LibreOffice RefCard

LibreOffice Basic
IDE

Beginner

v. 1.05 – 01/15/2019

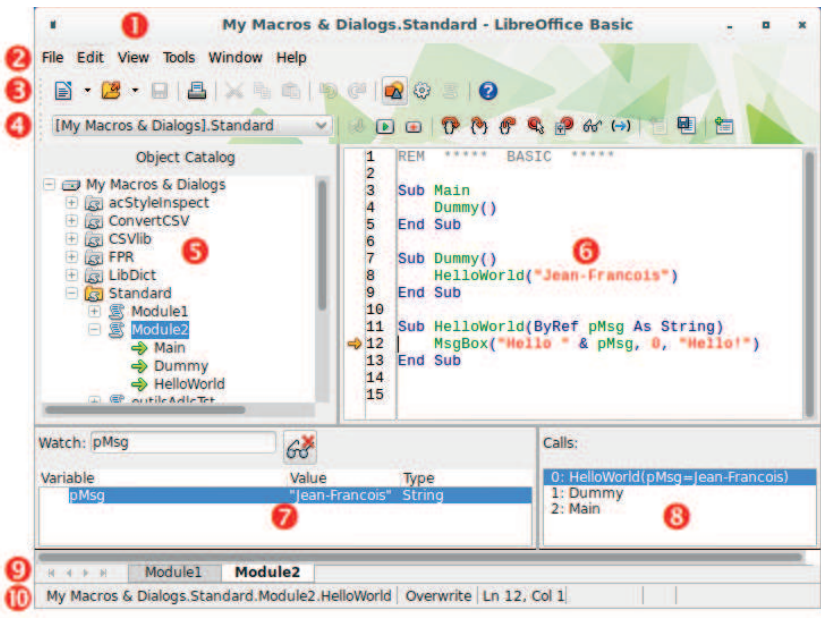
Written using LibreOffice v. 5.3.3 – Platform: All

☞ IDE : *Integrated development environment*.
A set of tools for writing and debugging your macros code.

Getting To The IDE

Menu **Tools > Macros > Organize macros > LibreOffice Basic** (**Alt** + **F11**) then select the desired module to edit (**Edit** button).

Overview



```
REM  *****  BASIC  *****

Sub Main
	Dummy()
End Sub

Sub Dummy()
	HelloWorld("Jean-Francois")
End Sub

Sub HelloWorld(ByRef pMsg As String)
	MsgBox("Hello " & pMsg, 0, "Hello!")
End Sub
```

The IDE window is made of 10 tools, most of them are described in this refcard.

❶ Title bar	Name of the container and of the current library.
❷ Menu Bar	Standard LibreOffice menus, dedicated to macros management.
❸ **Standard** toolbar	The LibreOffice **Standard** toolbar, showing only tools dedicated to code edition and debugging.
❹ **Macros** toolbar	Tools dedicated to code edition and debugging.
❺ Object catalog	Containers tree-view showing libraries, modules and subprograms.
❻ Code editor	Main area for code edition, with syntax highlighting, breakpoints management and execution control.
❼ Watch pane	Variable contents checking (watches).
❽ Call stack panel	Subprogram calls and their parameters.
❾ Tab bar	Lists and manages the current library modules.
❿ Status bar	Run-time status.

✧ The main panels (**Catalog** ❺, **Editor** ❻, **Watches** ❼ and **Call stack** ❽) can be detached.
✧ **F6** swaps from a tool to the next one.

Toolbars

If not visible : **View > Toolbars**.

Standard Toolbar

Two toolbuttons are of interest:

Select macro	Calls the macro selection dialog.
Modules	Calls the **Macros Organizer** dialog.

Macros Toolbar

Library selector	Selection of the library to use.
« Compile »	Syntax checking.
Run immediately F5	Run the subprogram under the cursor.
Stop Maj + F5	Stop the running program.
Step Over Maj + F8	Step-by-step execution, bypassing the called.
Step Into F8	Step into the called subprogram.
Step Out Ctrl + Maj + F8	Step out to the caller.
Breakpoint On/off F9	(De)activate the breakpoint on current line.
Manage Breakpoints	Opens the breakpoints dialog.
Enable Watch F7	Adds the selected var to the watches.
Find Parenthesis	
Import Basic	Import a macro source code.
Export Basic	Save the source code (macro or dialog).
Import Dialog	Import a dialog source code.

Dialog Toolbar

This toolbar takes place of the Macros toolbar when editing dialog boxes.

Insert Controls	Same contents as the controls toolbar (below).
Import Dialog	Import a dialog box source code.
Export Dialog	Export a dialog box source code. The files have a `.xdl` extension (XML).

Controls Toolbar

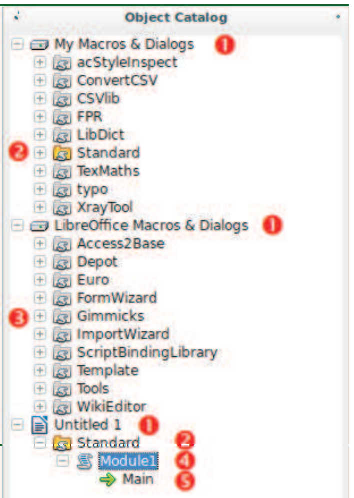
This toolbar is shown when in dialog box conception mode. It displays each available control to place on a dialog. These three buttons deserve an explanation:

Select Element	Get back to selection mode.
Manage Language	Multilingual dialog creation.
Preview Dialog	Dialog test mode (hit **Esc** to get back to edition mode).

Object Catalog

Displays the 3 **containers types** and their contents: libraries, modules and macros.

Containers ❶



My Macros & Dialogs

Specific to the user's account, for all documents. Can only be used by the user.

LibreOffice Macros & Dialogs

(aka **global macros**)

These macros are stored within the LibreOffice global container. As such, they can be viewed and used everywhere.

The ones on the capture are part of a standard LibreOffice install.

Untitled 1 (in the example)

In the current document.

Libraries

Loaded (colored) ❷ or not (grayed-out) ❸.

Standard Library

Apart from global libraries, each container comes with a **Standard** library.

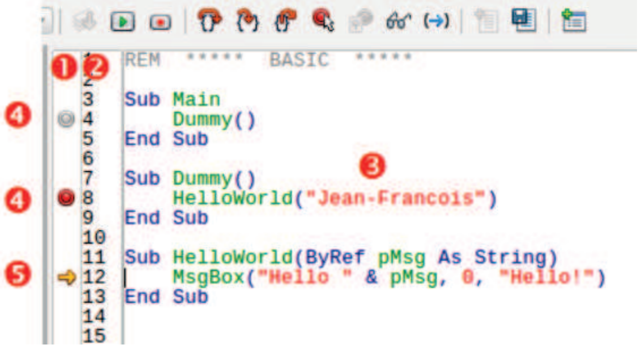
☞ **Standard** is always loaded at opening time (of the application or the document):
It can't be deleted.
It can't be overwritten through some code import.
It can't be encrypted.

Modules ❹

Subprograms (macros) ❺

Code Editor

Used for code writing (with syntax highlighting) and debugging (read on).



```
REM  *****  BASIC  *****

Sub Main
	Dummy()
End Sub

Sub Dummy()
	HelloWorld("Jean-Francois")
End Sub

Sub HelloWorld(ByRef pMsg As String)
	MsgBox("Hello " & pMsg, 0, "Hello!")
End Sub
```

Useful Items In the Edition Panel

❶ Gutter	A "margin" in which the breakpoints ❹ and the execution flag ❺ are displayed.
❷ Line numbers	For easier navigation (toggled using **View > Line Numbers**).
❸ Editor	Helps writing Basic code. Supports syntax highlighting.

(Un)Indenting lines

Press **Tab** / **Maj** + **Tab** to indent/unindent lines of code.

☞ These commands can be used when selecting a set of lines.

Syntax Highlighting

The colors used are specified in LibreOffice options: **Tools > Options > LibreOffice > Application Colors**. Check the **Basic Syntax Highlighting** section values.

Displaying/hiding line numbers

Toggle line numbers display with **View > Line Numbers**.

☞ You may add a button to the **Macros** toolbar (see **Category** `BASIC`).
☞ Breakpoint management and execution control: see **Debugging**.

Goto Line

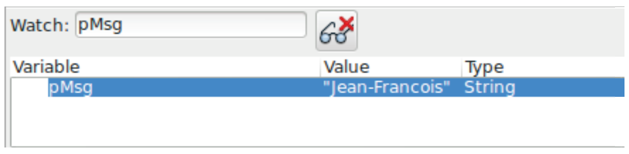
Ctrl + **L**

Watches Panel

Allows checking variable contents at runtime (more in **Debugging** below).

Three columns (adjustable widths):

- Variable name
- Value
- Type



Watch: pMsg

Variable	Value	Type
pMsg	"Jean-Francois"	String

Adding A Watch

1. In the code, select the variable to observe (you may enter its name in the **Watch** zone),
2. click the **Enable Watch** button **F7**.

Removing A Watch

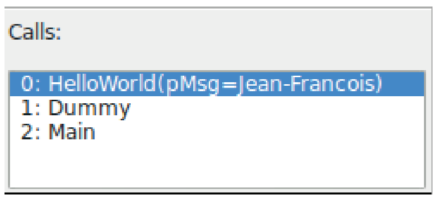
1. Select the watch to remove,
2. click **Remove Watch** (in the Watches panel)

Call Stack Panel

In step-by-step execution mode, allows to know the successive subprograms calls.

Read it bottom-up. All subprograms are listed along with their parameter values.

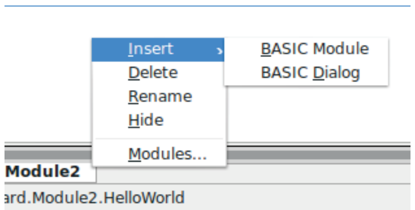
Subprogram `0` is the one in which the execution is currently. The others are numbered subsequently.



Calls:
0: HelloWorld(pMsg=Jean-Francois)
1: Dummy
2: Main

Tab Bar

Displays the modules names for the current library. Modules names are shown in alphabetical order, left to right.



Right-click on the bar to manage the modules:

Insert > Basic Module Create a new code module (defaults to **ModuleN**).
Insert > Basic Dialog Create a new dialog module (defaults to **DialogN**).
Delete Delete the selected module.
Rename Rename the selected module.
Hide Temporarily hide the selected module.
Modules Calls the **Macro Organizer**.

Status Bar

From let to right:

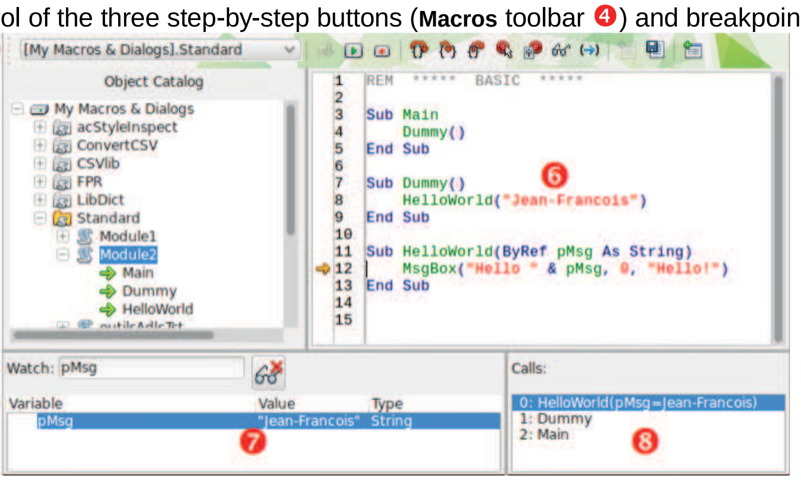
- The subprogram full name in which the cursor currently sits.
- The current code edition mode (switch to **Overwrite** using Ins).
- The line (**Ln**) and column (**Col**) (character) number where the cursor currently is.

Debugging

Debugging: checking a program execution.

☞ This operation is realized executing the program part to be checked in step-by-step mode while watching the execution context (variables values).

Debugging uses three panels: code editor ❻, call stack ❼ and watch pane ❽. It runs under the control of the three step-by-step buttons (**Macros** toolbar ❹) and breakpoints.



```
REM  *****  BASIC  *****

Sub Main
    Dummy()
End Sub

Sub Dummy()
    HelloWorld("Jean-Francois")
End Sub

Sub HelloWorld(ByRef pMsg As String)
    MsgBox("Hello " & pMsg, 0, "Hello!")
End Sub
```

```
pMsg    "Jean-Francois"  String
```

```
0: HelloWorld(pMsg=Jean-Francois)
1: Dummy
2: Main
```

Execution Modes

Five buttons define the way the code is run. The ⇨ mark displays the execution point.

☞ Note that when you start execution (any mode) while the cursor is in a subprogram that requires non-optional parameters, you get an error message: **Argument is not optional**.

Run Maj + F5 Runs the subprogram where the cursor sits. The execution stops at breakpoints only.
Stop F5 Stops the execution.
Step Over Maj + F8 Step-by-step execution without entering the subprograms.
Step Into F8 Step in subprograms.
Step Out Ctrl + Maj + F8 Step out to callers.

Managing Breakpoints

Breakpoint: a mark where the execution is paused. This allows to examine the execution context at that moment (watches values).

Execution is paused before executing the marked line.

☞ The actions below may be accomplished during a debugging session.

Adding A Breakpoint

Double-click in the gutter on the line to stop at when no breakpoint exists (F9).

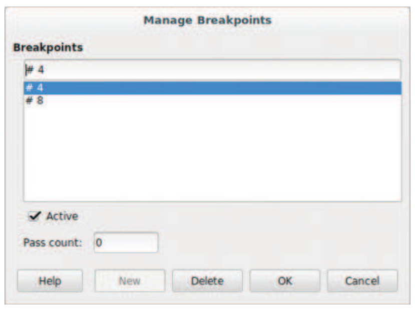
Deleting A Breakpoint

Double-click in the gutter on an existing breakpoint (F9).

Disabling/Re-enabling A Breakpoint

Right-click in the gutter on any existing breakpoint, then disable **Active**. The icon turns from ● (active) to ○ (inactive).

Breakpoints Manager



Call it by:

- right-clicking the execution mark or any existing breakpoint, option **Properties**
- clicking the **Manage Breakpoints** button. Top to bottom:

(list and edition area) Line numbers where breakpoints are set. Select or create a new one.
Active (Un)Check to (de)activate the selected breakpoint.
Pass Count Activates a breakpoint after a given number of passes.
New Click to add a new breakpoint on the specified line.
Delete Deletes the selected breakpoint.

Options

LibreOffice offers a set of IDE options in **Tools > Options > LibreOffice > Basic IDE**

☞ These options are available after having checked the **Enable experimental features** in **Tools > Options > LibreOffice > Advanced**

Enable code completion The IDE automatically completes the Basic objects methods
☞ Doesn't apply to custom objects.
☞ The **UseExtended types** option must be enabled.

Autocorrection Fixes keywords and variable names syntax.
Autoclose quotes When typing `"` (double quote), the IDE adds the closing quote.
Autoclose parenthesis When typing `(` (parenthesis), the IDE adds the closing parenthesis.
Autoclose procedures When typing `Sub Xxx` or `Function Xxx` (`Sub` or `Function` declaration), the IDE adds an `End Sub` or `End Function`.
UseExtended types (needed to get code completion) Allows using UNO object types as valid for Basic.

Hints

Copying A Library Between Containers

1. Open the **source** document/container,
2. Open the **Macro Organizer** (📜), **Libraries** tab,
3. Select the library **Location**,
4. **Export > Export as Basic library**,
5. Open the **target** document/container,
6. **Import**.

Copying Modules From A Library To Another

(in the same document or between documents/containers)

1. Open both source and target documents/containers,
2. Open the **Macro Organizer** (📜),
3. Drag/drop from source to target.

☠ By default, modules are **moved**. To copy: Ctrl + drag/drop.

Hiding Modules

Allows to (temporarily) simplify the tab row: right-click a tab then **> Hide**.

Encrypting Libraries

Encrypting a library makes a more secure code.

☞ Any library may be encrypted, except **Standard**.

1. Open the **Macro Organizer** (📜), **Libraries** tab,
2. Select the **Location**,
3. Select the library,
4. Click **Password**,
5. Enter the **New password** and confirm,
6. Validate.

☠ **Warning**!
The encryption is **very** strong (AES): **password loss is irremediable!**
☞ **Always** keep a non-encrypted version for yourself.

Cleaning MSOffice imports

Delete the unnecessary libraries and modules that came with the import.

☠ Always work on a copy of your file!

1. Save to the "flat" format (ex: `.fods`)
2. Open the "flat" file using a XML-capable text editor.
3. Search for the `<office:scripts>` markup.
4. Delete the whole section between (including) the `<ooo:library-embedded ooo:name="Standard">` and `</ooo:library-embedded>` marks.
5. Save.
6. Open with LibO and re-create the correct libraries/modules tree (copy/paste the wanted code from the original document to the current one).

IDE Keyboard Shortcuts

Macros dialog	Alt + F11	Stop execution	F5
Goto line	Ctrl + L	Step Over	Maj + F8
Add/remove breakpoint	F9	Step Into	F8
Activate a watch	F7	Step Out	Ctrl + Maj + F8
Immediate execution	Maj + F5	Switch IDE tool	F6

Credits

Author : Jean-François Nifenecker – jean-francois.nifenecker@laposte.net

We are like dwarves perched on the shoulders of giants, and thus we are able to see more and farther than the latter. And this is not at all because of the acuteness of our sight or the stature of our body, but because we are carried aloft and elevated by the magnitude of the giants (Bernard de Chartres [attr.])

History

Version	Date	Comments
1.04	09/10/2018	Minor updates
1.05	01/15/2019	Minor updates