

























B4x Booklets

B4A B4i B4J B4R

B4x I D E

Integrated Development Environment

1	General	5
2	Menu and Toolbar	6
2.1	Toolbar	6
2.2	File menu	7
2.3	Edit menu	7
2.4	Project menu	8
2.4.1	Add a new module	9
2.5	Tools menu	10
2.5.1	IDE Options	11
2.5.1.1	Themes	12
2.5.1.2	Font Picker	13
2.5.1.2.1	Word wrap	13
2.5.1.2.2	Configure Process Timeout	13
2.5.1.2.3	Disable Implicit Auto Completion	13
2.5.2	Take Screenshot B4A only	14
2.5.3	Create Video B4A only	15
2.5.4	Clean Files Folder (unused files)	16
2.5.5	Clean Project	16
2.6	Right click menu	17
2.7	Compiler mode	18
2.7.1	B4A and B4J	19
2.7.1.1	Release and Release (obfuscated) modes B4A and B4J	19
2.7.2	B4i	20
2.7.3	B4R	20
3	Code area	21
3.1	Split the code area	21
3.2	Code header Project Attributes / Activity Attributes	22
3.2.1	B4A	22
3.2.1.1	Project Attributes	22
3.2.1.2	Activity Attributes	22
3.2.1.3	Service Attributes	22
3.2.2	B4i	24
3.2.3	B4J	24
3.2.4	B4R	24
3.3	Undo – Redo  	25
3.4	Collapse a subroutine	25
3.5	Collapse a Region	26
3.6	Collapse the entire code	27
3.7	Toggle Outlining Ctrl + 0	28
3.8	Copy a selected bloc of text	29
3.9	Move line(s) up / down Alt + Up / Alt + Down	29
3.10	Find / Replace	30
3.11	Commenting and uncommenting code  	31
3.12	Bookmarks 	32
3.13	Indentation  	33
3.14	Auto format	35
3.15	Documentation tool tips while hovering over code elements	36
3.16	Auto Completion	37
3.17	Built in documentation	41
3.17.1	Copy code examples	42
3.18	Jump to a subroutine	43
3.19	Highlighting occurrences of words	44

3.20	Breakpoints	45
3.21	Color Picker  Color Picker	47
3.22	Icon Picker  Icon Picker	48
3.23	Colors in the left side	49
3.24	URLs in comments and strings are ctrl-clickable	50
4	Tabs	51
4.1	Floating Tab windows	52
4.2	Float 	53
4.3	Auto Hide 	56
4.4	Close	58
4.5	Modules and subroutine lists  Modules	59
4.5.1	Find Sub / Module / Line number (Ctrl + E)	60
4.6	Files Manager  Files Manager B4A, B4i and B4J only	61
4.7	Logs 	63
4.7.1	Compile Warnings	64
4.7.1.1	Ignoring warnings	65
4.7.1.2	List of warnings	66
4.8	Libraries Manager  Libraries Manager	72
4.9	Quick Search  Quick Search	73
4.10	Find All References (F7)  Find All References (F7)	75
5	Navigation in the IDE	76
5.1	Alt + Left / Alt + Right Move backwards and forwards	76
5.2	Alt + N Navigation stack menu 	76
5.3	Split the screen	76
5.4	Multiple windows	77
5.5	Ctrl + E Search for sub or module	77
5.6	Ctrl + Click on any sub or variable	77
5.7	F7 - Find all references	77
5.8	Ctrl + F Quick Search	77
6	Debugging B4A, B4i, B4J	78
6.1	B4A, B4i, B4J	78
6.1.1	Debug mode	79
6.1.1.1	Debug Toolbar	79
6.1.1.1.1	Run  F5	79
6.1.1.1.2	Step In  F8	80
6.1.1.1.3	Step Over  F9	81
6.1.1.1.4	Step Out  F10	81
6.1.1.1.5	Stop 	82
6.1.1.1.6	Restart  F11	82
6.1.2	Debug window	83
6.1.2.1	The status button	83
6.1.2.2	The breakpoint window	83
6.1.2.3	The Watch window	84
6.1.2.4	The object window	85
6.1.2.5	Breakpoints	86
6.1.2.6	With Logs	88
6.1.2.7	Modifying code in the Debugger	89
6.1.3	Debug (legacy) mode B4A only	90
6.2	Debugging B4R	91

6.2.1	Debug example with the TrafficLight project.....	91
-------	--	----

Main contributors: Klaus Christl (klaus), Erel Uziel (Erel)

To search for a given word or sentence use the Search function in the Edit menu.

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:

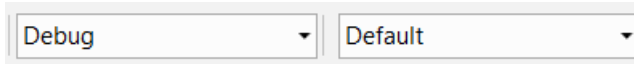
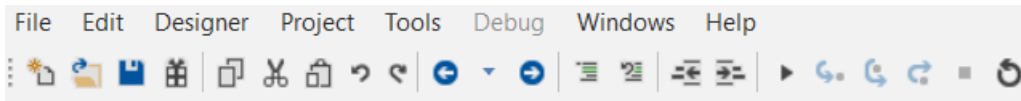
B4A version 7.00

B4i version 4.01


















B4J version 5.51






B4R version 1.80

2 Menu and Toolbar



2.1 Toolbar

-  Generates a new empty project [Ctrl + N].
-  Loads a project.
-  Saves the current project [Ctrl + S].
-  Export As Zip.
-  Copies the selected text to the clipboard [Ctrl + C].
-  Cuts the selected text and copies it to the clipboard [Ctrl + X].
-  Pastes the text in the clipboard at the cursor position [Ctrl + V].
-  Undoes the last operation [Ctrl + Z].
-  Redoes the previous operation [Ctrl + Shift + Z].
-  Navigate backwards [Alt + Left].
-  Navigation history [Alt + N].
-  Navigate forwards [Alt + Right].
-  [Block Comment \[Ctrl + Q\]](#).
-  [Block Uncomment \[Ctrl + W\]](#).
-  [Decrease the indentation of the selected lines.](#)
-  [Increase the indentation of the selected lines.](#)
-  Runs the compiler [F5].

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

These 5 functions are active only when the debugger is active.

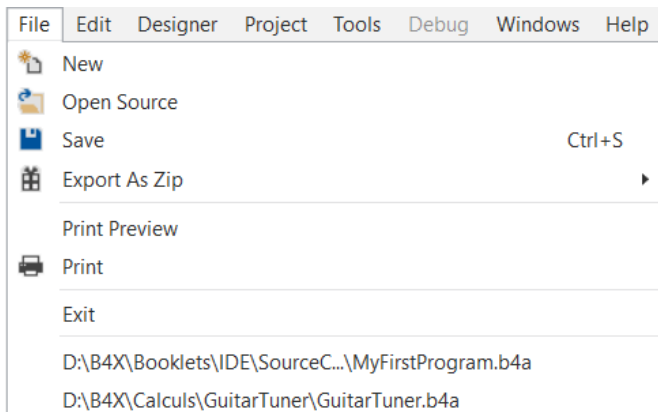


[Compiler options](#) list and [Debugging](#).



Conditional compiling options.

2.2 File menu



New Generates a new empty project.

Open Source Loads a project.

Save Saves the current project.

Export As Zip Exports the whole project in a zip file.

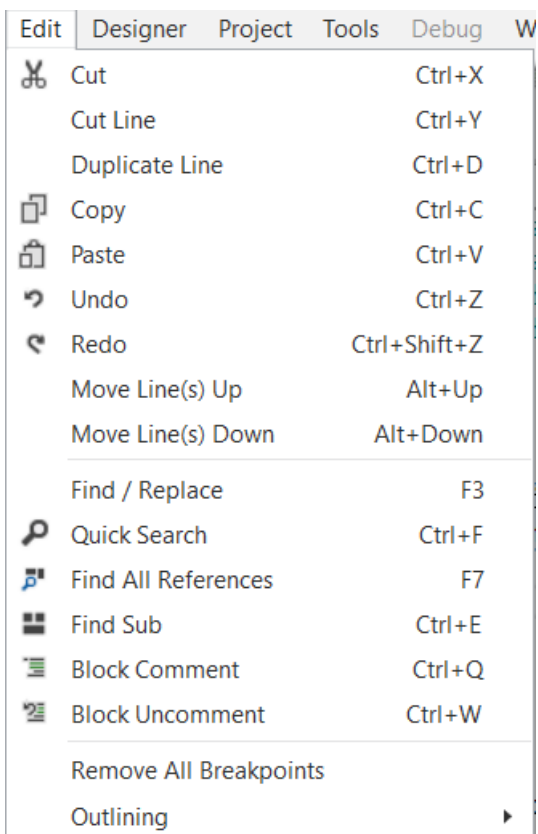
Print Preview Preview of the print.

Print Prints the whole code of the selected Module.

Exit Leaves the IDE.

List of last loaded programs.

2.3 Edit menu



Cut Cuts the selected text and copies it to the clipboard.

Cut Line Cuts the line at the cursor position.

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 upwards.

Move Line(s) Down Moves the selected lines downwards.

Find / Replace Activates the [Find and Replace](#) function.

Quick Search [Quick Search](#)

Find All References [Find All References](#)

Find Sub [Find Sub](#)

Block Comment

Block Uncomment

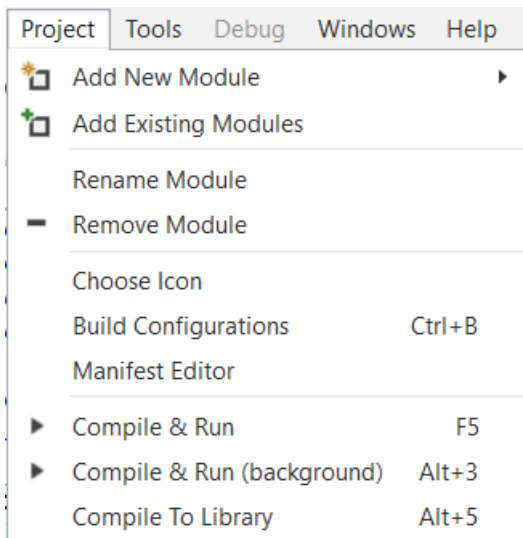
[Comment / Uncomment the selected lines.](#)

Remove All Breakpoints [Breakpoints.](#)

Outlining [Collapse the whole code.](#)

2.4 Project menu

B4A



Adds a new [module](#)

Adds an existing [module](#)

Changes the [module](#) name

Removes the current [module](#)

Chooses an icon for the program.

Changes the package name.

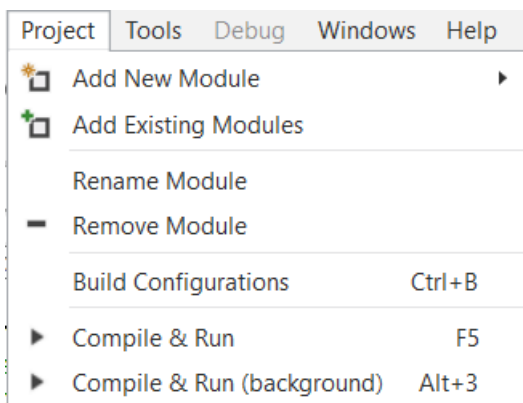
Runs the Manifest Editor.

Compile and run the project.

Compile and run the project in the background.

Compile to a library.

B4i, B4R



Adds a new [module](#)

Adds an existing [module](#)

Changes the [module](#) name

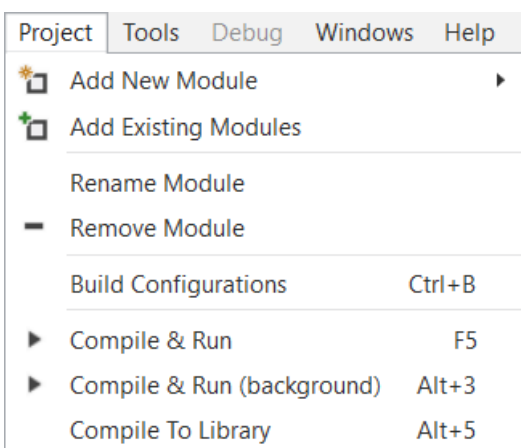
Removes the current [module](#)

Changes the package name.

Compile and run the project.

Compile and run the project in the background.

B4J



Adds a new [module](#)

Adds an existing [module](#)

Changes the [module](#) name

Removes the current [module](#)

Changes the package name.

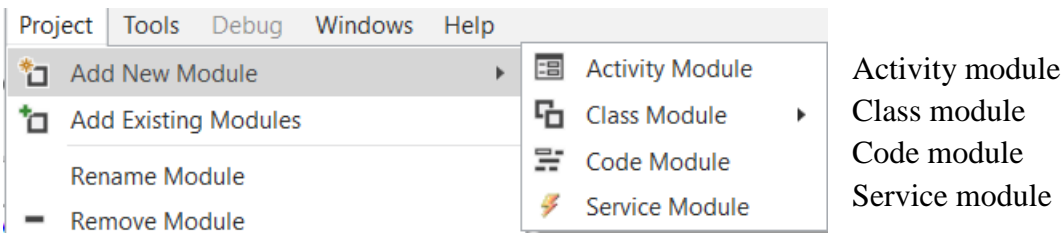
Compile and run the project.

Compile and run the project in the background.

Compile to a library.

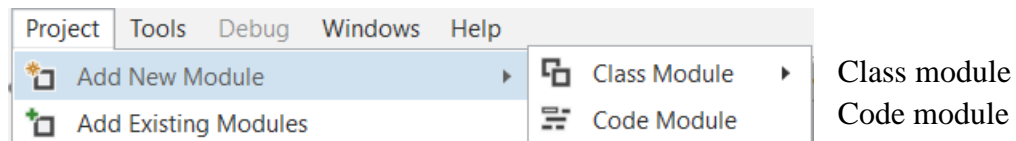
2.4.1 Add a new module

B4A



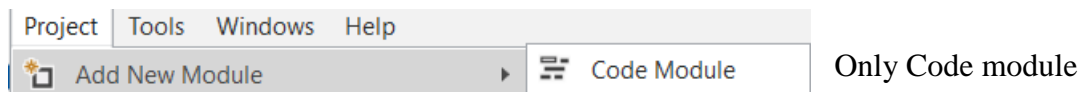
Activity module
Class module
Code module
Service module

B4i, B4J



Class module
Code module

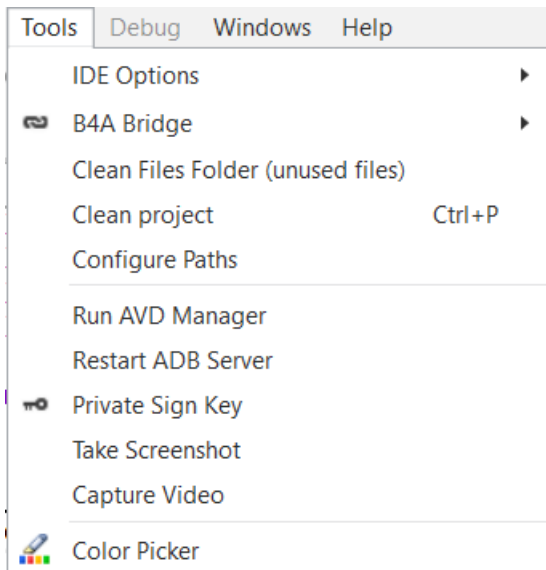
B4R



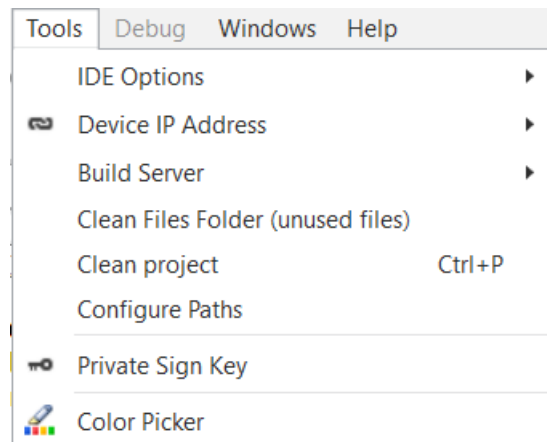
Only Code module

2.5 Tools menu

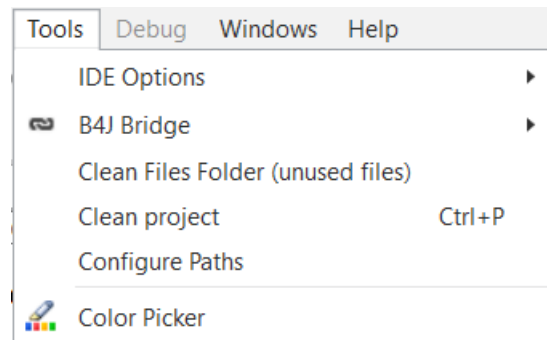
B4A



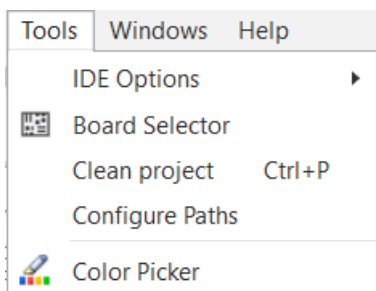
B4i



B4J



B4R



IDE Options see below

[B4A Bridge](#), connection with Bluetooth or Wifi B4A

[Clean Files Folder](#) (unused files) B4A, B4i, B4J

[Clean Project](#) All

[Configure Paths](#) All

[Run AVD Manager](#) B4A

[Take Screenshot](#) B4A

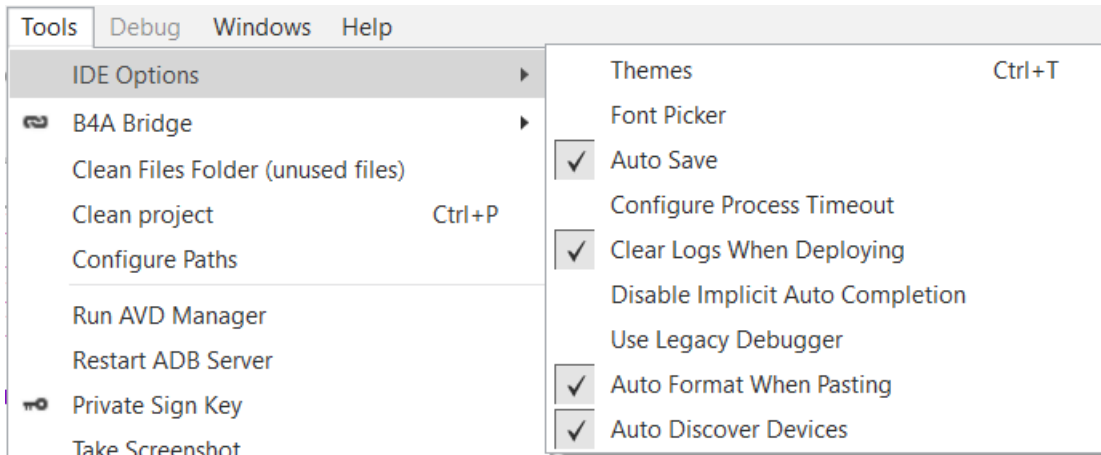
[Capture a video](#) B4A

Show the [Color Picker](#) All

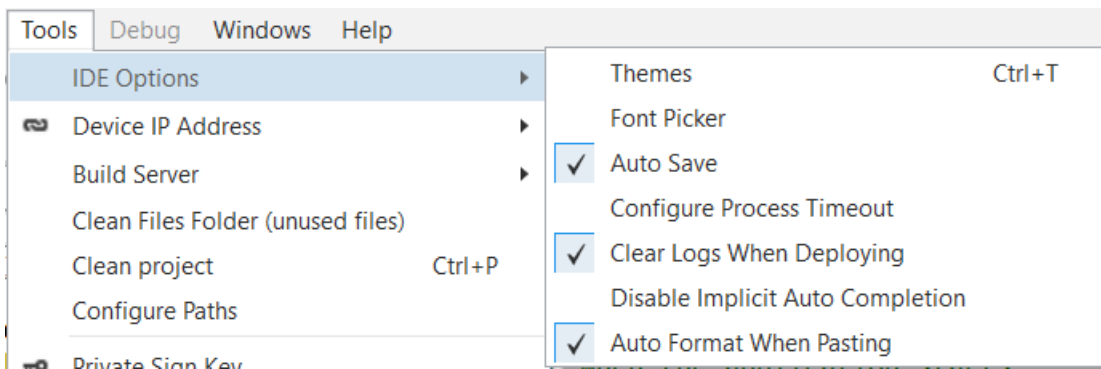
Board Selector B4R

2.5.1 IDE Options

B4A



B4i, B4J, B4R



All

[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.](#)

B4A only

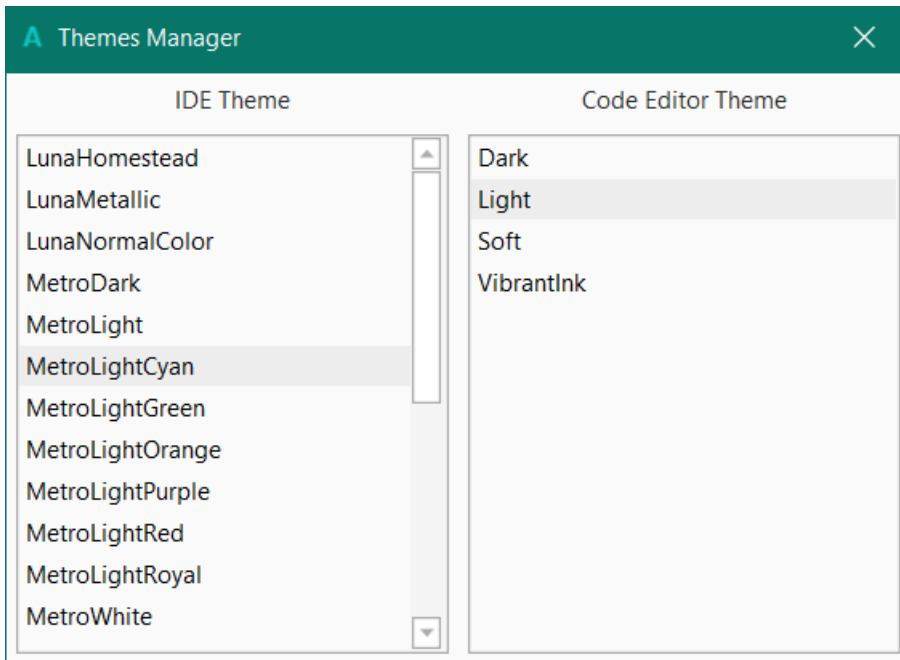
[Use Legacy Debugger](#)

Use the legacy Debugger instead of the rapid Debugger.

Auto Discover Devices

Detects automatically the connected devices.

2.5.1.1 Themes

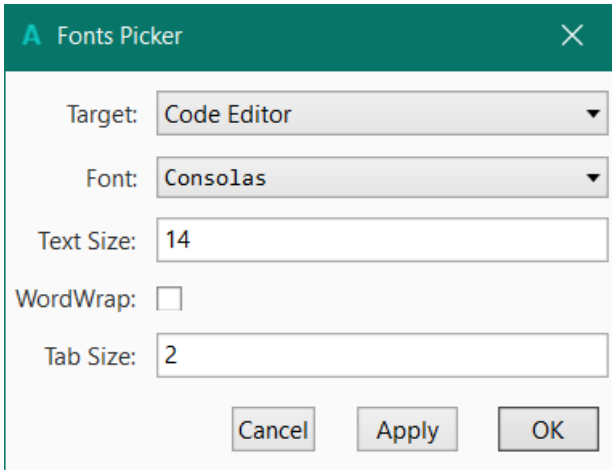


You can select different themes for the IDE.

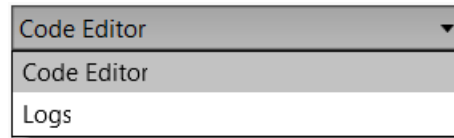
The default theme is MetroLight.

When you select one you see directly the new colors.

2.5.1.2 Font Picker



You can select the target Code Editor or Logs.



Different fonts.
Enter the text size.
Select WordWrap
Enter the Tab size.

2.5.1.2.1 Word wrap

```
53 | lblComments.Text = "Enter the result" & CRLF & "and click" | |
54 |
```

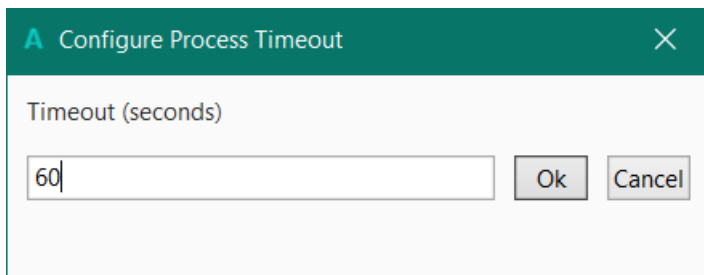
Without word wrap. The end of the line is hidden.

```
53 | lblComments.Text = "Enter the result" & CRLF & "and click" | |
    | on OK"
```

With word wrap.

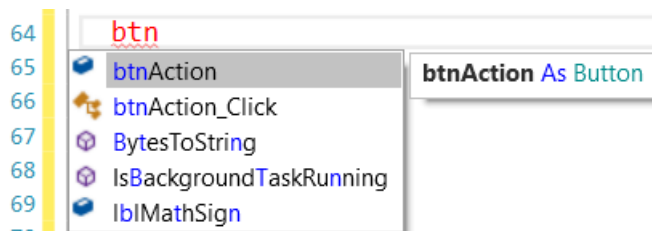
The end of the line is wrapped to the next line.

2.5.1.2.2 Configure Process Timeout



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

2.5.1.2.3 Disable Implicit Auto Completion



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

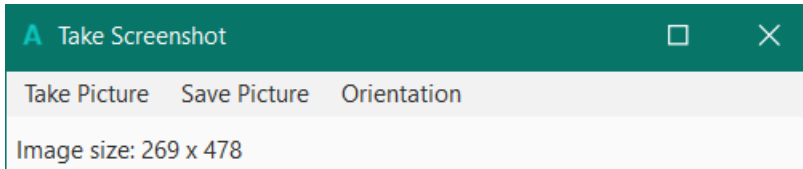
If checked Disable Implicit Auto Completion you won't see the auto completion list.

2.5.2 Take Screenshot B4A only

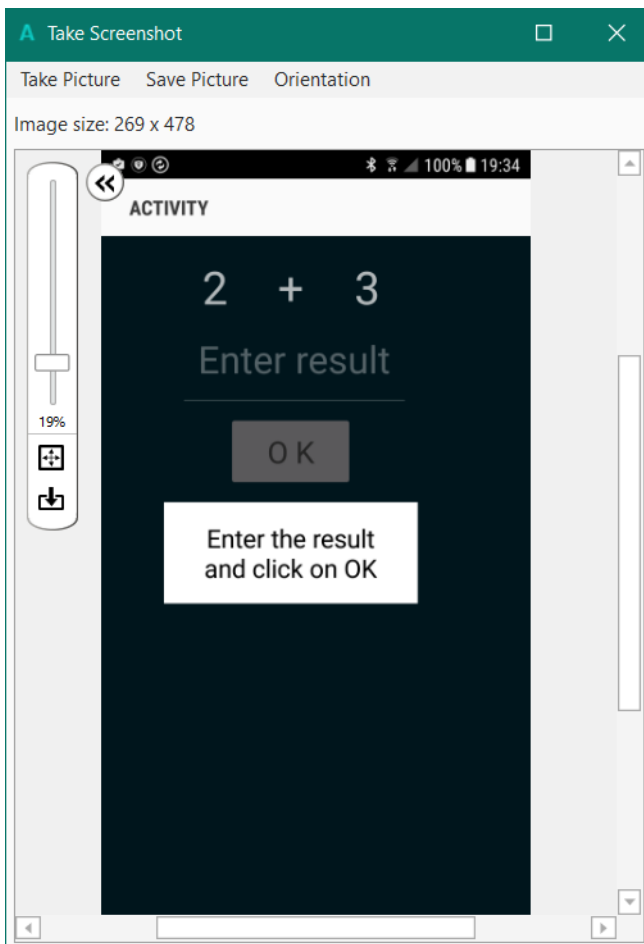
The **Take Screenshot** function can be called from the:

- Tools menu when the IDE is in edit mode
- Debug menu when the IDE is in debug mode

Note: This function works only with USB connection not with B4A-Bridge !



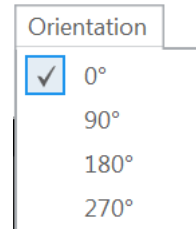
Clicking on **Take Screenshot** shows this window.



Click on **Take Picture** to take the screenshot picture from the device.

You can resize the image with the cursor on the left side.

You can save the image with **Save Picture** as a PNG file.



And you can change the orientation of the picture.

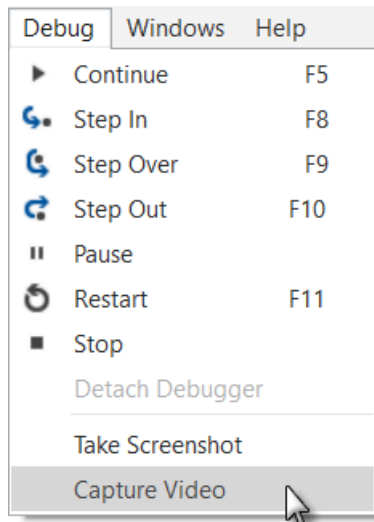
Copy To Clipboard

Right click on the image to copy the image to the clipboard.

2.5.3 Create Video **B4A only**

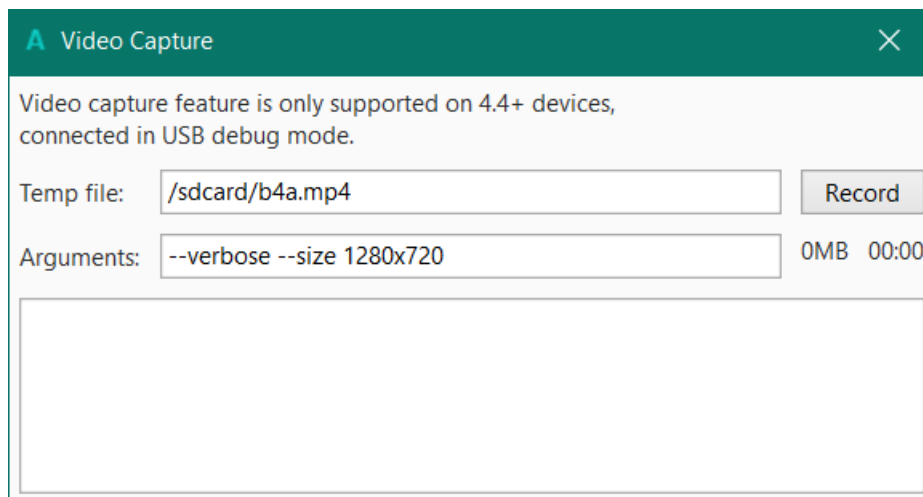
You can run your program and record a video when you use it.

Note: This function works only with USB connection not with B4A-Bridge !



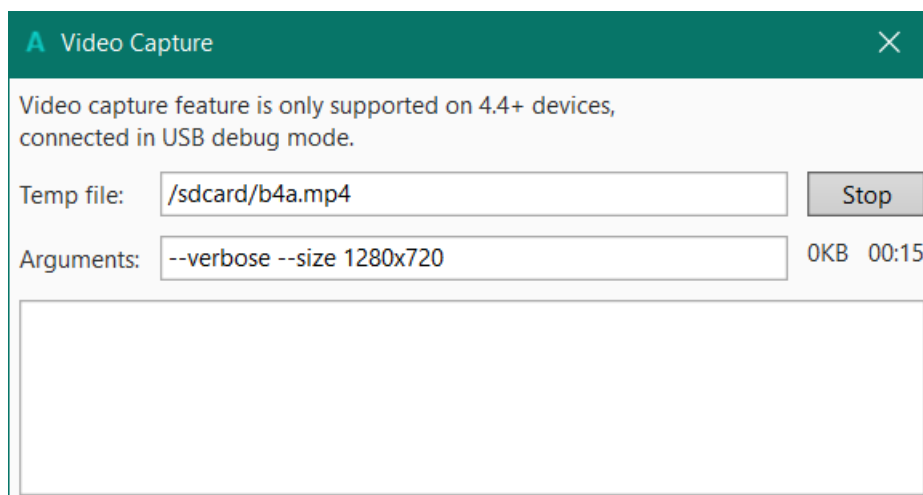
In the **Debug** menu click on **Capture Video**.

The screen below will be displayed:



Click on **Record** to begin recording.

A screen similar to this one will be displayed:



Click on **Stop** to stop recording.

You will be asked where you want to save the file on the computer.

2.5.4 Clean Files Folder (unused files)

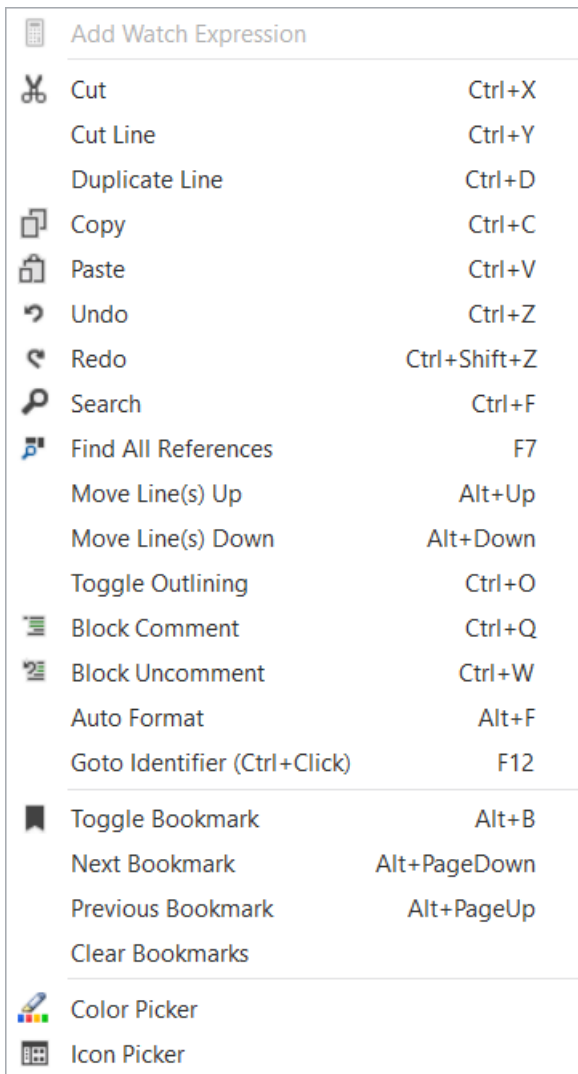
Deletes files that are located under the Files folder but are not used by the project (it will not delete any file referenced by any of the project layouts). A list of unused files will be displayed before deletion (and you may cancel the operation).

2.5.5 Clean Project

Deletes all files that are generated during compilation.

2.6 Right click menu

When you right click in the code area the menu below is displayed.



Cut

Cut Line

Duplicate Line

Copy

Paste

[Undo](#)

[Redo](#)

[Search](#)

[Find All References](#)

[Move Line\(s\) Up](#)

[Move Line\(s\) Down](#)

Toggle Outlining

[Block Comment](#)

[Block Uncomment](#)

[Auto Format](#)

[Goto identifier](#)

[Toggle Bookmark](#)

[Previous Bookmark](#)

[Next Bookmark](#)

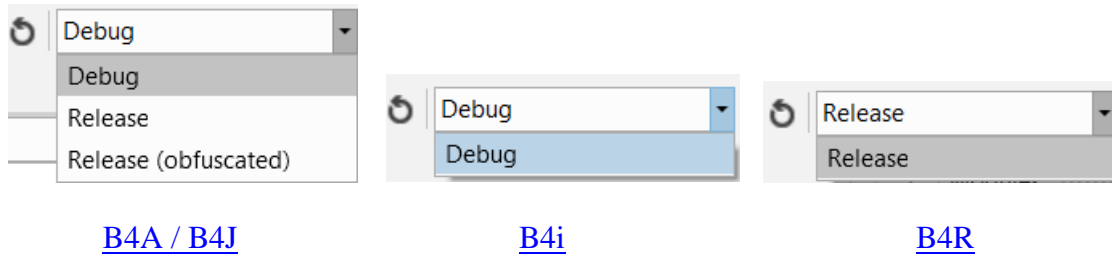
[Clear Bookmark](#)

[Color Picker](#)

[Icon Picker](#) Not in B4R.

2.7 Compiler mode

Besides the toolbar there is a drop down list to select the compiler mode.



Debugging is explained in detail in the [Debugging](#) chapter.

2.7.1 B4A and B4J

Compiling modes:

- [Debug](#)
- [Release](#)
- [Release \(obfuscated\)](#)

2.7.1.1 Release and Release (obfuscated) modes B4A and B4J

To distribute your project you must compile it with:

- Release
The debugger code will not be added to the apk file.
- Release (obfuscated)
The debugger code will not be added to the apk file, but the program file will be modified. See below.

During compilation B4A generates Java code which is then compiled with the Java compiler and converted to Dalvik (Android byte code format).

There are tools that allow decompilation of Dalvik byte code into Java code.

The purpose of obfuscation is to make the decompiled code less readable, harder to understand and make it more difficult to extract strings like developer account keys.

It is important to understand how the obfuscator works.

The obfuscator does two things:

Strings obfuscation

Any string written in Process_Globals sub (and only in this sub) will be obfuscated, making it much harder to extract important keys. The strings are deobfuscated at runtime.

Note that several keys are used during obfuscation including the package name, version name and version code. Modifying these values with the manifest editor will break the deobfuscation process.

Variables renaming

The names of global variables and subs are converted to meaningless strings. Local variables are not affected as their names are lost anyway during the compilation.

The following identifiers are **not** renamed:

- Identifiers that contain an underscore (required for the events handlers).
- Subs that appear in CallSub statements. When a sub name appears as a static string, the identifier be kept as it is.
- Designer views names.

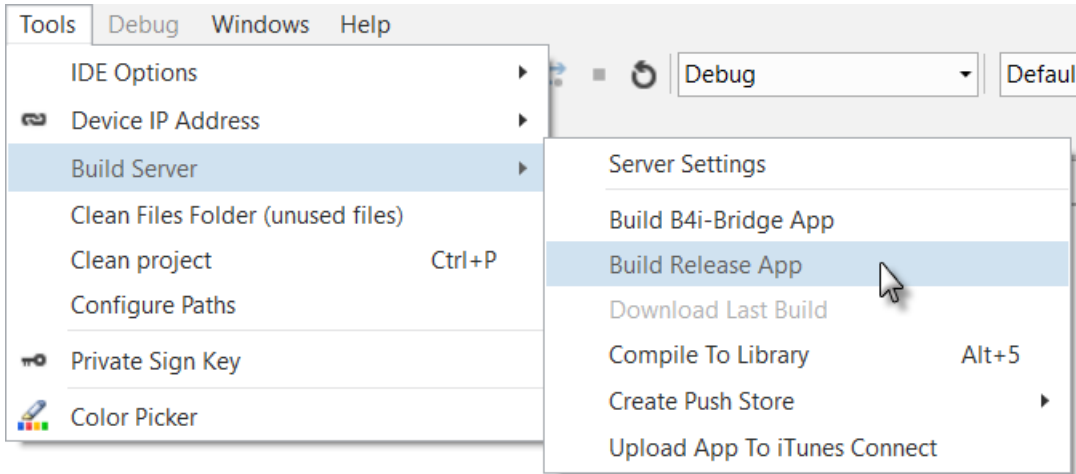
Tip: If, for some reason, you wish to prevent obfuscation of an identifier, include an underscore character in the name.

A file named ObfuscatorMap.txt will be created under the Objects folder. This file maps the original identifiers names to the obfuscated names. This mapping can be helpful in analysing crash reports.

2.7.2 B4i

To distribute a project you must compile it in Release mode.

Click on **Build Release App** in the Tools / Build Server menu.



2.7.3 B4R

Only Release mode.

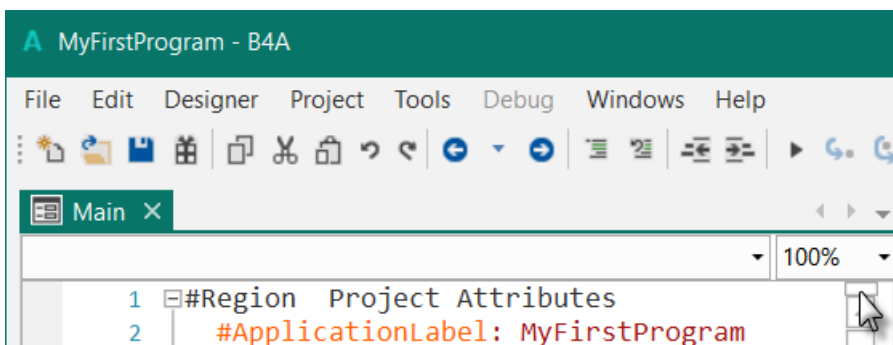
3 Code area

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.

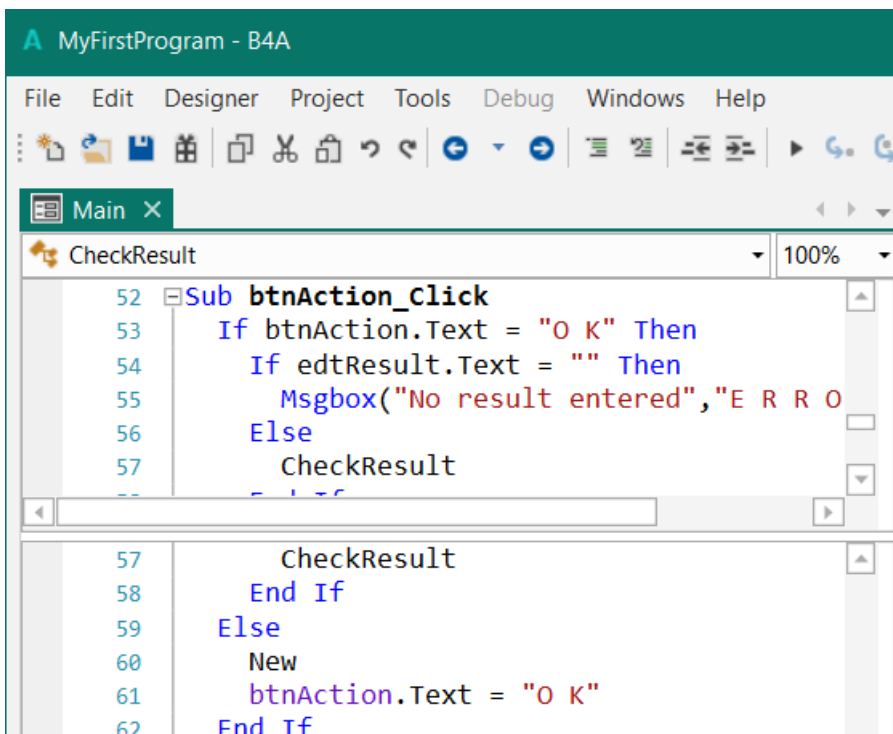
3.1 Split the code area

It is possible to split the code area into two parts allowing to edit two different code parts at the same time.

Move the small rectangle below the zoom level.



And the result.



3.2 Code header Project Attributes / Activity Attributes

A code header, with general settings, is added at the beginning of the code.

3.2.1 B4A

3.2.1.1 Project Attributes

Attributes that are valid for the whole project. Displayed only in the Main module.

```
#Region Project Attributes
#ApplicationLabel: SecondProgram
#VersionCode: 1
#VersionName:
'SupportedOrientations possible values: unspecified, landscape or portrait.
#SupportedOrientations: unspecified
#CanInstallToExternalStorage: False
#End Region
```

#ApplicationLabel: The name which will be displayed below the program icon on the device.

#VersionCode: The version of the code, it is not displayed.

#VersionName: You can add a name for the version.

#SupportedOrientations: You can limit the whole program to a given orientation.

#CanInstallToExternalStorage: If you want to install the program on an external storage card you must set this attribute to True.

You can add or change the values to your needs.

3.2.1.2 Activity Attributes

Valid for the current activity.

```
#Region Activity Attributes
#FullScreen: False
#IncludeTitle: True
#End Region
```

When you add a new Activity you'll find the Activity Attributes region on top.

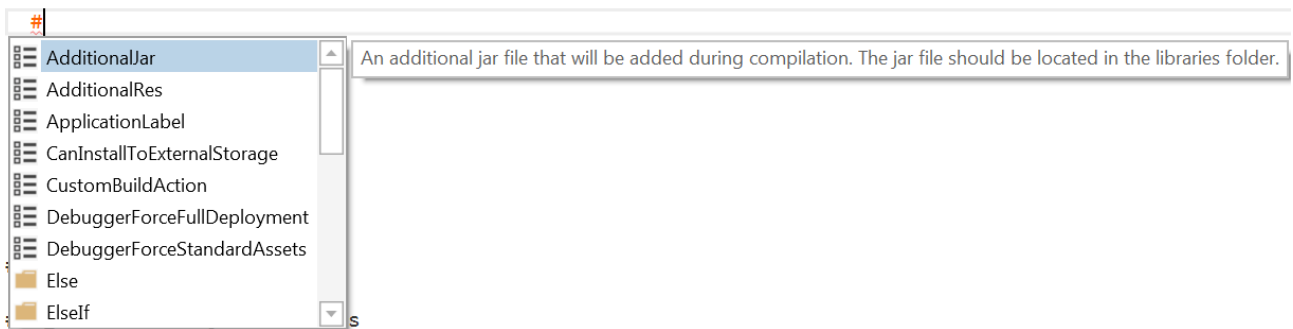
```
#Region Activity Attributes
#FullScreen: False
#IncludeTitle: True
#End Region
```

3.2.1.3 Service Attributes

When you add a new Service you'll find the Service Attributes header.


```
#Region Service Attributes
#StartAtBoot: False
#End Region
```

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



Note the two different icons:

 Attributes.

 Conditional compilation and region keywords.

When you load a project saved with a version of B4A older than 2.5 then the header will look like this:

```
#Region Module Attributes
  #FullScreen: False
  #IncludeTitle: True
  #ApplicationLabel: MyFirstProgram
  #VersionCode: 1
  #VersionName:
  #SupportedOrientations: unspecified
  #CanInstallToExternalStorage: False
#End Region
```

3.2.2 B4i

Only the Attributes below. No other Attributes in modules.

```
'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
  #Target: iPhone, iPad
  #ATSEnabled: True
  #MinVersion: 7
#End Region
```

3.2.3 B4J

Only the two Attributes below. No other Attributes in modules.

```
#Region Project Attributes
  #MainFormWidth: 600
  #MainFormHeight: 600
#End Region
```



3.2.4 B4R

Only the Attributes below. No other Attributes in modules.

```
#Region Project Attributes
  #AutoFlushLogs: True
  #CheckArrayBounds: True
  #StackBufferSize: 300
#End Region
```


3.3 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.

3.4 Collapse a subroutine

A subroutine can be collapsed to minimize the number of lines displayed.

```

Sub btnAction_Click
If btnAction.Text = "O K" Then
  If edtResult.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

```

The btnAction_Click routine expanded.

Click on  to collapse the subroutine.

```

Sub btnAction_Click

```

The btnAction_Click routine collapsed.

```

Sub btnAction_Click
Sub btnAction_Click
  If btnAction.Text = "O K" Then
  If edtResult.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

```

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

3.5 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 click on  to collapse the whole region.

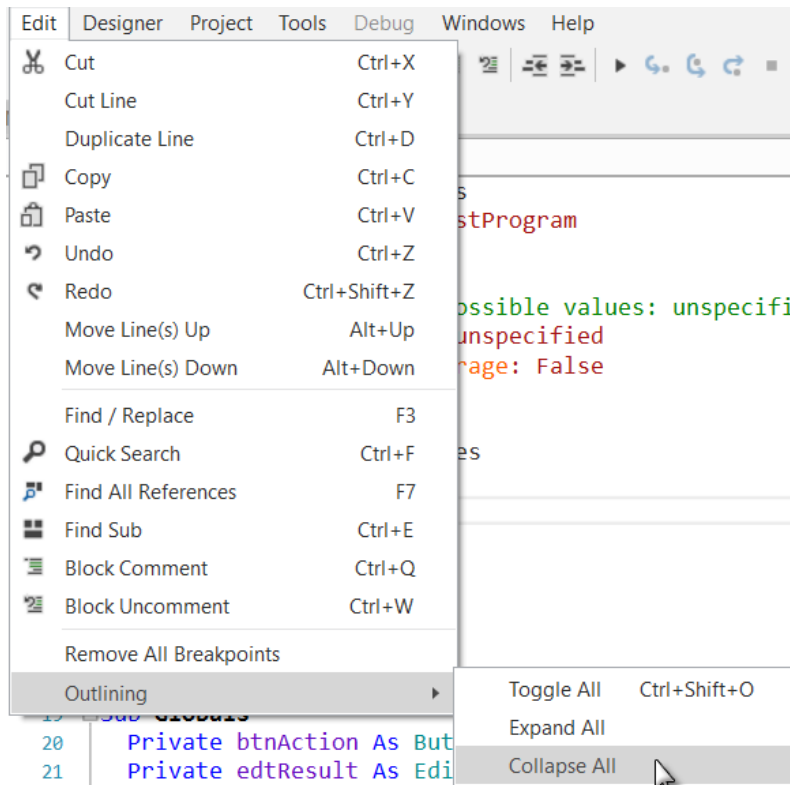
```
#Region GPS
```

Hovering over GPS

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

shows the code. For big regions not all the code is displayed.

3.6 Collapse the entire code



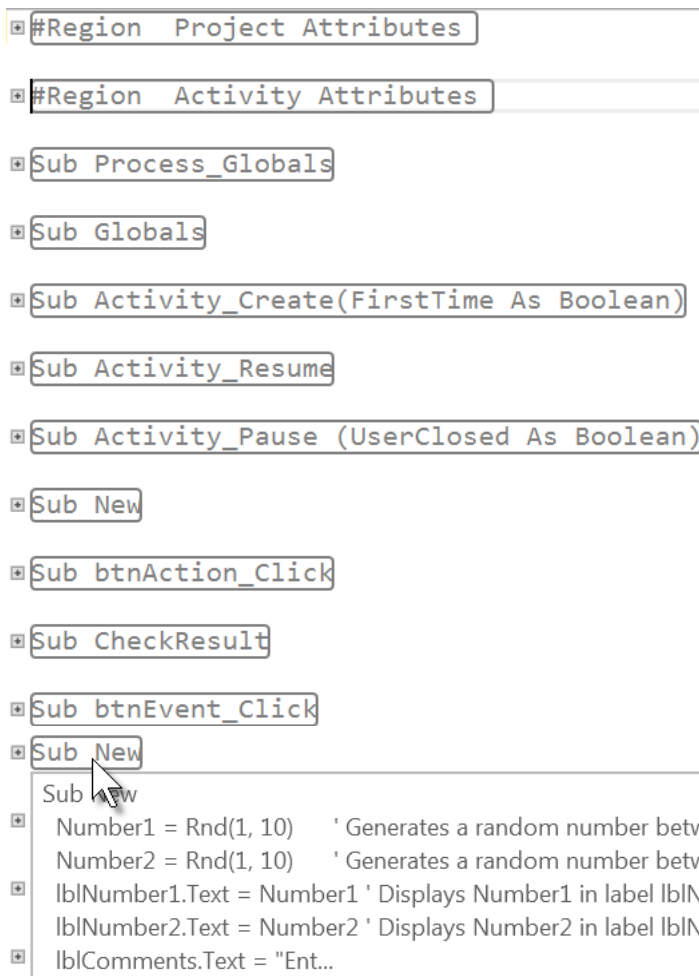
In the Edit / Outlining menu there are three functions:

- Toggle All
Expands the collapsed routines and collapses the expanded routines and regions.

- Expand All
Expands the entire code

- Collapse All
Collapses the entire code.

Click on **Collapse All**.



The whole code collapsed.

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

3.7 Toggle Outlining Ctrl + O

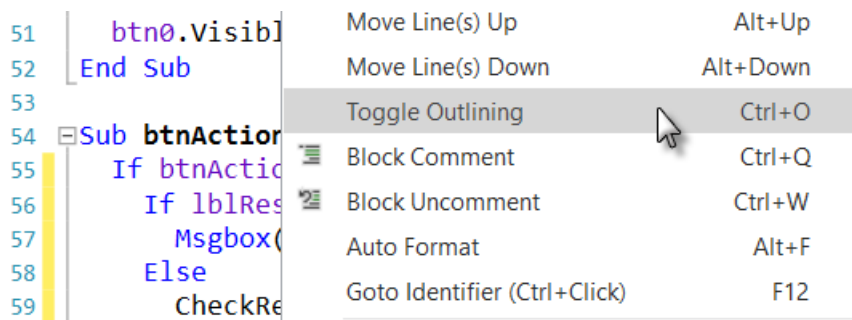
You can toggle code outlining.

Example:

```
Sub btnAction_Click
  If btnAction.Text = "O K" Then
    If lblResult.Text="" Then
      MsgBox("No result entered","E R R O R")
    Else
      CheckResult
    End If
  Else
    New
    btnAction.Text = "O K"
    lblResult.Text = "" & Chr(0xE632)
  End If
End Sub
```


Click inside the routine and press Ctrl + O.

Or right click inside the routine to show the pop-up menu and click on **Toggle Outlining** to collapse the routine.



And the result.

```
53
54 Sub btnAction_Click
67
```

It is the same as clicking on .

```
53
54 Sub btnAction_Click
55 If btnAction.Text = "O K" Then
56 If lblResult.Text="" Then
```

3.8 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.

```
19 Sub Globals
20 Private btnAction As Button
21 Private edtResult As EditText
22 Private lblComments As Label
23 Private lblMathSign As Label
24 Private lblNumber1 As Label
25 Private lblNumber2 As Label
```

3.9 Move line(s) up / down Alt + Up / Alt + Down

You can move selected lines up or down.

Either with Alt + Up or Alt + Down.

Or right click on the selected lines and select **Move Line(s) Up** or **Move Line(s) Down**.

3.10 Find / Replace

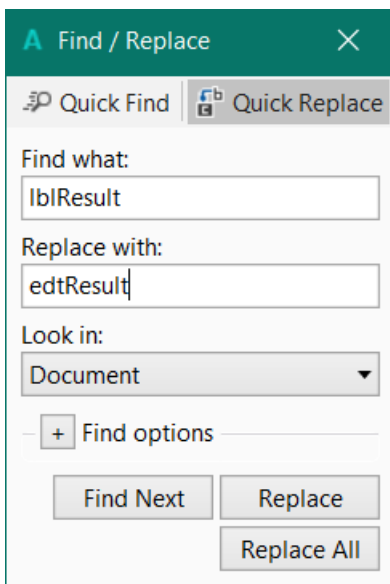
The example uses the code from the SecondProgram project.

Let's replace `lblResult` by `edtResult`.

```
19 Sub Globals
20   Dim btnAction, btn0 As Button
21   Dim lblResult As Label
22   Dim lblComments As Label
```

In the code select `lblResult`.

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

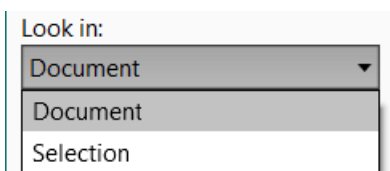


This window will be displayed

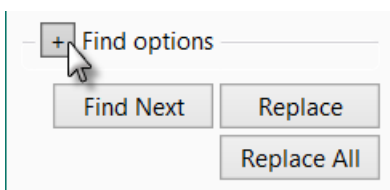
Enter `edtResult` in the 'Replace with' field.

Now, you can either:

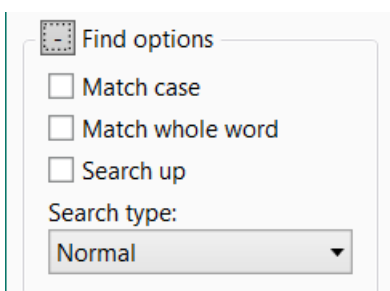
- **Find Next** find the next occurrence.
- **Replace** replace the current occurrence and find the next one.
- **Replace All** replace all occurrences.



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



You can select Find options, click on **+**.



These options are self-explanatory.

3.11 Commenting and uncommenting code

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

```
20 Private btnAction, btn0 As Button
21 Private lblResult As Label
22 Private lblComments As Label
23 Private lblMathSign As Label
24 Private lblNumber1 As Label
25 Private lblNumber2 As Label
26 Private Number1, Number2 As Int
```

Original code


```
20 Private btnAction, btn0 As Button
21 Private lblResult As Label
22 Private lblComments As Label
23 Private lblMathSign As Label
24 Private lblNumber1 As Label
25 Private lblNumber2 As Label
26 Private Number1, Number2 As Int
```

Select the code.

Click on  or Ctrl + Q.

```
20 ' Private btnAction, btn0 As Button
21 ' Private lblResult As Label
22 ' Private lblComments As Label
23 ' Private lblMathSign As Label
24 ' Private lblNumber1 As Label
25 ' Private lblNumber2 As Label
26 ' Private Number1, Number2 As Int
```

The selected lines set as comments.

To set the lines to normal,
select the lines and click on  or Ctrl + W.

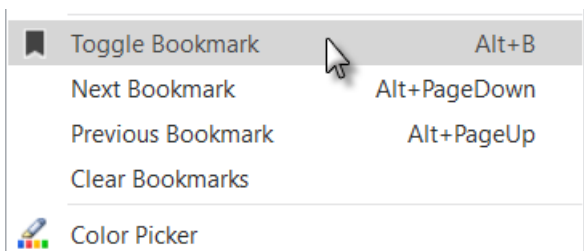
Or right click on the selected code and select  Block Comment or  Block Uncomment.


3.12 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:

```

46 | lblNumber1.Text = Number1 ' Di
47 | lblNumber2.Text = Number2 ' Di
48 | lblComments.Text = "Enter the re
49 | edtResult.Text = "" ' Sets e

```

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`

3.13 Indentation

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

You should also have a look at Auto Format.

```

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

```

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

```

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

```

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

```
20 Dim btnAction, btn0 As Button
21 Dim lblResult As Label
22 Dim lblComments As Label
23 Dim lblMathSign As Label
24 Dim lblNumber1 As Label
25 Dim lblNumber2 As Label
```

Original code.

```
20 Dim btnAction, btn0 As Button
21 Dim lblResult As Label
22 Dim lblComments As Label
23 Dim lblMathSign As Label
24 Dim lblNumber1 As Label
25 Dim lblNumber2 As Label
```

Select the code block.

Click on .

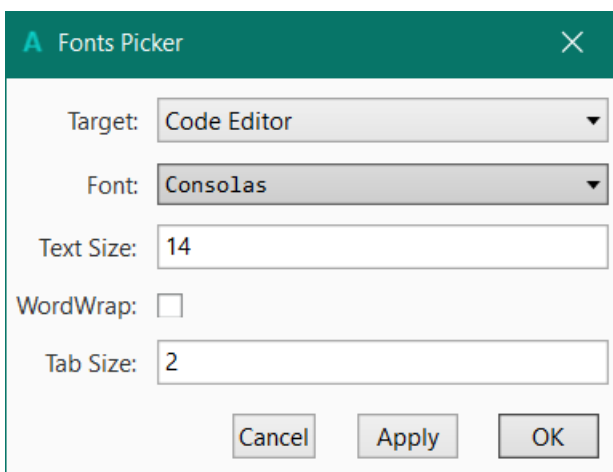
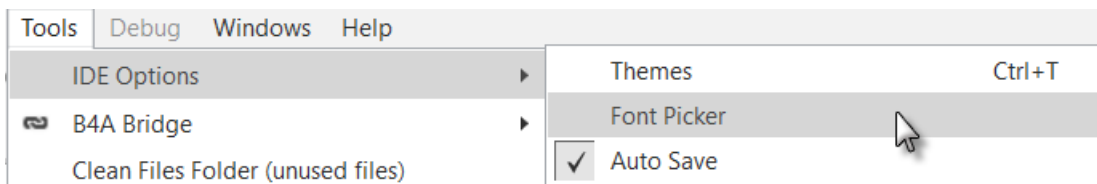
```
20 Dim btnAction, btn0 As Button
21 Dim lblResult As Label
22 Dim lblComments As Label
23 Dim lblMathSign As Label
24 Dim lblNumber1 As Label
25 Dim lblNumber2 As Label
```

The whole block has moved one tabulation to the right.

To move a block to the left.

Select the code block and click on .

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



Enter the value and click on .

3.14 Auto format

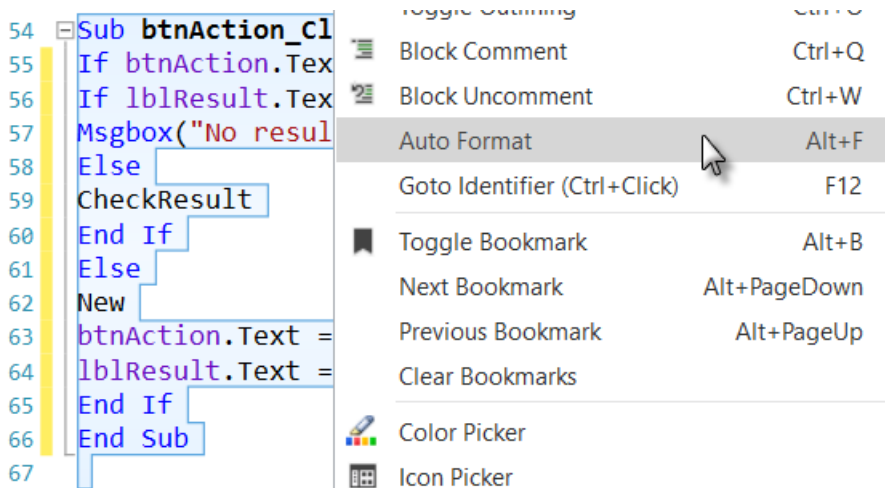
You can auto format the code.

This code is not easy to read.

```

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

```



Select the code.

Right click in the code area to show this pop-up menu.

And click on **Auto Format**.

```

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

```

And the result.

The Tab size depends on your settings, see previous page.

3.15 Documentation tool tips while hovering over code elements

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

Examples:

Hovering over Globals:

```
19 Sub Globals
20     Private Globals As String, n As Button
21     Private editresult As EditText
```

Hovering over Private:

```
20 Private btnAction As Button
21 Private Dim
22 Private Declares a variable.
23 Private Syntax:
24 Private Declare a single variable:
25 Private Dim variable name [As type] [= expression]
26 Private The default type is String.
27 Private
28 Private End Declare multiple variables. All variables will be of the specified type.
29 Private Dim [Const] variable1 [= expression], variable2 [= expression], ..., [As type]
30 Sub Activity_Create(FIRSTTIME AS Boolean)
```

3.16 Auto Completion

A very useful tool is the Auto Completion function.

Example with the SecondProgram code:

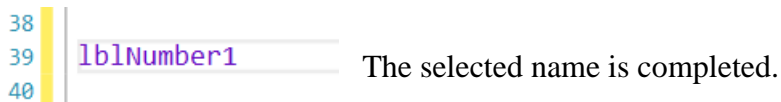


Let us write lblN.

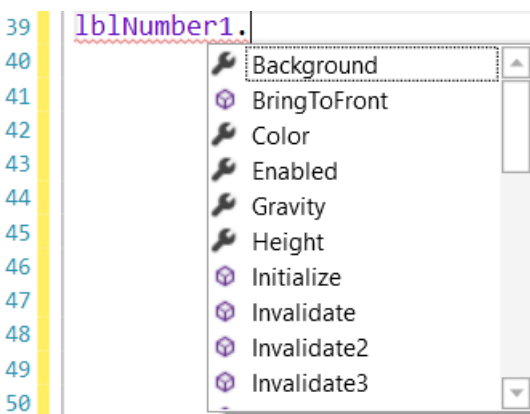
All variables, views and property names beginning with the letters already written are shown in a popup menu with the online help for

the highlighted variable, view or property name.

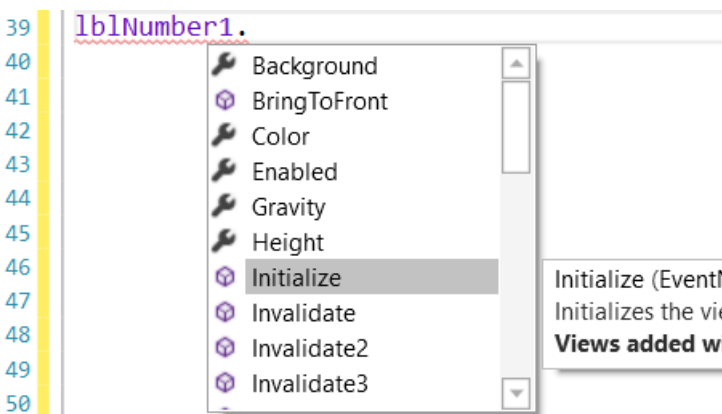
To choose lblNumber1 press Return.



To choose lblNumber2 double click on it or press the down arrow and press Return.



After pressing "." all properties and methods of the view are displayed in a popup menu.



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.

Structures are also completed.

Examples:

For / Next

<pre>38 Fo</pre>	<pre>39 ConfigureHomeWidget 40 Floor 41 For 42 ForEach 43 NumberFormat 44 NumberFormat2 45 SmartStringFormatter 46 </pre>	<pre>39 For 40 Syntax: 41 For variable = value1 To value2 [Step 42 ... 43 Next 44 If the iterator variable was not declar</pre>	<p>Type Fo</p> <p>You get For with the help.</p> <p>Press Return.</p>
--------------------	--	---	---

<pre>38 For</pre>	<p>For is completed.</p>
---------------------	--------------------------

<pre>38 For i = 0 to 9</pre>	<p>Write the rest of the instruction. And press Return.</p>
--------------------------------	---

<pre>38 For i = 0 To 9 39 40 Next</pre>	<p>Next is automatically added and the cursor is in the next line indented.</p>
---	---

If / Then

<pre>37 if 38 39 40 If 41 SmartStringFormatter 42 43 44 </pre>	<pre>40 If 41 Single line: 42 If condition Then true-statement [E 43 Multiline: 44 If condition Then</pre>	<p>Type 'if'.</p> <p>You get If with the help.</p> <p>Press Return and continue typing like in the example.</p>
---	--	---

<pre>37 If i = 0 th</pre>	<p>After th you get Then with its help.</p>
-----------------------------	---

<pre>38 39 40 Catch 41 IblMathSign 42 Logarithm 43 Then 44 ToastMessageShow</pre>	<p>Press Return.</p>
---	----------------------

<pre>41 Then</pre>	<p>And press Return again.</p>
----------------------	--------------------------------

<pre>37 If i = 0 Then</pre>	<p>End If is automatically added and the cursor is in the next line indented.</p>
-------------------------------	---

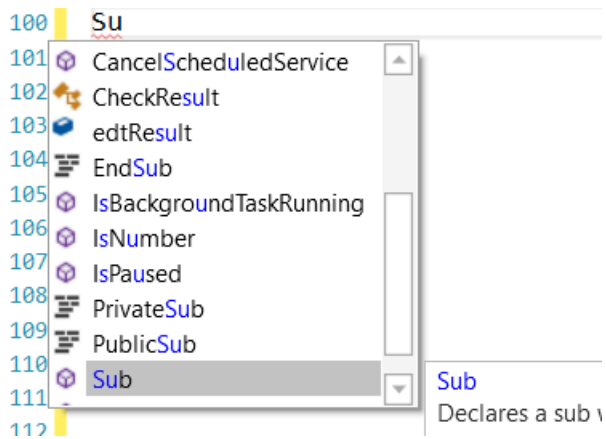
<pre>37 If i = 0 Then 38 39 End If</pre>	
--	--

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

Another very powerful Autocomplete, function allows you to create event subroutines.

In the example below we want to create the Click event for the bntOK button.

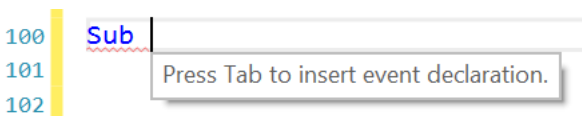
Write 'su' and the Auto Completion displays all keywords containing the two characters.



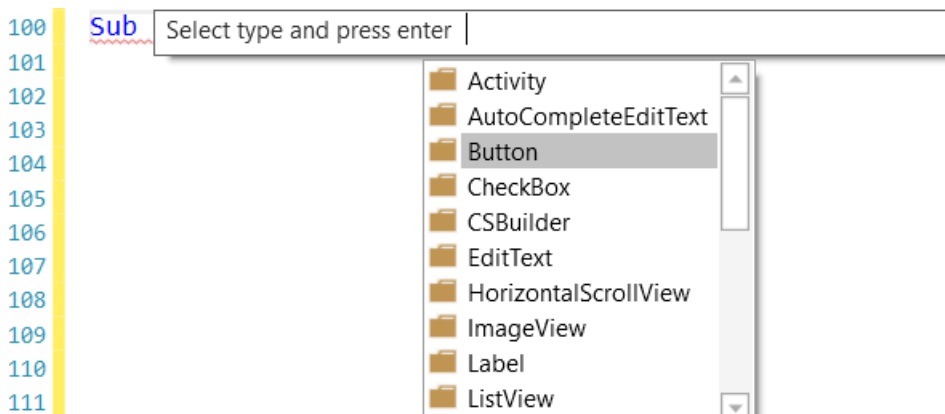
Press Return to select Sub.



Press blank.



Press Tab and select the view type, select Button.



All events for a Button are displayed, select Click.



The subroutine frame is generated.

```
100  Sub EventName_Click  
101  
102  End Sub
```

Modify 'EventName' to the event name of the button, in our example btnOK.

```
100  Sub btnOK_Click  
101  
102  End Sub
```

Press Return and the routine is ready.

```
100  Sub btnOK_Click  
101  
102  End Sub
```


3.17 Built in documentation

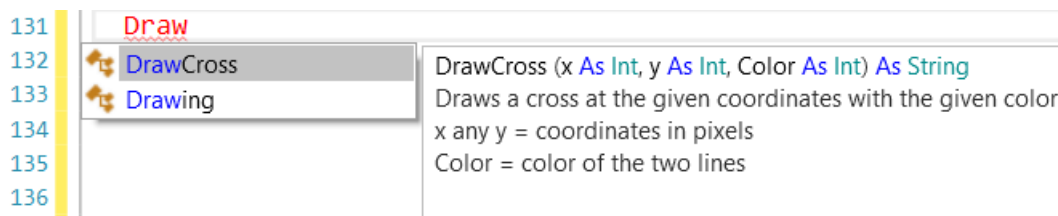
Another useful function is the built-in documentation.

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:

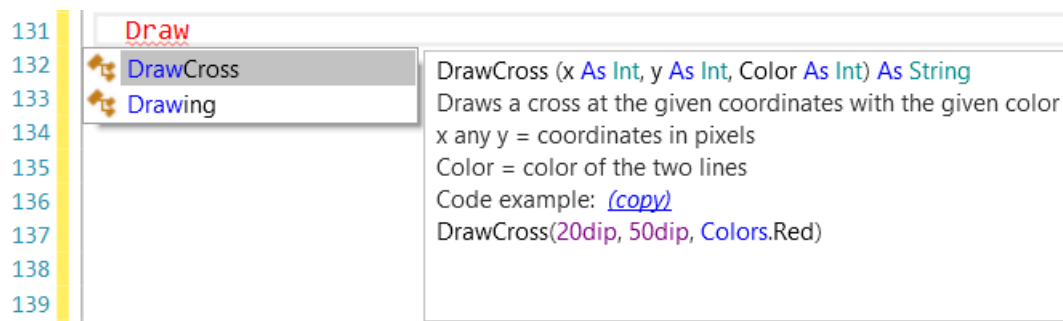


If you want to add a code example you can use `<code>` `</code>` tags:

```
'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
```

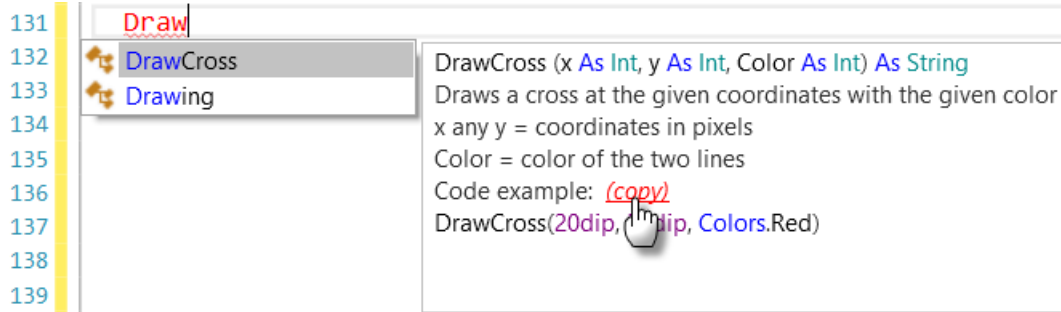
The code will be syntax highlighted:



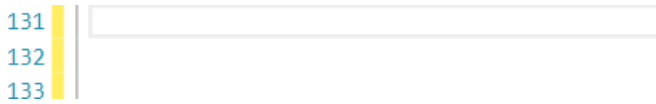
3.17.1 Copy code examples

You can copy the code example in your code.

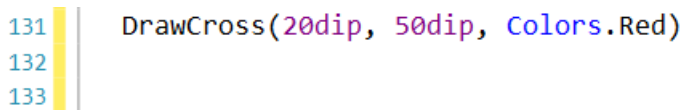
When hovering over (copy) you can copy the code example to the clipboard.



Remove `Draw`



And copy.



3.18 Jump to a subroutine

Sometimes it is useful to jump from a subroutine call to the subroutine definition. This can easily be done :

```

61 Else
62     New
63     btnAction.Text = "O K"
64     lblResult.Text = ""
65 End If

```

Select the text of the subroutine call.

Press Ctrl and Click.

```

43 Sub New
44     Number1 = Rnd(1, 10)
45     Number2 = Rnd(1, 10)
46     lblNumber1.Text = Number1
47     lblNumber2.Text = Number2

```

And you are there.

```

55 If b
56 If
57
58 El
59
60 En
61 Else
62 Ne

```

Another method.
Select the text of the subroutine call.
Right click on the selected text.
Click on **Goto Identifier (Ctrl+Click)**.

```

43 Sub New
44     Number1 = Rnd(1, 10)
45     Number2 = Rnd(1, 10)
46     lblNumber1.Text = Number1
47     lblNumber2.Text = Number2

```

And you are there.

3.19 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.

```
    lblComments.Color = Colors.RGB(255,235,128) ' yellow color
    lblResult.Text = ""           ' Sets lblResult.Text to empty
    btn0.Visible = False
End Sub

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

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"
    End If
End Sub
```

3.20 Breakpoints

Clicking on a line in the left margin adds a breakpoint. When the program is running it stops at the first breakpoint.

Breakpoints are ignored in Globals, Process_Globals and Activity_Pause.

The IDE behaves differently depending on the debug mode. The examples below are for the *rapid debug* mode.

```

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

```

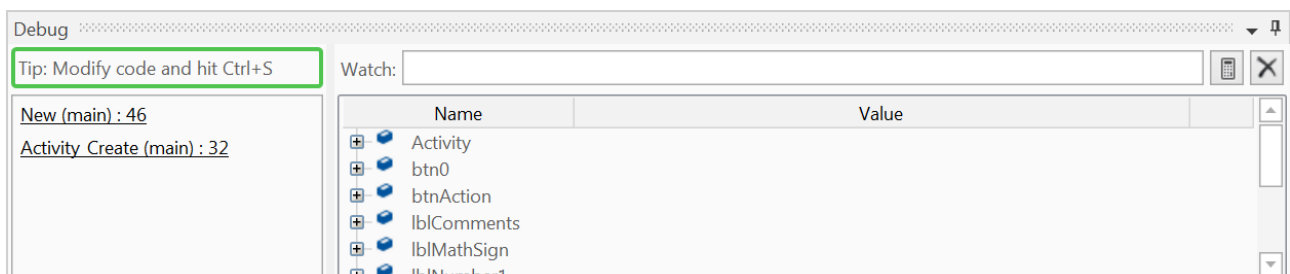
Run the program, the program stops at the breakpoint and the IDE looks like below. The line where the program stops is highlighted in yellow.

```

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

```

At the bottom of the IDE you find other information.



The Debugger is connected. In the left part of the Debugger window we find:

- [Tip: Modify code and hit Ctrl+S](#) A button to update the program after a code modification.
- [New \(main\) : 46](#) The name of the routine where the Debugger stopped the program. New in the module Main in line 46.
- [Activity Create \(main\) : 32](#) Caller of the “New” routine:
Activity_Create in the module Main routine in line 32.

Clicking on these links moves the cursor to the given line.

In the right part of the Debugger window we find the list of all Views and Variables with their values.



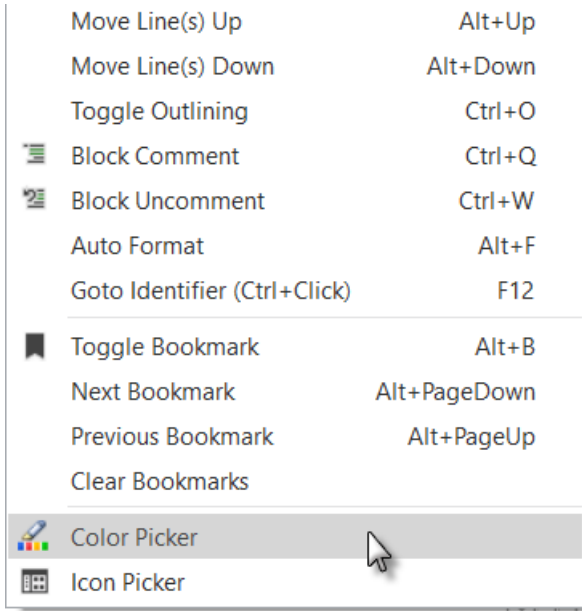
In the Toolbar, at the top of the IDE the navigation buttons are enabled.



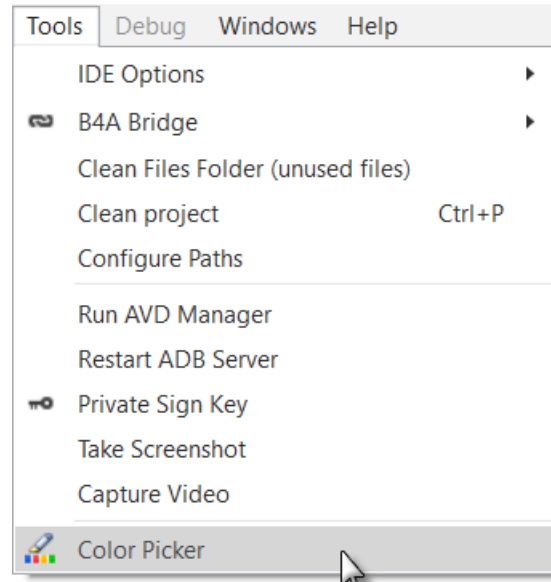
▶	Run the program		Runs the program, no action in Debug (rapid)
↳	Step In	F8	Executes the next statement.
↳	Step Out	F9	Leaves the current subroutine.
↳	Step Over	F10	Steps over the subroutine call.
■	Stop		Stops the program.
↻	Restart	F11	Restarts the program.

3.21 Color Picker Color Picker

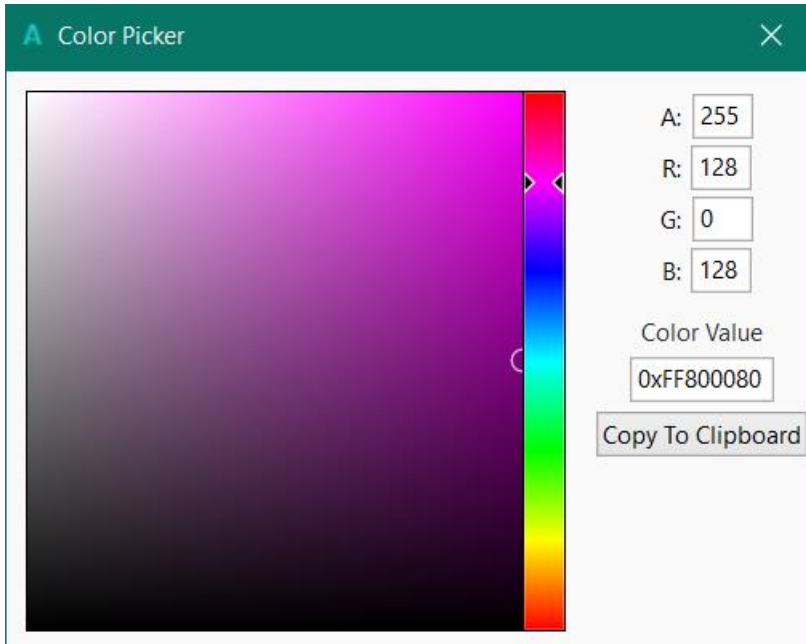
In the code, right click to show the popup menu below.



Or, in the menu Tools.



Click on  Color Picker to show the Color Picker.



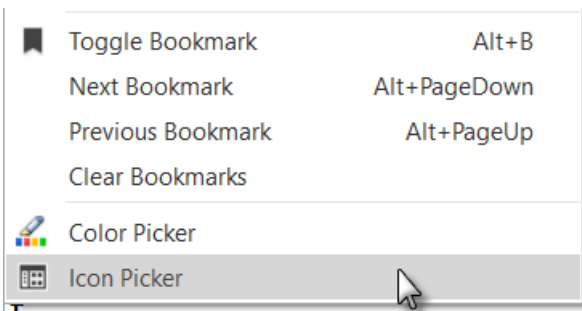
You can move the cursor in the square and the rectangular areas.

Or enter the A R G B values.

Copy the value to the Clipboard.

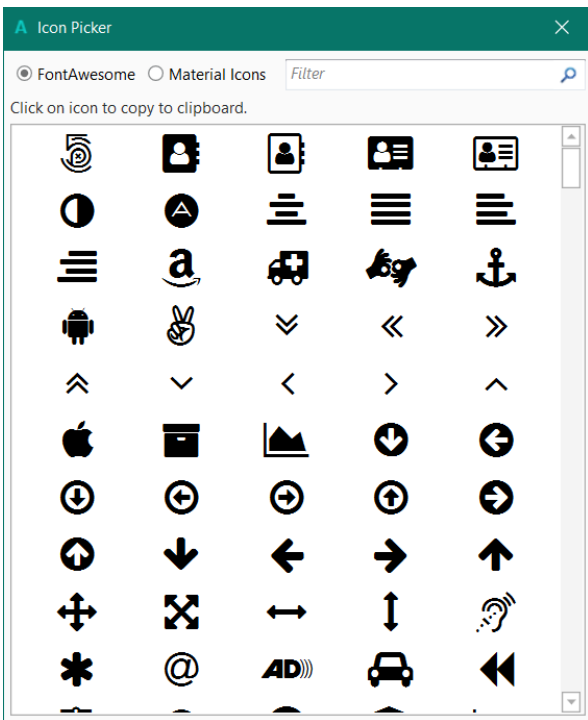
You can then paste the value into the code.

3.22 Icon Picker Icon Picker

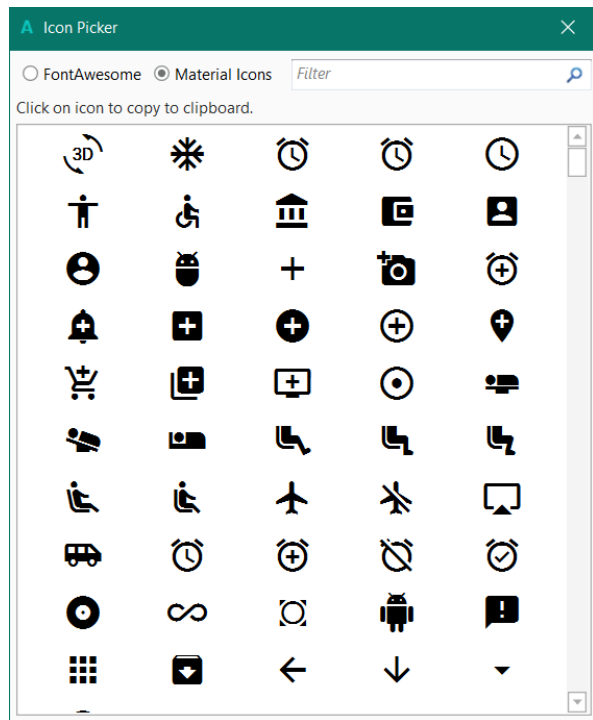


Right click in the IDE code area to show the pop-up menu and click on **Icon Picker**.

You can choose between Font Awesome and Material icons.



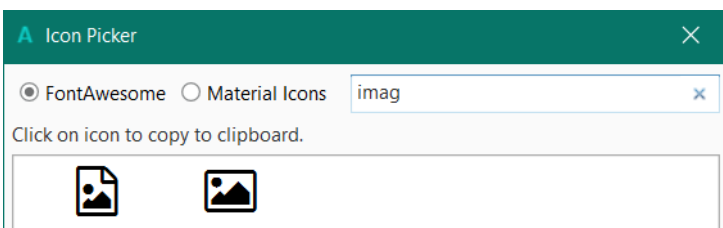
Font Awesome icons.



Material icons.

Click on an icon to copy it to the clipboard.
 Then you can paste it into the code like below.
 The icon is given with its character number, `Chr(0xE632)`.

```
lblResult.Text = Chr(0xE632)
```




You can filter the icons.

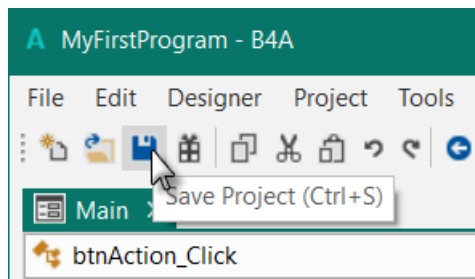
3.23 Colors in the left side

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

```
67 Sub CheckResult
68     If edtResult.Text = N
69         lblComments.Text =
70         btnAction.Text = "N
71     Else
72         lblComments.Text =
73     End If
74 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.



```
67 Sub CheckResult
68     If edtResult.Text = N
69         lblComments.Text = '
70         btnAction.Text = "N
71     Else
72         lblComments.Text = '
73     End If
74 End Sub
```

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

3.24 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 is executed. Hovering over the line with Ctrl pressed does also highlight the url.

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

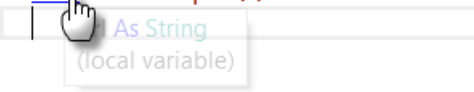


In a String:

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

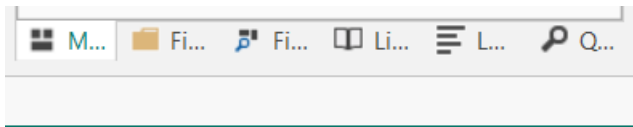
The cursor must be over the String variable and not over text.

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



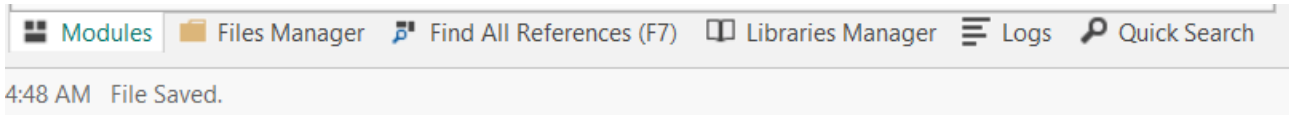
4 Tabs

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



The short version.

The wide version.



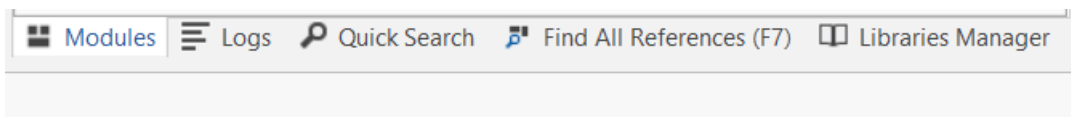
The 6 Tabs are:

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

Each Tab has its own window.

By default they are displayed in the Tab area on the right side of the IDE, only one at the same time. These windows can be closed, hidden or floating, see next chapter.

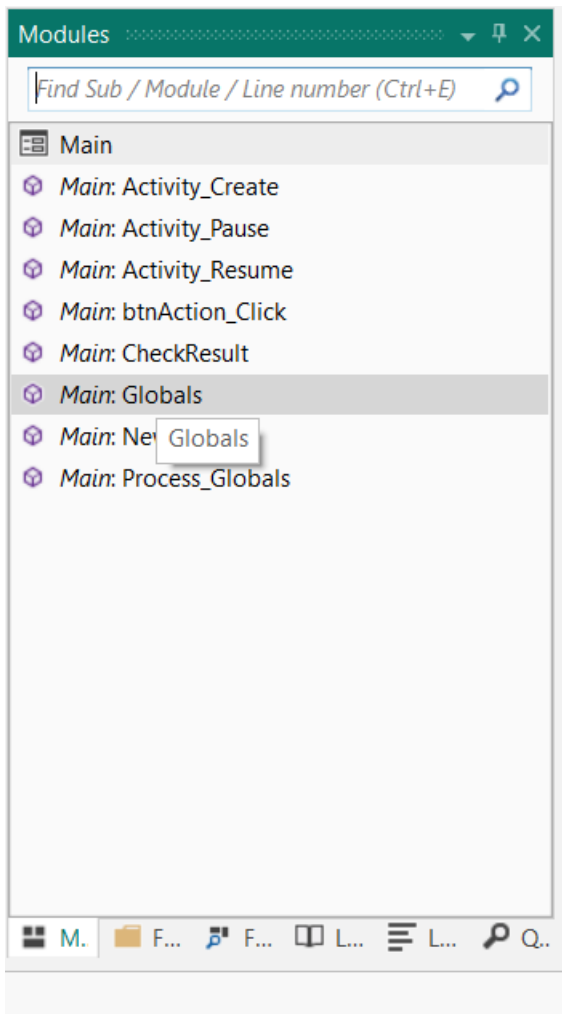
B4R



Only 5 Tabs, no Files Manager Tab


4.1 Floating Tab windows

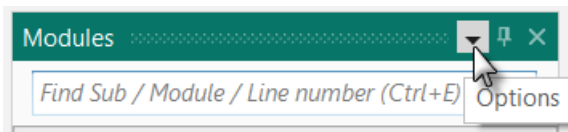
When you start the default IDE all Tab windows are docked in the Tab area.

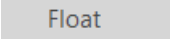


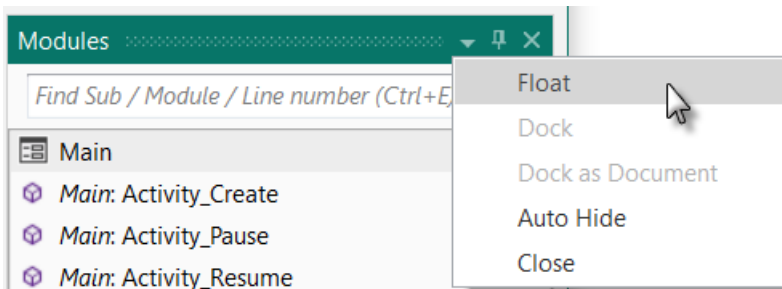
You can set each Tab window as a separate floating window.

4.2 Float

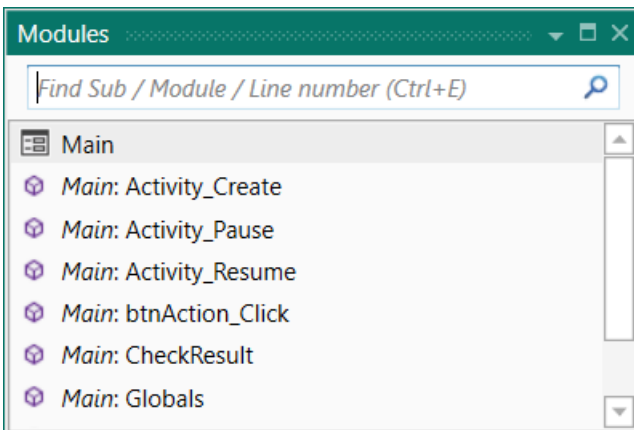
To set the Modules Tab window to floating click in the title on .

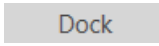


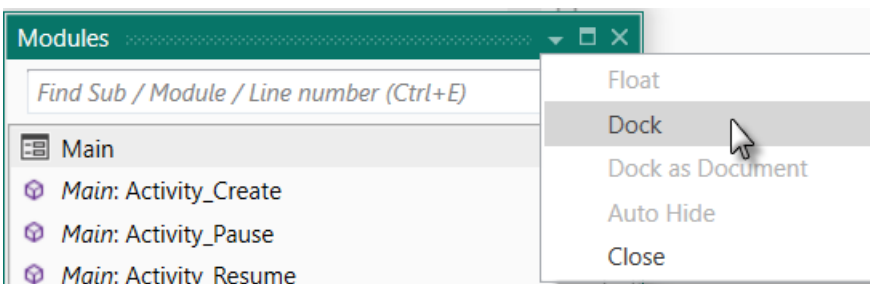
Click on .



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

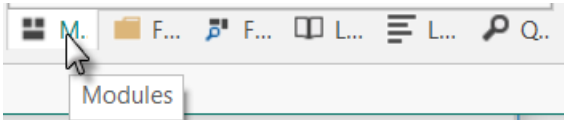


To dock it back to the Tab area click on .

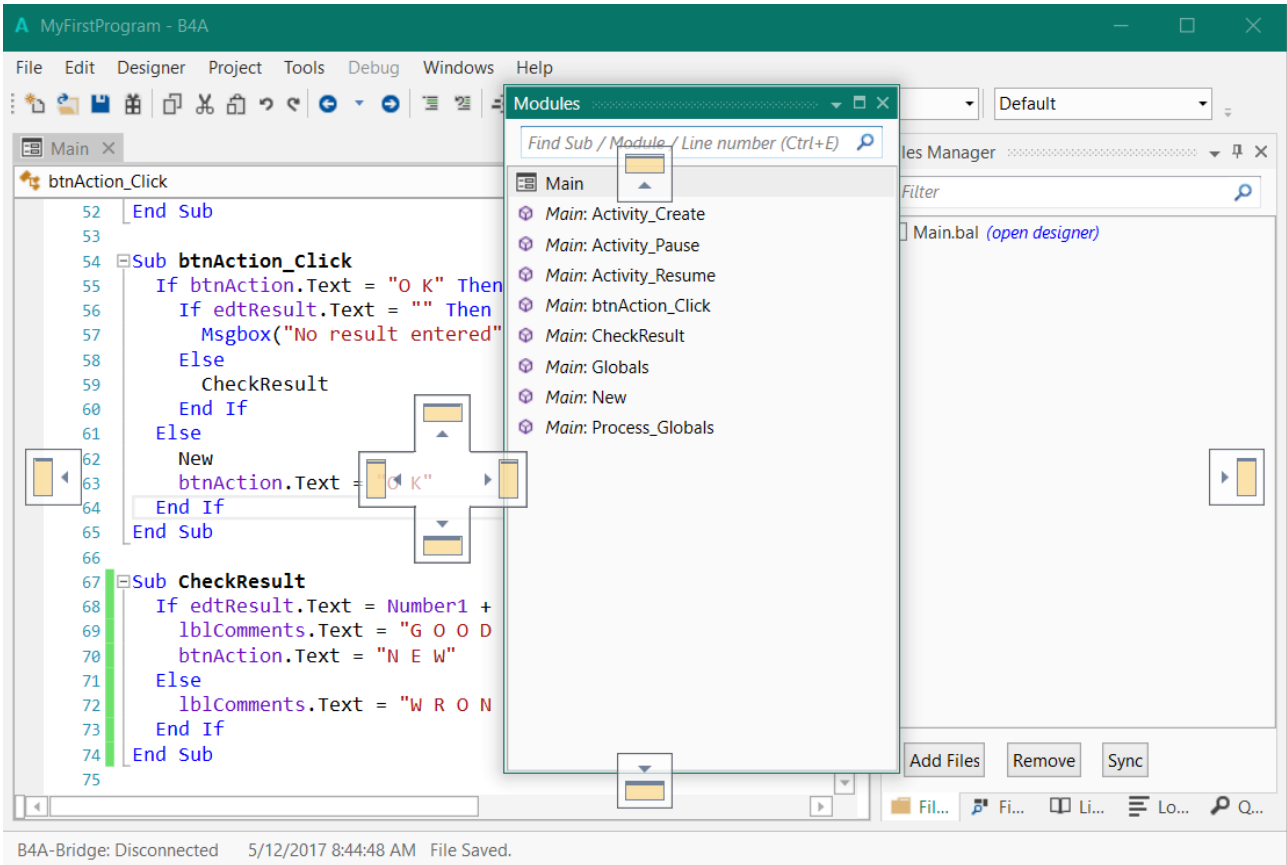


To show the Tabs again click either on Dock in the Options or on Reset in the IDE Window menu.

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

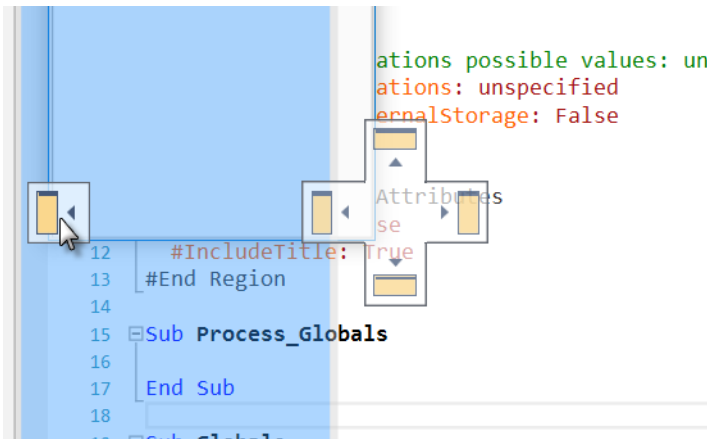


Right

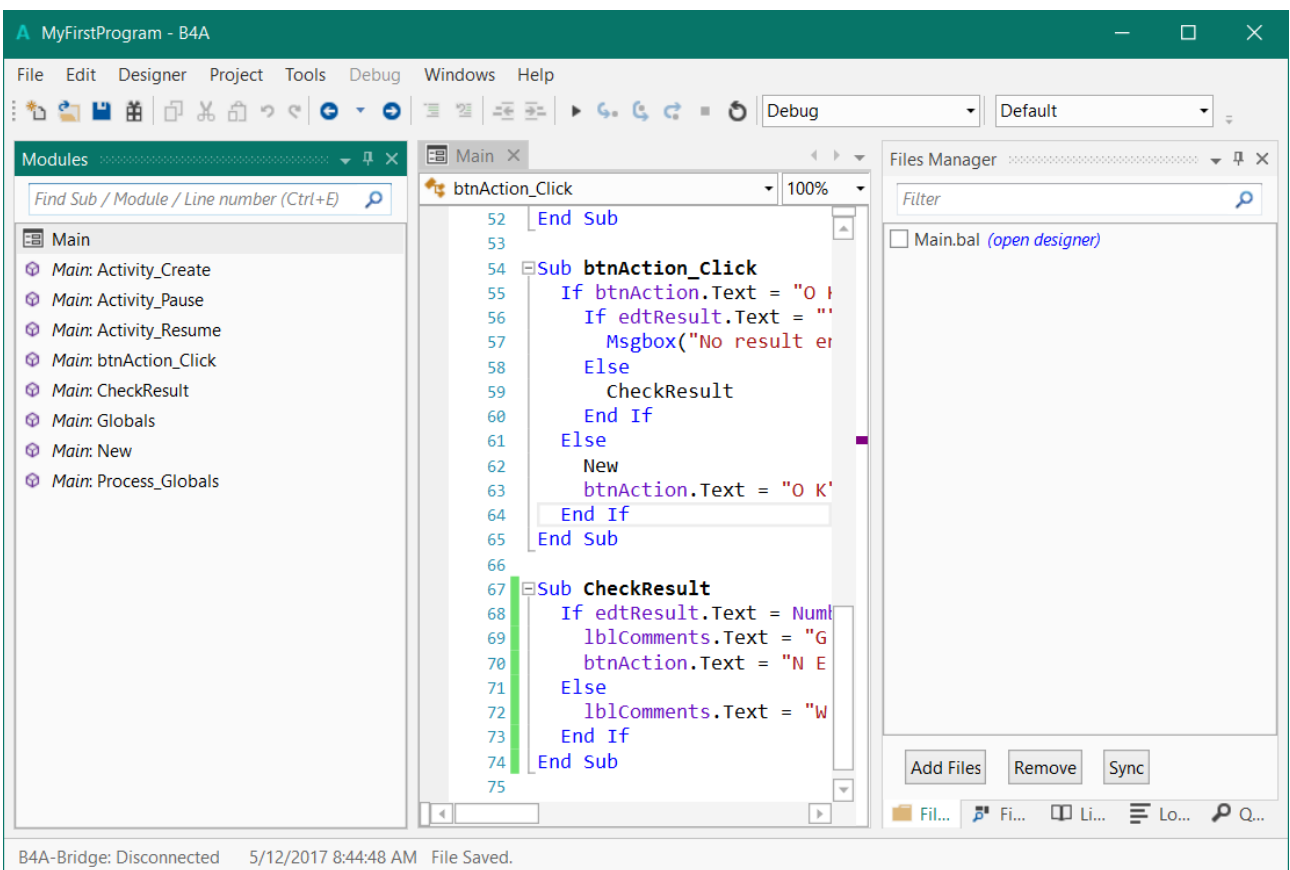


Bottom

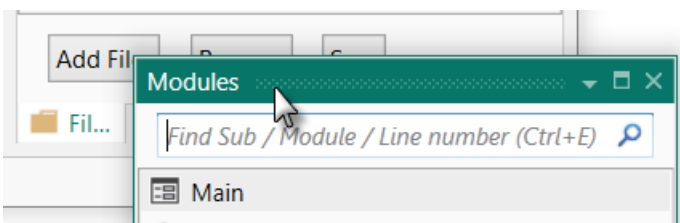
If you mouve the mouse onto one of the docking area symbol, the Tab window will be either on top, on the left, the right or on the bottom.




And the result.

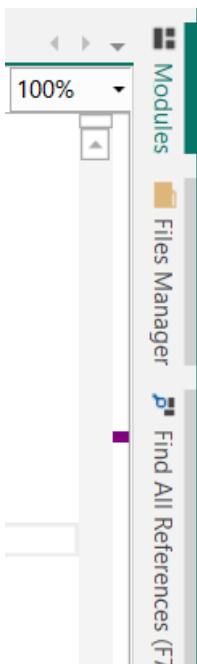
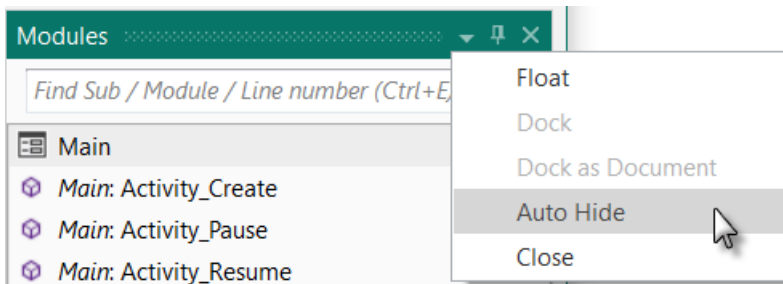


To bring it back to the Tabs, click on the window title and move it back to the Tabs.



4.3 Auto Hide

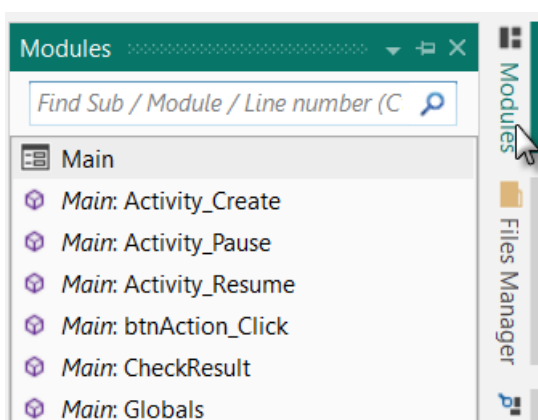
Click on  in the title or click on **Auto Hide** in the Options.



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

Hovering over a Tab highlights it in green.

Click on a Tab to show it.

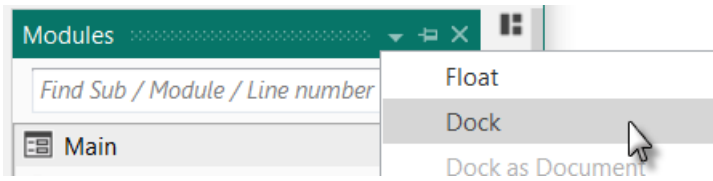


The selected Tab is displayed.

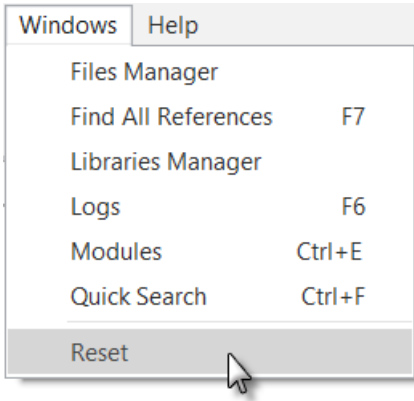
As soon as you click on something in the IDE the Tab is hidden again.

To move the Tabs back to the lower right corner:

Click on **Dock** in the Options.


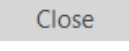


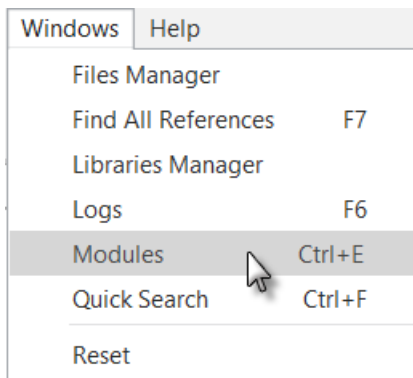
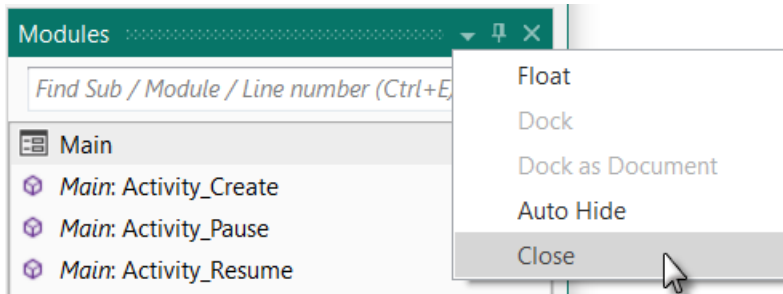
Or click on **Reset** in the IDE Windows menu.



4.4 Close

You can close a window, hide it.

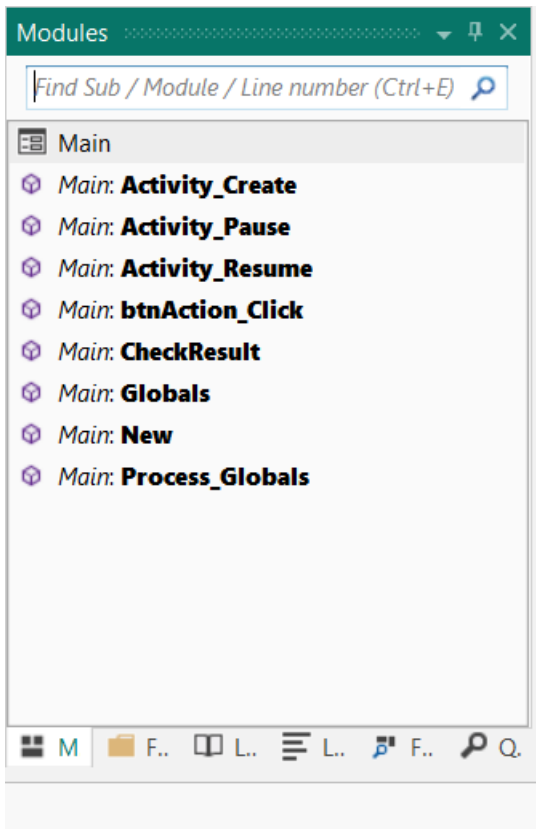
Click on  in the title or on  in the Options.



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

4.5 Modules and subroutine lists Modules

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.



[Find Sub / Module / Line number \(Ctrl + E\)](#)

Module list on top.

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

Find Sub Tool (Ctrl + E) see below

Find All References (F7) see below

Subroutine list of the selected module.

Clicking on a subroutine shows its code in the middle of the code area.

In the IDE, in the bottom right corner.

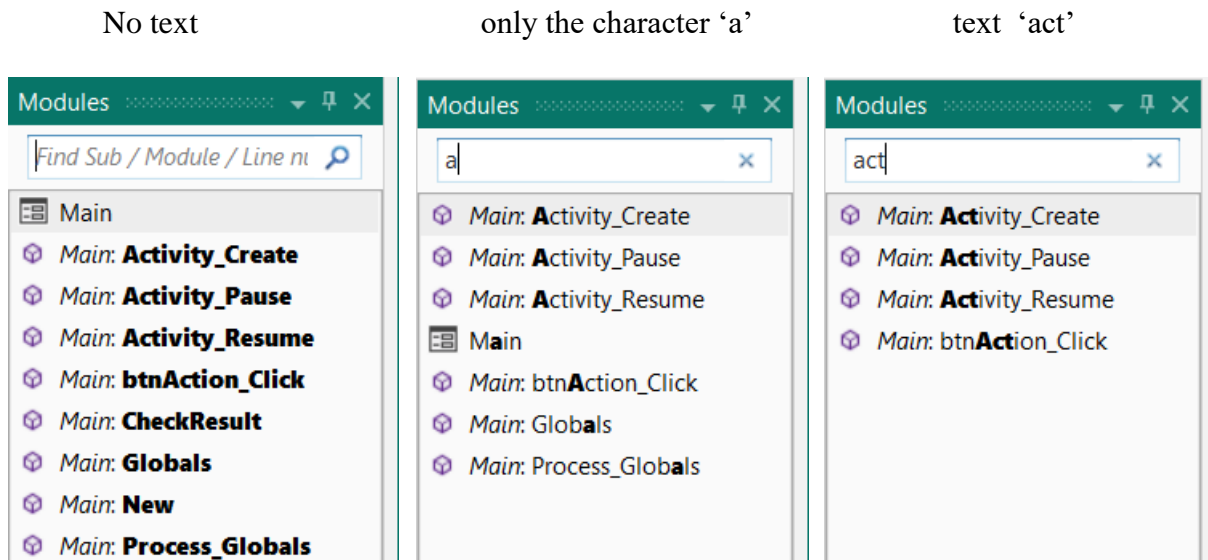
To show a hidden module, click on the module name in the module list.

4.5.1 Find Sub / Module / Line number (Ctrl + E)

The *Find Sub / Module / Line number* 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.



Shows all modules and all routines of the selected Module.

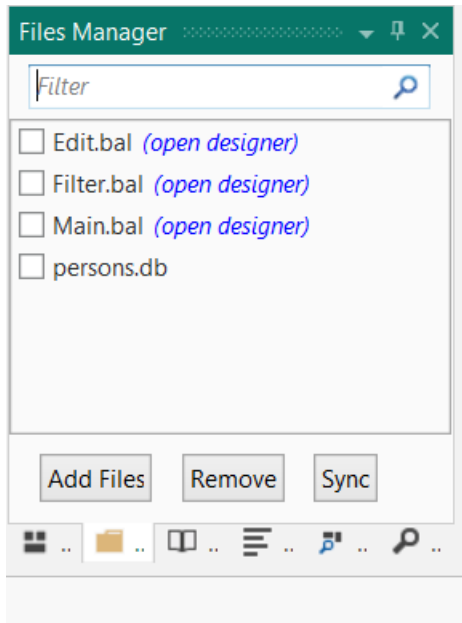
Shows all modules and routines containing 'a'.

Shows all modules and routines containing 'act'.

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

4.6 Files Manager Files Manager B4A, B4i and B4J only

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. These can be any kind of files: layouts, images, texts, etc.

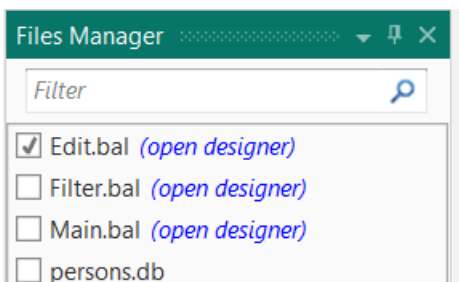


Click on [\(open designer\)](#) to open the Designer with the selected file.

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.

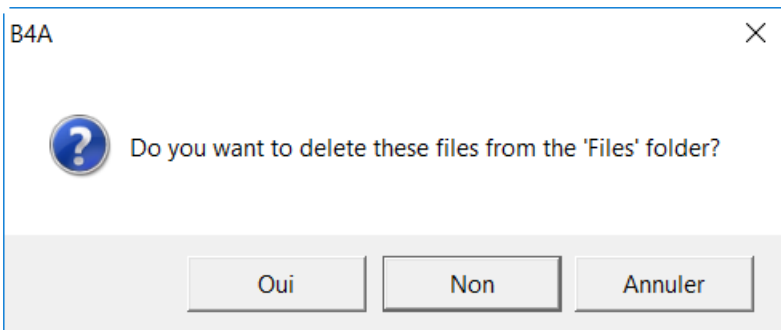
In the IDE, in the bottom right corner.



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.

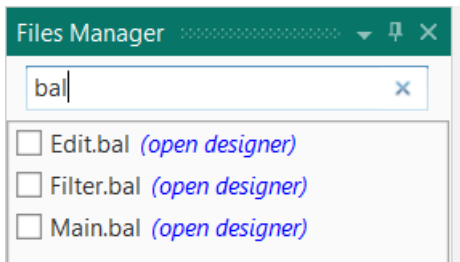
Oui = Yes

Non = No

Annuler = Cancel

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.

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



Enter '.bal' to filter all layout files,

4.7 Logs Logs

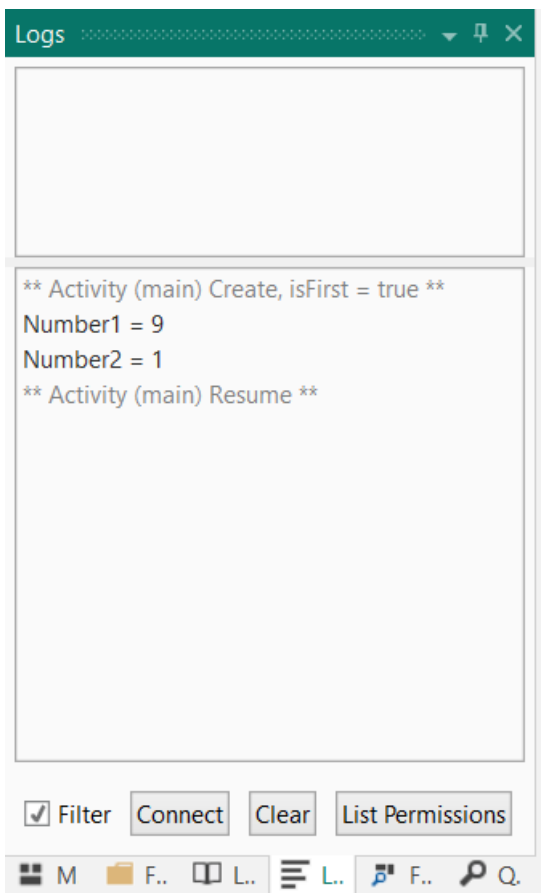
Display of Log comments generated by the program when it is running.

We add the two lines 44 and 46 in the program 'SecondProgram' in the 'New' routine. The number of the lines may be different from yours.

```

43 Sub New
44     Number1 = Rnd(1, 10)      ' Generates a random number between 1 and 9
45     Log("Number1 = " & Number1)
46     Number2 = Rnd(1, 10)    ' Generates a random number between 1 and 9
47     Log("Number2 = " & Number2)
48     lblNumber1.Text = Number1 ' Displays Number1 in label lblNumber1
49     lblNumber2.Text = Number2 ' Displays Number2 in label lblNumber2
50     lblComments.Text = "Enter the result" & CRLF & "and click on OK"
51     lblComments.Color = Colors.RGB(255,235,128) ' yellow color
52     lblResult.Text = ""      ' Sets lblResult.Text to empty
53     btn0.Visible = False
54 End Sub

```



Run the program.

Click on to connect the logger.

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.

```

** Activity (main) Create, isFirst = true **
Number1 = 9           First log message
Number2 = 1           Second log message
** Activity (main) Resume **

```

Filter When *Filter* 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

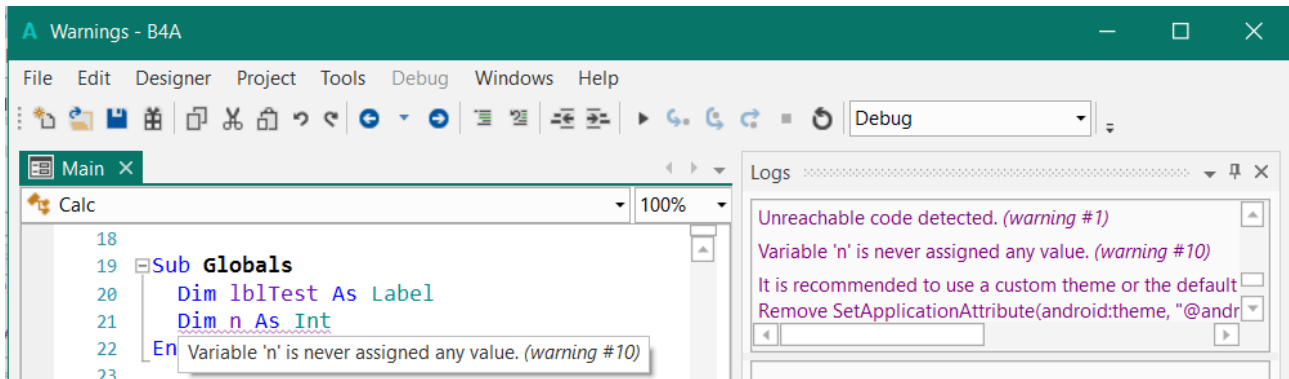
Click on to clear the Logs window.

4.7.1 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.

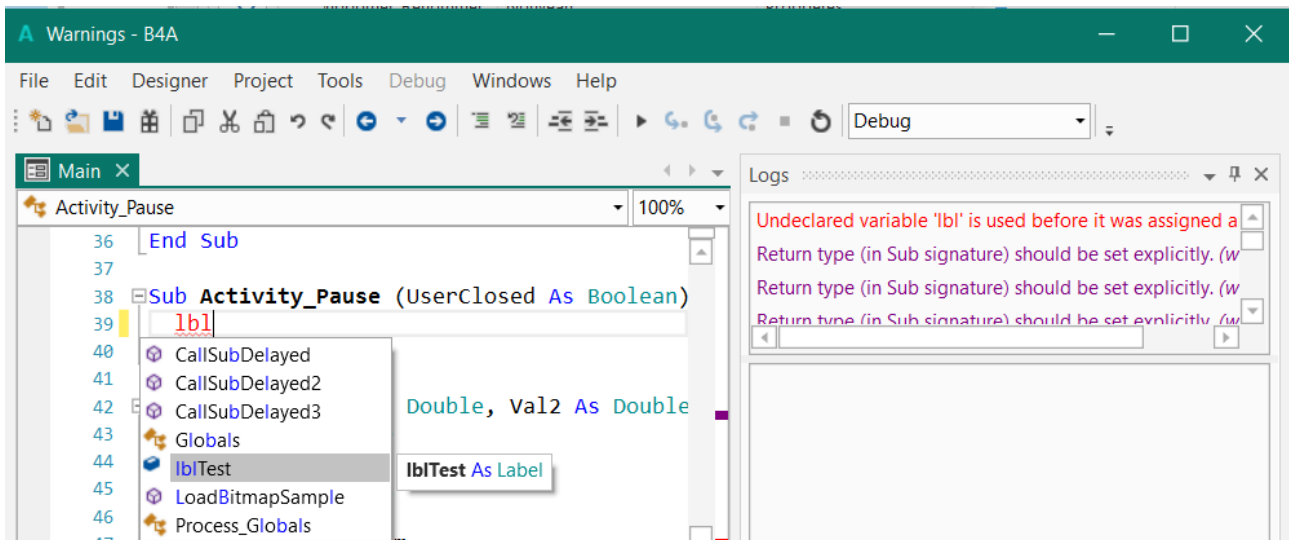
The compile-time warnings appear above the logs and in the code itself when hovering with the cursor above the code line.

The code lines which cause a warning are underlined like this `Dim 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 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.

4.7.1.1 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:

```
50 | Sub Test                                     50 | Sub Test 'ignore
51 | Dim h As Int                               51 | Dim h As Int
```

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

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

```
#Region Project Attributes
  #ApplicationLabel: Warnings
  #VersionCode: 1
  #VersionName:
  'SupportedOrientations possible values: unspecified, landscape or portrait.
  #SupportedOrientations: unspecified
  #CanInstallToExternalStorage: False
  #IgnoreWarnings: 10, 12
#End Region
```

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

4.7.1.2 List of warnings

The warning may be different in the four products.

- 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: The following value misses screen units ('dip' or %x / %y): {1}.
- 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 Activity.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: Modal dialogs are not allowed in Sub Activity_Pause. It will be ignored.
- 24: Accessing fields from other modules in Sub Process_Globals can be dangerous as the initialization order is not deterministic.
- 28: It is recommended to use a custom theme or the default theme.
Remove SetApplicationAttribute(android:theme, “@android:style/Theme.Holo”) from the manifest editor.
- 32: Library ‘xxxx’ is not used.

'Runtime warnings

- 1001: Panel.LoadLayout should only be called after the panel was added to its parent.
- 1002: The same object was added to the list. You should call Dim again to create a new object.
- 1003: Object was already initialized.
- 1004: FullScreen or IncludeTitle properties in layout file do not match the activity attributes settings.

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 !

Other example:

Wrong code

```

Sub Activity_KeyPress(KeyCode As Int) As Boolean
  Private Answ As Int
  Private Txt As String

  If KeyCode = KeyCodes.KEYCODE_BACK Then' Checks if the KeyCode is BackKey
    Txt = "Do you really want to quit the program ?"
    Answ = MsgBox2(Txt,"A T T E N T I O N","Yes","","No",Null) ' MessageBox
    If Answ = DialogResult.POSITIVE Then ' If return value is Yes then
      Return False ' Return = False the Event will not be consumed
    Else
      ' we leave the program
      Return True ' Return = True the Event will be consumed to avoid
    End If
  End If
End Sub

```

Correct code

```

Sub Activity_KeyPress(KeyCode As Int) As Boolean
  Private Answ As Int
  Private Txt As String

  If KeyCode = KeyCodes.KEYCODE_BACK Then' Checks if the KeyCode is BackKey
    Txt = "Do you really want to quit the program ?"
    Answ = MsgBox2(Txt,"A T T E N T I O N","Yes","","No",Null) ' MessageBox
    If Answ = DialogResult.POSITIVE Then ' If return value is Yes then
      Return False ' Return = False the Event will not be consumed
    Else
      ' we leave the program
      Return True ' Return = True the Event will be consumed to avoid
    End If
  Else
    Return True ' Return = True the Event will be consumed to avoid
  End If
End Sub

```

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

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

    Sum = Val1 + Val2
    Return
End Sub
```

Correct code

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

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

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

Wrong code

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

Correct code

```
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 Private, Public or Dim declarations it's allowed, in the line below both variables are Doubles:

```
Private Val1, Val2 As Double
```

6: The following value misses screen units ('dip' or %x / %y): {1}.

Wrong code

```
Activity.AddView(lblTest, 10, 10, 150, 50)
```

Correct code

```
Activity.AddView(lblTest, 10dip, 10dip, 150dip, 50dip)
```

In the example above you will get four warnings, one for each value.

For view dimensions you should use dip, %x or %y values.

See chapter [5.1 Special functions like 50%x, 50dip](#)

7: Object converted to String. This is probably a programming mistake.**8: Undeclared variable '{1}'.**

Wrong code

```
Sub SetHeight
    h = 10dip
End Sub
```

Correct code

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

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

9: Unused variable '{1}'.

```
Sub SetHeight
  Private h As Int
  h = 10dip
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:

```
Sub SetHeight
  Private h As Int
  h = 10dip
  lblTest.Height = h
End Sub
```

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

```
Sub Test
  Private 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

```
Private lst As List
lst.Add("Test1")
```

Correct code

```
Private 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.

Wrong code :

```
Sub Globals
  Public Timer1 As Timer
  Public GPS1 As GPS
```

Correct code :

```
Sub Process_Globals
  Public Timer1 As Timer
  Public GPS1 As GPS
```

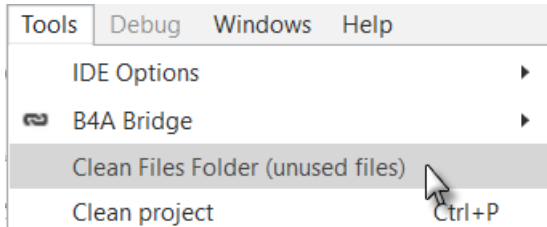
Certain objects like Timers and GPS should be declared in Process_Globals, not in Globals.

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.

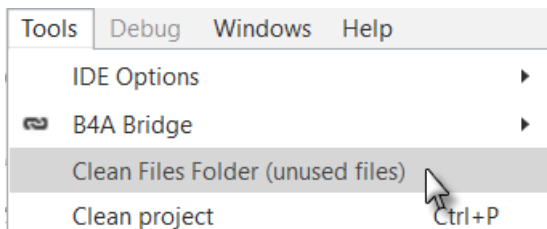
Or you can clean the Files folder from within the Tools menu (see above).

16: Layout file '{1}' is not used. Are you missing a call to Activity.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 [4.3.2 Files](#)

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

Wrong code

```
lblTest.TextSize = 16dip
```

Correct code

```
lblTest.TextSize = 16
```

TextSize values are pixel and density independent. Their unit is the [typographic point](#), a typographic unit, and must be given absolute values and not dip values.

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

Wrong code

```
Try
    imageView.Bitmap = LoadBitmap(File.DirRootExternal, "image.jpg")
Catch

End Try
```

Correct code

```
Try
    imageView.Bitmap = LoadBitmap(File.DirRootExternal, "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.**

Modal dialogs like `MessageBox` should not be used in the `Activity_Pause` routine.

24: Accessing fields from other modules in Sub Process_Globals can be dangerous as the initialization order is not deterministic.**28: It is recommended to use a custom theme or the default theme.**

Remove `SetApplicationAttribute(android:theme, "@android:style/Theme.Holo")` from the manifest editor.

This was set automatically in older versions of B4A. No more needed.

32: Library 'xxxx' is not used.

Remove the unused library.

4.8 Libraries Manager Libraries Manager

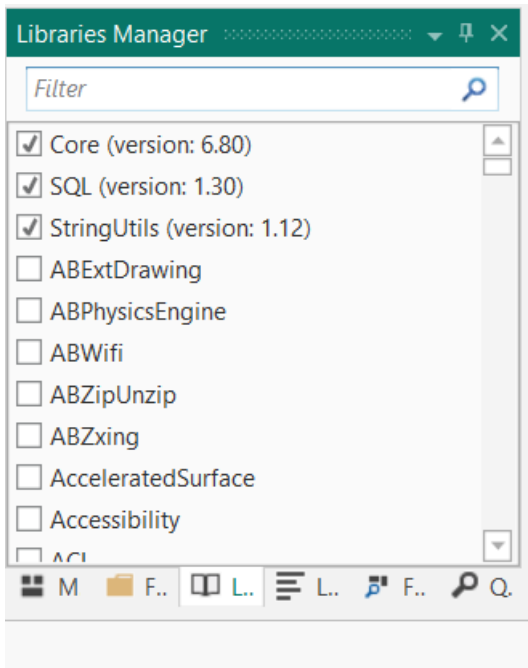
The “Libraries Manager” Tab contains a list of the available libraries that can be used in the project.

The images are an example with B4A.

The libraries in the list depends on the available libraries in the given IDE.

Check the libraries you need for your project.

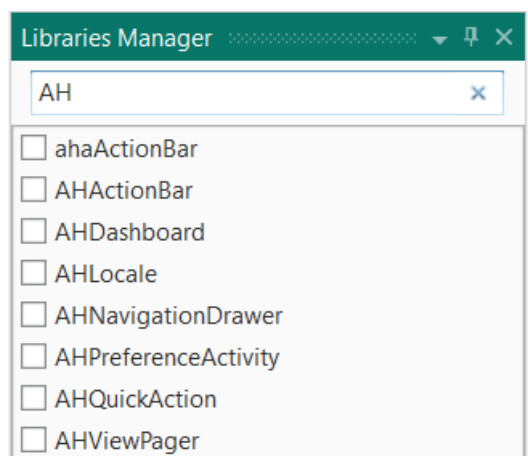
Make sure that you have the latest version of the libraries.



The used libraries are shown on top of the list.
As soon as you select one it moves to the top of the list.



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



Enter ‘AH’ and you get all libraries beginning with AH.

The list of all additional libraries can be found here:
[B4A](#), [B4i](#), [B4J](#), [B4R](#)



Clicking on a link in the list shows the documentation.

Libraies are explained in detail in the
B4x Basic Language booklet.

4.9 Quick Search 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:

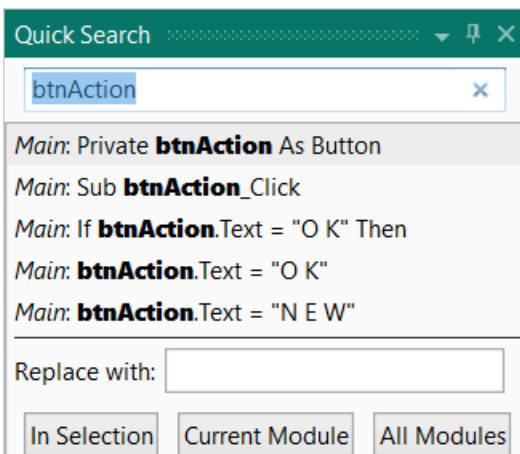
- Ctrl + F, the easiest and most efficient way.
- Click on the  Quick Search Tab in the lower right corner of the IDE.
- Click on  Quick Search Ctrl+F in the Edit menu.

Example:

```
Sub Globals
Private btnAction As Button
Private edtRe btnAction As Button
Private lblCo (global variable)
Private lblMainSign 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.

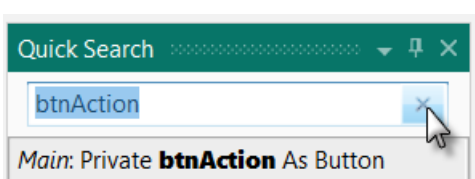
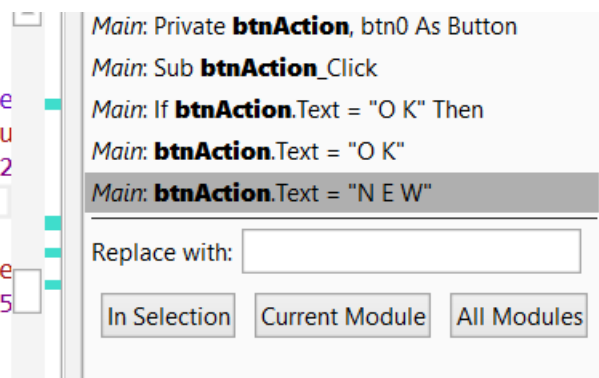



The list shows the occurrences in all Modules.

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

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

```
Sub CheckResult
If lblResult.Text = Number1 + Numbe
lblComments.Text = "G O O D resu
lblComments.Color = Colors.RGB(12
btnAction.Text = "N E W"
Else
lblComments.Text = "W R O N G re
lblComments.Color = Colors.RGB(25
End If
End Sub
```



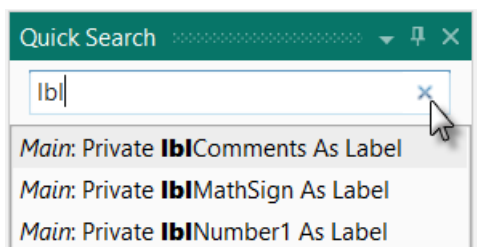
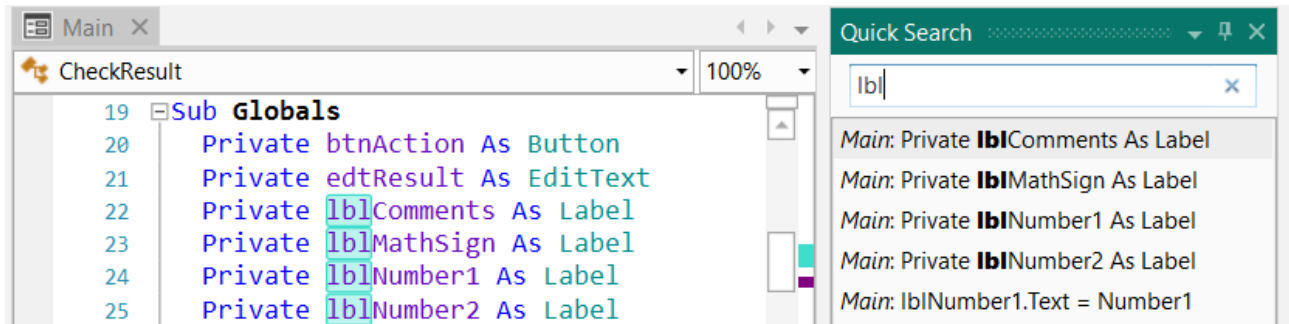
To remove the selection click on  on the top right corner of the Quick Search window.


You can also enter any text in the search field:

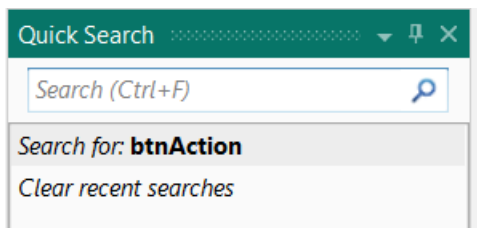
As an example, enter *lbl* in the Search field and you get the window below where you find all lines containing the text you entered, *lbl* in this example.

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

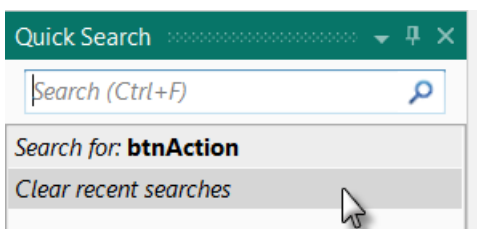
Clicking on one of the lines in the list jumps directly to this line in the IDE.



Click on  to remove a search.



You will see a list of the last searches.



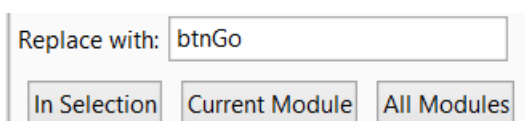
Click on **Clear recent searches** to remove all recent searches.

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.


You can replace an object either in the selected code, in the current module or in all modules.

Enter the new name and click either on **In Selection**, **Current Module** or **All Modules**.



4.10 Find All References (F7) Find All References (F7)

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.

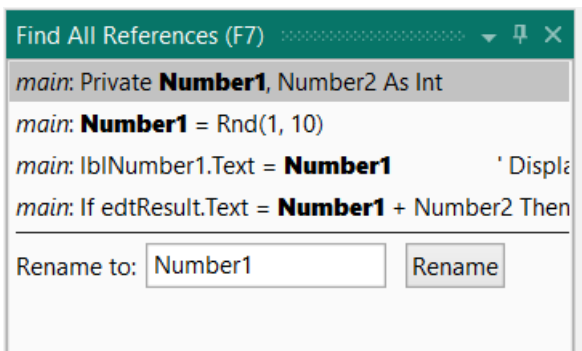
Select in the code in line 49 `Number1`.

```


45 Sub New
46     Number1 = Rnd(1, 10) ' Generates a random number between 1 and 9
47     Number2 = Rnd(1, 10) ' Generates a random number between 1 and 9
48     lblNumber1.Text = Number1 ' Displays Number1 in label lblNumber1
49     lblNumber2.Text = Number2 ' Displays Number2 in label lblNumber2

```

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.

You can change the name of the selected object. Enter a new name and click on .

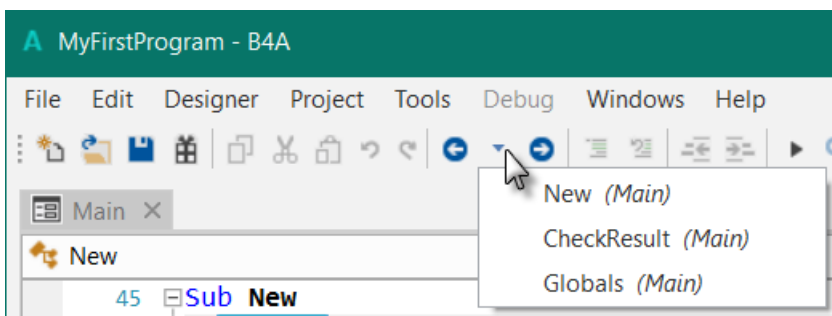
5 Navigation in the IDE

5.1 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.

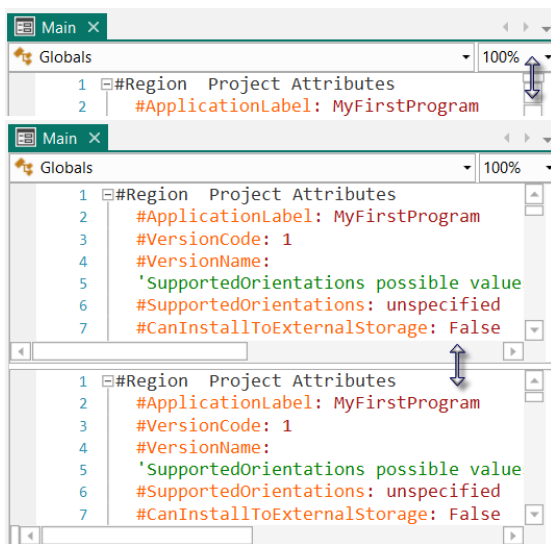
5.2 Alt + N Navigation stack menu

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

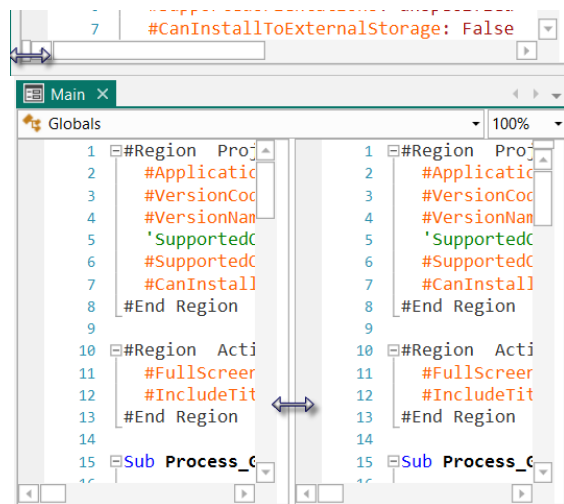


5.3 Split the screen

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

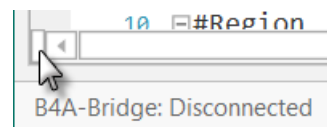
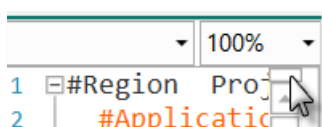


Horizontally



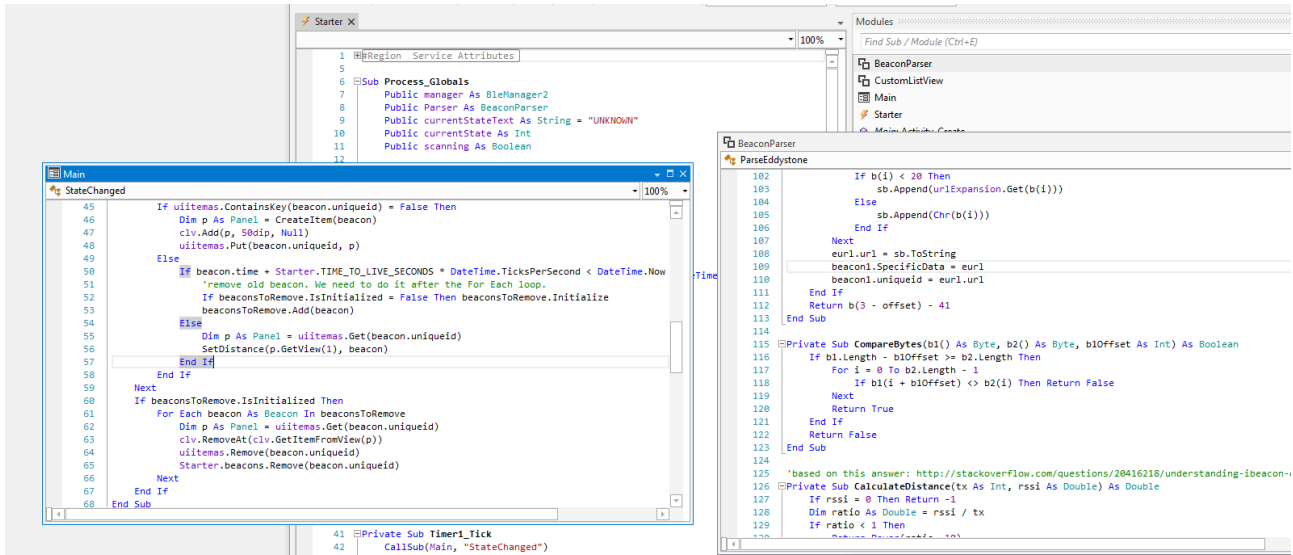
Vertically

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



5.4 Multiple windows

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



5.5 Ctrl + E Search for sub or module

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

5.6 Ctrl + Click on any sub or variable

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

5.7 F7 - Find all references

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

5.8 Ctrl + F Quick Search

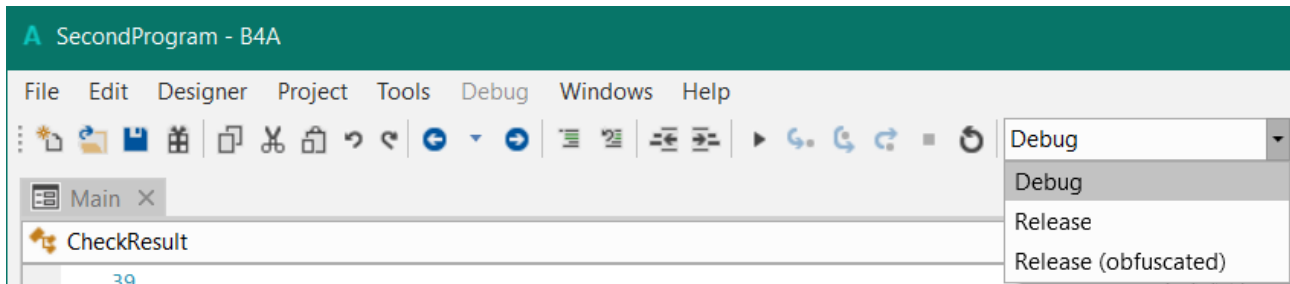
Ctrl + F - Index based quick search. Details in [Quick Search](#).

6 Debugging B4A, B4i, B4J

Debugging is an important part when developing.
Debugging is different in B4R than in B4A, B4i and B4J.

6.1 B4A, B4i, B4J

To allow debugging you must activate the debugging mode *Debug* on top of the IDE.



Notes about the debugger (B4A only):

- Breakpoints in the following subs will be ignored: Globals, Process_Globals and Activity_Pause.
- Services - Breakpoints that appear after a call to StartService will be ignored. Breakpoints set in Service_Create and Service_Start will pause the program for up to a specific time (about 12 seconds). This is to prevent the OS from killing the Service.
- Events that fire when the program is paused will be executed. Breakpoints in the event code will be ignored (only when the program is already paused).
- The data sent from the device to the IDE is limited in size. Long strings may be truncated.
- When the debugger is running in *rapid* mode, you can change the code and run the changes.
- When the debugger is running in *legacy* mode, the IDE is read-only. The user cannot change any of the program text.

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.

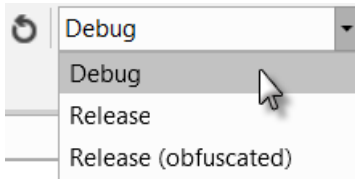
The program will pause when it reaches a breakpoint and will allow you to inspect the current state.

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.

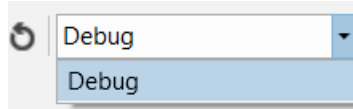
Note that the log is maintained by the device. When you connect to a device you will also see previous messages.

6.1.1 Debug mode

The debugging modes are different in the in the products:



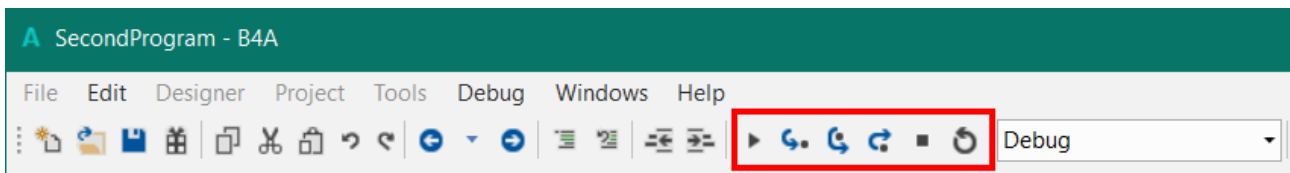
B4A, B4J



B4i, only Debug

6.1.1.1 Debug Toolbar

The debug toolbar is at the right side of the IDE 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 are shown in the SecondProgram project.

6.1.1.1.1 Run ▶ F5

Runs the program,

If the program is stopped at a breakpoint the program runs until the next breakpoint or completes running.

6.1.1.1.2 Step In F8

The debugger executes the code step by step.

```

29 Sub Activity_Create(FirstTime As Boolean)
30     Activity.LoadLayout("Main")
31     New
32 End Sub

```

In the SecondProgram project we set a Breakpoint at line 32 New.

```

29 Sub Activity_Create(FirstTime As Boolean)
30     Activity.LoadLayout("Main")
31     New
32 End Sub


```

We run the program, it will stop executing at line 31 New.

```

42 Sub New
43     Number1 = Rnd(1, 10) ' Generates
44     Number2 = Rnd(1, 10) ' Generates
45     lblNumber1.Text = Number1 ' Displays N
46     lblNumber2.Text = Number2 ' Displays N
47     lblComments.Text = "Enter the result"
48     lblComments.Color = Colors.RGB(255,235
49     lblResult.Text = "" ' Sets lblRe
50     btn0.Visible = False
51 End Sub


```

Click on  .
The debugger executes the next line, Sub New in this case.

```

42 Sub New
43     Number1 = Rnd(1, 10) ' Generates a
44     Number2 = Rnd(1, 10) ' Generates a
45     lblNumber1.Text = Number1 ' Displays Nu


```

Click once more on  .
The debugger executes the next line, Number1 =...

```

42 Sub New
43     Number1 = Rnd(1, 10) ' Generates a
44     Number2 = Rnd(1, 10) ' Generates a
45     lblNumber1.Text = Number1 ' Displays Nu

```

Click once more on  .
The debugger executes the next line, Number2 =...

6.1.1.1.3 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.

```

29 Sub Activity_Create(FirstTime As Boolean)
30     Activity.LoadLayout("Main")
31     New
32 End Sub

```

In the SecondProgram project we set a Breakpoint at line 31 New.

```

29 Sub Activity_Create(FirstTime As Boolean)
30     Activity.LoadLayout("Main")
31     New
32 End Sub

```

We run the program, it will stop executing at line 31 New.

```

29 Sub Activity_Create(FirstTime As Boolean)
30     Activity.LoadLayout("Main")
31     New
32 End Sub

```

Click on .

The debugger executes the code in New and jumps directly to the next line which is End Sub of Activity_Create.

6.1.1.1.4 Step Out F10

If the current line is in a subroutine the debugger finishes executing the rest of the code and jumps to the next line after the subs' calling line.

```

29 Sub Activity_Create(FirstTime As Boolean)
30     Activity.LoadLayout("Main")
31     New
32 End Sub

```

In the SecondProgram project we set a Breakpoint at line 32 New.

```

29 Sub Activity_Create(FirstTime As Boolean)
30     Activity.LoadLayout("Main")
31     New
32 End Sub


```

We run the program, it will stop executing at line 32 New.

```

42 Sub New
43     Number1 = Rnd(1, 10) ' Generates a
44     Number2 = Rnd(1, 10) ' Generates a
45     lblNumber1.Text = Number1 ' Displays Nu

```

We go step by step with  to a line in the subroutine.

```

29 Sub Activity_Create(FirstTime As Boolean)
30     Activity.LoadLayout("Main")
31     New
32 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 Activity_Create.

6.1.1.1.5 Stop ■

Stops the program and leaves the Rapid Debugger.

6.1.1.1.6 Restart 🔄 **F11**

Restarts the program remaining in the Rapid Debugger.

Executes:

B4A Process_Globals, Globals, Activity_Create and reloads the layout.

B4i Process_Globals,

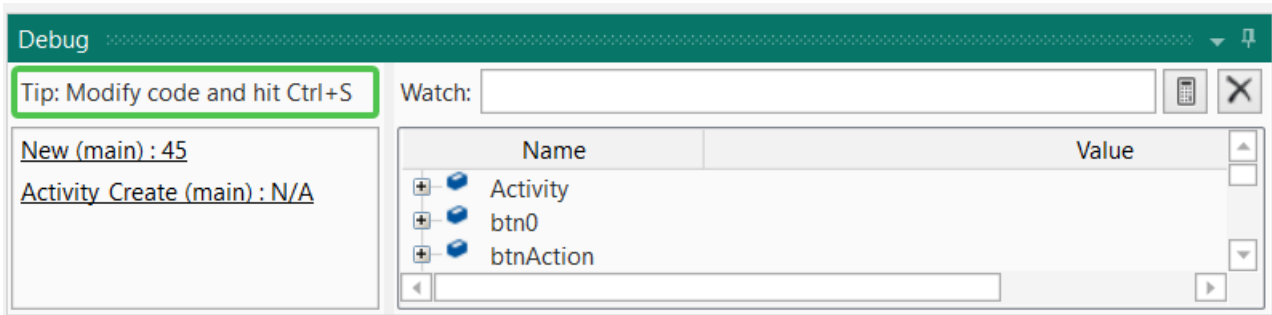
B4J Process_Globals,

This is useful if you changed a layout file.

It is different from  explained in the next chapter.

Code changed
Hit Ctrl+S to update.

6.1.2 Debug window



In the debug window we have (example with the SecondProgram, and a breakpoint in line 45:

6.1.2.1 The status button

Tip: Modify code and hit Ctrl+S

Shows that the program is running, the button border is green.

Code changed
Hit Ctrl+S to update.

When you change the code the button border changes to red.

To update the code click on the button or hit Ctrl + S.

6.1.2.2 The breakpoint window

New (main) : 45
Activity Create (main) : N/A

The breakpoint window shows where the program has stopped.

```

42 Sub New
43     Number1 = Rnd(1, 10) ' Generates
44     Number2 = Rnd(1, 10) ' Generates
45     lblNumber1.Text = Number1 ' Displays
46     lblNumber2.Text = Number2 ' Displays
47     lblComments.Text = "Enter the result"
48     lblComments.Color = Colors.RGB(255,231,208)
49     lblResult.Text = "" ' Sets lblR
50     btn0.Visible = False
51 End Sub

```

New (main) : 45

The program stopped in line 45, in routine New in the main module.

```

29 Sub Activity_Create(FirstTime As Boolean)
30     Activity.LoadLayout("Main")
31     New
32 End Sub

```

AppStart (main) : N/A

The calling routine is AppStart, and the calling line is not shown.

New (main) : 45
Activity Create (main) : N/A

When you click on one of the lines the cursor jumps to that line.

6.1.2.3 The Watch window



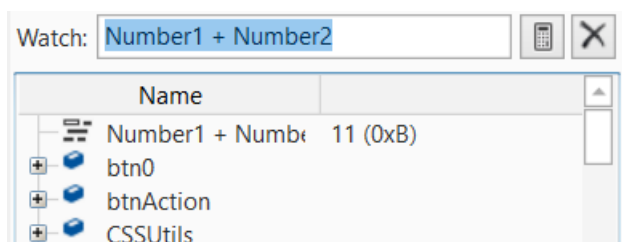
The Watch window allows to check more complex functions for testing and debugging.

```

42 Sub New
43   Number1 = Rnd(1, 10)
44   Log(Number1)
45   Number2 = Rnd(1, 10)
46   Log(Number1)
47   lblNumber1.Text = Number1
48   lblNumber2.Text = Number2
49   lblComments.Text = "Enter
  
```


In the SecondProgram code add two Log lines and set a breakpoint in line 47.

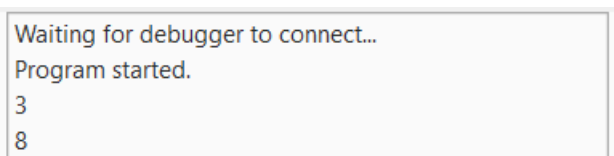
Run the program.



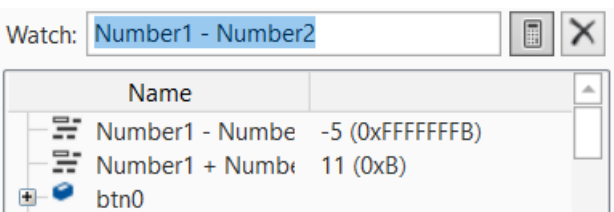
In the Add Watch field enter:

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.

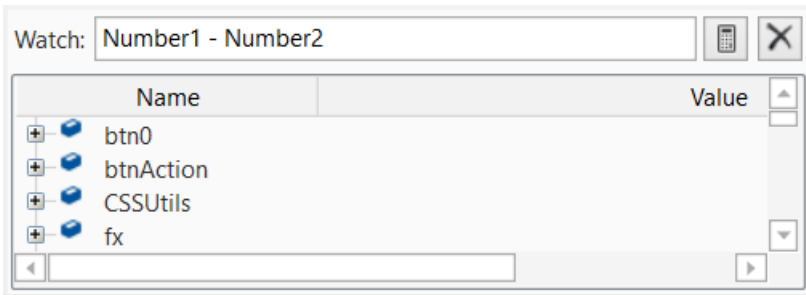


You can enter a new watch line
Number1 + Number2
and show it.


Click on  to remove the watch functions. This removes all the functions.

We could, of course, also have done this test with a Log.

6.1.2.4 The object window

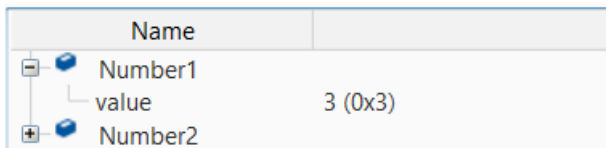


Shows all variables and objects in the list ordered by alphabetical order.

Click on  to show the details of the object:

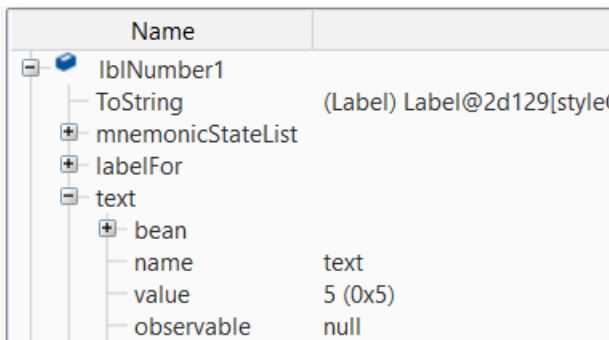
Examples:

- Number1

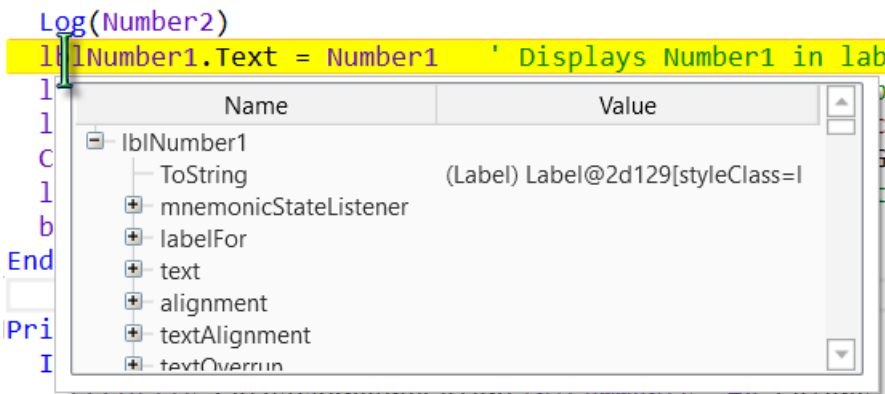


Shows the current value (3).

- lblNumber1

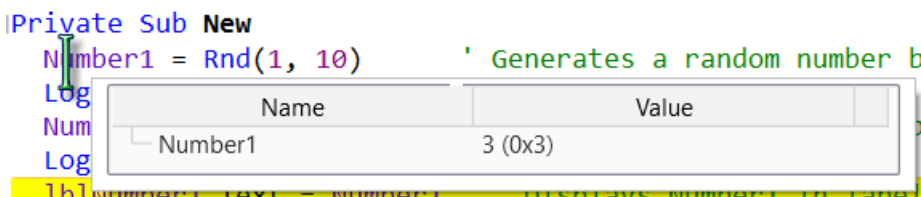


Shows all properties of the object, a Label in the example.



You get the same information when you hover over the object in the code:

lblNumber1



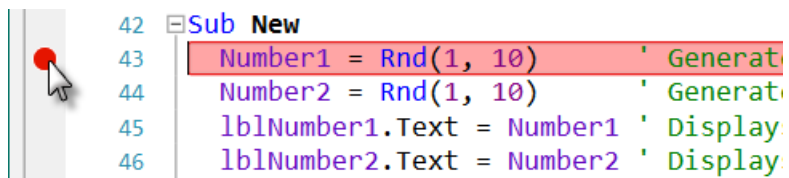
Number1

6.1.2.5 Breakpoints

One important feature to make debugging easier are breakpoints. You can set breakpoint almost wherever you want in the code.

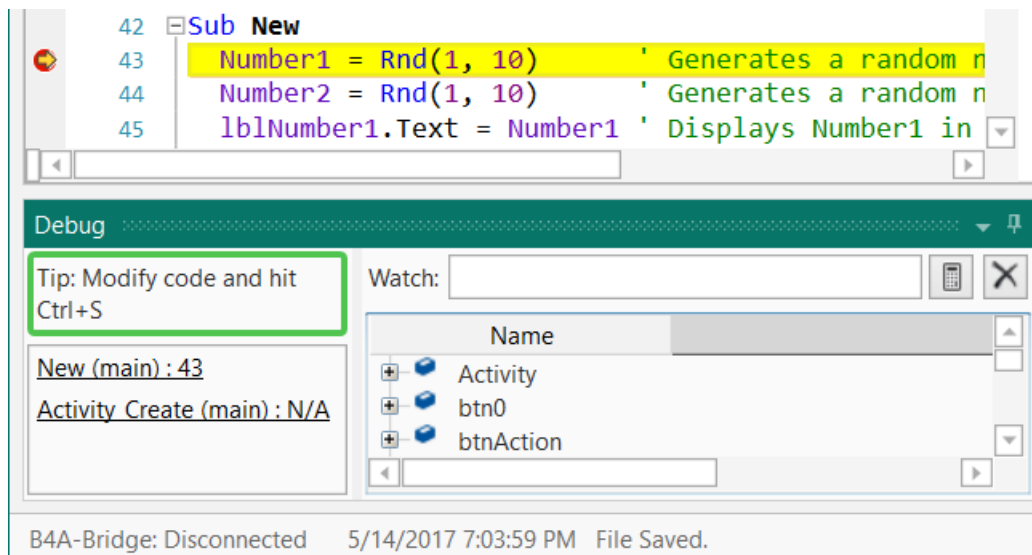
Breakpoints, in Process_Globals are ignored.

Clicking on a line in the left margin adds a breakpoint. When the program is running it stops at the first encountered breakpoint.



```
42 Sub New
43   Number1 = Rnd(1, 10) ' Generates a random number
44   Number2 = Rnd(1, 10) ' Generates a random number
45   lblNumber1.Text = Number1 ' Displays Number1 in the label
46   lblNumber2.Text = Number2 ' Displays Number2 in the label
```

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.

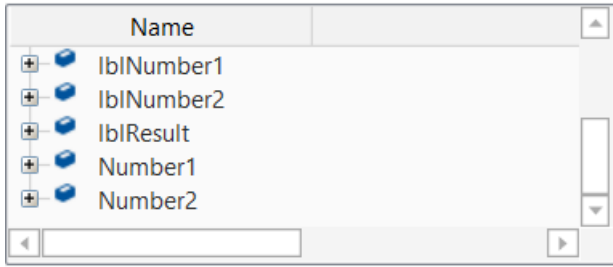
```

41
42 Sub New
43   Number1 = Rnd(1, 10)
44   Number2 = Rnd(1, 10)
45   lblNumber1.Text = Number1

```

Example with the SecondProgram:

Set a breakpoint in line 43 and run the program.




In the variable window look at Number1 and Number2:

```

42 Sub New
43   Number1 = Rnd(1, 10)
44   Number2 = Rnd(1, 10)

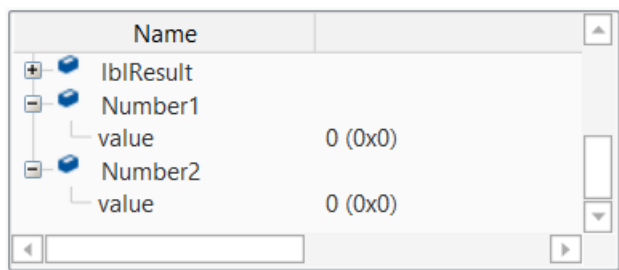
```


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.

Click on .

The program jumps to the next line.



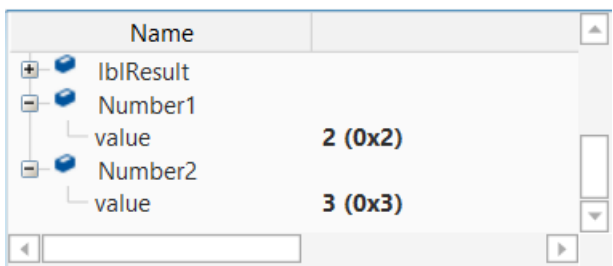
Click on .

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.

Click on .

You see that the value of Number2 has changed.

The best way to learn debugging is testing, testing and testing!

6.1.2.6 With Logs

Example with the SecondProgram.

```

42 Sub New
43   Number1 = Rnd(1, 10)
44   Log(Number1)
45   Number2 = Rnd(1, 10)
46   Log(Number2)
47   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.

```

42 Sub New
43   Number1 = Rnd(1, 10)
44   Log(Number1)
45   Number2 = Rnd(1, 10)
46   Log(Number2)
47   lblNumber1.Text = Number1

```

Run the program, it stops at line 43.

```

** Activity (main) Create, isFirst = true **

```

Filter Connect Clear List Permissions


M... Fil... Li... L... Fi... Q...

In the Log Tab we see at the moment only Waiting for debugger to connect... and `** Activity (main) Create, isFirst = true **` telling that the program has started.

```

42 Sub New
43   Number1 = Rnd(1, 10)
44   Log(Number1)
45   Number2 = Rnd(1, 10)
46   Log(Number2)
47   lblNumber1.Text = Number1
48   lblNumber2.Text = Number2

```


Click four times on  till the program reaches line 47.

```

** Activity (main) Create, isFirst = true **
2
2

```

In the Log Tab we see the values of the two variables.

Click on  to run to the end. Nothing new is displayed

```

** Activity (main) Create, isFirst = true **
1
5
** Activity (main) Resume **
6
2

```

When you are using the program the two new values will be shown every time the program runs the New routine.

6.1.2.7 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 47.

```

42 Sub New
43   Number1 = Rnd(1, 10)
44   Log(Number1)
45   Number2 = Rnd(1, 10)
46   Log(Number2)
47   lblNumber1.Text = Number1
48   lblNumber2.Text = Number2

```

Run the program till it stops at the breakpoint.

```

42 Private Sub New
43   Number1 = Rnd(1, 20)
44   Log(Number1)
45   Number2 = Rnd(1, 20)
46   Log(Number2)
47   lblNumber1.Text = Number1
48   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.

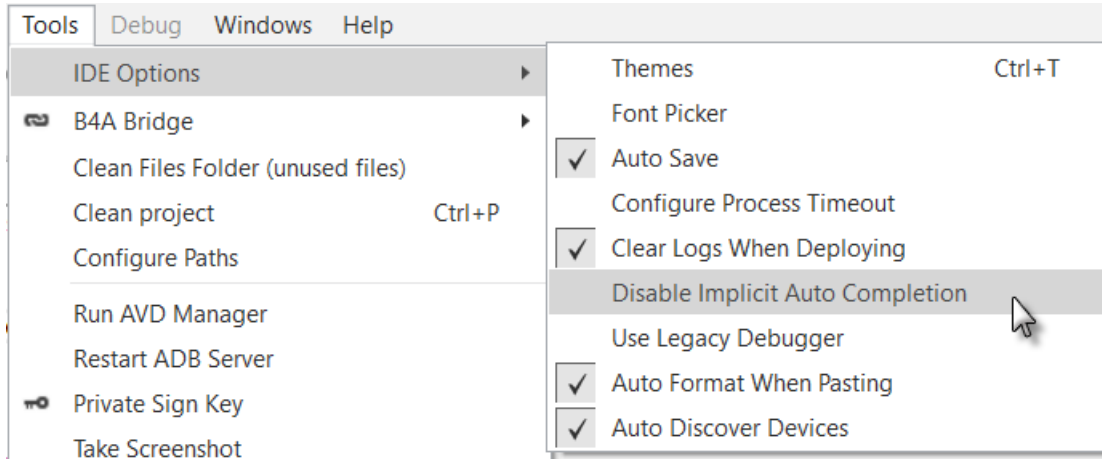
```

** Activity (main) Create, isFirst = true **
2
6
5
17

```

6.1.3 Debug (legacy) mode B4A only

In some cases the legacy Debugger can be useful, can select it in the Tools menu under IDE options.

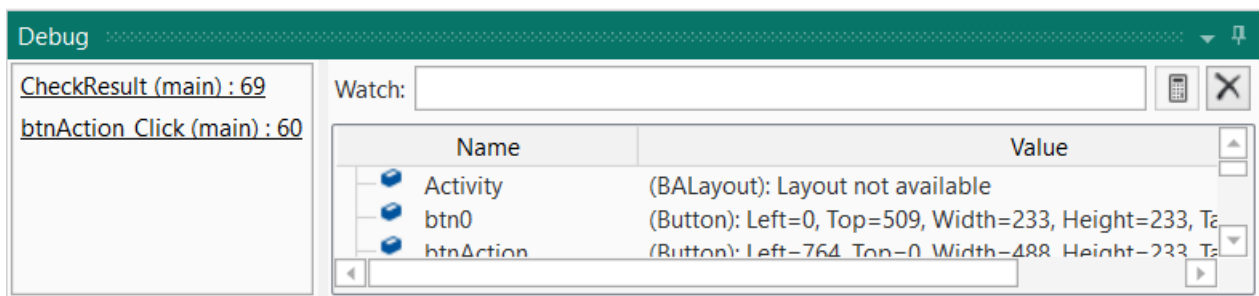


Debug(legacy): When this option is selected then the compiled code will contain debugging code. The debugging code allows the IDE to connect to the program and inspect it while it runs. When the program starts, it will wait for up to 10 seconds for the IDE to connect. Usually the IDE will connect immediately. However if you run your program manually from the phone you will see it waiting.

The name of the compiled APK file will end with `_DEBUG.apk`. You should not distribute this apk file as it contains the debugging code which adds a significant overhead.

To distribute files you must select the *Release* or the *Release (obfuscated)* option.

When we run the program with the Debug (legacy) option and setting a Breakpoint, the IDE will open the debugger module at the bottom of the screen:



The navigation buttons in the Toolbar are enabled
These work similar to the Debug (rapid) mode.



6.2 Debugging B4R

Debugging is an important part when developing.

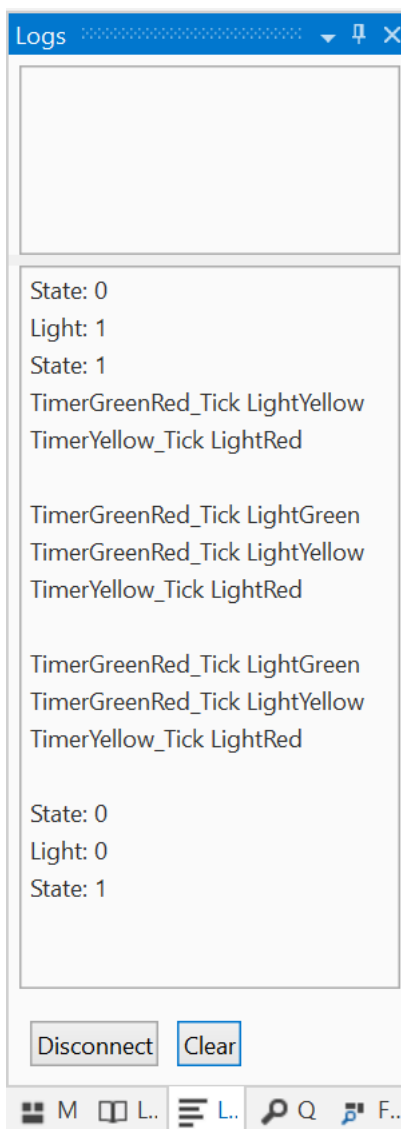
In B4R there is no Debug mode like in the other B4x languages.

Debugging can only be done with [Logs](#).

The Logs tab in the right pane 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.

6.2.1 Debug example with the TrafficLight project

In the TrafficLight project I added several Log statements which show the evolution of the program.



When we run the program the Logs is empty.

Then:

```
We press the button           > State: 0
The light is set to ON        > Light: 1 red light ON
We release the button         > State: 1
```

The logs of the timers:

In TimerGreenRed_Tick change from red to green:

```
> TimerGreenRed_Tick LightGreen
```

We set the light to yellow and enable TimerYellow:

```
> TimerGreenRed_Tick LightYellow
```

In TimerYellow_Tick we set yellow OFF and red ON

```
> TimerYellow_Tick LightRed
```

End of first cycle

```
Begin of next cycle: > TimerGreenRed_Tick LightGreen
```

```
We press the button           > State: 0
```

```
The light is set to OFF       > Light: 0
```

```
We release the button         > State: 1
```