.

No programs yet.

15 Graphics / Drawing

15.1 Overview

To draw graphics we need to use a Canvas object.

Explanation from the help file.
Canvas is used for drawing over other views.
The drawings will only be updated after a call to Canvas.Refresh. Note that you should call Canvas.Release when it is no longer used.
If the hosting is resized then the canvas should be released and initialized again.

The most useful views to draw on are:

- ImageView
- Panel
- **Don't use a Canvas with Page.RootPanel, you will get strange behaviors.**

A Canvas must be dimed in Process_Globals and initialized in Page_Resize before it can be used.
Example:

```
Sub Process_Globals
   Private pnlLayout0 As Panel
   Private cvsLayout0 As Canvas
```

And then

```
Private Sub Page1_Resize(Width As Int, Height As Int)
   cvsLayer0.Initialize(pnlLayer0)
```

In the following methods you will find a number of common parameters.

- Bitmap1 as Bitmap — a bitmap
- x, y. x1, y1, x2, y2 As Float — are coordinates, Float variables.
- Color as Int — are color variables. Int variables
- DestRect, Rect1 As Rect — are rectangles, Rect objects
- Filled As Boolean — flag if the surface is filled (True) or not (False)

The most common drawing functions are:

- **DrawBitmap** (Bitmap1 As Bitmap, DestRect As Rect).
  Draws the given bitmap or only a part of it.
  DestRect = destination rectangle, can be any size.
  Do draw with the same size both rectangles must have same width and same height.

- **Draw BitmapRotated** (Bitmap1 As Bitmap, DestRect As Rect, Degrees As Float)
  Same function as DrawBitmap, but with a rotation of the given Degrees angle around the center of the bitmap.

- **DrawCircle** (x As Float, y As Float, Radius As Float, Color as Int, Filled As Boolean, StrokeWidth As Float)
  Draws a circle.
  x and y are the center coordinates of the circle and Radius the circles radius.

- **DrawColor** (Color As Int)
  Fills the entire canvas with the given color. Note that you can use ClipPath to clip the drawings to a specific region.
  The color can be Colors.Transparent making the whole view transparent.

- **DrawLine** (x1 As Float, y1 As Float, x2 As Float, y2 As Float, Color as Int, StrokeWidth As Float)
  Draws a straight line between two points.

- **DrawPath**(Path1 As Path, Color As Int, Filled As Boolean, StrokeWidth as Float)
  Draws or fills the given path, color, filled or not and line width.
  Path1=Font the Path that will be filled or drawn.

- **DrawRect** (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth As Float)
  Draws a rectangle with given size, color, filled or not and line width.

- **DrawRectRounded** (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth As Float, CornerRadius As Float)
  Same as DrawRect but with rounded corners with the given radius.

- **DrawRectRotated** (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth As Float, CornerRadius as Float, Degrees As Float)
  Draws a rotated rectangle with given size, color, filled or not, line width, corner radius and rotation angle.

- **DrawText** (Text As String, x As Float, y As Float, Font1 As Font, Color As Int, Align1 As Align).
  Draws the given text.
  Font1 = Font object
  Align1 can be either : LEFT, CENTER or RIGHT

- **DrawTextRotated** (Text As String, x As Float, y As Float, Font1 As Font, Color As Int Align1 As Align, Degrees As Float)
  Similar to DrawText. Draws rotated text.
  Font1 = Font object

- **DrawView** (View1 As View, DestRect As Rect)
  Draws the given view in the DestRect rectangle.

- **FillGradient**(x1 As Float, y1 As Float, x2 As Float, y2 As Float, Colors As List)
  Paints a gradient fill along the two points.
  x1, y1 - Starting point.
  x2, y2 - End point.
  Colors - A list (or array) with the gradient colors.

15.2 Coordinates

Coordinate values are different in B4i from those in B4A.

Extract from the iOS documentation:

Points Versus Pixels

In iOS there is a distinction between the coordinates you specify in your drawing code and the pixels of the underlying device. When using native drawing technologies such as Quartz, UIKit, and Core Animation, the drawing coordinate space and the view's coordinate space are both **logical coordinate spaces**, with distances measured in **points**. These logical coordinate systems are decoupled from the device coordinate space used by the system frameworks to manage the pixels onscreen.

The system automatically maps points in the view's coordinate space to pixels in the device coordinate space, but this mapping is not always one-to-one. This behavior leads to an important fact that you should always remember:

One point does not necessarily correspond to one physical pixel.

The purpose of using points (and the logical coordinate system) is to provide a consistent size of output that is device independent. For most purposes, the actual size of a point is irrelevant. The goal of points is to provide a relatively consistent scale that you can use in your code to specify the size and position of views and rendered content. How points are actually mapped to pixels is a detail that is handled by the system frameworks. For example, on a device with a high-resolution screen, a line that is one point wide may actually result in a line that is two physical pixels wide. The result is that if you draw the same content on two similar devices, with only one of them having a high-resolution screen, the content appears to be about the same size on both devices.

B4i **B4A**

1 Point = **1 dip** (**d**evice **i**ndependent **p**ixel)

	B4i	**B4A**
Coordinates	Point	Pixel
View dimensions	Point	Pixel
Drawing coordinates	Point	Pixel
Adding a view	`AddView(10, 10, 150, 50)`	`AddView(10dip, 10dip, 150dip, 50dip)`
Standard dpi (dots per inch)	160	16'
`GetDeviceLayoutValues` Example 320 dpi screen		
Scale	Always 1	2
NonnormalizedScale	2	---
Width	Point	Pixel
Height	Point	Pixel
1 Pixel	1 Point / NonnormalizedScale	---
1 Point	---	1 Pixel * Scale

15.3 Transparency

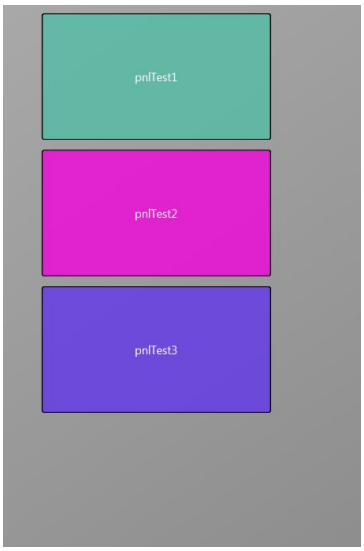
There are a few differences between B4A and B4i with transparency.

The test project, Transparency, is in the SourceCode folder shipped with the Beginner's Guide.

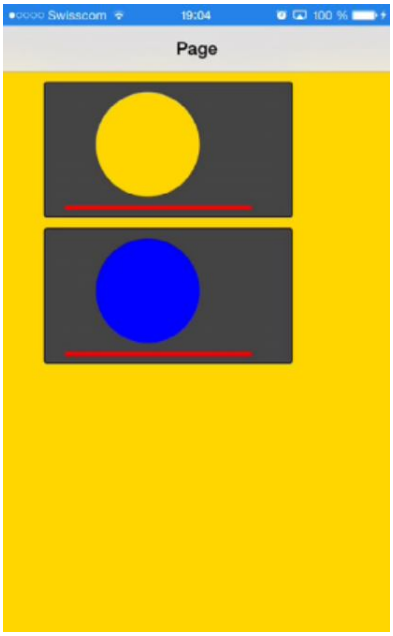
Let's consider three examples with a Panel defined in the Designer and a Canvas for each and the same drawing functions:

```
cvsTest1.DrawColor(Colors.DarkGray)
cvsTest1.DrawCircle(100, 60, 50, Colors.Transparent, True, 1)
cvsTest1.DrawLine(20, 120, 200, 120, Colors.Red, 4)
cvsTest1.Refresh
```

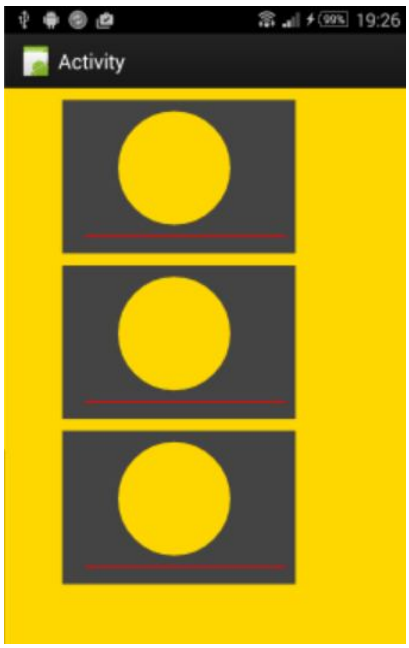
	Case 1	Case 2	Case 3
Original color	Transparent	Blue	Transparent
Alpha	1	1	0
Page background color	Yellow	Yellow	Yellow
DrawColor	DarkGray	DarkGray	DarkGray
DrawCircle color	Transparent	Transparent	Transparent
DrawLine color	Red	Red	Red
Panel color on screen	DarkGray	DarkGray	Yellow
Line color on screen	Red	Red	Not visible
Circle color on screen	Yellow Page background color	Blue Original color	Not visible



Abstract designer.



Result on the device.



Same project in Android.

15.4 Example programs

15.4.1 First steps Example program

The project is in: SourceCode\Graphics\GraphicsFirstSteps.b4t

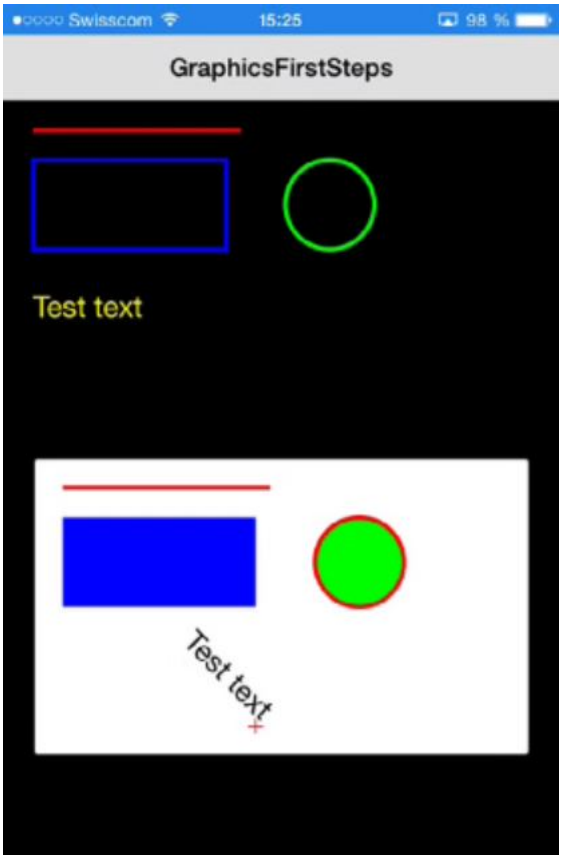
To draw something we need a Canvas object which is simply a drawing tool.
The Canvas draws onto a Bitmap. This Bitmap can be the background bitmap of views.

The most common views to draw on are: Panel, ImageView or a Bitmap.
The Canvas must be 'connected' to a view background image in the Initialize method.

- Initialize(Target View)

If we want to draw on different views at the same time we need one Canvas for each view.

In the example program we'll use several drawing functions and draw onto the Activity and onto a Panel `pnlGraph` defined in the 'main' layout file. Here we need two canvases.



15.4.1.1 Start Dim and Initialisation

First we must dim the different views and objects:
We have:

- the Panel `pnlGraph`
- the Canvas `cvsPage` for the Page
- the Canvas `cvsPanel` for the Panel

```
Sub Process_Globals
   Public App As Application
   Public NavControl As NavigationController
   Private Page1 As Page

   Private pnlGraph As Panel
   Private cvsPage, cvsGraph As Canvas
End Sub
```

Then we must load the layout file.

```
Private Sub Application_Start (Nav As NavigationController)
   NavControl = Nav
   Page1.Initialize("Page1")
   ' load the layout file
   Page1.RootPanel.LoadLayout("main")
   ' show page 1
   NavControl.ShowPage(Page1)
End Sub
```

And we initialize the two Canvases and call the Drawing routine.
We need to initialize the canvases in the Page1_Resize routine and not in Application_Start to make sure that the sizes are correct!

```
Private Sub Page1_Resize(Width As Int, Height As Int)
   ' initialize the Canvas for the activity
   cvsPage.Initialize(Page1.RootPanel)

   ' initialize the Canvas for the pnlGraph panel
   cvsGraph.Initialize(pnlGraph)

   Drawing
End Sub
```

15.4.1.2 Draw a line

Then in the Drawing routine we draw a horizontal line onto Page1: 
The function is:
DrawLine (x1 As Float, y1 As Float, x2 As Float, y2 As Float, Color as Int, StrokeWidth As Float)
Where:

- x1, y1 are the coordinates of the start point in pixels
- x2, y2 are the coordinates of the end point in pixels
- Color is the line color
- StrokeWidth the line thickness in pixels

And the code:

```
' draw a horizontal line onto the Activity
cvsPage.DrawLine(20, 20, 160, 20, Colors.Red, 3)
```

Then we draw a horizontal line onto pnlGraph with the same coordinates:
The coordinates are relative to the upper left corner of the view we draw on, the Panel `pnlGraph` in this case.

```
' draw a horizontal line onto pnlGraph
cvsGraph.DrawLine(20, 20, 160, 20, Colors.Red, 3)
```

15.4.1.3 Draw a rectangle

Then we draw an empty rectangle onto Page1: 
The function is:
DrawRect (Rect1 As Rect, Color As Int, Filled As Boolean, StrokeWidth as Float)
Where:

- Rect1 is a rectangle object
- Color is the border or rectangle color
- Filled: False = only the border is drawn True = the rectangle is filled
- StrokeWidth is the line thickness of the border, not relevant when Filled = True

To draw a rectangle we need a Rect object.
We:

- Define it with the name `rect1`.
- Initialize it with the coordinates of the upper left corner and the coordinates of the lower right corner.
- Draw it

```
' draw an empty rectangle onto the Activity
Dim rect1 As Rect
rect1.Initialize(20, 40, 150, 100)
cvsPage.DrawRect(rect1, Colors.Blue, False, 3)
```

Then we draw a filled rectangle onto pnlGraph with the same coordinates: 

We don't need to define nor initialize a new rectangle because the coordinates are the same so we use the same Rect object.

```
' draw a filled rectangle onto pnlGraph
cvsPage.DrawCircle(220, 70, 30, Colors.Green, False, 3)
```

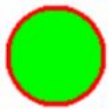
15.4.1.4 Draw a circle

Then we draw an empty circle onto Page1: 
The function is:
DrawCircle (x As Float, y As Float, Radius As Float, Color as Int, Filled As Boolean, StrokeWidth As Float)
Where:

- x, y are the coordinates of the center in pixels.
- Radius is the radius in pixels.
- Color is the border or circle color
- Filled: False = only the border is drawn True = the circle is filled
- StrokeWidth is the line thickness of the border, not relevant when Filled = True

And the code:

```
' draw an empty circle onto the Activity
cvsPage.DrawCircle(220, 70, 30, Colors.Green, False, 3)
```

Then we draw a filled circle with a border with a different color on the panel. 

There is no direct function to draw a filled circle with a border with a different colors.
So we first draw the filled circle and then the circle border in two steps.
Instead of using fixed values like 220 we can also use variables like in the code below.
When a same value is used several times it's better to use variables because if you need to change the value you change it only once the value of the variable all the rest is changed automatically by the variable.

```
' draw a filled circle with a boarder onto pnlGraph
Dim centerX, centerY, radius As Float
centerX = 220
centerY = 70
radius = 30
cvsGraph.DrawCircle(centerX, centerY, radius, Colors.Green, True, 3)
cvsGraph.DrawCircle(centerX, centerY, radius, Colors.Red, False, 3)
```

15.4.1.5 Draw a text

Then we draw a text onto Page1. Test text
The function is:
DrawText (Text As String, x As Float, y As Float, Font1 As Font, Color As Int, Align1 As Align)
Where:

- Text is the text to draw
- x, y are the coordinates of the reference point (depending on the Align1 value) in pixels. The reference point is on the texts baseline.
- Font1 is the Font object
- Color is the text color
- Align1 is the alignment of the text according to the reference point. Possible values: "LEFT", "CENTER", "RIGHT".

And the code:

```
' draw a text onto the Page
cvsPage.DrawText("Test text", 20, 150, Font.CreateNew(20), Colors.Yellow, "LEFT")
```

Then we draw a rotated text onto pnlGraph.

And we draw a cross on the reference point to show where it is and how the alignment does work.
The function is DrawTextRotated, it's the same as DrawText but with an additional parameter Degrees, the rotation angle.
Instead of using fixed values in the routine we define three variables:

refX and refY	the coordinates of the reference point
hl	the half of the cross line length

```
Dim refX, refY, hl As Float
refX = 150
refY = 180
hl = 5
' draw a rotated text onto pnlGraph
cvsGraph.DrawTextRotated("Test text", refX, refY, Font.CreateNew(20), Colors.Black,
"RIGHT", 45)

' draw a cross on the reference point
cvsGraph.DrawLine(refX - hl, refY, refX + hl, refY, Colors.Red, 1)
cvsGraph.DrawLine(refX, refY - hl, refX, refY + hl, Colors.Red, 1)
```

And at the end we need to ‘refresh’ the canvases to show the result.

```
cvsPage.Refresh
cvsGraph.Refresh
```

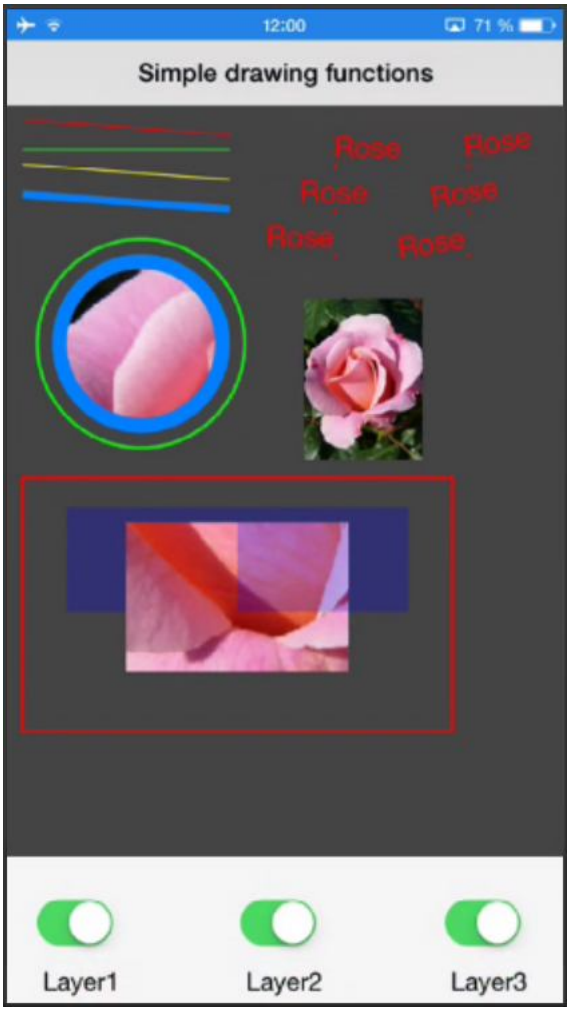
15.4.2 Simple draw functions Example program

The project is in: SourceCode\Graphics\SimpleDrawFunctions\SimpleDrawFunctions.b4i

In the second drawing program, SimpleDrawFunctions, we use the other common drawing functions.

The program has no other purpose than show what can be done with drawings.

The program has four Panels which we use as layers, one for the background and three layers with three Switches allowing to show or hide the different layers.
The background layer has an image, Layer(0) has a grey background and the two other layers have a transparent background.

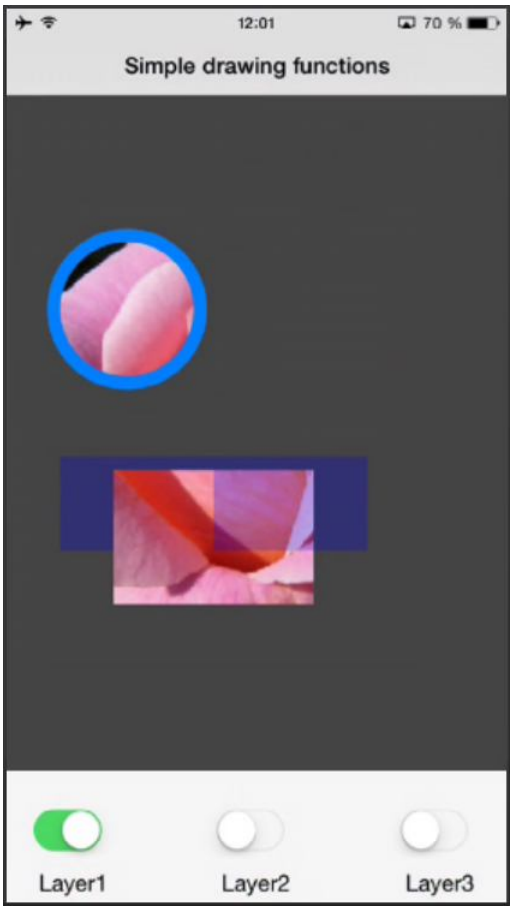


You can play with the switches to observe the different combinations of visible and hidden layers.

The layout is defined in the Designer.

In this screenshot we solely see the background image of the background layer.

We use the Switches to either show or hide the different layers.

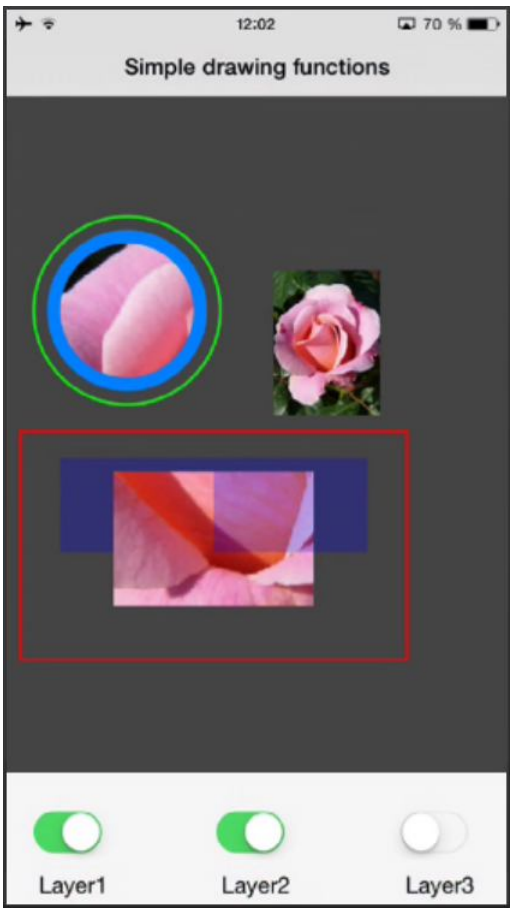


Here we show layer(0).

The panel has a dark gray background with:

- a blue circle.
- a transparent circle, the background image appears inside this circle.
- two blue, semi-transparent rectangles, the left one is drawn before the transparent rectangle and the second one is drawn after the transparent rectangle.
- a transparent rectangle, the background image appears inside this rectangle.

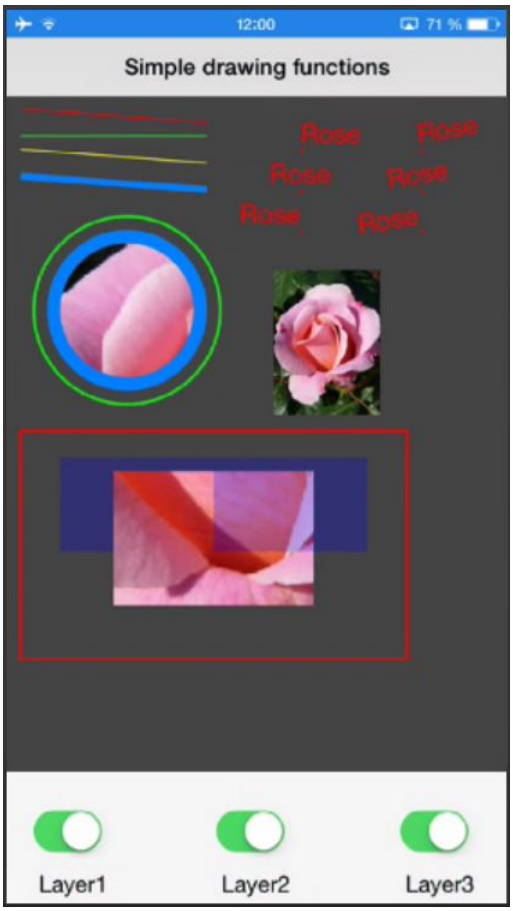
Touching the screen and moving the finger moves the blue and transparent circles on layer(0).



Here we show layer(1) on top of layer(0).

The panel has a transparent background with:

- a green circle.
- a small copy of the background image.

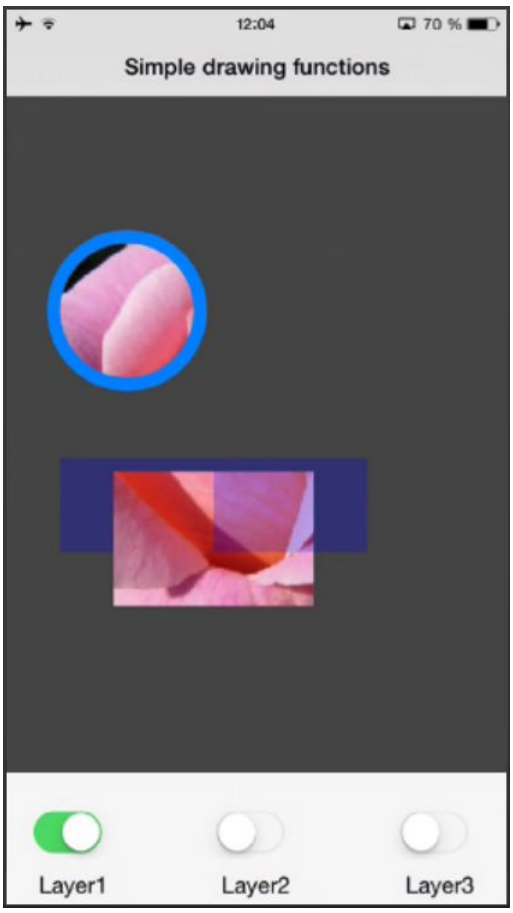


Here we show layer(2) on top of layer(0) and layer(1).

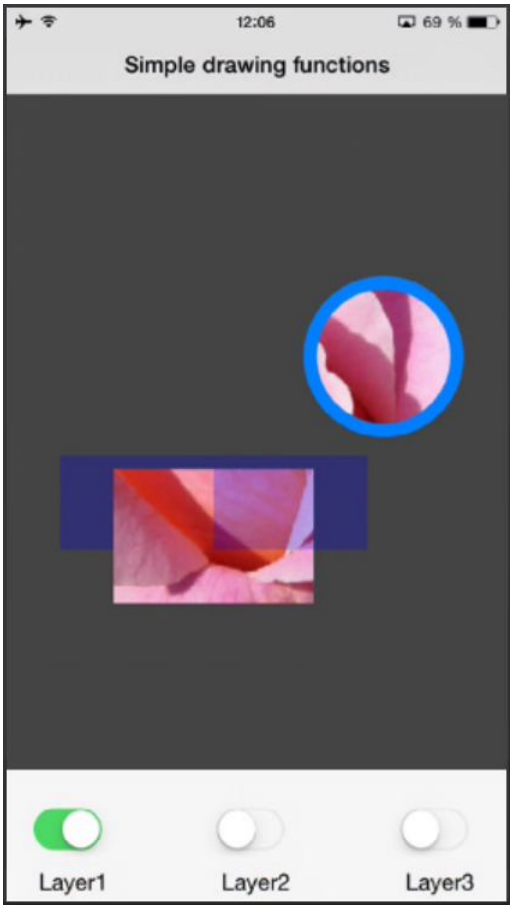
The panel has a transparent background with:

- 4 lines on top.
- 3 horizontal texts with the three different alignments.
- 3 rotated texts with the three different alignments.
- a point for each text showing the position of the text reference point.

You can play with the buttons to show the different combinations of visible and hidden layers.



Touching the screen with the finger and moving it, moves the blue and transparent circles.



On each move, the background image of the activity appears.

Analysis of the code:

In the Sub Process_Globals routine we have:

- 3 application variables.
- 4 Panels, one background and 3 layers.
- 4 Canvases, one for each panel.
- 2 Rects, one for the background and the other is used to draw rectangles.
- 1 Bitmap, holding the background image
- different variables used for the drawings.

Note that we use arrays of views for the three layer panels and the canvases.

```
Sub Process_Globals
   Public App As Application
   Public NavControl As NavigationController
   Private Page1 As Page

   Private pnlLayer(3) As Panel
   Private pnlBackground, pnlLayer1, pnlLayer2, pnlLayer3 As Panel
   Private cvsLayer(3) As Canvas
   Private cvsBackground As Canvas
   Private rectBG, rect1 As Rect
   Private bmpBackground As Bitmap
   Private xc, yc, x1, y1, x2, y2, r1, r2, h, w As Float
End Sub
```

In the Sub Application_Start we:

- Set the layers array.
- Show page 1
- Set the UserInteractionEnabled property to false.
- Load the `rose2.jpg` image file into the bitmap.
- Initialize the background image of the activity.

```
Private Sub Application_Start (Nav As NavigationController)
   NavControl = Nav
   Page1.Initialize("Page1")
   Page1.RootPanel.LoadLayout("Main")

   ' set the layer panel array
   pnlLayer = Array As Panel(pnlLayer0, pnlLayer1, pnlLayer2)

   NavControl.ShowPage(Page1)

   ' set the two upper layers user unterfaces disabled
   ' otherwise pnlLayer0_Touch wouldn't work when covered layer.
   ' by one overlaying
   pnlLayer(1).UserInteractionEnabled = False
   pnlLayer(2).UserInteractionEnabled = False

   ' initialize the background image
   bmpBackground.Initialize(File.DirAssets, "rose2.jpg")
End Sub
```

In the Sub Page1_Resize routine we:

- Calculate and set the panel heights.
  The panel horizontal anchors are set to BOTH, we must calculate the height of the panels according to the new width.
- Initialize the canvases.
  The canvases must be declared here because of the dimension change of the panels.
- Call the Drawing routine.

```
Private Sub Page1_Resize(Width As Int, Height As Int)
   Dim i As Int
   Dim Scale As Double

   Scale = pnlBackground.Width / bmpBackground.Width
   pnlBackground.Height = bmpBackground.Height * Scale
   For i = 0 To 2
      pnlLayer(i).Height = bmpBackground.Height  * Scale
      cvsLayer(i).Initialize(pnlLayer(i))
   Next

   cvsBackground.Initialize(pnlBackground)

   Drawing
End Sub
```

In the Sub Drawing routine we:

- Initialize rectBG, the background rectangle to draw the background image.
- Draw the background image and Refresh the canvas.
- Draw the layout(0) background dark gray.
- Draw the layout(1) and layout(2) background transparent.

```
Sub Drawing
   ' draw the background image
   rectBG.Initialize(0, 0, pnlBackground.Width, pnlBackground.Height)
   cvsBackground.DrawBitmap(bmpBackground, rectBG)
   cvsBackground.Refresh

   ' set the layer backgrounds
   cvsLayer(0).DrawColor(Colors.DarkGray)
   cvsLayer(1).DrawColor(Colors.Transparent)
   cvsLayer(2).DrawColor(Colors.Transparent)
```

- Draw the three circles on layer(0) and layer(1)

```
   xc = 90
   yc = 160
   r1 = 70
   cvsLayer(1).DrawCircle(xc, yc, r1, Colors.Green, False, 2)
   r1 = 60
   cvsLayer(0).DrawCircle(xc, yc, r1, Colors.RGB(0, 128, 255), True, 3)
   r2 = 50
   cvsLayer(0).DrawCircle(xc, yc, r2, Colors.Transparent, True, 1)
```

- Draw the rectangles on layer(0) and layer(1)

```
rect1.Initialize(10, 250, 300, 420)
cvsLayer(1).DrawRect(rect1, Colors.Red, False, 2)
rect1.Initialize(40, 270, 155, 340)
cvsLayer(0).DrawRect(rect1, Colors.ARGB(128, 0, 0, 255), True, 2)
rect1.Initialize(80, 280, 230, 380)
cvsLayer(0).DrawRect(rect1, Colors.Transparent, True, 2)
rect1.Initialize(155, 270, 270, 340)
cvsLayer(0).DrawRect(rect1, Colors.ARGB(128, 0, 0, 255), True, 2)

rect1.Initialize(200, 130, 280, 238)
cvsLayer(1).DrawBitmap(bmpBackground,rect1)
```

- Draw four lines onto layer(2).

```
x1 = 10
y1 = 10
x2 = 150
y2 = 20
cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Red, 1)
y1 = 30
y2 = 30
cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Green, 0.99)
y1 = 40
y2 = 50
cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.Yellow, 0.99)
y1 = 60
y2 = 70
cvsLayer(2).DrawLine(x1, y1, x2, y2, Colors.RGB(0, 128, 255), 5)
```

- Draw the three horizontal texts.
  DrawRect draws the reference point of the text.

```
x1 = 220
y1 = 40
cvsLayer(2).DrawText("Rose", x1, y1, Font.CreateNew(20), Colors.Red, "LEFT")
rect1.Initialize(x1, y1, x1 + 2, y1 + 2)
cvsLayer(2).DrawRect(rect1, Colors.Red, True, 1)
y1 = 70
cvsLayer(2).DrawText("Rose", x1, y1, Font.CreateNew(20), Colors.Red, "CENTER")
rect1.Initialize(x1, y1, x1 + 2, y1 + 2)
cvsLayer(2).DrawRect(rect1, Colors.Red, True, 1)
y1 = 100
cvsLayer(2).DrawText("Rose", x1, y1, Font.CreateNew(20), Colors.Red, "RIGHT")
rect1.Initialize(x1, y1, x1 + 2, y1 + 2)
cvsLayer(2).DrawRect(rect1, Colors.Red, True, 1)
```

- Draw the three rotated texts.
  Similar to the code above, but rotated with the rotation angle as the last parameter-.

```
cvsLayer(2).DrawTextRotated("Rose", x1, y1, Font.CreateNew(20), Colors.Red, "LEFT", -
10)
```

Looking closer on the displayed texts we see the reference point for each text.

```
cvsLayer(2).DrawText("Rose", x1, y1, Font.CreateNew(20), Colors.Red, "LEFT")
rect1.Initialize(x1, y1, x1 + 2, y1 + 2)
cvsLayer(2).DrawRect(rect1, Colors.Red, True, 1)
```

These are the x1 and y1 coordinates used to display the texts.

LEFT alignment.

CENTER alignment.

RIGHT alignment.

In the Sub swtLayer_Checked routine we:

- Dim swt as a local Switch to get the view that raised the event.
- Set swt to Sender, which is the view that raised the event.
- Get the index of the Switch from the Tag property, the Tag property is set in the Designer.
- Change the Visible property from True to False or from False to True.

```
Sub swtLayer_ValueChanged(Value As Boolean)
   Dim swt As Switch
   Dim index As Int

   swt = Sender
   index = swt.Tag
   pnlLayer(index).Visible = Value
End Sub
```

In the Sub pnlLayer0_Touch routine we:

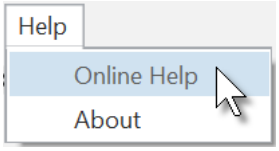
- Draw a dark gray circle to erase the previous blue and transparent circle.
- Set xc and yc to the new coordinates of the circle centers.
- Draw a blue and transparent circle on layer(1).
- Refresh cvsLayout(1) to force the update of the drawing.

```
Sub pnlLayer0_Touch (Action As Int, X As Float, Y As Float)
   cvsLayer(0).DrawCircle(xc, yc, r1 + 2, Colors.DarkGray, True, 3)
   xc = X
   yc = Y
   cvsLayer(0).DrawCircle(xc, yc, r1, Colors.RGB(0, 128, 255), True, 3)
   cvsLayer(0).DrawCircle(xc, yc, r2, Colors.Transparent, True, 1)
   cvsLayer(0).Refresh
End Sub
```

16 Help Tools

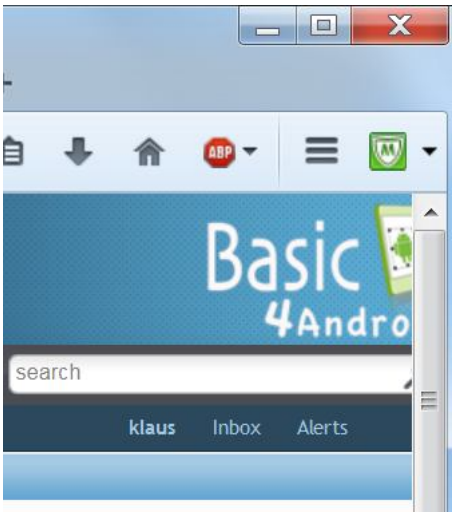
To find answers to questions about B4i the following tools are very useful.

16.1 Online Help link in the IDE



In the IDE Help menu click on Online Help.

16.2 Search function in the forum



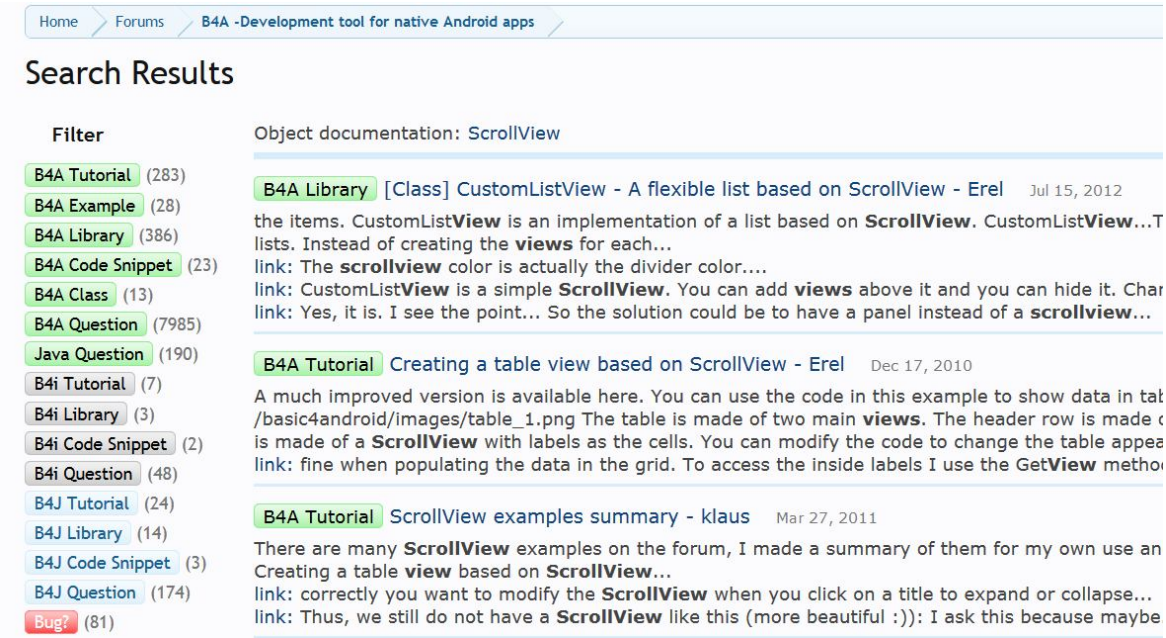
In the upper right corner you find the search box for the forum.

Enter a question or any keywords and press 'Return'.

The function shows you the posts that match your request.

Example: Enter the keyword ScrollView:

And the result:

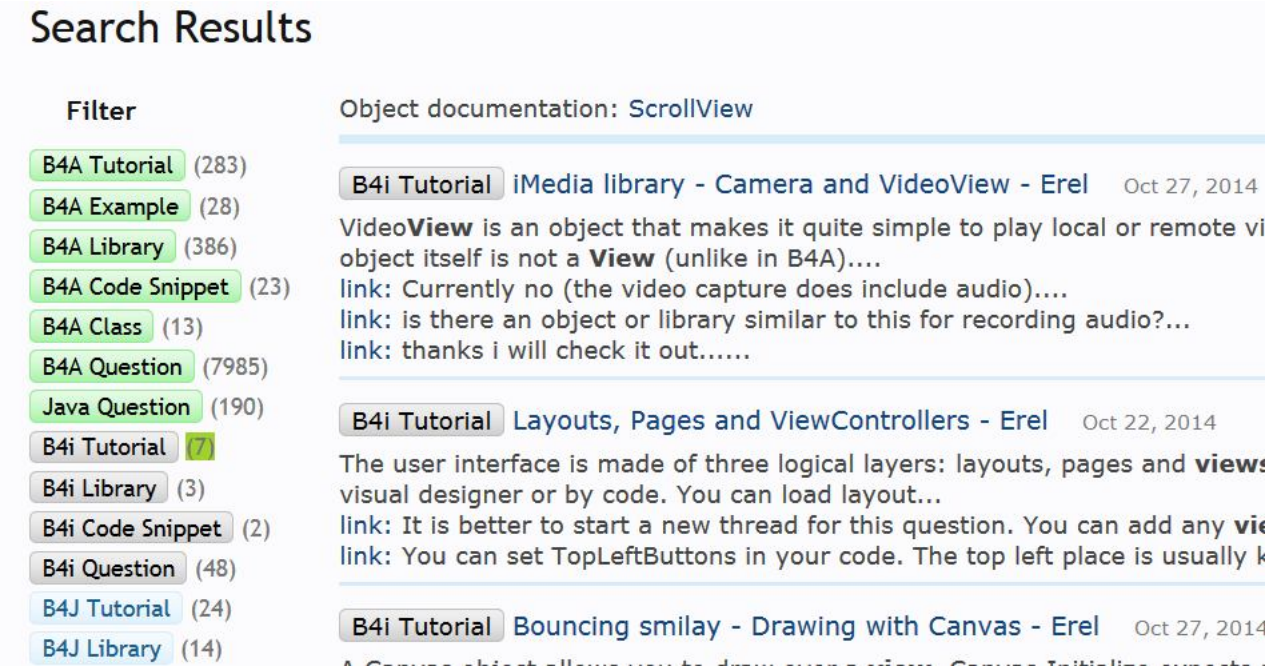


Click on the title to show the selected post.
On the left side of the screen you see buttons allowing to filter the result.

You can then filter the search with the buttons on the left.

Example: Click on B4i Tutorial (7)

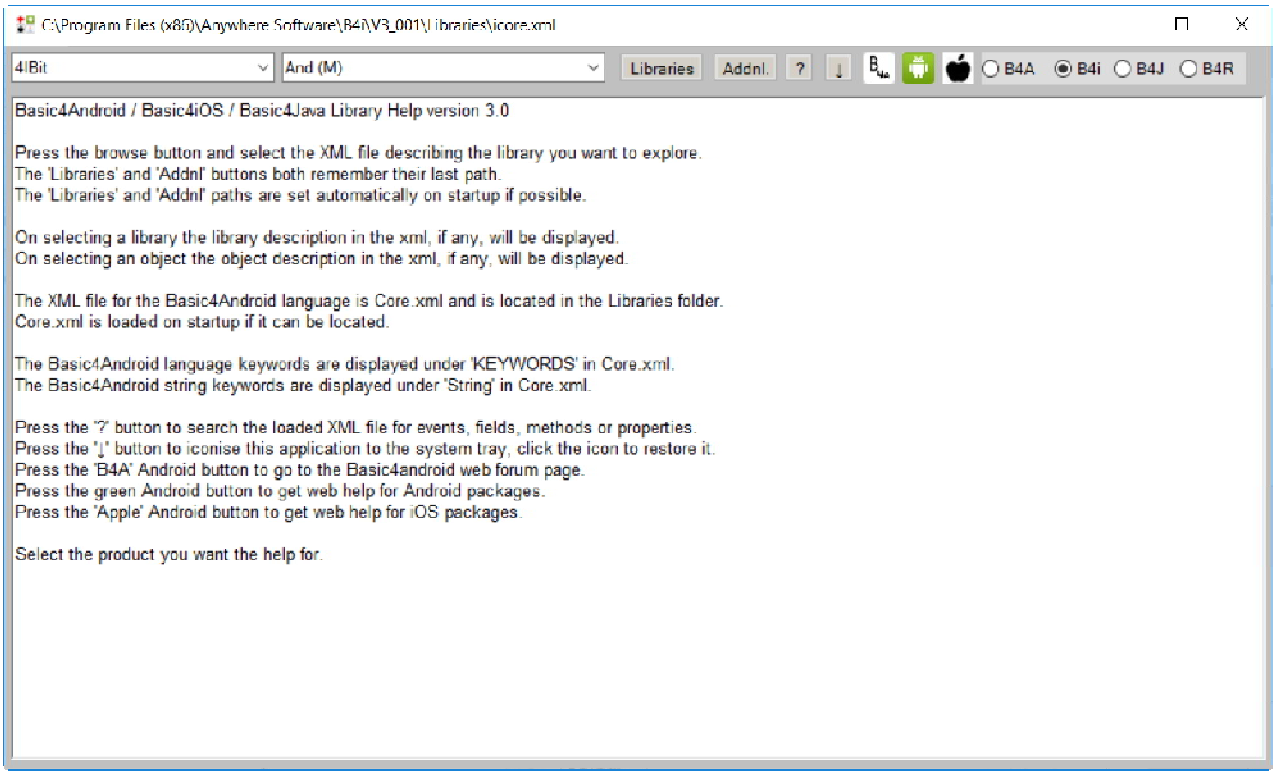
The result may look like this:



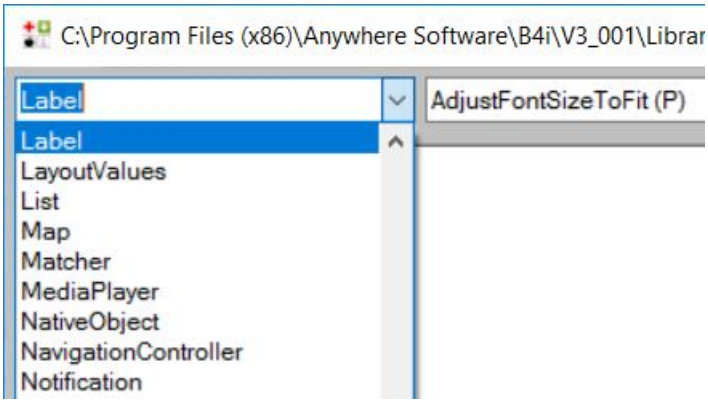
16.3 B4x Help Viewer

This program shows xml help files. It was originally written by Andrew Graham (agrham) for B4A. I modified it, with Andrews' agreement, to show B4A, B4J, B4i and B4R xml help files.

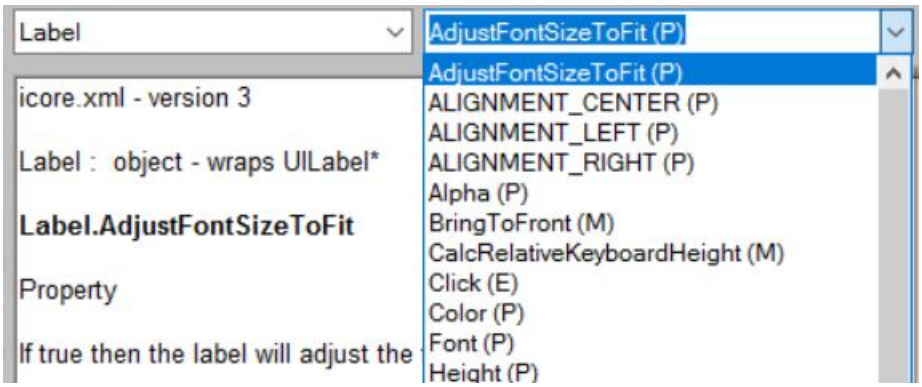
The program can be downloaded from the forum.



On top we find:



In the upper left corner a drop down list shows the different objects included in the selected library.



Besides the objects list you find another drop down list with the

- methods(M)
- events(E)
- properties(P)
- fields(F) constants for the selected object.

Select an object.

Select a property.

Select a standard library.

Select an additional library.

Search for a given text.

Closes B4xHelp

Link to the B4i / B4A forum.

Link to Android documentation.

Link to iOS documentation.

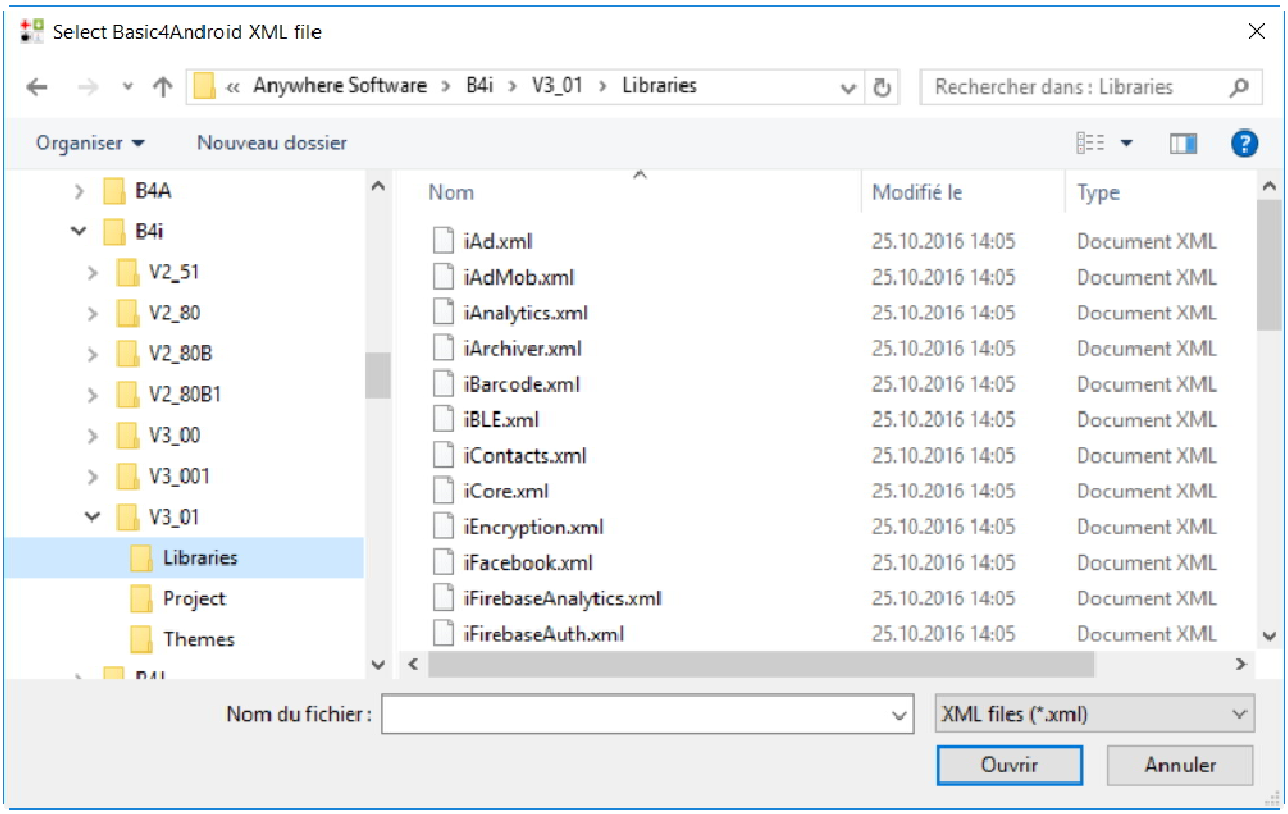
B4A help files.

B4i help files.

B4J help files.

B4R help files.

Libraries Standard libraries



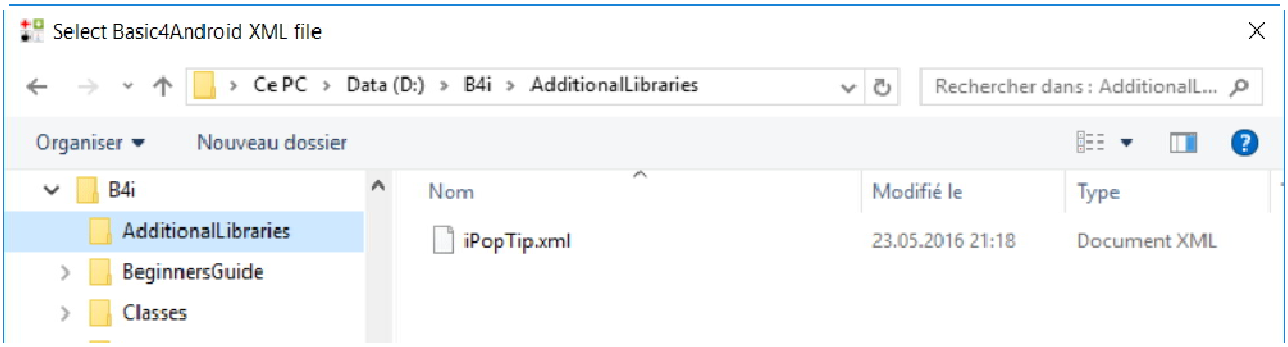
Select the library to display and click on Ouvrir (Open).

Here « Anywhere Software › B4i › V3_01 › Libraries you can select the directory where the standard libraries are saved.

Once selected the directory is saved for the next start of the program.

Additional libraries.

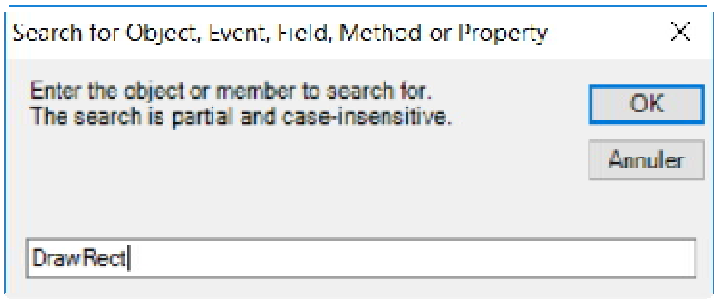
The same also for the additional libraries.



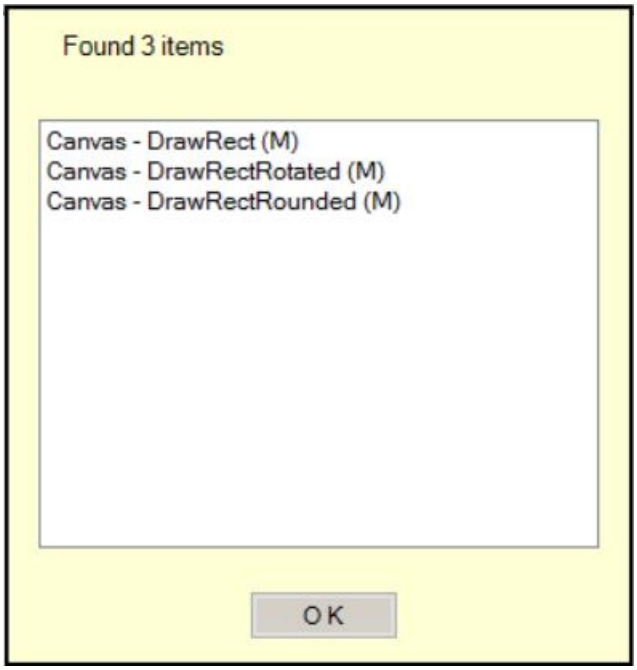
Here 

you can select the directory for the additional libraries.

Search engine for the selected library.

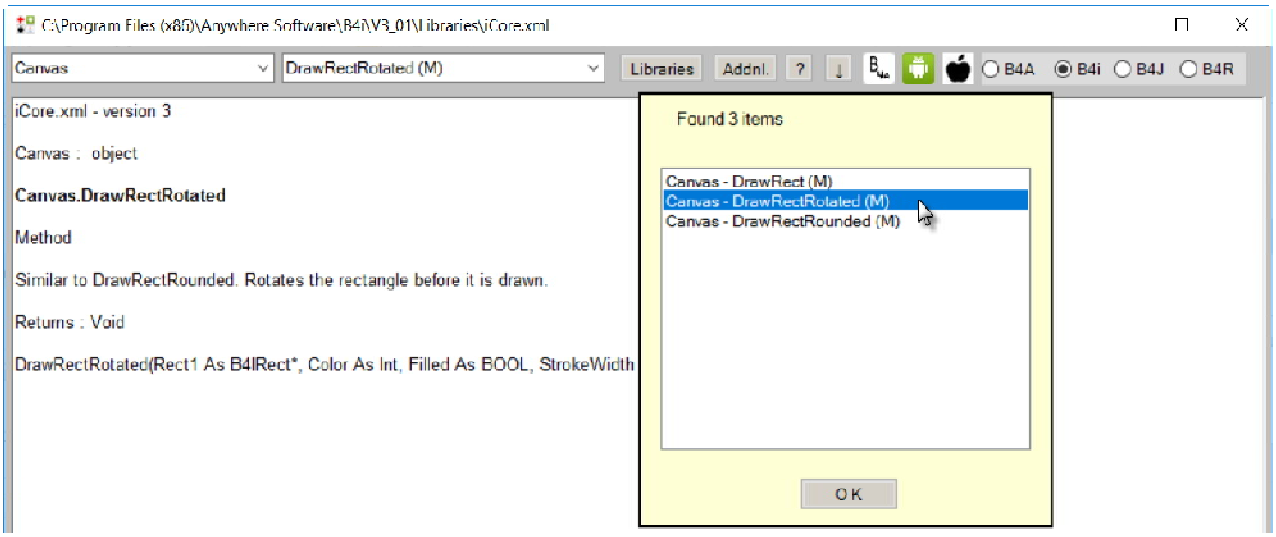


Example:
Selected library: iCore
Enter *DrawRect*



And the result.

We get the object Canvas and three methods.



Click on an item in the list to show its help.

Click on  to leave the search result list.

16.4 Asking a question in the forum

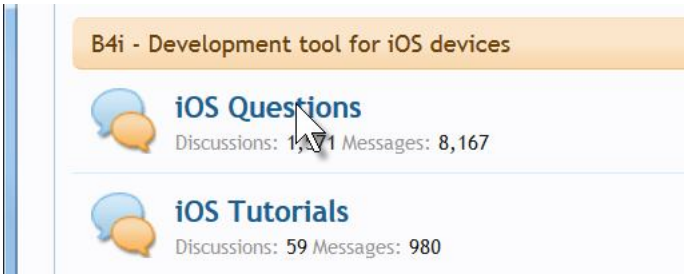
If you cannot find the answer with the previous tools you can ask the question in the forum.

Guidelines to ask a question:

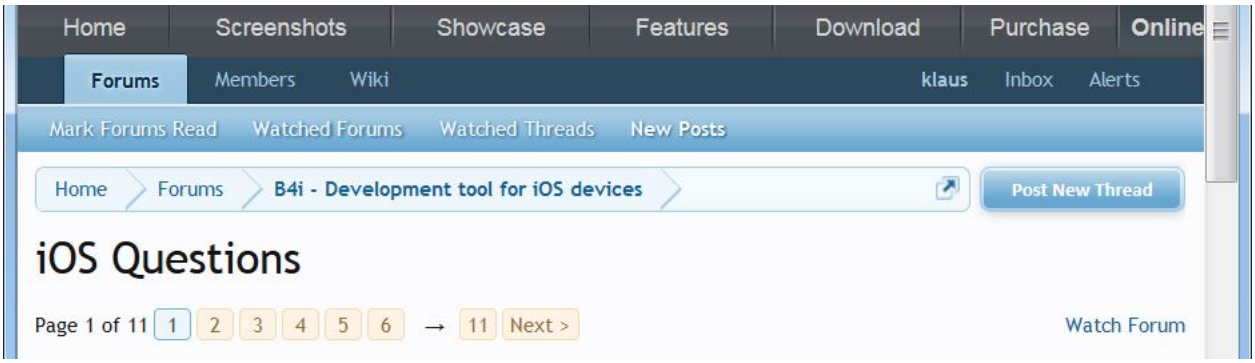
- Open a new thread for each problem with a meaningful title.
  This will make researches easier for other users.
- Explain in detail your problem.
  - What you want to do.
  - What you have done and how.
  - What is not working as expected?
- Post the error message in the Log if you get one.
- Post the relevant code or better a small project showing the problem.
- Click on Like in the answer, or post a confirmation when the problem is solved.

16.5 Creating a new thread

Example:

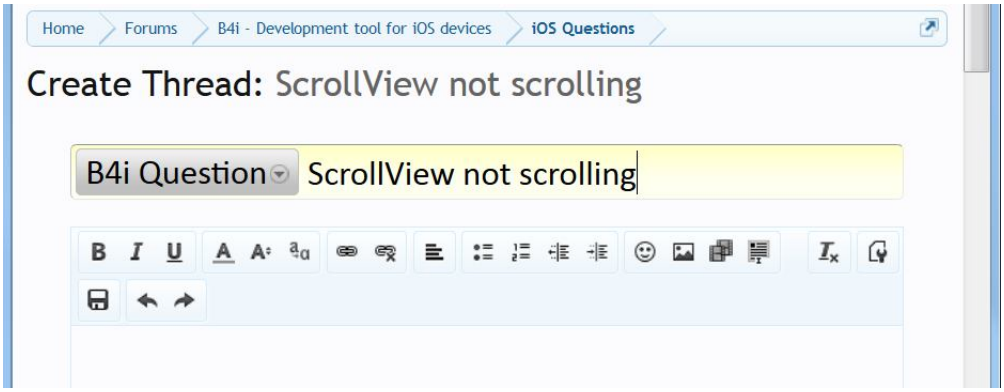


In the B4x Community page select the correct forum, depending on the product.



Then click on 

.



Enter a meaningful title.

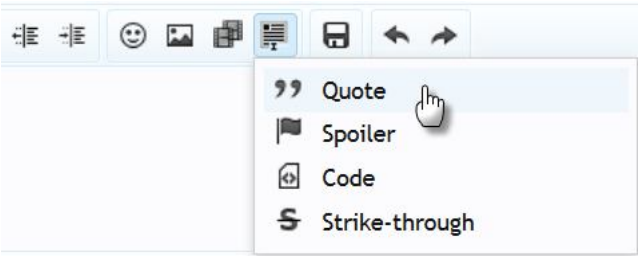
16.5.1 Editing a new thread or post

The editor:

Enter your text in the editor.

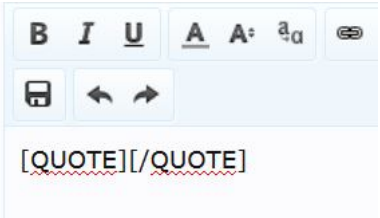
There are some useful features to edit the text:

16.5.2 QUOTE tags



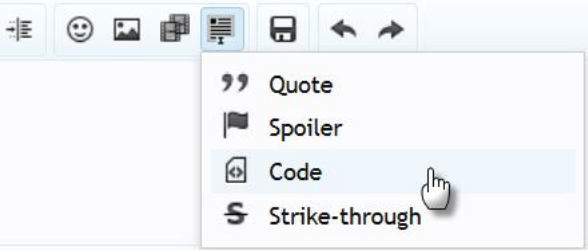
Click on the Insert button and select Quote.

You will see [QUOTE][/ QUOTE] in the editor between][, for example [QUOTE]Test text for quote.[/QUOTE]



Enter your text

16.5.3 CODE tags

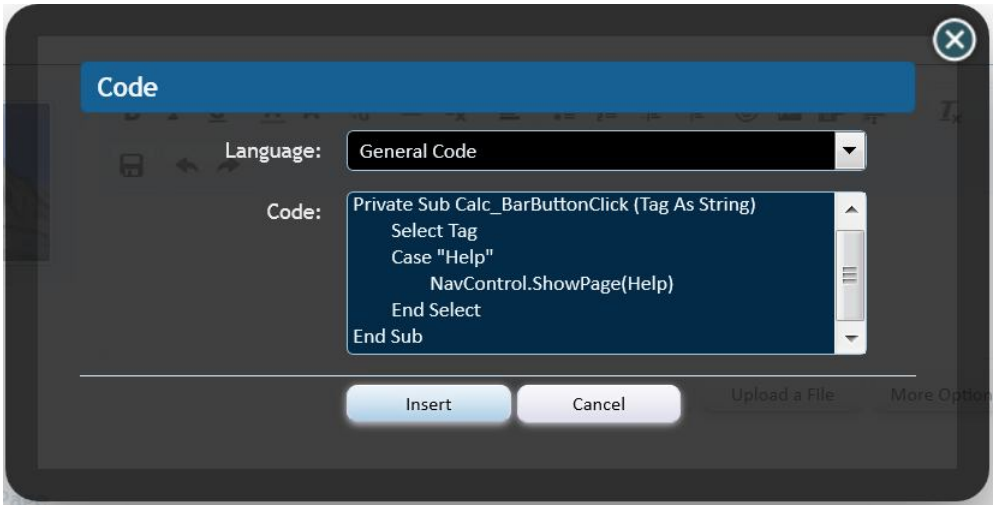


Click on the Insert button and select Code.

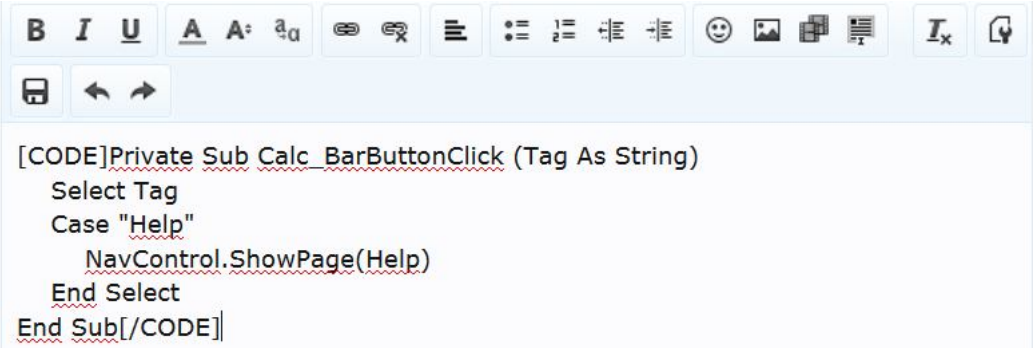
You will get this window.

Copy the code from the IDE, or enter it, and click on 

.

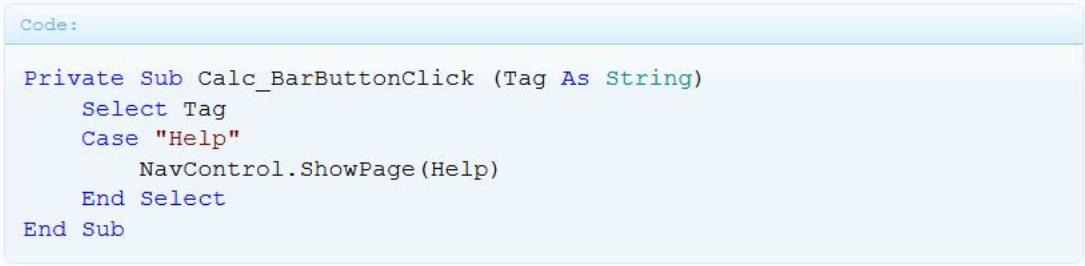


You will see this in the editor:



The code is added between the CODE tags [CODE][/CODE] and formatted.
You can also add the CODE tags and the code directly in the editor.

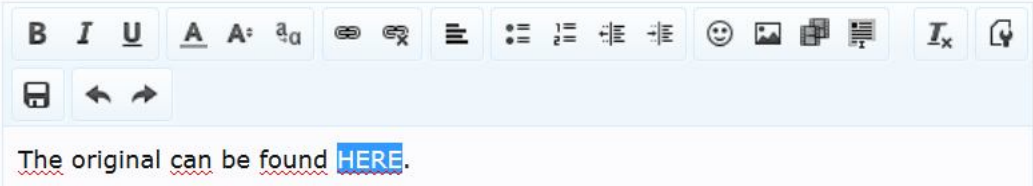
And the result in the forum.



16.5.4 Links

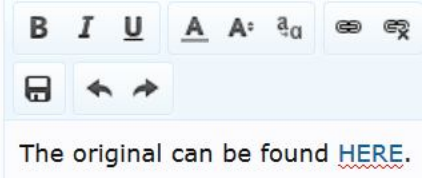
You can add links in the editor.

Select the text and click on .



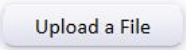
Enter or copy the link and click on 

.



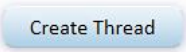
And the result

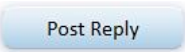
16.5.5 Add files.

To add files to the thread or post click on 

Select the files in the files explorer.

16.5.6 Send the thread or post

Click on 

to create the thread or on 

to send the post.

17 Code snippets

17.1 Picker change text color

```
Private Sub Application_Start (nav As NavigationController)
   NavControl = nav
   Page1.Initialize("Page1")
   NavControl.ShowPage(Page1)
   Page1.RootPanel.LoadLayout("1")
   Picker1.SetRowsHeight(35)
   Dim items As List
   items.Initialize
   For i = 1 To 100
     Dim s As AttributedString
     s.Initialize("Item #" & i, Font.CreateNew(30), Rnd(0x8fffffff, -1))
     items.Add(s)
   Next
   Picker1.SetItems(0, items)
End Sub
```

17.2 NavigationBar button change color

```
Sub SetNavigationBarTintColor(clr As Int)
   Dim no As NativeObject = NavControl
   no.GetField("navigationBar").SetField("tintColor", no.ColorToUIColor(clr))
End Sub
```

17.3 NavigationBar title style

```
Private Sub Application_Start (nav As NavigationController)
   SetNavigationBarTitleStyle(Colors.Red, Font.CreateNewBold(20))
   Dim nav As NavigationController
   nav.Initialize("nav")
   App.KeyController = nav
   NavControl = nav
   Page1.Initialize("Page1")
   NavControl.ShowPage(Page1)
   Page1.RootPanel.LoadLayout("1")
End Sub

Sub SetNavigationBarTitleStyle(Color As Int, Fnt As Font)
   Dim attributes As NativeObject
   attributes = CreateMap("NSFont": Fnt, "NSColor": attributes.ColorToUIColor(Color))
   Dim no As NativeObject
   no.Initialize("UINavigationBar").RunMethod("appearance", Null) _
     .RunMethod("setTitleTextAttributes:", Array(attributes.RunMethod("ToDictionary", N
ull)))
End Sub
```

17.4 ToolBarButton Replace text

```
Sub ReplaceBarButtonText(Tag As String, NewText As String)
   Dim buttons As List = Page1.ToolbarButtons
   For i = 0 To buttons.Size - 1
     Dim bb As BarButton = buttons.Get(i)
     If bb.Tag = Tag Then
       Dim newButton As BarButton
       newButton.InitializeText(NewText, Tag)
       buttons.Set(i, newButton)
       Exit
     End If
   Next
   Page1.ToolbarButtons = buttons
End Sub
```

17.5 Segmented Control add image

```
Private Sub Application_Start (Nav As NavigationController)
   NavControl = Nav
   Page1.Initialize("Page1")
   NavControl.ShowPage(Page1)
   Page1.RootPanel.LoadLayout("1")
   SetImageSegments(SegmentedControl1, Array(LoadBitmap(File.DirAssets, "smiley.png"),
_
     LoadBitmap(File.DirAssets, "icon-40.png")))
End Sub

Sub SetImageSegments(sc As SegmentedControl, Images As List)
   Dim no As NativeObject = sc
   no.RunMethod("removeAllSegments", Null)
   For i = 0 To Images.Size - 1
     no.RunMethod("insertSegmentWithImage:atIndex:animated:", Array(Images.Get(i), i, T
rue))
   Next
End Sub
```

17.6 Get language

```
Sub GetPreferredLanguage As String
    Dim no As NativeObject
    Return no.Initialize("NSLocale") _
        .RunMethod("preferredLanguages", Null).RunMethod("objectAtIndex:", Array(0)).As
String
End Sub
```

17.7 Keep alive

```
Dim App As Application
App.IdleTimerDisabled = True
```

17.8 Get device info

```
Dim device As NativeObject
device = device.Initialize("UIDevice").RunMethod("currentDevice", null)
Log(device.GetField("name").AsString)
Log(device.GetField("model").AsString)
Log(device.GetField("systemName").AsString)
```

17.9 Set full screen

Add this line the Application attributes:

```
#PlistExtra: <key>UIViewControllerBasedStatusBarAppearance</key><false/>
```

And add this code in Application_Start:

```
Dim no As NativeObject = app
no.RunMethod("setStatusBarHidden:animated:", Array(True, False))
```

17.10 Get parent view

```
Dim no As NativeObject = panel1
Dim parent As View = no.GetField("superview")
```

17.11 Set WebView ScrollBars

```
Dim wv As WebView
wv.Initialize("wv")
Dim no As NativeObject = wv
no.GetField("scrollView").SetField("showsHorizontalScrollIndicator", False)
no.GetField("scrollView").SetField("showsVerticalScrollIndicator", False)
```

17.12 Get battery level

```
Sub GetBatteryLevel As Float
   Dim no As NativeObject
   no = no.Initialize("UIDevice").RunMethod("currentDevice", Null)
   If no.GetField("batteryMonitoringEnabled").AsBoolean = False Then
     no.SetField("batteryMonitoringEnabled", True)
   End If
   Return no.GetField("batteryLevel").AsNumber
End Sub
```

17.13 Screen orientation

```
Sub DeviceOrientation As String
   Dim no As NativeObject
   Dim o As Int = no.Initialize("UIDevice").RunMethod("currentDevice", Null).RunMethod(
"orientation", Null).AsNumber
   Select o
     Case 0
       Return "Unknown"
     Case 1
       Return "Portrait"
     Case 2
       Return "PortraitUpsideDown"
     Case 3
       Return "LandscapeLeft"
     Case 4
       Return "LandscapeRight"
     Case 5
       Return "FaceUp"
     Case 6
       Return "FaceDown"
   End Select
   Return "Unknown"
End Sub
```

17.14 Set screen brightness

```
'value between 0 to 1
Sub SetScreenBrightness (value As Float)
   Dim no As NativeObject
   no.Initialize("UIScreen").RunMethod("mainScreen",Null).SetField("brightness", value)
End Sub
```

17.15 Add an Input Accessory View

Add an Input Accessory View, which allow to add toolbar, panel,... to the top of the keyboard.
https://developer.apple.com/library...TextAndWebiPhoneOS/InputViews/InputViews.html
In the code below, it's a button.
You can also add a Panel with more views.

```
Dim b As Button
b.Initialize("b", b.STYLE_SYSTEM)
b.Text = "Click me"
b.Width = 100
b.Height = 50
Dim no As NativeObject = TextField1
no.SetField("inputAccessoryView", b)
```