B4x Booklets





B4x I D E

**I**ntegrated **D**evelopment **E**nvironment

Copyright: © 2017 Anywhere Software Edition 1.0

Last update : 2017.05.25

[1 General 5](#_Toc482888428)

[2 Menu and Toolbar 6](#_Toc482888429)

[2.1 Toolbar 6](#_Toc482888430)

[2.2 File menu 7](#_Toc482888431)

[2.3 Edit menu 7](#_Toc482888432)

[2.4 Project menu 8](#_Toc482888433)

[2.4.1 Add a new module 9](#_Toc482888434)

[2.5 Tools menu 10](#_Toc482888435)

[2.5.1 IDE Options 11](#_Toc482888436)

[2.5.1.1 Themes 12](#_Toc482888437)

[2.5.1.2 Font Picker 13](#_Toc482888438)

[2.5.1.2.1 Word wrap 13](#_Toc482888439)

[2.5.1.2.2 Configure Process Timeout 13](#_Toc482888440)

[2.5.1.2.3 Disable Implicit Auto Completion 13](#_Toc482888441)

[2.5.2 Take Screenshot B4A only 14](#_Toc482888442)

[2.5.3 Create Video B4A only 15](#_Toc482888443)

[2.5.4 Clean Files Folder (unused files) 16](#_Toc482888444)

[2.5.5 Clean Project 16](#_Toc482888445)

[2.6 Right click menu 17](#_Toc482888446)

[2.7 Compiler mode 18](#_Toc482888447)

[2.7.1 B4A and B4J 19](#_Toc482888448)

[2.7.1.1 Release and Release (obfuscated) modes B4A and B4J 19](#_Toc482888449)

[2.7.2 B4i 20](#_Toc482888450)

[2.7.3 B4R 20](#_Toc482888451)

[3 Code area 21](#_Toc482888452)

[3.1 Split the code area 21](#_Toc482888453)

[3.2 Code header Project Attributes / Activity Attributes 22](#_Toc482888454)

[3.2.1 B4A 22](#_Toc482888455)

[3.2.1.1 Project Attributes 22](#_Toc482888456)

[3.2.1.2 Activity Attributes 22](#_Toc482888457)

[3.2.1.3 Service Attributes 22](#_Toc482888458)

[3.2.2 B4i 24](#_Toc482888459)

[3.2.3 B4J 24](#_Toc482888460)

[3.2.4 B4R 24](#_Toc482888461)

[3.3 Undo – Redo   25](#_Toc482888462)

[3.4 Collapse a subroutine 25](#_Toc482888463)

[3.5 Collapse a Region 26](#_Toc482888464)

[3.6 Collapse the entire code 27](#_Toc482888465)

[3.7 Toggle Outlining Ctrl + 0 28](#_Toc482888466)

[3.8 Copy a selected bloc of text 29](#_Toc482888467)

[3.9 Move line(s) up / down Alt + Up / Alt + Down 29](#_Toc482888468)

[3.10 Find / Replace 30](#_Toc482888469)

[3.11 Commenting and uncommenting code   31](#_Toc482888470)

[3.12 Bookmarks  32](#_Toc482888471)

[3.13 Indentation   33](#_Toc482888472)

[3.14 Auto format 35](#_Toc482888473)

[3.15 Documentation tool tips while hovering over code elements 36](#_Toc482888474)

[3.16 Auto Completion 37](#_Toc482888475)

[3.17 Built in documentation 41](#_Toc482888476)

[3.17.1 Copy code examples 42](#_Toc482888477)

[3.18 Jump to a subroutine 43](#_Toc482888478)

[3.19 Highlighting occurrences of words 44](#_Toc482888479)

[3.20 Breakpoints 45](#_Toc482888480)

[3.21 Color Picker  47](#_Toc482888481)

[3.22 Icon Picker  48](#_Toc482888482)

[3.23 Colors in the left side 49](#_Toc482888483)

[3.24 URLs in comments and strings are ctrl-clickable 50](#_Toc482888484)

[4 Tabs 51](#_Toc482888485)

[4.1 Floating Tab windows 52](#_Toc482888486)

[4.2 Float  53](#_Toc482888487)

[4.3 Auto Hide  56](#_Toc482888488)

[4.4 Close 58](#_Toc482888489)

[4.5 Modules and subroutine lists  59](#_Toc482888490)

[4.5.1 Find Sub / Module / Line number (Ctrl + E) 60](#_Toc482888491)

[4.6 Files Manager  B4A, B4i and B4J only 61](#_Toc482888492)

[4.7 Logs  63](#_Toc482888493)

[4.7.1 Compile Warnings 64](#_Toc482888494)

[4.7.1.1 Ignoring warnings 65](#_Toc482888495)

[4.7.1.2 List of warnings 66](#_Toc482888496)

[4.8 Libraries Manager  72](#_Toc482888497)

[4.9 Quick Search  73](#_Toc482888498)

[4.10 Find All References (F7)  75](#_Toc482888499)

[5 Navigation in the IDE 76](#_Toc482888500)

[5.1 Alt + Left / Alt + Right Move backwards and forwards 76](#_Toc482888501)

[5.2 Alt + N Navigation stack menu  76](#_Toc482888502)

[5.3 Split the screen 76](#_Toc482888503)

[5.4 Multiple windows 77](#_Toc482888504)

[5.5 Ctrl + E Search for sub or module 77](#_Toc482888505)

[5.6 Ctrl + Click on any sub or variable 77](#_Toc482888506)

[5.7 F7 - Find all references 77](#_Toc482888507)

[5.8 Ctrl + F Quick Search 77](#_Toc482888508)

[6 Debugging B4A, B4i, B4J 78](#_Toc482888509)

[6.1 B4A, B4i, B4J 78](#_Toc482888510)

[6.1.1 Debug mode 79](#_Toc482888511)

[6.1.1.1 Debug Toolbar 79](#_Toc482888512)

[6.1.1.1.1 Run  F5 79](#_Toc482888513)

[6.1.1.1.2 Step In  F8 80](#_Toc482888514)

[6.1.1.1.3 Step Over  F9 81](#_Toc482888515)

[6.1.1.1.4 Step Out  F10 81](#_Toc482888516)

[6.1.1.1.5 Stop  82](#_Toc482888517)

[6.1.1.1.6 Restart  F11 82](#_Toc482888518)

[6.1.2 Debug window 83](#_Toc482888519)

[6.1.2.1 The status button 83](#_Toc482888520)

[6.1.2.2 The breakpoint window 83](#_Toc482888521)

[6.1.2.3 The Watch window 84](#_Toc482888522)

[6.1.2.4 The object window 85](#_Toc482888523)

[6.1.2.5 Breakpoints 86](#_Toc482888524)

[6.1.2.6 With Logs 88](#_Toc482888525)

[6.1.2.7 Modifying code in the Debugger 89](#_Toc482888526)

[6.1.3 Debug (legacy) mode B4A only 90](#_Toc482888527)

[6.2 Debugging B4R 91](#_Toc482888528)

[6.2.1 Debug example with the TrafficLight project 91](#_Toc482888529)

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

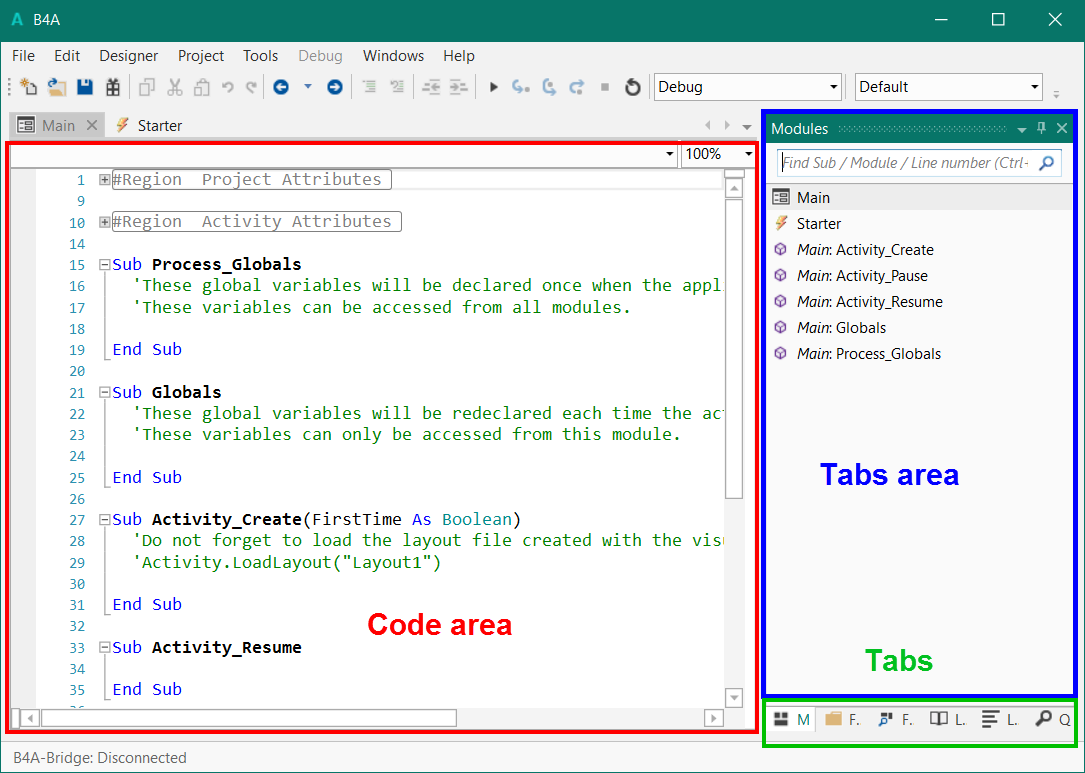
B4J version 5.51

B4R version 1.80

# General

The **I**ntegrated **D**evelopment **E**nvironment.

When you run the IDE you will get a form like the image below.



All the images are made with the B4A IDE.

The IDEs of the other products look similar with different themes.

Specific images are shown if needed.

You see 3 main areas:

* Code area The code editor
* Tab area The content of this area depends on the selected Tab.
* [Tabs](#_Tabs) Tabs for different settings.

# Menu and Toolbar





## 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]](#_Commenting_and_uncommenting).

 [Block Uncomment [Ctrl + W]](#_Commenting_and_uncommenting).

 [Decrease the indentation of the selected lines](#_Indentation).

 [Increase the indentation of the selected lines](#_Indentation).

Runs the compiler [F5].



Step In [F8].

 Step Over [F9].

 Step Out [F10]. These 5 functions are active only when the debugger is active.

 Stop.

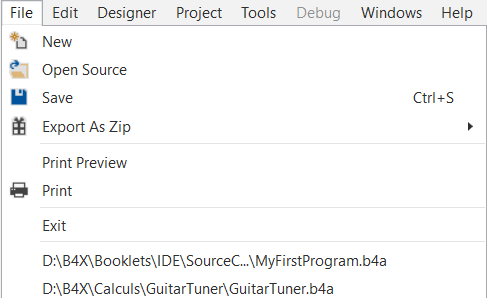
Restart [F11].

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



Conditional compiling options.

## 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.

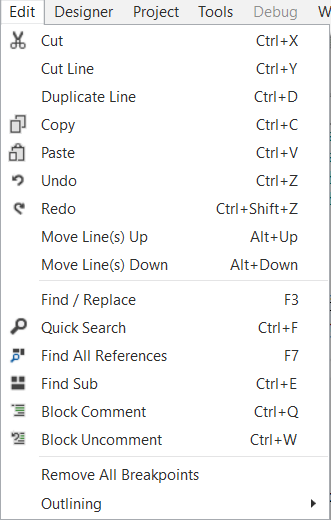
**Print Preview** Preview of the print.

**Print** Prints the whole code of the selected Module.

**Exit** Leaves the IDE.

List of last loaded programs.

## 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](#_Find_/_Replace) function.

**Quick Search** [Quick Search](#_Quick_Search)

**Find All References** [Find All References](#_Find_All_References)

**Find Sub** [Find Sub](#_Find_Sub_/)

**Block Comment  
Block Uncomment**

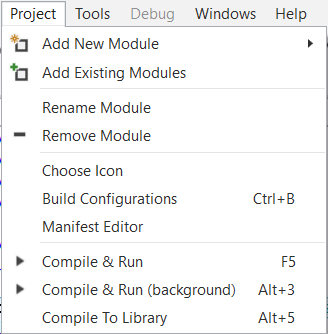
[Comment / Uncomment the selected lines](#_Commenting_and_uncommenting).

**Remove All Breakpoints** [Breakpoints](#_Breakpoints).

**Outlining** [Collapse the whole code](#_Collapse_the_whole).

## Project menu

**B4A**



Adds a new [module](#_Modules)

Adds an existing [module](#_Modules)

Changes the [module](#_Modules) name

Removes the current [module](#_Modules)

Chooses an icon for the program.

Changes the package name.

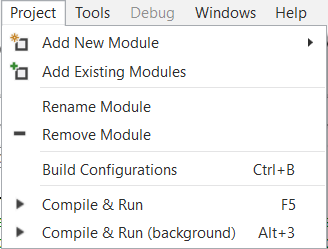
Runs the Manifest Editor.

Compile and run the project.

Compile and run the project in the background.

Compile to a library.

**B4i, B4R**



Adds a new [module](#_Modules)

Adds an existing [module](#_Modules)

Changes the [module](#_Modules) name

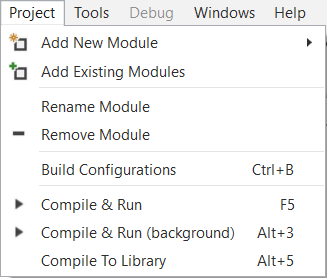
Removes the current [module](#_Modules)

Changes the package name.

Compile and run the project.

Compile and run the project in the background.

**B4J**



Adds a new [module](#_Modules)

Adds an existing [module](#_Modules)

Changes the [module](#_Modules) name

Removes the current [module](#_Modules)

Changes the package name.

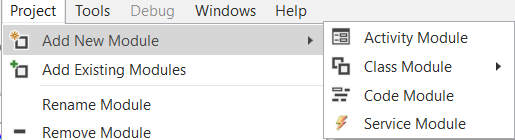
Compile and run the project.

Compile and run the project in the background.

Compile to a library.

### Add a new module

**B4A**



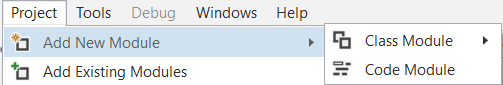
Activity module

Class module

Code module

Service module

**B4i, B4J**



Class module

Code module

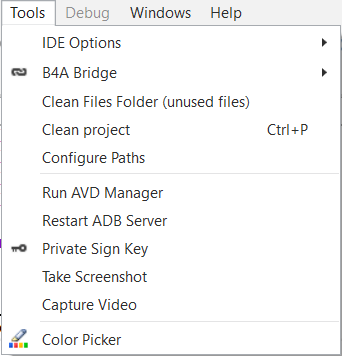
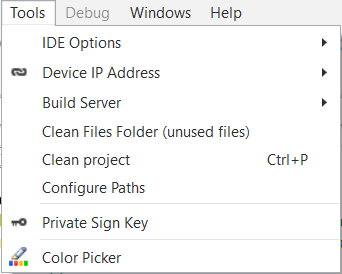
**B4R**



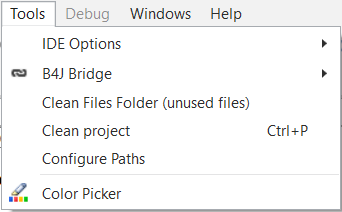
Only Code module

## Tools menu

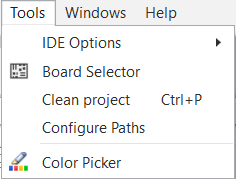
**B4A B4i**



**B4J**



**B4R**



IDE Options see below

[B4A Bridge](#_B4A_Bridge_1), connection with Bluetooth or Wifi B4A

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

[Clean Project](#_Clean_Project) All

[Configure Paths](#_Install_and_configure) All

[Run AVD Manager](#_Launch_an_Emulator) B4A

[Take Screenshot](#_Take_Screenshot) B4A

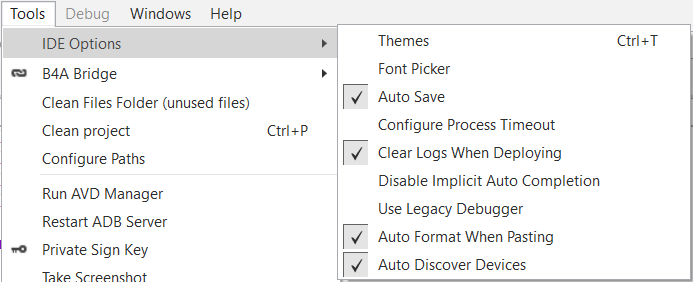
[Capture a video](#_Create_Video) B4A

Show the [Color Picker](#_Color_Picker) All

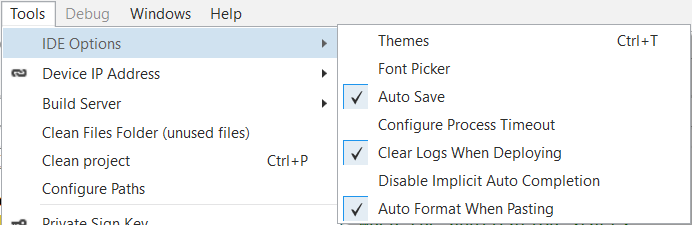
Board Selector B4R

### IDE Options

**B4A**



**B4i, B4J, B4R**



**All**

[Themes](#_Themes).

[Font Picker](#_Font_Picker).

Auto Save Saves the program every time you run it.

[Configure Process Timeout](#_Configure_Process_Timeout)

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

[Disable Implicit Auto Completion](#_Disable_Implicit_Auto).

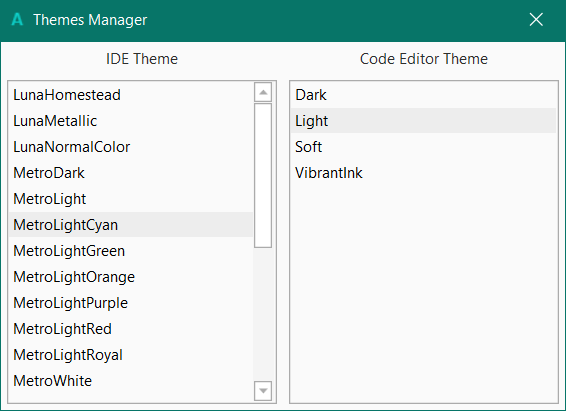
**B4A only**

[Use Legacy Debugger](#_Debug_(legacy)_mode) Use the legacy Debugger instead of the rapid Debugger.

Auto Discover Devices Detects automatically the connected devices.

#### 

#### Themes

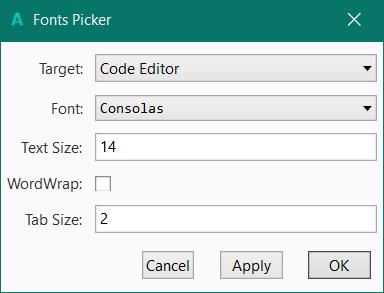


You can select different themes for the IDE.

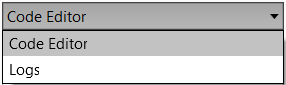
The default theme is MetroLight.

When you select one you see directly the new colors.

#### Font Picker



You can select the target Code Editior or Logs.



Different fonts.

Enter the text size.

Select WordWrap

Enter the Tab size.

##### Word wrap

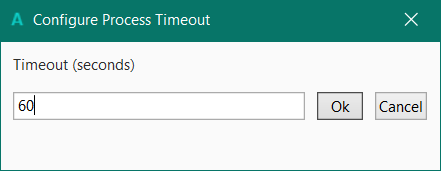


Without word wrap. The end of the line is hidden.



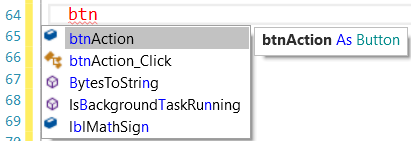
With word wrap. The end of the line is wrapped to the next line.

##### Configure Process Timeout



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

##### Disable Implicit Auto Completion



If 

is unchecked you will see a drop down list with possible words during typing.

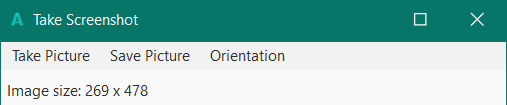
If checked  you won’t see the auto completion list.

### Take Screenshot B4A only

The function can be called from the:

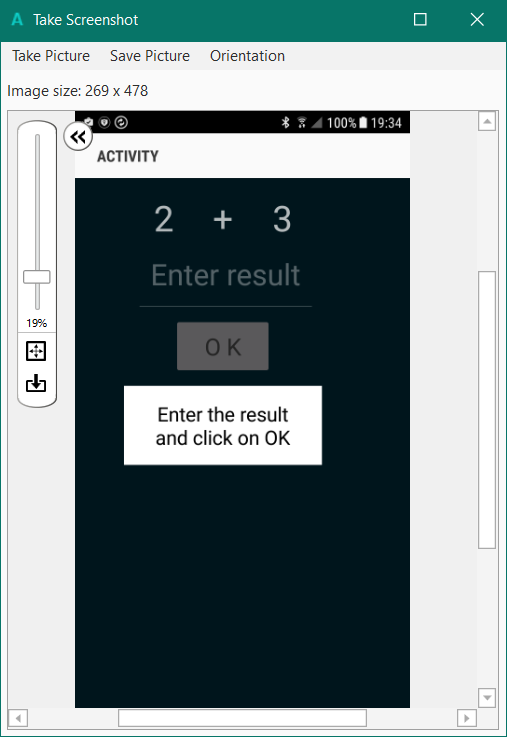
* Tools menu when the IDE is in edit mode
* Debug menu when the IDE is in debug mode

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



Clicking on 

shows this window.

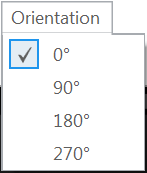


Click on  to take the screenshot picture from the device.

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

You can save the image with 

as a PNG file.



And you can change the orientation

of the picture.

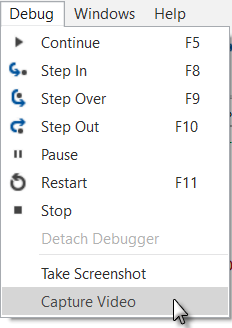


Right click on the image to copy the image to the clipboard.

### Create Video B4A only

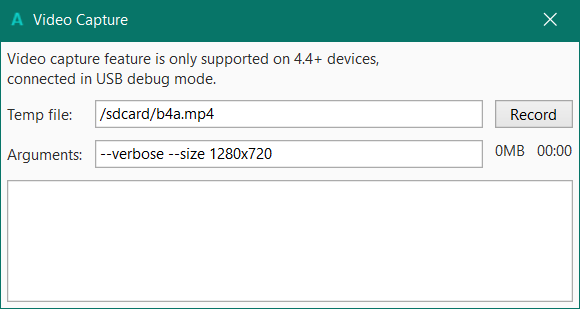
You can run your program and record a video when you use it.

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



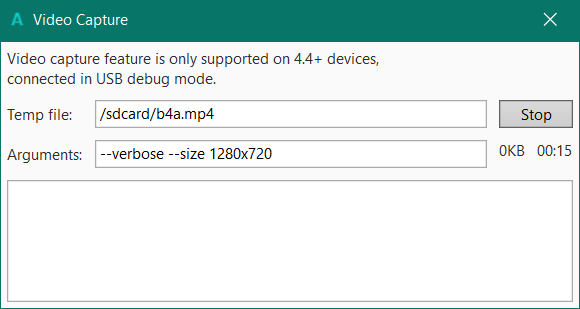
In the  menu click on .

The sceen below will be dispayed:



Click on  to begin recording.

A screen similar to this one will be dispaled:



Click on  to stop recording.

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

#### 

### Clean Files Folder (unused files)

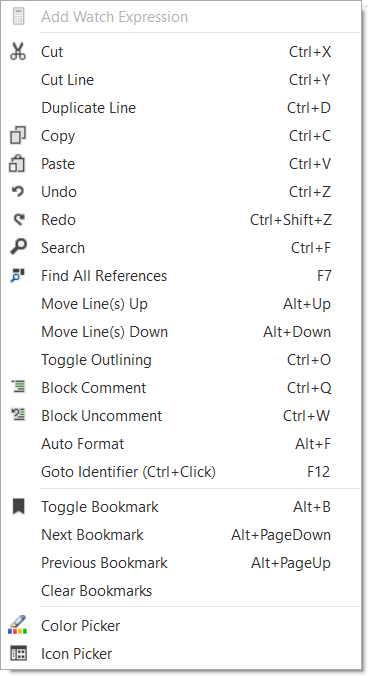
Deletes files that are located under the Files folder but are not used by the project (it will not delete any file referenced by any of the project layouts). A list of unused files will be displayed before deletion (and you may cancel the operation).

### Clean Project

Deletes all files that are generated during compilation.

## Right click menu

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



Cut

Cut Line

Diplicate Line

Copy

Paste

[Undo](#_Undo_–_Redo)

[Redo](#_Undo_–_Redo)

[Search](#_Find_All_References)

[Find All References](#_Find_All_References)

[Move Line(s) Up](#_Move_lines_up_1)

[Move Line(s) Down](#_Move_lines_up_1)

Toggle Outlining

[Block Comment](#_Commenting_and_uncommenting_1)

[Block Uncomment](#_Commenting_and_uncommenting_1)

[Auto Format](#_Auto_format_1)

[Goto identifier](#_Jump_to_a)

[Toggle Bookmark](#_Bookmarks_1)

[Previous Bookmark](#_Bookmarks_1)

[Next Bookmark](#_Bookmarks_1)

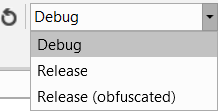
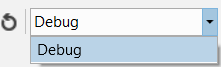
[Clear Bookmark](#_Bookmarks_1)

[Color Picker](#_Color_Picker)

[Icon Picker](#_Icon_Picker_1) Not in B4R.

## Compiler mode

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

[B4A / B4J](#_B4A_and_B4J) [B4i](#_B4i) [B4R](#_B4R)

Debugging is explained in detail in the [Debugging](#_Debugging_1) chapter.

### B4A and B4J

Compiling modes:

* [Debug](#_Debug_(rapid)_mode)
* [Release](#_Release_and_Release)
* [Release (obfuscated)](#_Release_and_Release)

#### Release and Release (obfuscated) modes B4A and B4J

**To distribute your project you must compile it with:**

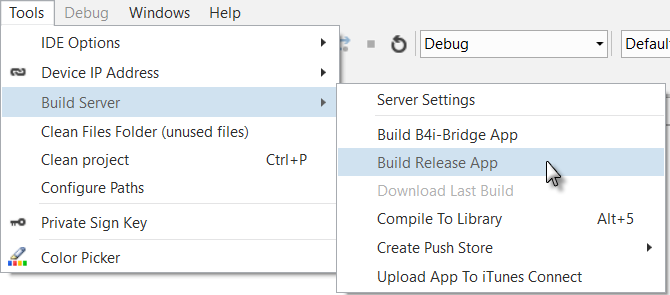
* Release  
  The debugger code will not be added to the apk file.
* Release (obfuscated)   
  The debugger code will not be added to the apk file,  
  but the program file will be modified. See below.

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.

### B4i

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

Click on  in the Tools / Build Server menu.



### B4R

Only Release mode.

# Code area

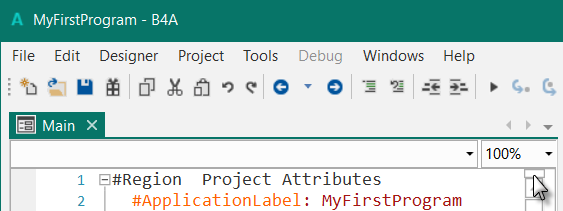
The code of the selected module is displayed in this area and can be edited.

The examples below are based on the code of the SecondProgram.

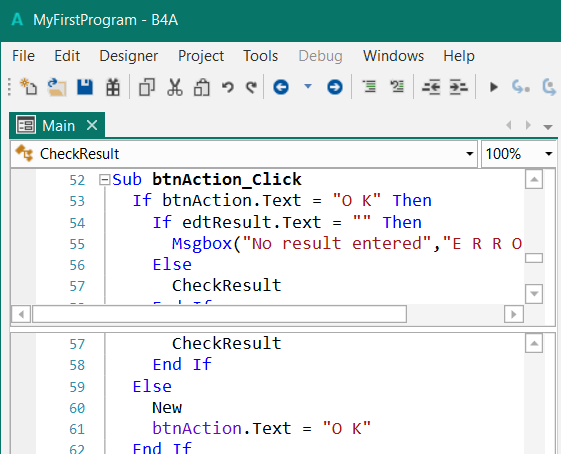
## Split the code area

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.



## Code header Project Attributes / Activity Attributes

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

### B4A

#### 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.

#### 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

#### Service Attributes

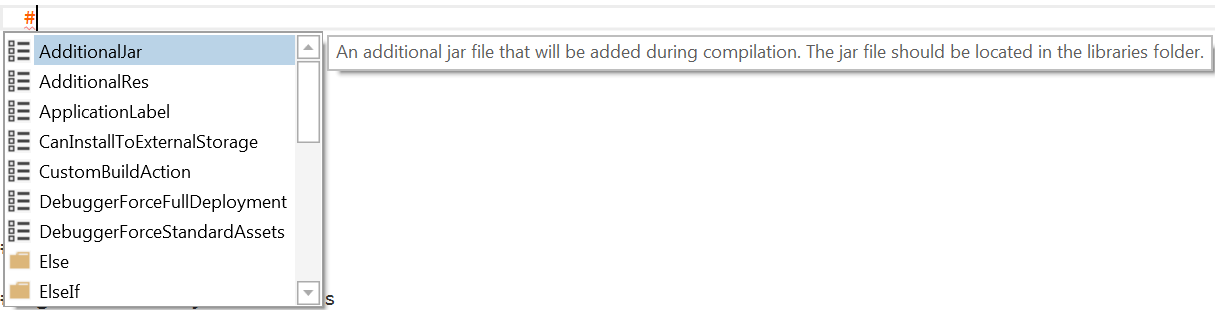
When you add a new Service you'll find the Service Attributes header.

#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

### 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

### B4J

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

#Region Project Attributes

#MainFormWidth: 600

#MainFormHeight: 600

#End Region

### B4R

Only the Attributes below. No other Attributes in modules.

#Region Project Attributes

#AutoFlushLogs: True

#CheckArrayBounds: True

#StackBufferSize: 300

#End Region

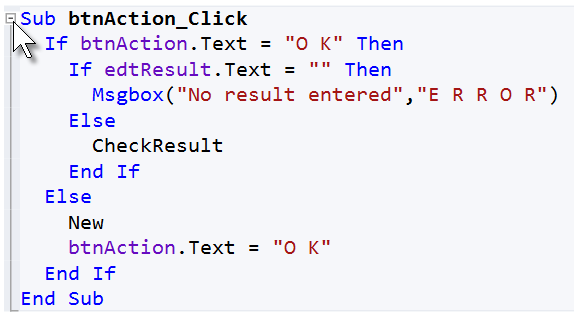
## Undo – Redo

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

Click on  to undo and on  to redo.

## Collapse a subroutine

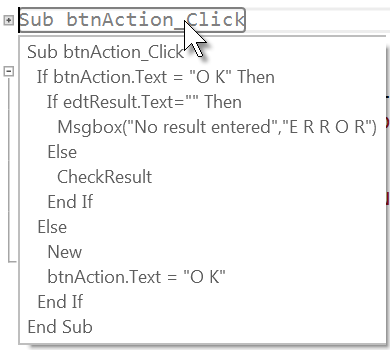
A subroutine can be collapsed to minimize the number of lines displayed.

The btnAction\_Click routine expanded.

Click on  to collapse the subroutine.

The btnAction\_Click routine collapsed.



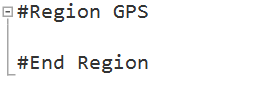


Hovering with the mouse over the collapsed routine name shows its content.

## Collapse a Region

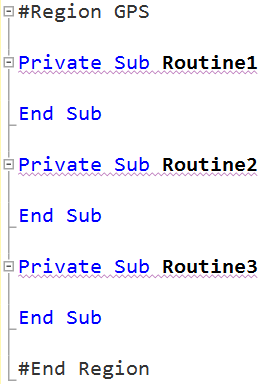
You can define ‘Regions’ in the code, which can be collapsed.

Example:

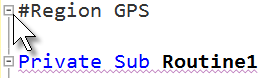


#Region GPS sets the beginning of a region and

#End Region the end



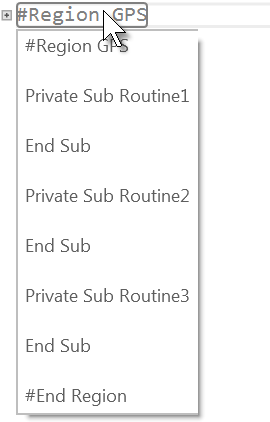
Then you can add the subroutines between the two limits.



Then click on  to collapse the whole region.

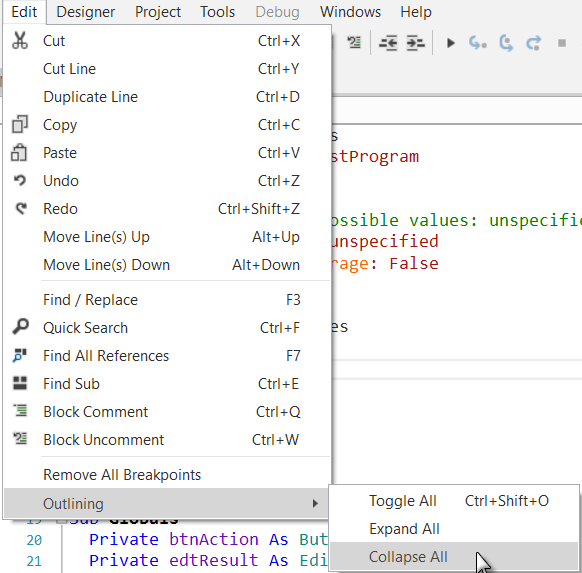


Hovering over GPS



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

## 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.

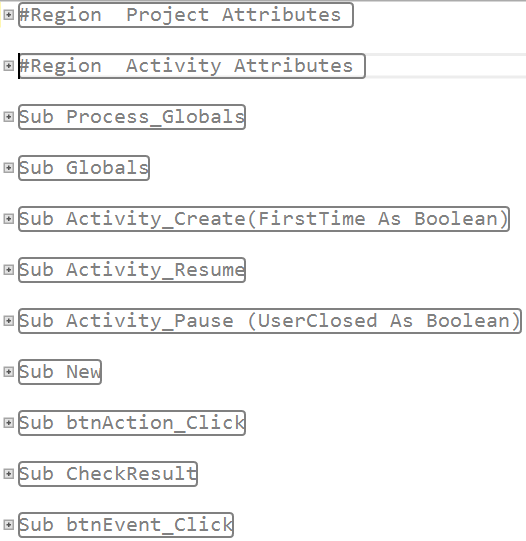
- Expand All

Expands the entire code

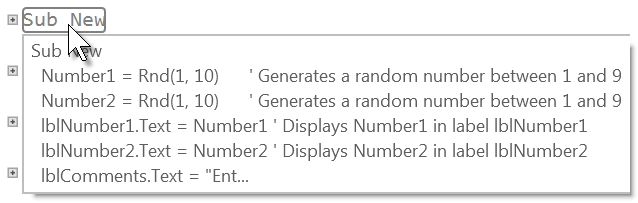
- Collapse All

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.

## Toggle Outlining Ctrl + 0

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"

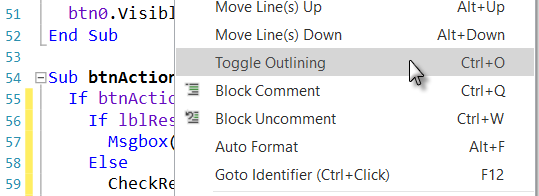
lblResult.Text = "" & Chr(0xE632)

End If

End Sub

Click insides the routine and press Ctrl + 0.

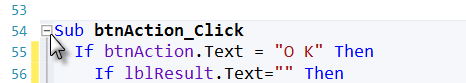
Or right click insides the routine to show the pop-up menu and click on  to collapse the routine.



And the result.



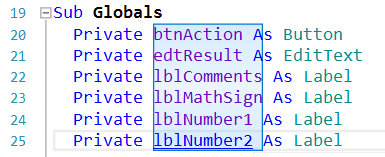
It is the same as clicking on .



## 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.



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

You can move selected lines up or down.

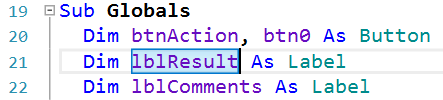
Either with Alt + Up or Alt + Down.

Or right click on the selected lines and select  or .

## Find / Replace

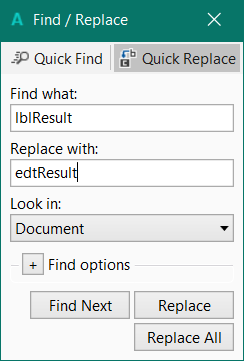
The example uses the code from the SecondProgram project.

Let’s replace lblResult by edtResult.



In the code select lblResult.

Press F3 or click on  in the  menu.

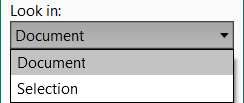


This window will be displayed

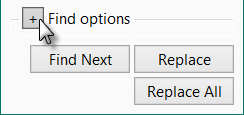
Enter edtResult in the ‘Replace with’ field.

Now, you can either:

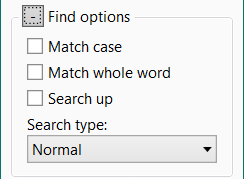
*  find the next occurrence.
*  replace the current occurrence and find the next one.
*  replace all occurrencies.



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



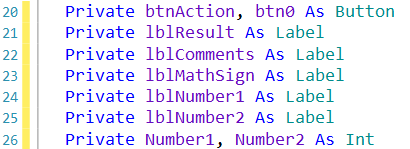
You can select Find options, click on .



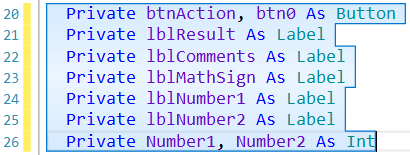
These options are self-explanatory.

## Commenting and uncommenting code

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

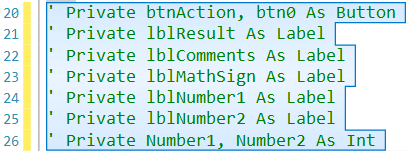


Original code



Select the code.

Click on  or Ctrl + Q.



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  or .

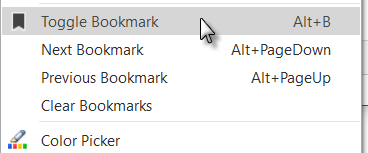
### 

## 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.

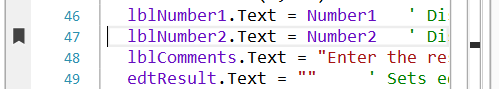
Or right click on the line where you want to set a bookmark.

You will get a pop up menu, click on



to activate or deactivate a bookmark.

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



To jump to the next bookmark press Alt + PageDown

or right click and click on 

To jump to the previous bookmark press on Alt + PageUp

or right click and click on 

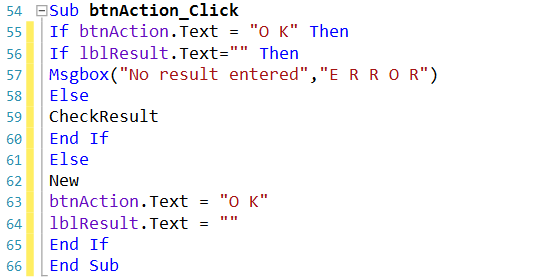
To clear all bookmarks right click and click on 

## Indentation

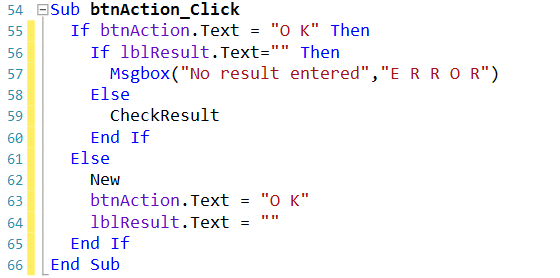
A good practice is to use indentation of code parts.

For example for subroutines, loops, structures etc.

You should also have a look at Auto Format.

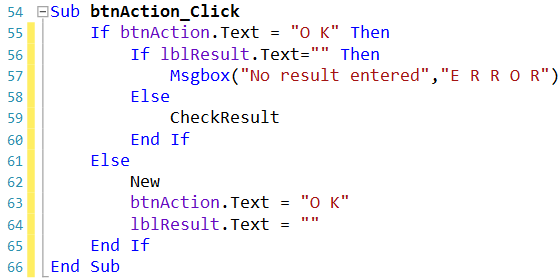


This code is difficult to read because the structure of the code is not obvious.



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.

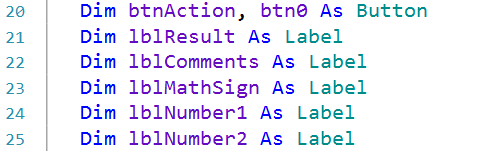


Example with an indentation of 4

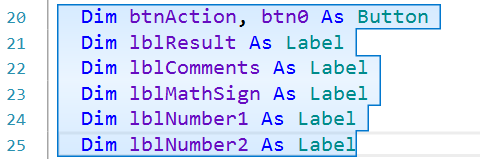
Personally,

I prefer a value of 2.

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

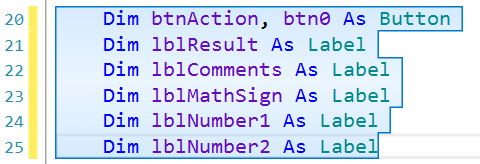


Original code.



Select the code block.

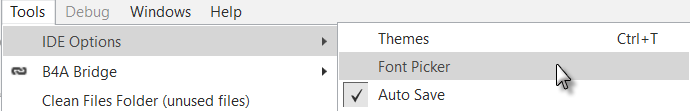
Click on .

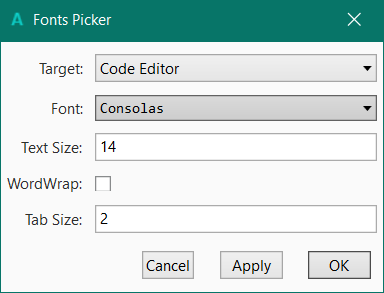
The whole block has moved one tabulation to the right.

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.



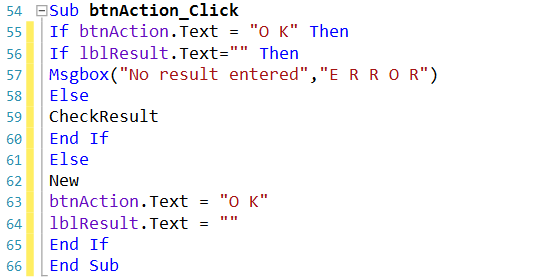


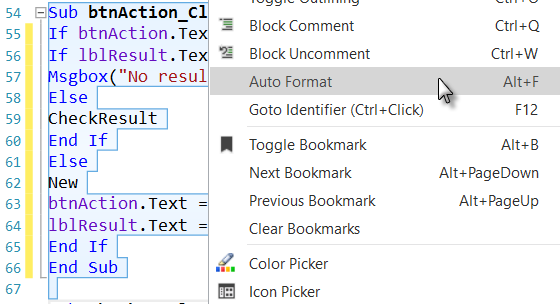
Enter the value and click on .

## Auto format

You can auto format the code.

This code is not easy to read.

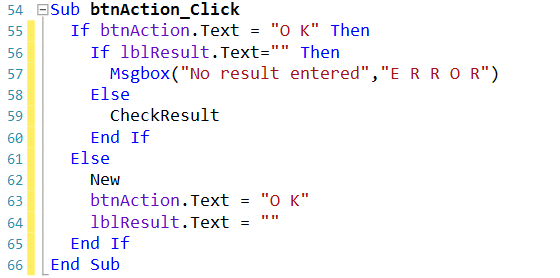




Select the code.

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

And click on .



And the result.

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

## Documentation tool tips while hovering over code elements

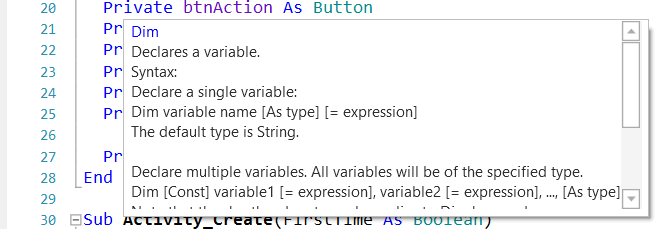
When you hover over code elements the on line help is displayed.

Examples:

Hovering over Globals:



Hovering over Private:



## Auto Completion

A very useful tool is the Auto Completion function.

Example with the SecondProgram code:



Let us write lblN.

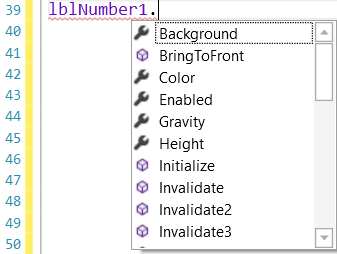
All variables, views and property names beginning with the letters already written are shown in a popup menu with the online help for the highlighted variable, view or property name.

To choose lblNumber1 press Return.

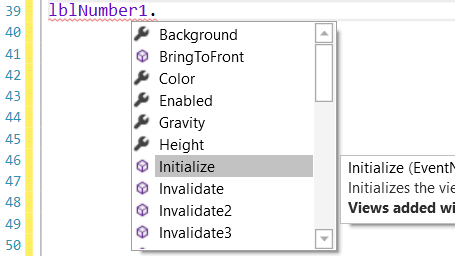


The selected name is completed.

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

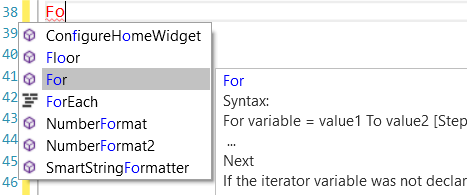
Pressing on the up / down arrows selects the previous or next item with its help.

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

Structures are also completed.

Examples:

**For / Next**

Tpye Fo

You get For with the help.

Press Return.



For is completed.

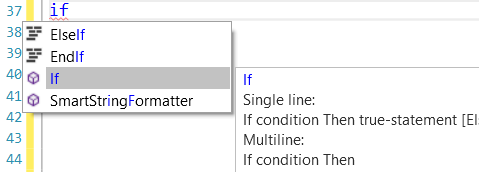
Write the rest of the instruction.

And press Return.



Next is automatically added and the cursor is in the next line idented.

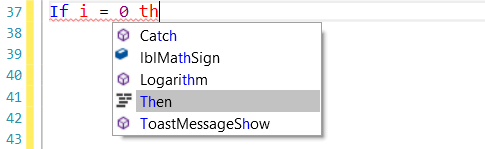
**If / Then**



Type ‘if’.

You get If with the help.

Press Return and continue typing like in the example.

After th you get Then with its help.

Press Return.

And press Return again.



End If is automatically added

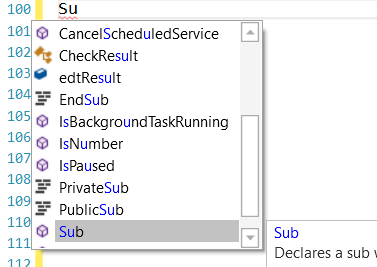
and the cursor is in the next line idented.

**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.

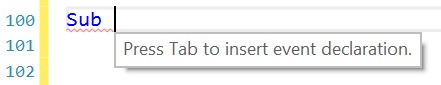
Write ’Su’ and the Auto Completion displays all keywords containing the two characters.



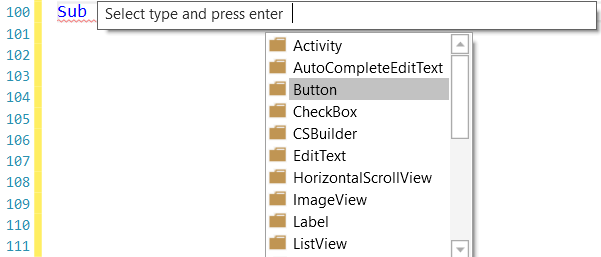
Press Return to select Sub.



Press blank.



Press Tab and select the view type, select Button.



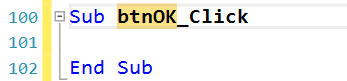
All events for a Button are displayed, select Click.



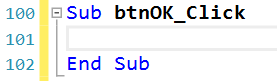
The subroutine frame is generated.



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



Press Return and the routine is ready.



## 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

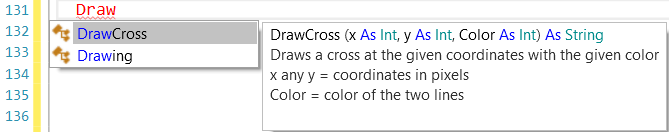
'Color = color of the two lines

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>

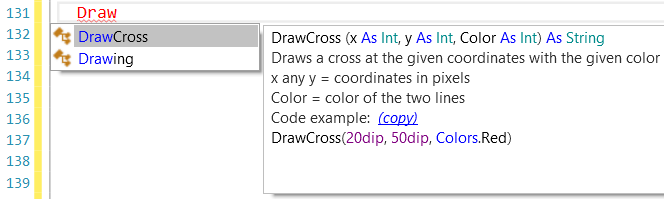
Sub **DrawCross**(x As Int, y As Int, Color As Int)

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)

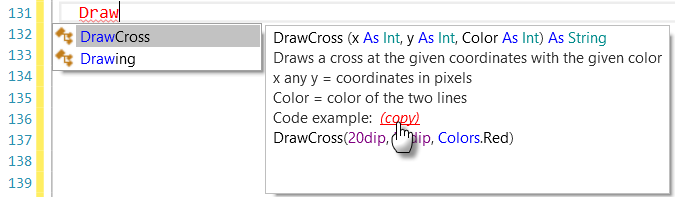
End Sub

 The code will be syntax highlighted:

### Copy code examples

You can copy the code example in your code.

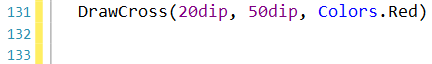
When hovering over (copy) you can copy the code example to the clipboard.



Remove Draw



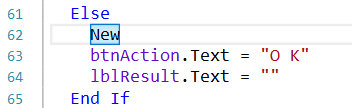
And copy.



## Jump to a subroutine

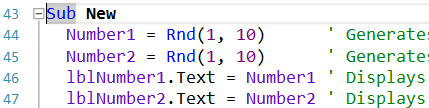
Sometimes it is useful to jump from a subroutine call to the subroutine definition.

This can easily be done :



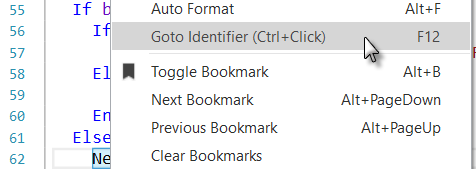
Select the text of the subroutine call.

Press Ctrl and Click.

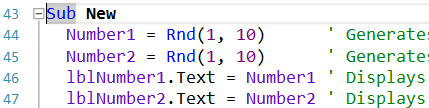


And you are there.

Another method.

Select the text of the subroutine call.

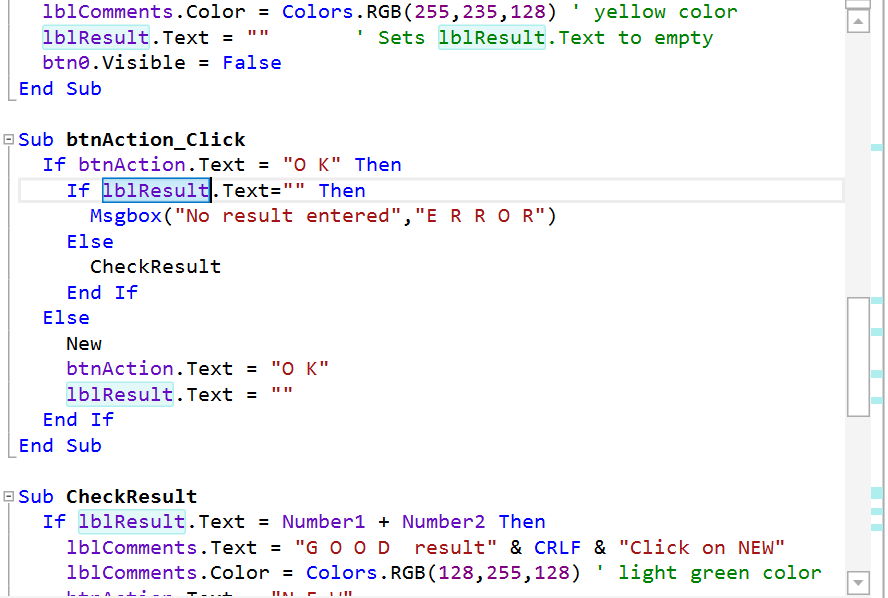
Right click on the selected text.  
  
Click on .

  
  
  
  
And you are there.

## 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.



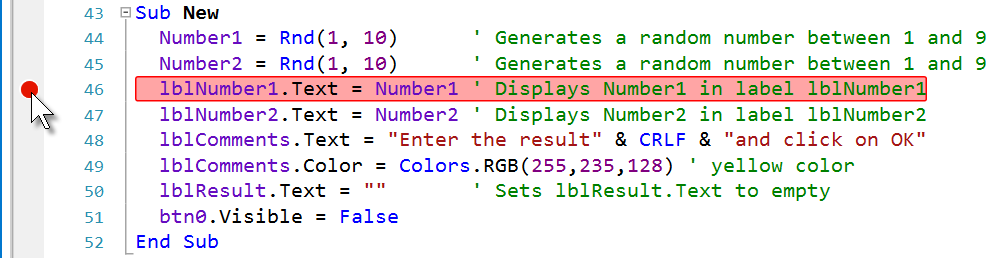
## Breakpoints

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

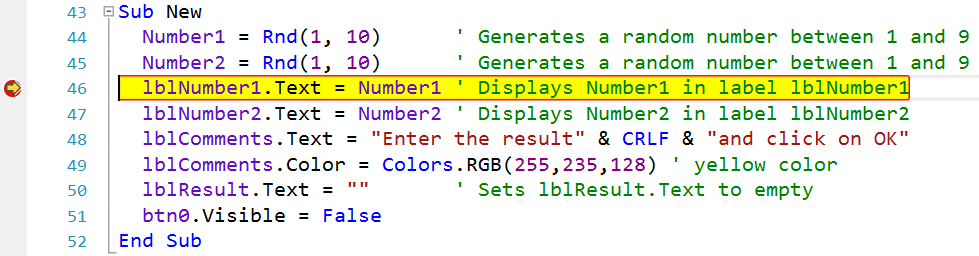
**Breakpoints are ignored in Globals, Process\_Globals and Activity\_Pause.**

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

*rapid debug* mode.



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.



At the bottom of the IDE you find other information.

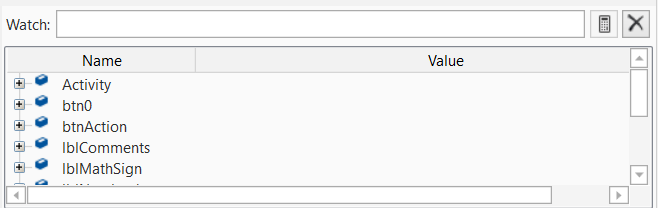


The Debugger is connected. In the left part of the Debugger window we find:

*  A button to update the program after a code modification.
*  The name of the routine where the Debugger stopped the   
   program. New in the module Main in line 46.
*  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.

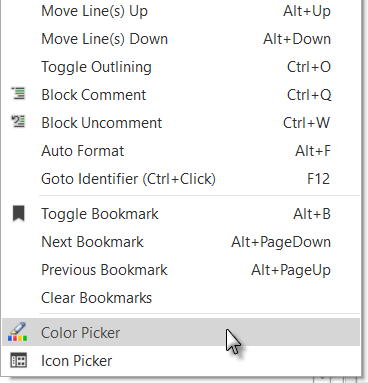
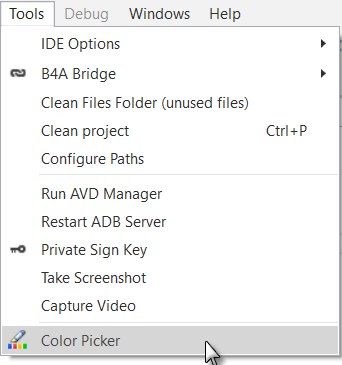
 Stop Stops the program.

 Restart F11 Restarts the program.

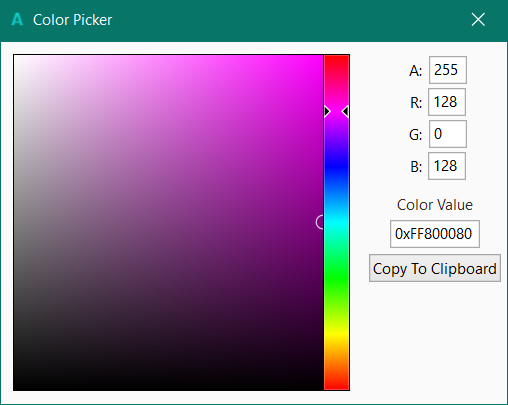
## Color Picker

In the code, right click to show the popup

menu below. Or, in the menu Tools.

Click on  to show the Color Picker.



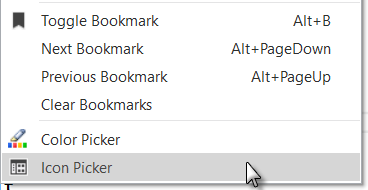
You can move the cursor in the square and the rectangular areas.

Or enter the A R G B values.

Copy the value to the Clipboard.

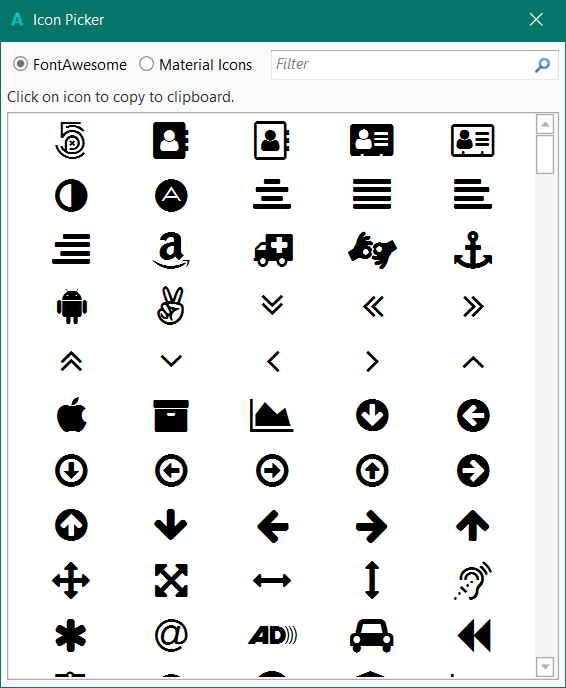
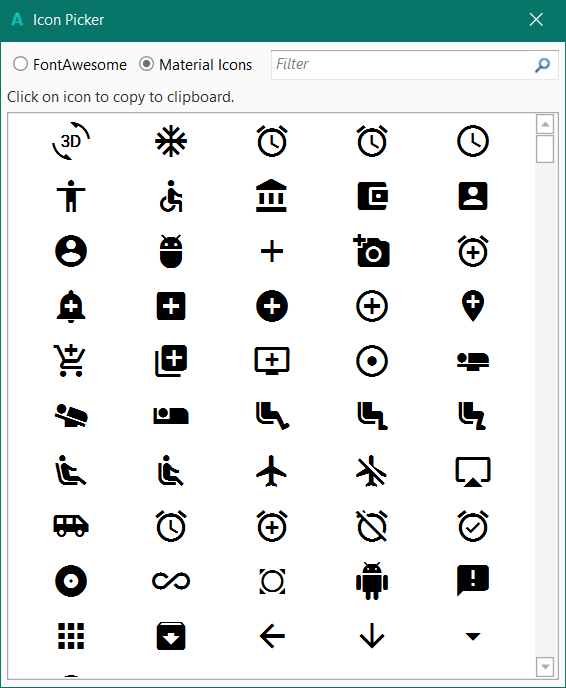
You can then paste the value into the code.

## Icon Picker



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

You can schoose between Font Awesome and Material icons.

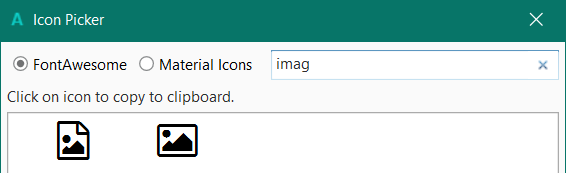
Font Awesome icons. Material icons.

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)

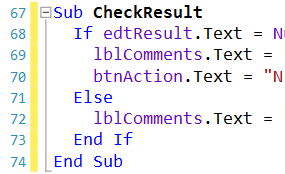


You can filter the icons.

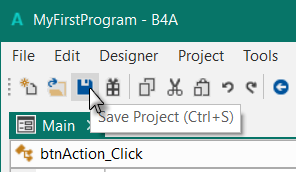
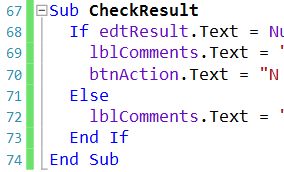
## Colors in the left side

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

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



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.

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

## URLs in comments and strings are ctrl-clickable

URLs in comments and strings are ctrl-clickable.

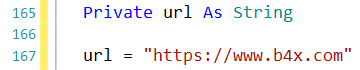
In a comment:



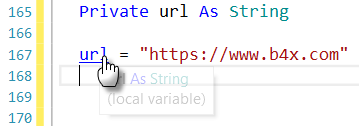
If the cursor is on the line and you press Ctrl the url is highlighted in blue and if you click on it the url it is executed. Hovering over the line with Ctrl pressed does also highlight the url.



In a String:

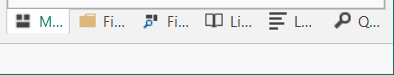


The cursor must be over the String variable and not over text.



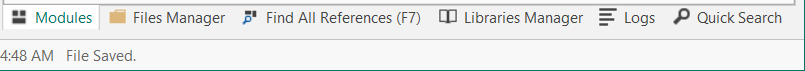
# Tabs

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:

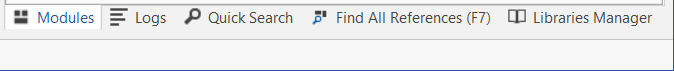
* Modules
* Files Manager
* Libraries Manager
* Logs
* Quick Search
* Find All References

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.

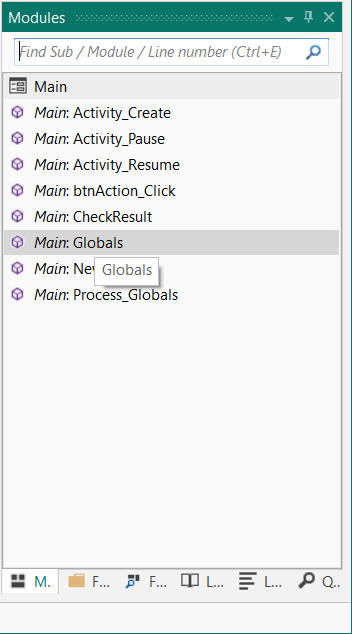
**B4R**



Only 5 Tabs, no Files Manager Tab

## Floating Tab windows

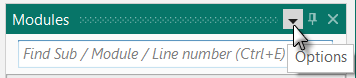
When you start the default IDE all Tab windows are docked in the Tab area.



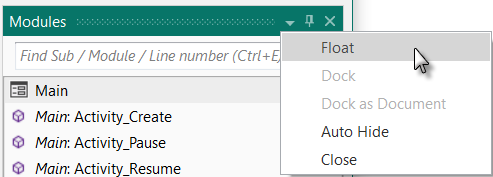
You can set each Tab window as a separate floating window.

## Float

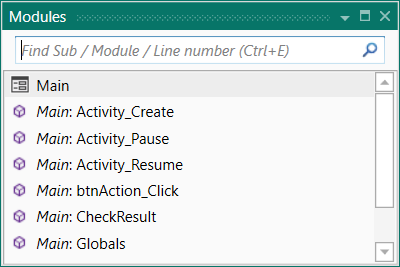
To set the Modules Tab window to floating click in the title on .



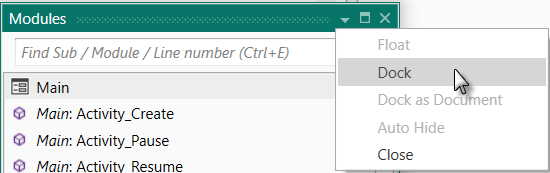
Click on .



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

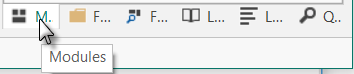


To dock it back to the Tab area click on .

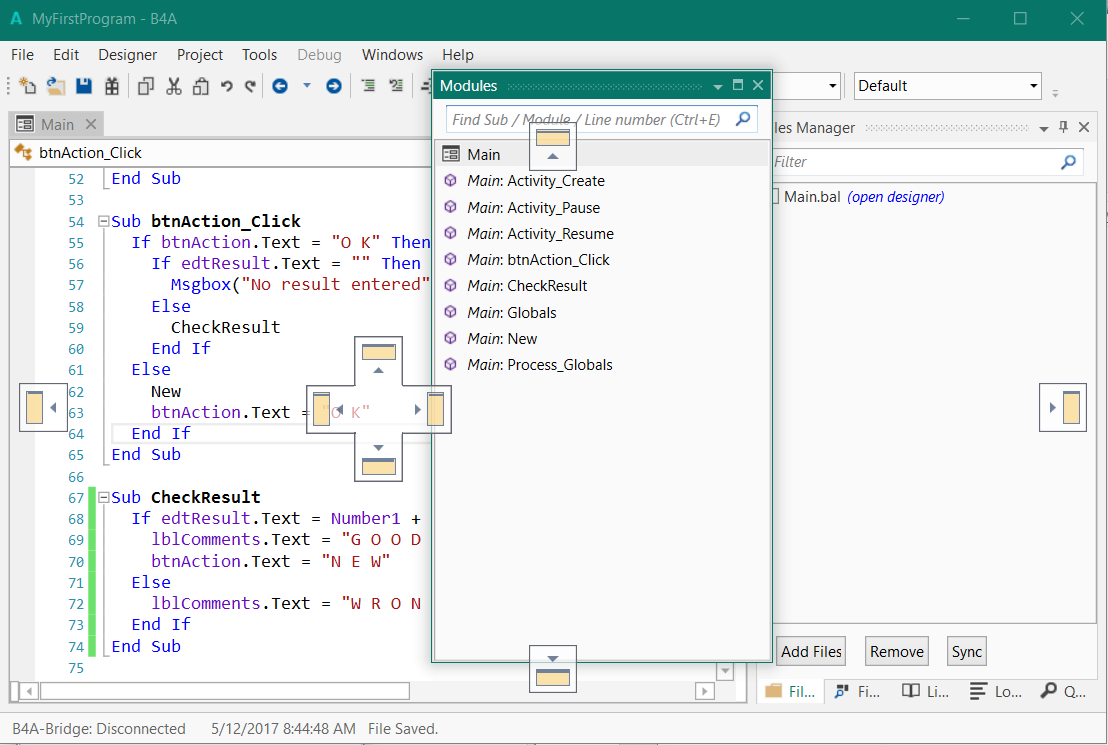


To show the Tabs again click either on Dock in the Options or on Reset in the IDE Window menu.

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

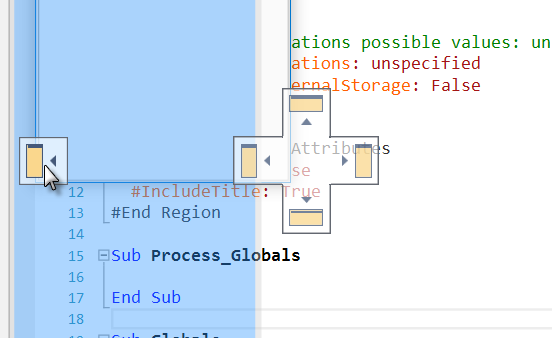


Right

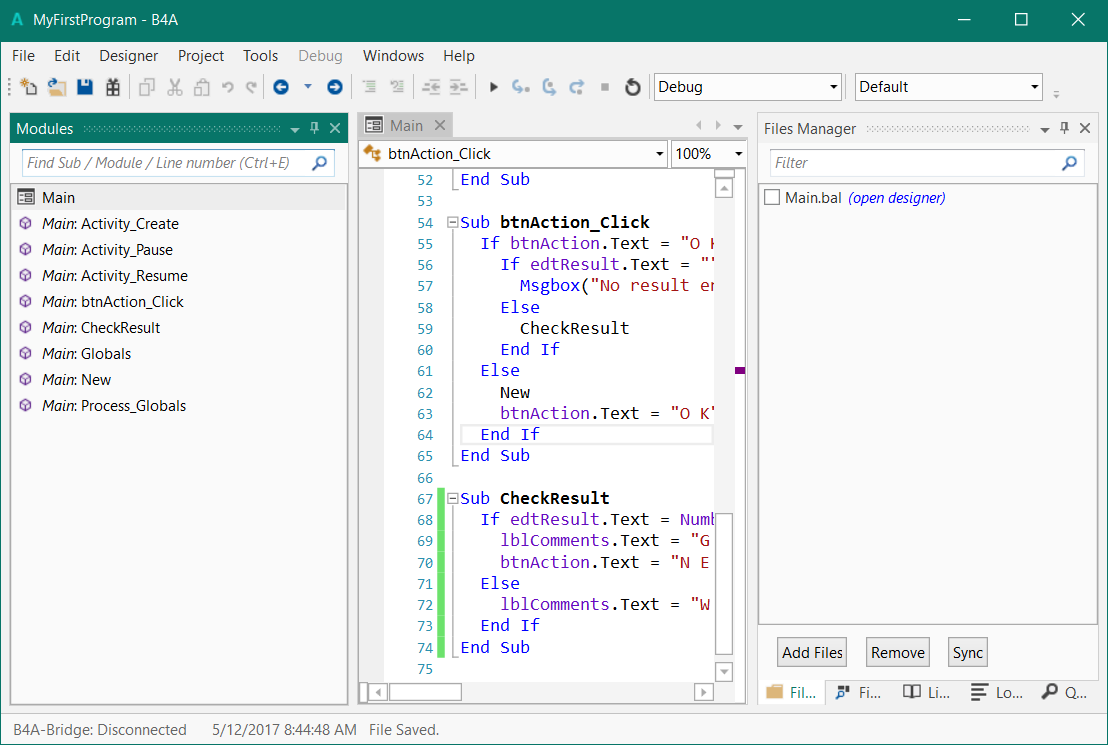


Bottom

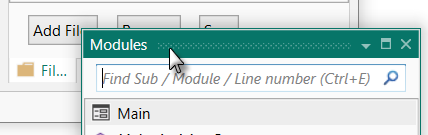
If you mouve the mouse onto one of the docking area symbol, the Tab window will be either on top, on the left, the right or on the bottom.



And the result.

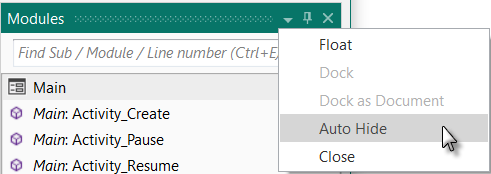


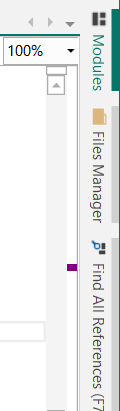
To bring it back to the Tabs, click on the window title and move it back to the Tabs.



## Auto Hide

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

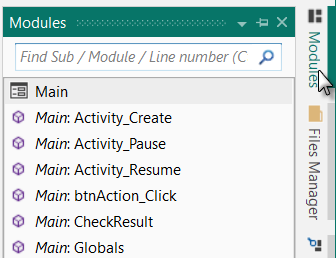




The Tabs move from the bottom of the screen vertically on the right side of the screen and the Tab window is hidden.

Hovering over a Tab highlights it in green.

Click on a Tab to show it.

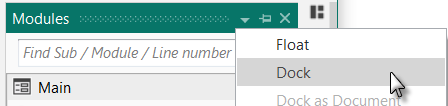


The selected Tab is displayed.

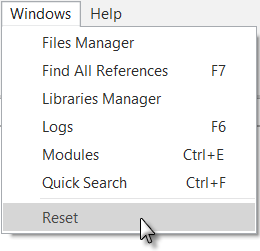
As soon as you click on something in the IDE the Tab is hidden again.

To move the Tabs back to the lower right corner:

Click on  in the Options.



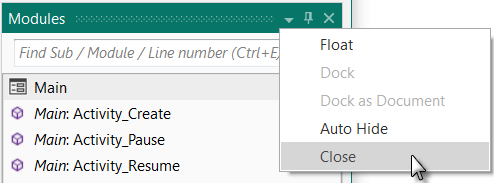
Or click on  in the IDE Windows menu.

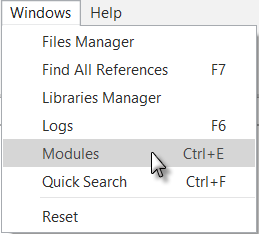


## Close

You can close a window, hide it.

Click on  in the title or on  in the Options.

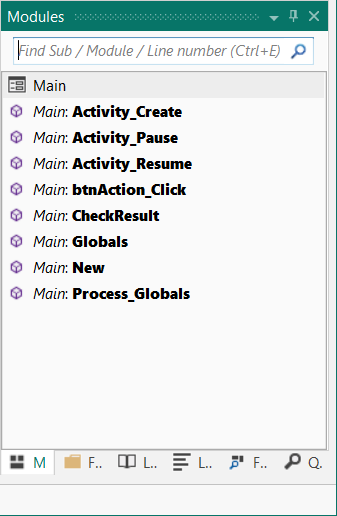




To show it again, in the Windows menu click on the module name you want to show,  in our example.

## Modules and subroutine lists

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)](#_Find_Sub_/)

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.

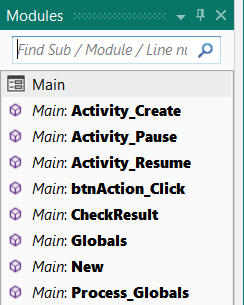
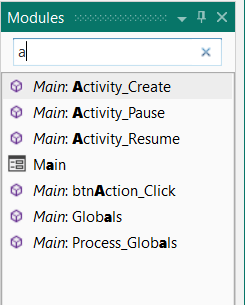
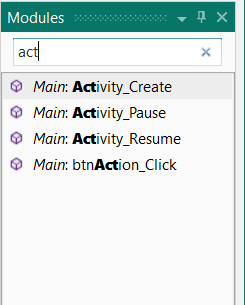
### Find Sub / Module / Line number (Ctrl + E)

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.

No text only the character ‘a’ text ‘act’

Shows all modules and all Shows all modules and Shows all modules and

routines of the selected routines containing ‘a’. routines containing ‘act’.

Module.

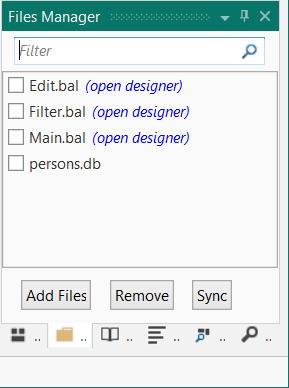
Clicking on one item shows the code of the selected module or routine, even if it's in another module than the current one.

## 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.



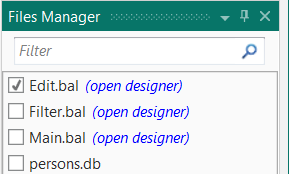
Click on  to open the Designer with the selected file.

Click on  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  to add all the files from the projects Files folder into the File Tab.

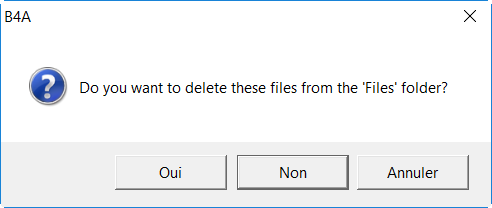
In the IDE, in the bottom right corner.



Checking one or more files enables the

 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.

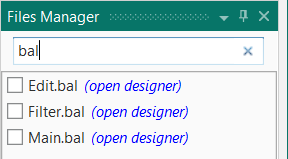
Oui = Yes

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.



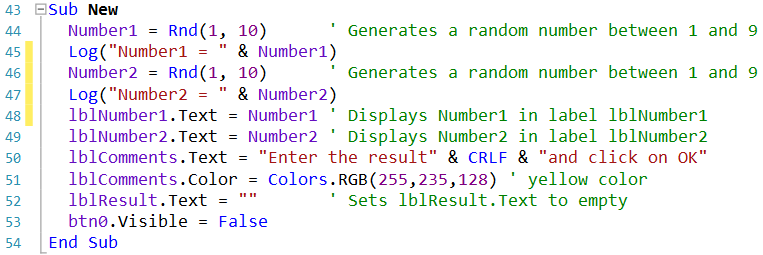
Enter ‘.bal’ to filter all layout files,

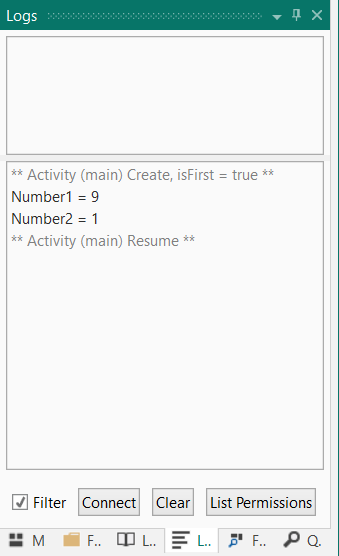
## Logs

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.



Run the program.

Click on  to connect the logger.

The top area of the window shows [Compile Warnings](#_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 \*\*

 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

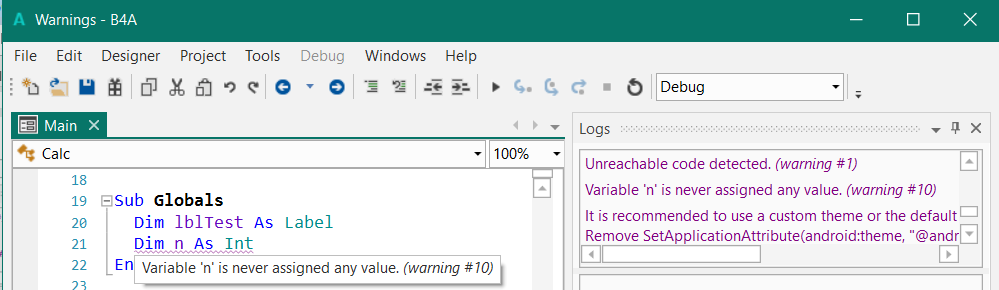
Click on  to clear the Logs window.

#### 

### Compile Warnings

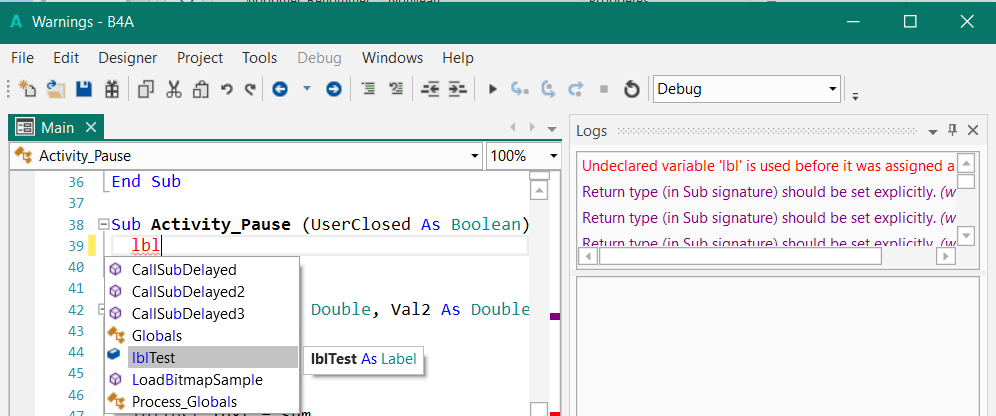
B4A includes a warning engine. The purpose of the warning engine is to find potential programming mistakes as soon as possible. The examples are from the Warnings project.

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 .  
  


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 
* the auto complete pop up window with suggestion containing the written characters.

#### 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:

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.

#### 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 editior.

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 = DialogResponse.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 ' leaving the program

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 = DialogResponse.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 ' leaving the program

Else

Return True ' Return = True the Event will be consumed to avoid

End If ' leaving the program

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](#_Special_functions_)

**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.

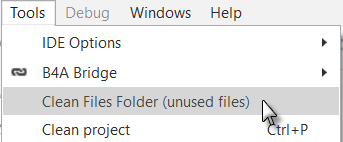
You should:

- 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.

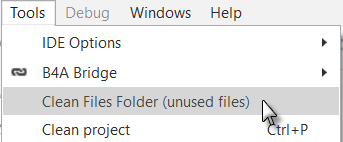
Or you can clean the Files folder from within the Tools menu (see above).

**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](#_Files_1)

**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](http://en.wikipedia.org/wiki/Point_(typography)), 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

imvImage.Bitmap = LoadBitmap(File.DirRootExternal, "image.jpg")

Catch

End Try

Correct code

Try

imvImage.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 editior**.

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

**32: Library ‘xxxx’ is not used.**

Remove the unused library.

## Libraries Manager

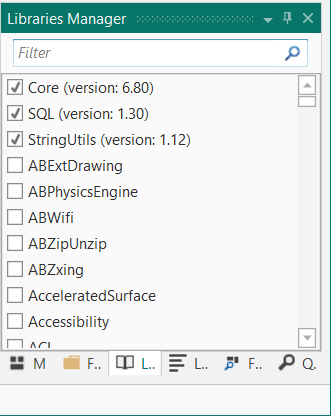
The “Libraries Manager” Tab contains a list of the available libraries that can be used in the project.

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.

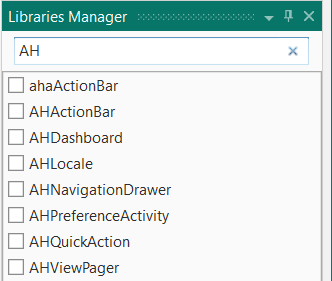
Make sure that you have the latest version of the libraries.



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](https://www.b4x.com/android/documentation.html), [B4i](https://www.b4x.com/b4i/documentation.html), [B4J](https://www.b4x.com/b4j/documentation.html), [B4R](https://www.b4x.com/b4r/documentation.html)

Clicking on a link in the list shows the documentation.

Libraies are explained in detail in the

B4x Basic Language booklet.

## Quick Search

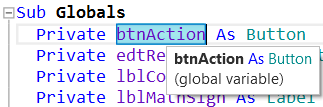
Quick Search allows to search for any text occurrences in the code of the whole project.

Examples with the SecondProgram code.

Several possibilities to select the Quick Search function:

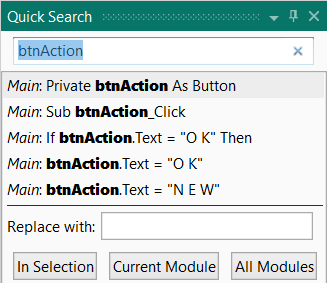
* Ctrl + F, the easiest and most efficient way.
* Click on the  Tab in the lower right corner of the IDE.
* Click on  in the Edit menu.

Example:



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

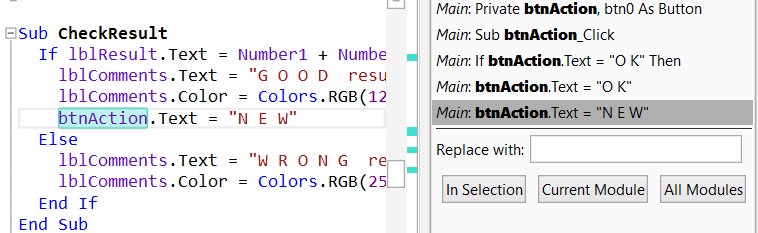
You get the window below in the Tab area.

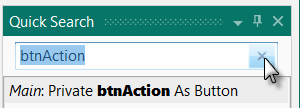


The list shows the occurrences in all Modules.

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.





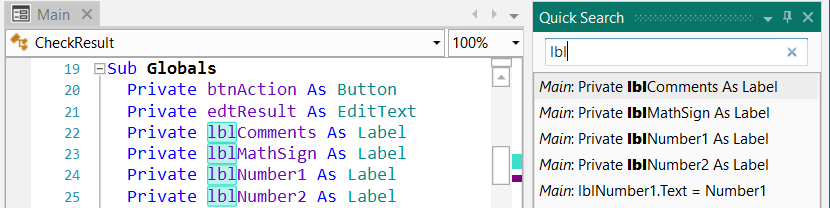
To remove the selection click on on the top right corner of the Quick Search window.

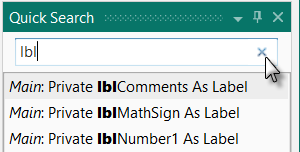
You can also enter any text in the search field:

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.

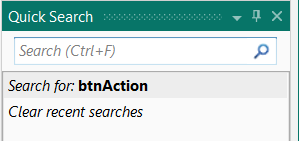
The search text is highlighted in all code Clicking on one of the lines in the list

lines containing this text. jumps directly to this line in the IDE.

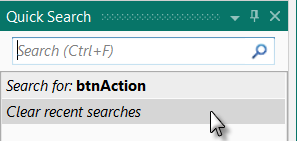




Click on  to remove a search.



You will see a list of the last searches.

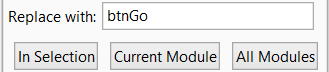


Click on  to remove all recent searches.

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 ,  or .



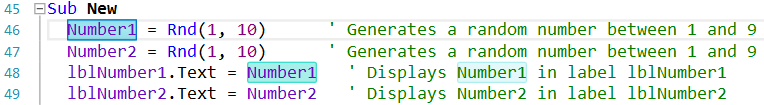
## Find All References (F7)

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

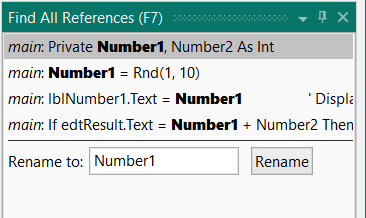
Click on the  Tab or press F7 to get the screen below showing a list of all code lines with the selected reference or the first object in the current line.

Example with the code of SecondProgram.

Select in the code in line 49 Number1.



Click on  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 .

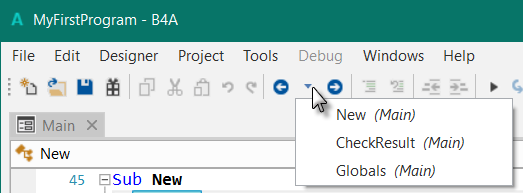
# Navigation in the IDE

## 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.

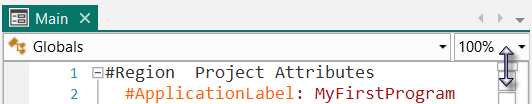
## Alt + N Navigation stack menu

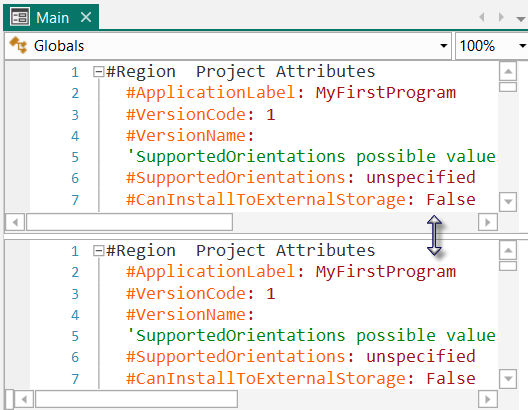
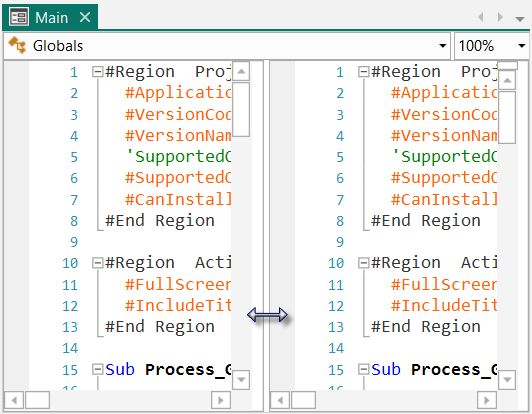
Opens the navigation stack menu. You can then choose the location with the up and down keys.



## Split the screen

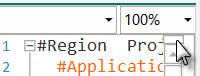
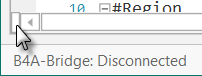
If you are working on two locations in the same module then you can split the code editor (it can be split again vertically):

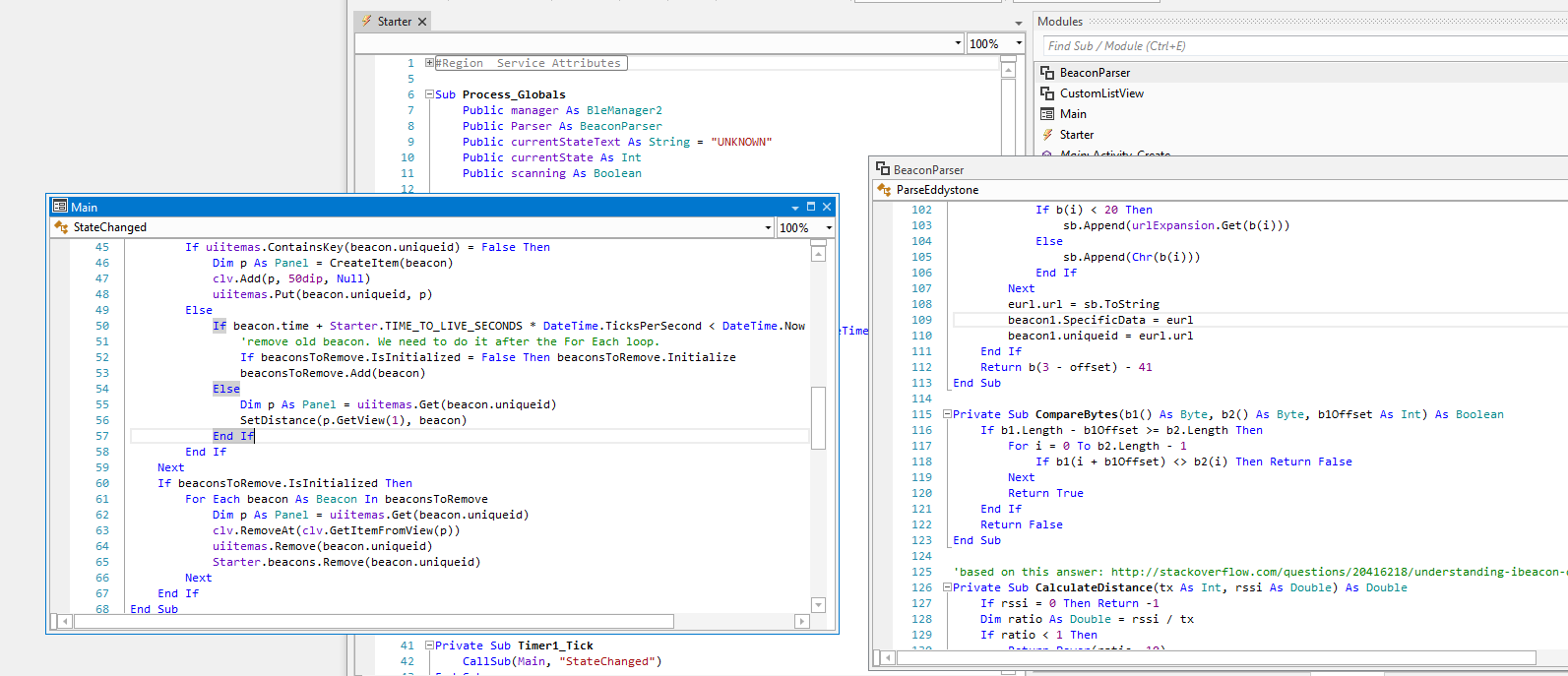
Horizontally Vertically

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



## Multiple windows

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



## Ctrl + E Search for sub or module

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

## Ctrl + Click on any sub or variable

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

## F7 - Find all references

Not exactly related to navigation but is also useful when working with large projects.

Details in [Find all references](#_Find_All_References).

## Ctrl + F Quick Search

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

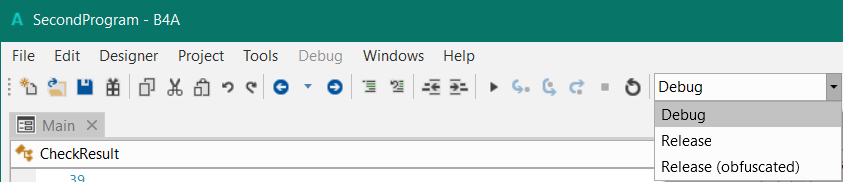
# Debugging B4A, B4i, B4J

Debugging is an important part when developing.

Debugging is different in B4R than in B4A, B4i and B4J.

## 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](#_Breakpoints) - You can mark lines of codes as breakpoints. This is done by pressing on the grey area left of the line.

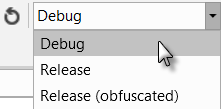
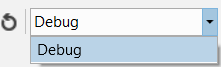
The program will pause when it reaches a breakpoint and will allow you to inspect the current state.

[Logging](#_Logs) - 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.

### Debug mode

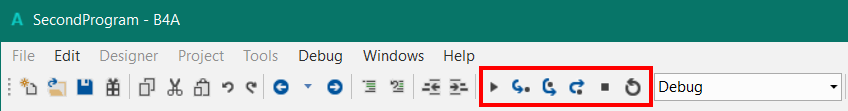
The debugging modes are different in the in the products:

B4A, B4J B4i, only Debug

#### 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.

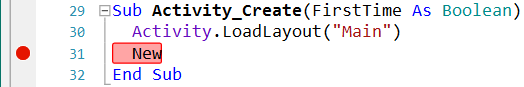
##### Run F5

Runs the program,

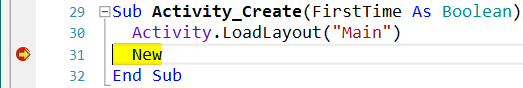
If the program is stopped at a breakpoint the program runs until the next breakpoint or completes running.

##### Step In F8

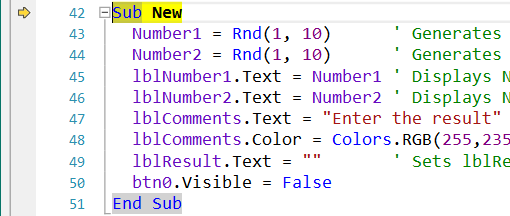
The debugger executes the code step by step.



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

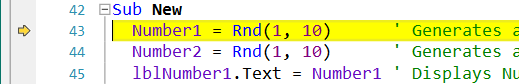


We run the program, it will stop executing at line 31 New.



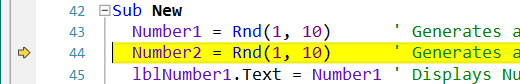
Click on .

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



Click once more on .

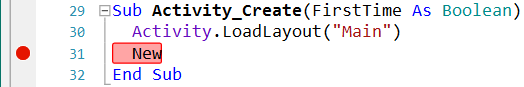
The debugger executes the next line, Number1 =…

Click once more on .

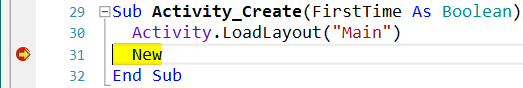
The debugger executes the next line, Number2 =…

##### 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.

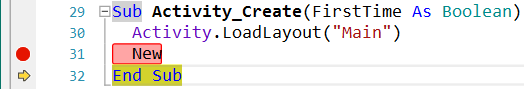


In the SecondProgram project we set a Breakpoint at line 31 New.



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

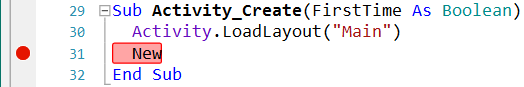
Click on .

The debugger executes the code in New and jumpes directly to the next line which is

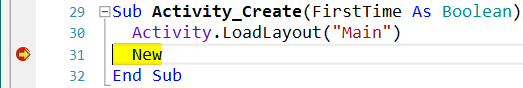
End Sub of Activity\_Create.

##### Step Out F10

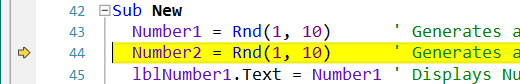
If the current line is in a subroutine the debugger finishes executing the rest of the code and jumps to the next line after the subs’ calling line.



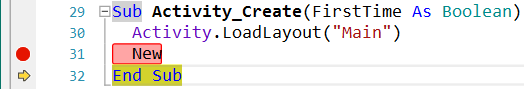
In the SecondProgram project we set a Breakpoint at line 32 New.



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



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

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.

##### Stop

Stops the program and leaves the Rapid Debugger.

##### Restart F11

Restarts the program remaining in the Rapid Debugger.

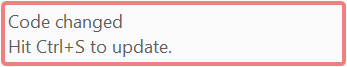
Executes:

**B4A** Process\_Globals, Globals, Activity\_Create and reloads the layout.

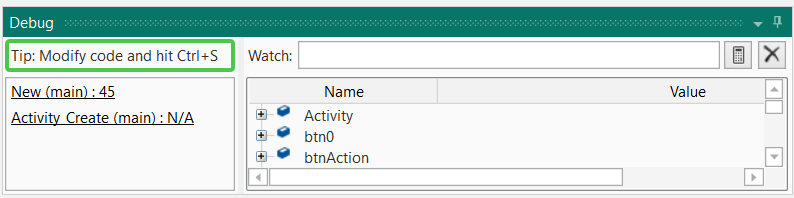
**B4i** Process\_Globals,

**B4J** Process\_Globals,

This is useful if you changed a layout file.

It is different from  explained in the next chapter.

### Debug window



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

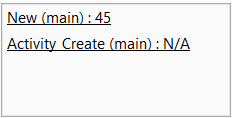
#### The status button



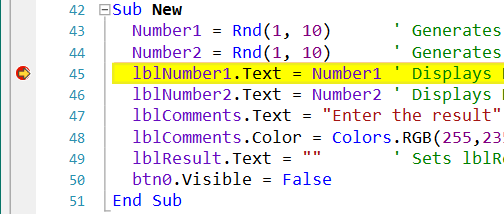
Shows that the program is running, the button border is green.

When you change the code the button border changes to red.  
To update the code click on the button or hit Ctrl + S.

#### The breakpoint window

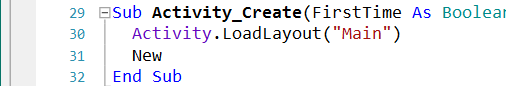


The breakpoint window shows where the program has stopped.



  
The program stopped in line 45,

in routine New in the main module.

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

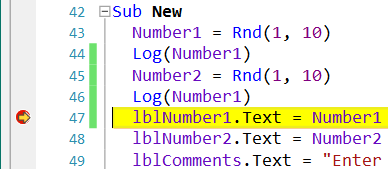


When you click on one of the lines the cursor jumps to that line.

#### The Watch window

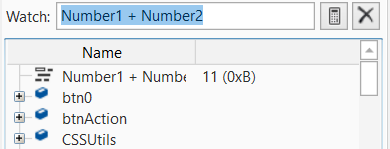


The Watch window allows to check more complex functions for testing and debugging.



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

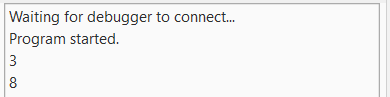
Run the program.



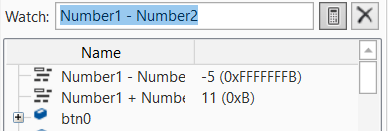
In the Add Watch field enter:

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.

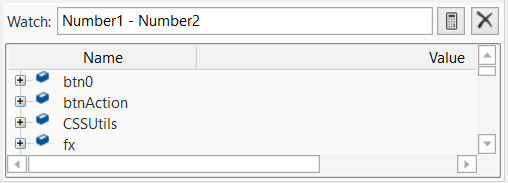


You can enter a new watch line   
Number1 + Number2   
and show it.

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

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

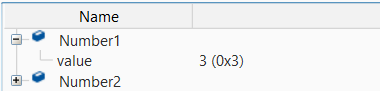
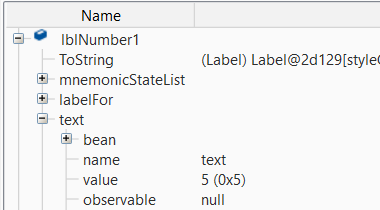
#### The object window

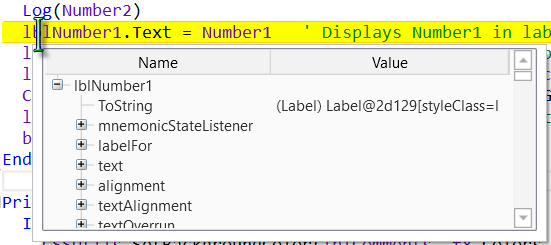


Shows all variables and objects in the list ordered by alphabetical order.

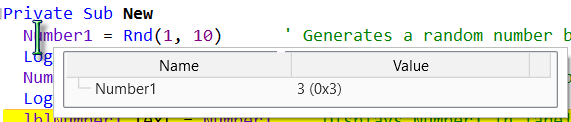
Click on  to show the details of the object:

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

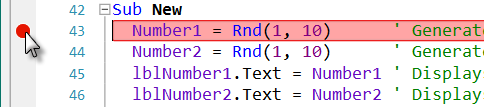
Number1

#### Breakpoints

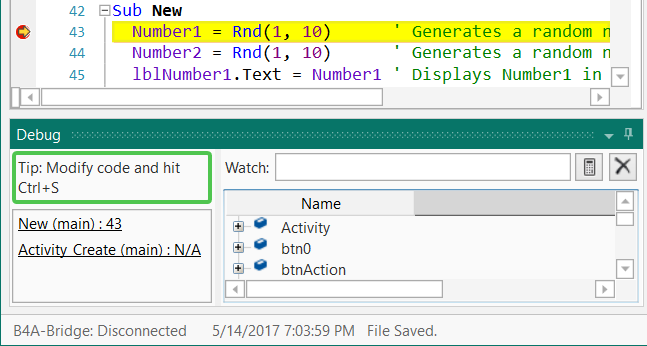
One important feature to make debugging easier are breakpoints. You can set breakpoint almost wherever you want in the code.

**Breakpoints, in Process\_Globals are ignored.**

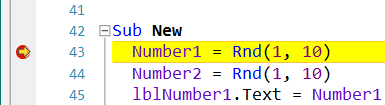
Clicking on a line in the left margin adds a breakpoint. When the program is running it stops at the first encountered breakpoint.



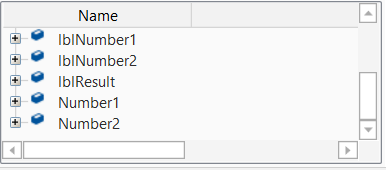
Run the program, the program stops at the breakpoint and the IDE looks like below. The breakpoint line is highlighted in yellow.



On the bottom of the window you see the debug window.

Example with the SecondProgram:

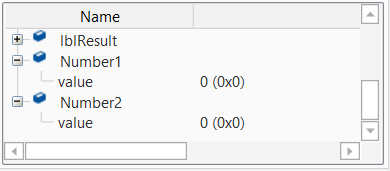
Set a breakpoint in line 43 and run the program.



In the variable window look at Number1 and Number2:



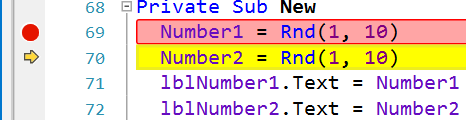
The values are 0 for both.

If you see this  at the left side of Number1 or Number2 click on it to show the details.

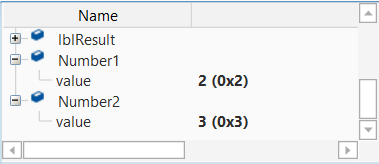
Click on .

The program jumps to the next line.

Click on .

You see that the value of

Number1 has changed.



Click on  again.

The program jumps to the next line.

Click on .

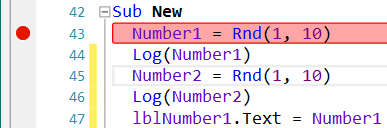
You see that the value of

Number2 has changed.

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

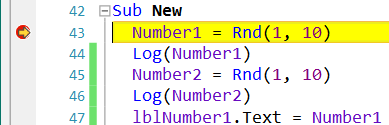
#### With Logs

Example with the SecondProgram.

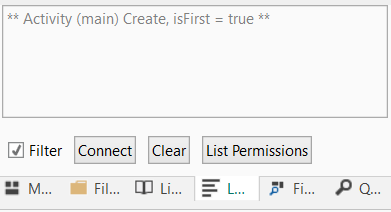


We add the two lines with the Log keyword to display the two numbers in the Log Tab.

We and add a breakpoint in line 69 to watch what happens.

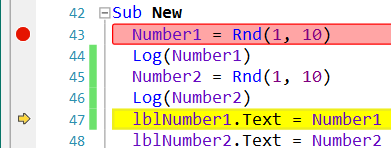


Run the program, it stops at line 43.

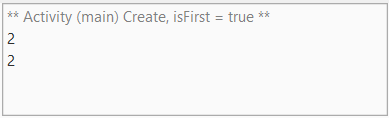


In the Log Tab we see at the moment only  
Waiting for debugger to connect...

and telling that the program has started.

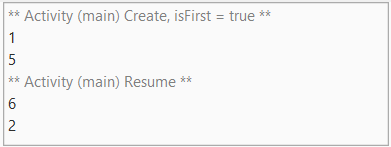


Click four times on  till the program reaches line 47.

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

Click on  to run to the end.

Nothing new is displayed

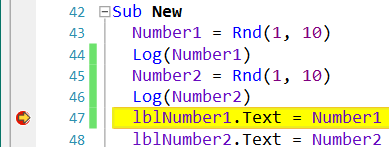


When you are using the program the two new values will be shown every time the program runs the New routine.

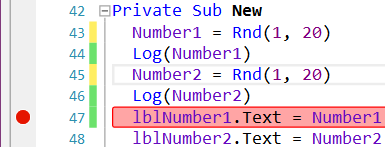
#### Modifying code in the Debugger

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.



Run the program till it stops at the breakpoint.



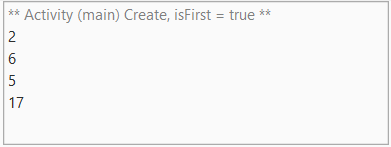
We change the two numbers 10 to 20.



The status button color has changed confirming a code change.

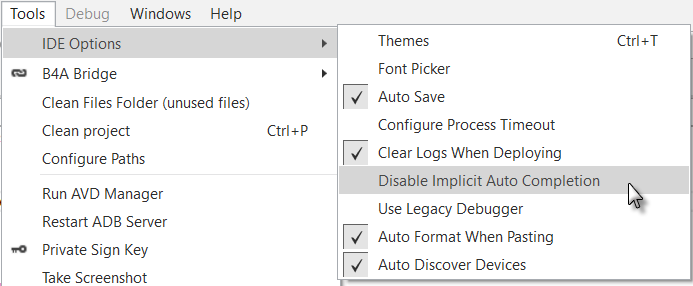
To rerun the program click on Ctrl + S.

Using the program we see now that the numbers can be between 1 and 19.



### 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.

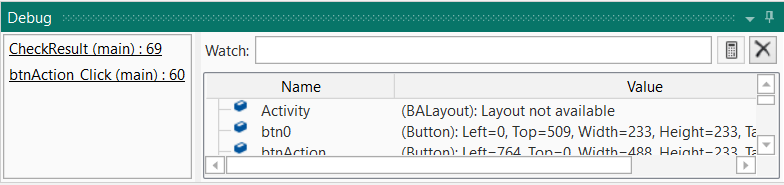
The debugging code allows the IDE to connect to the program and inspect it while it runs.

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.

## 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](#_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.

### 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