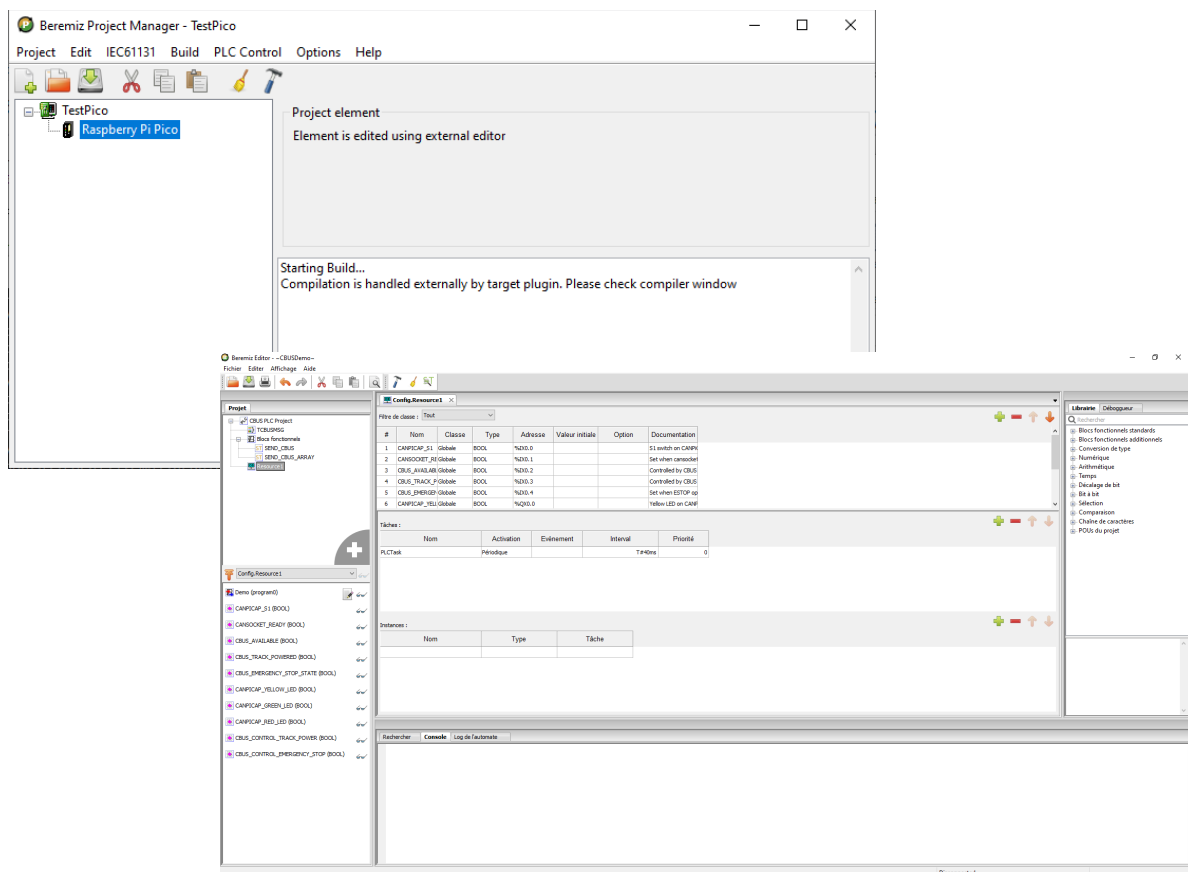


BEREMIZ PROJECT MANAGER

USER'S MANUAL



Contents

1 - Introduction.....	<u>3</u>
1.1 - IMPORTANT REMINDER.....	<u>3</u>
2 - IEC61131-3 terminology.....	<u>3</u>
3 - Software installation.....	<u>4</u>
3.1 - Installation of Beremiz Project Manager project manager.....	<u>4</u>
3.2 - First configuration.....	<u>4</u>
4 - Key concepts of Beremiz Project Manager.....	<u>7</u>
4.1 - Main user interface.....	<u>7</u>
4.2 - Project elements.....	<u>7</u>
4.3 - Processus of a project.....	<u>8</u>
5 - Writing a PLC application.....	<u>9</u>
5.1 - Creating a new project.....	<u>9</u>
5.1.1 - Target selection.....	<u>9</u>
5.2 - Loading an existing project.....	<u>10</u>
5.3 - Target configuration.....	<u>11</u>
5.4 - Target framework generation.....	<u>11</u>
5.5 - Writing / editing the PLC source code.....	<u>12</u>
5.5.1 - Edition of program properties.....	<u>13</u>
5.5.2 - Writing your first IEC61131-3 PLC program.....	<u>13</u>
5.5.3 - From the program to a task.....	<u>15</u>
5.5.4 - Generation of PLC code.....	<u>16</u>
5.6 - Cleaning project files.....	<u>17</u>
5.7 - Project compilation.....	<u>17</u>
6 - Document revisions.....	<u>19</u>

1 - Introduction

Beremiz Project Manager is a software suite, built around Beremiz, the Open Source IDE for IEC61131-3 based machine automation. It acts as a "top level" management tool allowing you to create powerful Programmable Logic Controller applications on multiple targets, programmed in any of the IEC61131-3 languages :

- Ladder Diagram
- Function Block Diagram
- Structured Text
- Instruction List
- Sequential Function Chart

Beremiz Project Manager is built in a modular way, in order to support multiple PLC targets. The Project Manager itself is not related to any target: code generation and target control are performed by dedicated plugins. By adding new plugins, you can then add as many new targets as wanted.

The software suite is operated from Beremiz Project Manager application, which acts as a control tower. The Manager is in charge of project generation and configuration (including I/O configuration), which is then sent into Beremiz IDE where you can write the PLC programs.

Once IEC61131-3 code has been written, Beremiz Project Manager will generate all project files needed by the compiler. With a simple click, your PLC program will be compiled and sent to your target, without needing to write a single line of code within Arduino IDE.

1.1 - IMPORTANT REMINDER

It must be understood that applications written using Beremiz and/or Beremiz Project Manager shall not be used in any safety-critical application without a full and competent review, from a certified authority.

Beremiz Project Manager is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY, and without even the implied warranty of merchantability or fitness for a particular purpose.

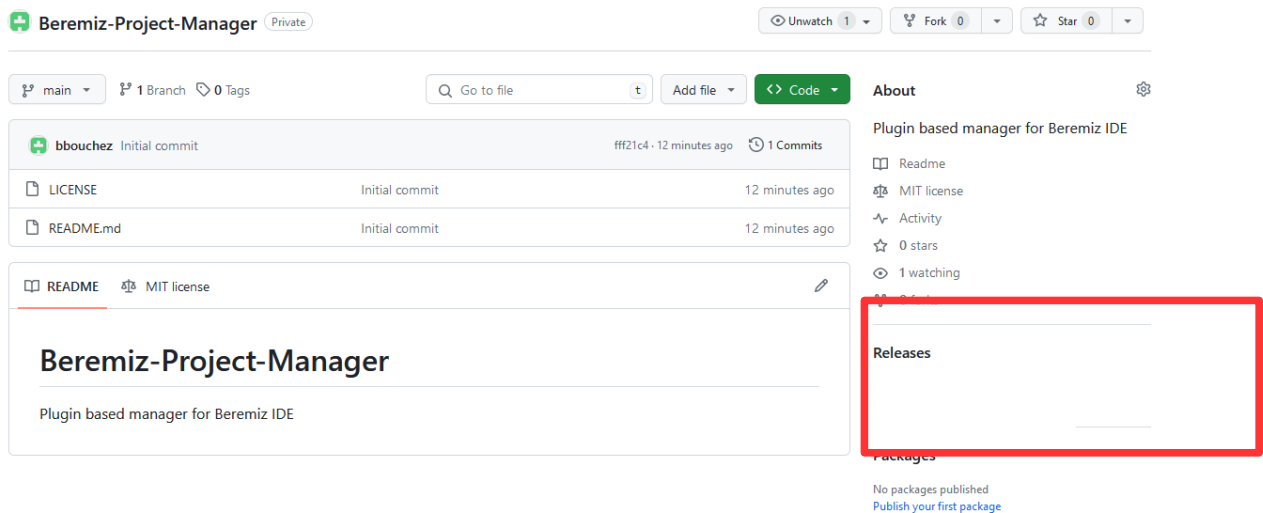
2 - IEC61131-3 terminology

- CONFIGURATION : set of files required by the Programmable Logic Controller function to perform
- RESOURCE : group of PLC programs processing PLC inputs and outputs
- PROGRAM : element written in one of the IEC61131-3 language (Ladder Diagram, Function Block Diagram, Structured Text, Instruction List, Sequential Function Chart) or in C language
- TASK : condition to start execution of a program object. A task can execute a program on a cyclic basis or when a given condition is met (e.g : interrupt)

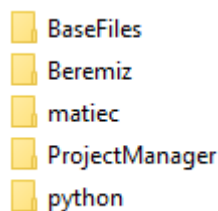
3 - Software installation

3.1 - Installation of Beremiz Project Manager project manager

- Open the Beremiz Project Manager Github page : <https://github.com/bbouchez/Beremiz-Project-Manager>
- Download the latest release. The release is a zip file containing all needed files, including the complete Python environment needed by Beremiz



- On your computer, create a folder in which the complete toolset will be stored. Exact name does not matter, but **make sure there are no spaces and/or special characters in the name** (a name like BeremizProjectManager is a good one)
- Unzip the content of the release file into the folder you have created. Once unzipped, you **must** see the following file structure in the folder. If some folders are missing, then repeat the installation process. You will not be able to use Beremiz Project Manager if the folder structure is not correct.



IMPORTANT : do not change the structure of this folder in any way! Do not rename, move or delete any file or folder, as Beremiz Project Manager relies on relative location of files within this folder.

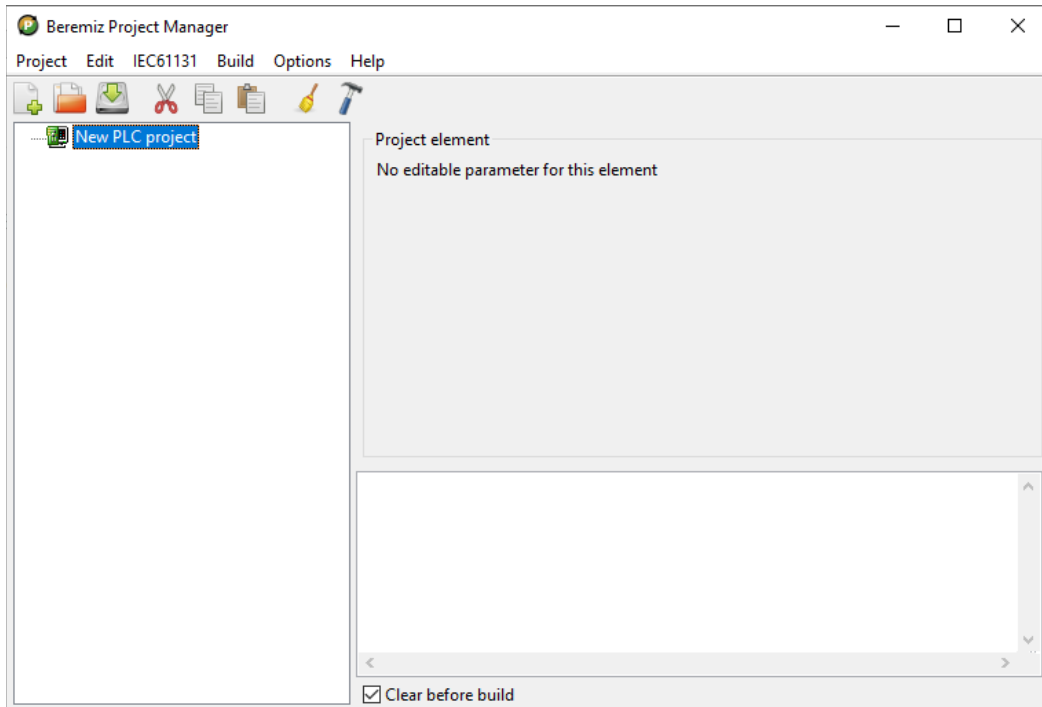
3.2 - First configuration

Once you have copied all Beremiz Project Manager files on your hard disk, you must configure the application.

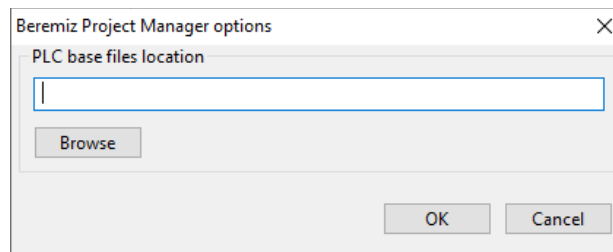
DO NOT ATTEMPT TO CREATE OR EDIT ANY BEREMIZ PROJECT IF THE PROJECT MANAGER HAS NOT BEEN CONFIGURED FIRST!

This configuration step is required only one time after the installation of the toolset. It has not to be done again except if you move the Beremiz Project Manager folder on your system.

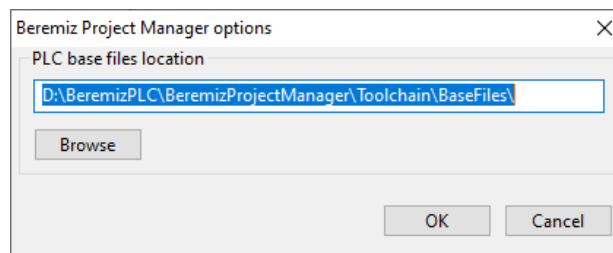
- Go into Project Manager folder, then launch "Beremiz Project Manager" and wait until application starts



- Open "Options" / "General" menu. The following dialog then opens



- Click on Browse button. In the file dialog, select "BaseFiles" folder in the Beremiz Project Manager installation directory and click on OK
- You should now see a complete path like the following (the path can also be entered manually). The path shown here is an example, it may be different on your computer



- Click on OK

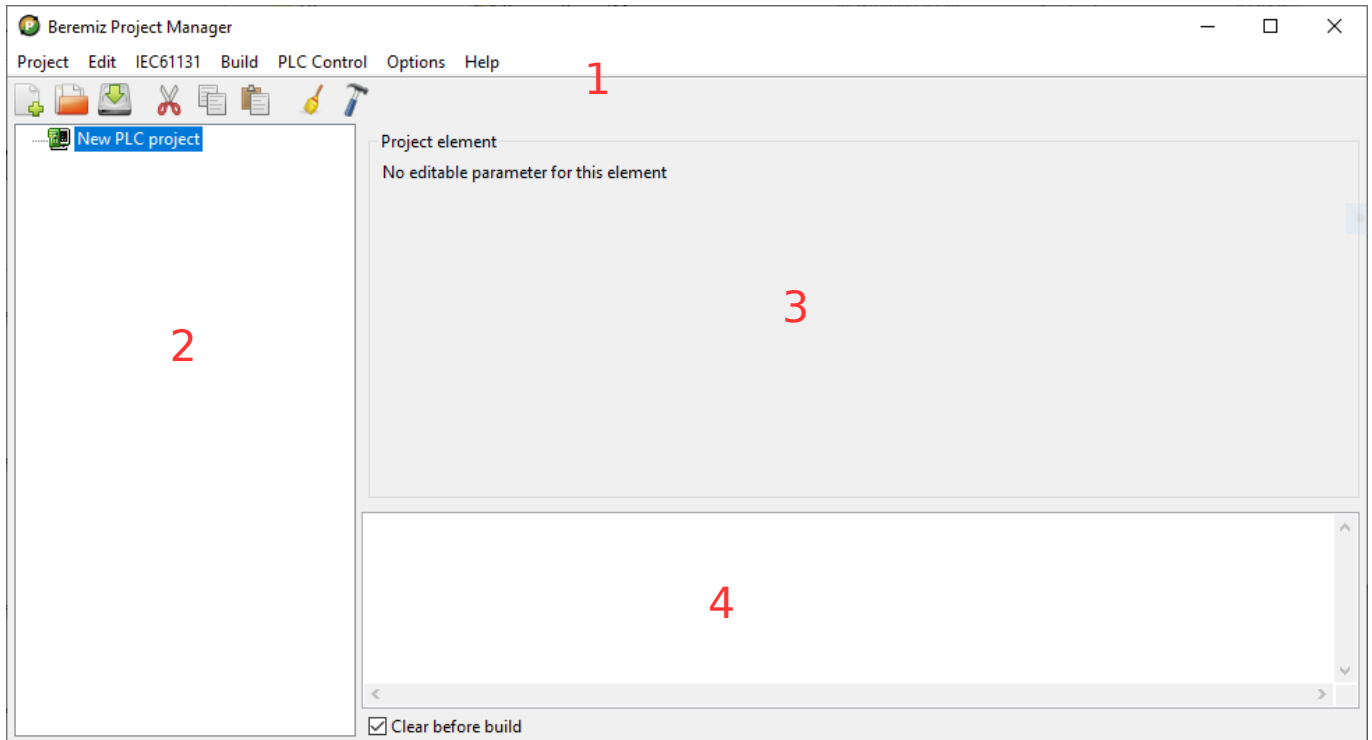
- Close Beremiz Project Manager to store the new configuration

You can now open and use Beremiz Project Manager.

4 - Key concepts of Beremiz Project Manager

4.1 - Main user interface

Beremiz Project Manager main user's interface is split in various areas :



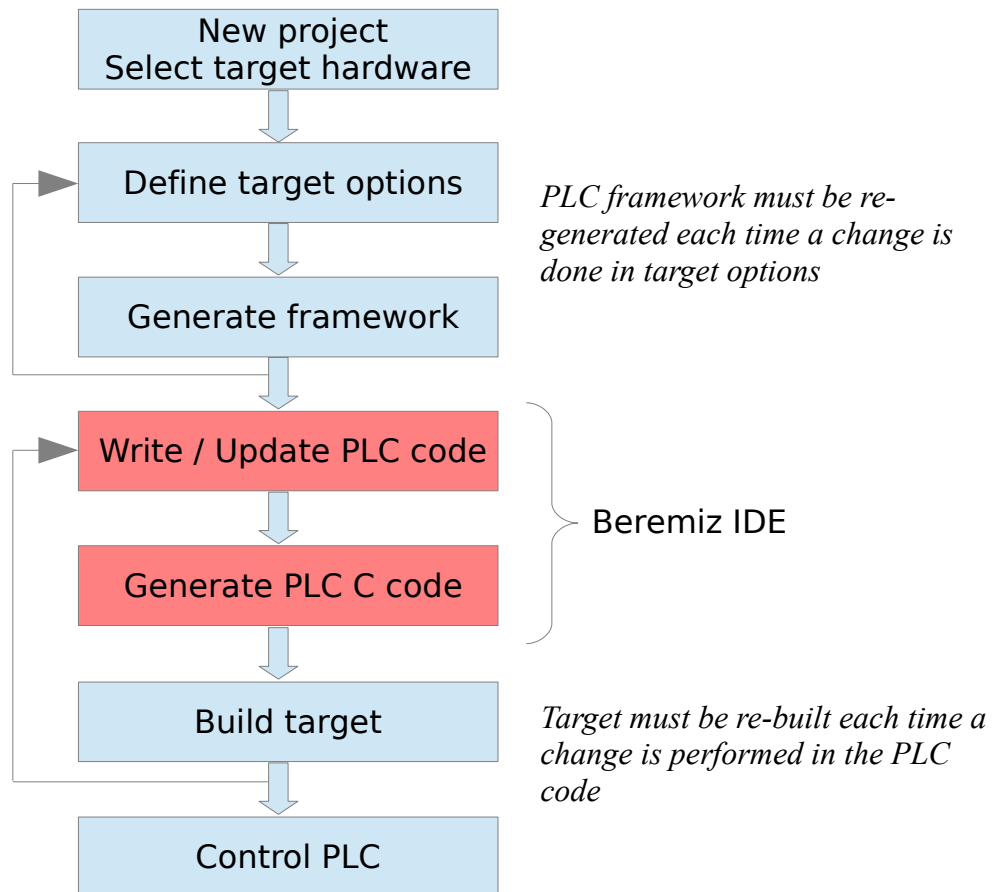
1. The menu / button area is used to trigger the various actions. It must be noted that the buttons only provide a limited subset of commands available in the menus.
2. The main project tree displays the various components from the current project. Top level of the tree represents the PLC project, which will contain target CPU, communication components, I/O submodules, etc...
3. The editor area is used to display and edit parameters from the selected project element in the tree
4. The log window displays various informations, like compilation reports or error messages

4.2 - Project elements

A PLC project managed by Beremiz Project Manager is a technically a folder on your hard disk. This folder contains multiple files (project configuration files, runtime and framework for a given target, source files generated by Beremiz and Beremiz Project Manager, etc...)

IMPORTANT : the files generated and located in a project directory shall not be edited, modified, moved, etc... manually! Only Beremiz Project Manager toolchain shall modify these files! If you don't respect this basic rule, you may break your whole project and make it totally impossible to use

4.3 - Processus of a project

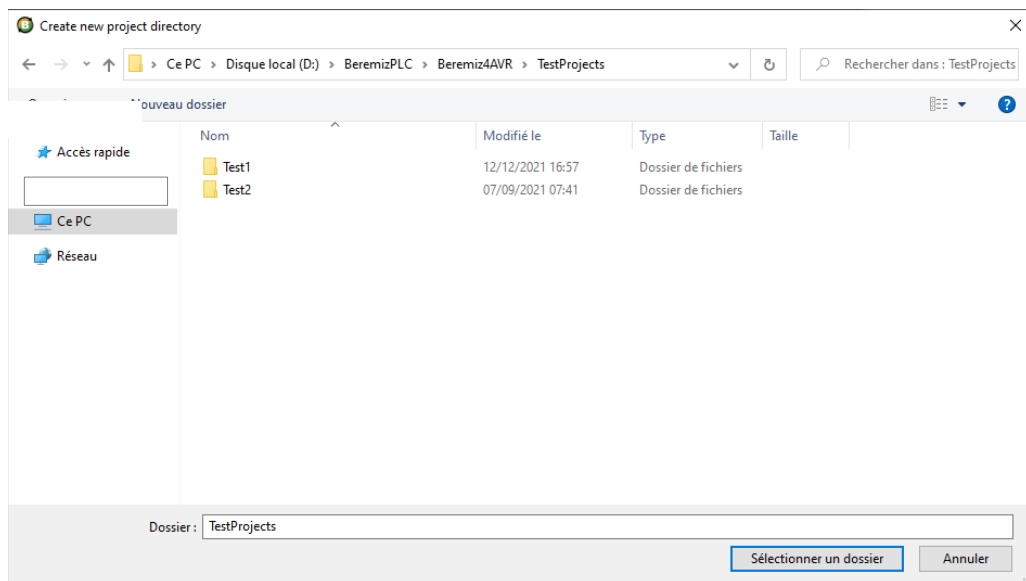


5 - Writing a PLC application

5.1 - Creating a new project

As explained in previous chapter, a Beremiz project is a set of folders and files located in a project folder (the top folder name being the the project name). Nothing useful can be done until a project is created or loaded in Beremiz Project Manager.

Click on "New project" command in File menu in order to open the dialog below.



Projects can be created anywhere on the hard disk. Using this dialog, select the folder in which you want to create the project. Click then on "New folder" button and enter the desired name for the project. Make sure that selection bar is on the newly created project.

It is highly recommended to avoid spaces and/or special characters in the project names.

Click then on "Select folder" button (bottom right of the dialog) to confirm project creation and close the dialog.

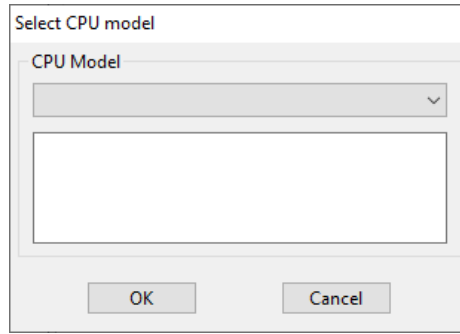
5.1.1 - Target selection

When you create a new project, you must specify on which target the PLC will be run. This step is mandatory as Beremiz Project Manager must perform some specific operation for each target.

Once a target has been set for a PLC project, it can not be changed (future versions of Beremiz Project Manager may include an "Import project" function)

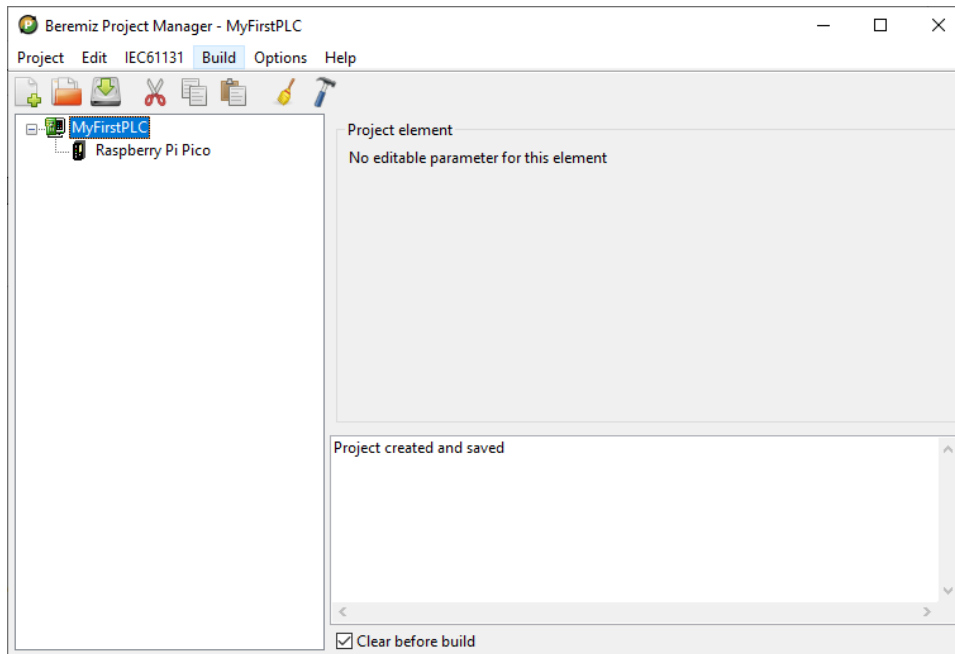
This step is the reason you need to configure location of PLC base files before you can use Beremiz Project Manager: each target uses different files and Beremiz Project Manager needs to know where they can be found before it can create the project.

Once the project folder has been created, the following dialog is displayed



Select the PLC CPU for your project using the dropdown list then click on OK (in our example, we have chosen a Raspberry Pico target)

Once the PLC CPU has been selected, the Beremiz Project Manager copies automatically a first set of files into the project folder.

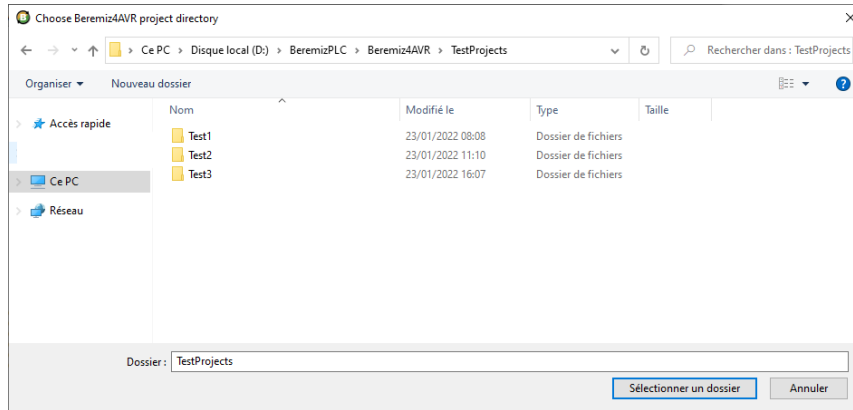


Make sure that the log window indicates "Project created and saved" and that no error is being reported.

5.2 - Loading an existing project

To load a Beremiz Project Manager project, click on "Load project" in "File menu"

In the dialog that opens, browse to the folder with name of the project you want to load. The following screenshot shows an example with 3 projects.



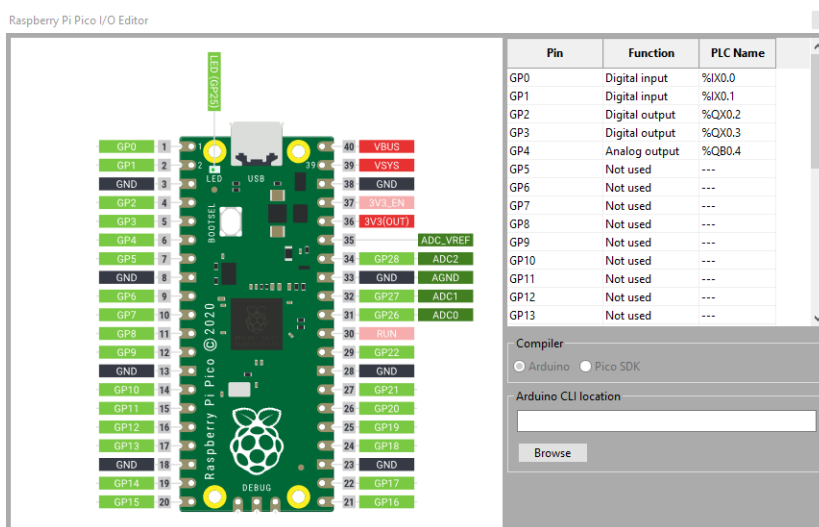
Select the project folder containing your project (Do not go inside the project folder! Remember that folder is the project...), then click on "Select a folder" button. This will load your project in Beremiz Project Manager.

5.3 - Target configuration

Each PLC target supported by Beremiz Project Manager has specific options and capabilities. The key concept of Beremiz Project Manager is the use of "plugins" dedicated to each PLC target. These plugins are in charge of the management of the configuration and generation of related source files for each target.

Double click on the CPU icon in the project tree to open the target editor. Note that each target has a dedicated editor, with specific commands and capabilities. **Please report to the plugins user's manuals for details for each target.**

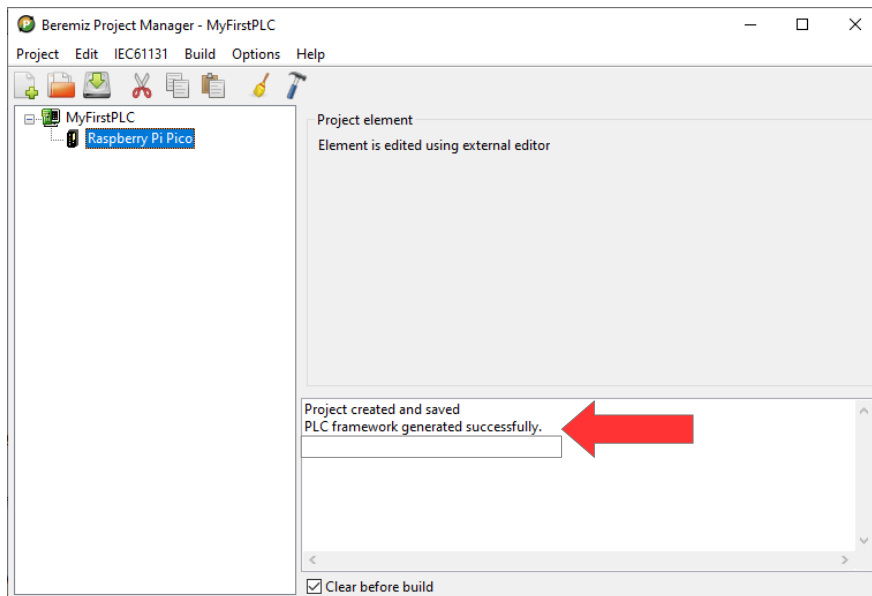
You can see here the Raspberry Pi Pico I/O editor for our example project :



5.4 - Target framework generation

Once a project has been created, it is mandatory to generate what is called the "Target PLC framework". The framework is a set of source files which can be seen as the "PLC Core" needed to run the user PLC application on a given target.

Click on "IEC61131" / "Generate framework" menu. Make sure "PLC framework generated successfully" is indicated in the log window



You can repeat the framework generation step multiple time without any consequences. But this step must be done at least one time before you can generate any PLC application.

IMPORTANT : if you perform any change in a target configuration, you must generate again the PLC framework in order to reflect the changes, before the project can be built.

5.5 - Writing / editing the PLC source code

Once your PLC project is created or loaded in Beremiz Project Manager you can open the Beremiz programming environment. To do that, click on "Open Beremiz Editor" command in "IEC61131" menu. This will launch Beremiz (note that first startup of Beremiz after installation can take a few minutes).

When a new project is loaded in Beremiz, you will see that it contains pre-defined elements ("Project" window in upper left corner) :

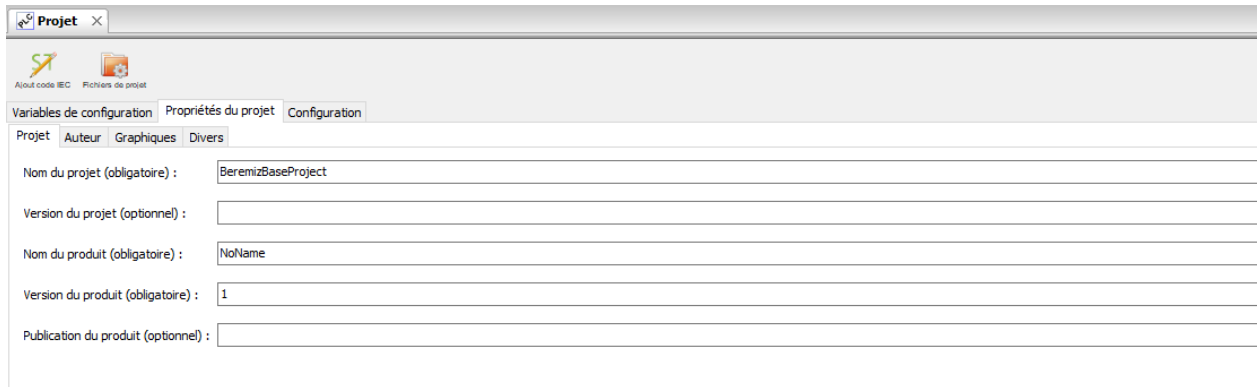
- a project name
- a resource called Resource1. **Never remove or rename this resource object, otherwise it will be impossible to compile your PLC program!**



5.5.1 - Edition of program properties

PLC program properties can be edited by double-clicking on program name (top level of the tree in "Project" cell). Click then on "Project properties" tab (middle window cell)

IMPORTANT : do not change ANY of the values in the "Configuration" tab

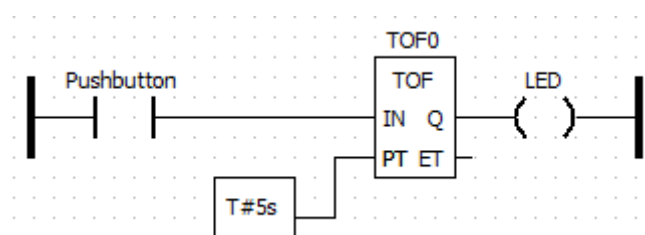


5.5.2 - Writing your first IEC61131-3 PLC program

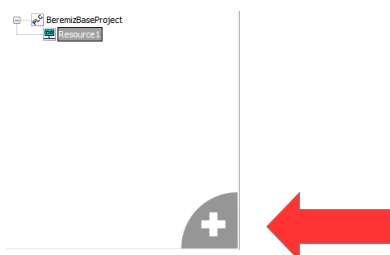
Note a very important thing before going further : Beremiz strictly follows the IEC61131-3 norm. You must have at least basic knowledge of Programmable Logic Controllers before you start to write a PLC program using Beremiz, but be aware that IEC61131-3 is extremely powerful and is able to deal with very complex programs, with complex features.

Our goal here is not explain how IEC61131-3 programming works, we will only take a look at some very basic steps in order to write a first PLC program.

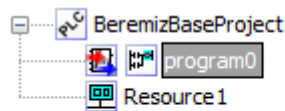
Let's see how to write a simple program using Ladder Diagram (LD) language. This program is simply a monostable timer which will lit a LED when a push-button is depressed. The light will stay active as long as the button is pressed, and will remain active 5 seconds after the button is released.



1 – Click on the "+" symbol in the Project cell and select "Program" in the list



2 – In the dialog that opens, enter the name you want for the program (you can keep the default name if you want) and select LD (Ladder Diagram) as language. Click on “Accept” to close the dialog. The program is now visible in the project, and the edition window opens automatically on the main pane (if not, just double click on the program name in the tree)



3 – We will now create the variables needed by our program. In our case, we need two variables :

- the pushbutton input
- the LED output

Click on the green “+” arrow in the upper right corner twice to create the lines in the variable table. This will create two default local variables.

#	Nom	Classe	Type	Adresse	Valeur initiale	Option	Documentation
1	LocalVar0	Locale	DINT				
2	LocalVar1	Locale	DINT				

Edit the cells in order to get these entries in the table

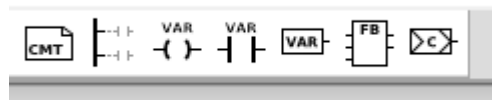
#	Nom	Classe	Type	Adresse	Valeur initiale	Option	Documentation
1	Pushbutton	Locale	BOOL	%IX0.0			
2	LED	Locale	BOOL	%QX0.0			

Note the addresses given (%IX0.0 and %QX0.0) in this table. The “%” sign in IEC61131-3 represents a physical I/O line. These lines are directly related to target hardware and they are different for each target!

These addresses are defined by the target plugin, not by the Beremiz editor.

Please refer to chapter “Target configuration” for details about the I/O configuration process.

4 – We can now “draw” the Ladder Diagram, using the LD toolbar (on top of the screen)



Click on the Contact button to open the Contact dialog. Select “Normal” type and choose “Pushbutton” in the variable list. Drag the contact wherever you want on the edition grid.

Click on the Coil button to open the Coil dialog. Select “Normal” type and choose “LED” in the variable list. Drag the coil on the edition grid near the contact, with some space between the two. Your program should now look like this :

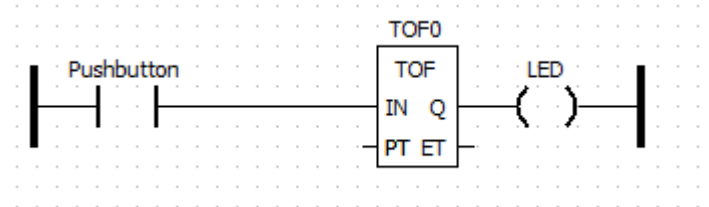


Click now on the “Supply bar” button. Draw a leftside supply bar and rightside supply bar near the contact

and the coil. Using the mouse, drag a connection between the leftside bar and the button, and a second connection between the coil output and the rightside bar.



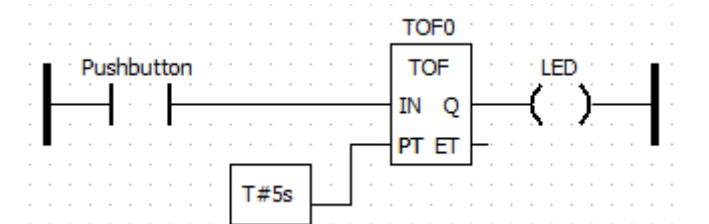
5 – Click on FB button on LD toolbar, then click on edition area where you want to put the function block. Select "Standard function block" in the dialog. Scroll down to "TOF" and drag the function between the contact and the coil. Connect IN input to the contact and Q output to coil using the mouse.



6 – To finish the program, click on VAR button (rectangle with VAR word) in the LD toolbar, then click on edition area. In the dialog that opens, select "Input" for the class and enter the following string in the Expression box : **T#5s**

This expression represents a time of 5 seconds in IEC61131-3 syntax

Drag the variable near PT input (PT = Preset Time) and connect it to TOF0 block. Your program should now look like this :



CONGRATULATIONS : you have written your first IEC61131-3 program in Ladder Diagram language !

7 – **Don't forget to save your work !**

5.5.3 - From the program to a task

Before going further, it is very important to understand what a program is (in IEC61131-3 concepts), since it easily tricks programmers.

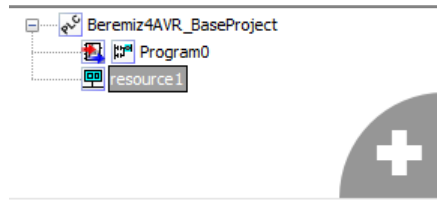
In IEC61131-3, a program is just a top level entity for the PLC source code (in IEC61131-3, a Program is a special kind of POU : Program Organization Unit). There are 3 levels of POU : Program, Function Blocks and Functions. Programs are top level, and they can call Functions Blocks and Functions (which are then "sub programs")

If you want master IEC61131-3 PLC programming, you will need to understand what POUs are (but once you know how to use them, writing complex PLC applications becomes much simpler).

A Program object **does not run by itself**. It will run only if it is triggered by a task !

This may look strange at first sight, but this approach allows to run the same program by multiple tasks if needed, amongst other things.

Let's go back to our program and see how we can run it. Double-click on "resource1" in the project tree to open the resource configuration tool



In the configuration tool, you will see that Beremiz Project Manager Manager has created a default periodic task called PLCTask. Period is set to 40ms (don't worry for now about this value, the PLC tasks are defined and controlled by the Target Plugins in most cases and default value can be kept in Beremiz Editor)

Tâches :				
Nom	Activation	Evènement	Interval	Priorité
PLCTask	Périodique		T#40ms	0

Click on the green "+" sign on "Instances" line (there should be no instances for now) to create a new empty instance.

Instances :		
Nom	Type	Tâche

In the "name" column, enter any name (it must be a IEC61131-3 compliant label, so no spaces or symbols...). In this example, we will call it "Instance1"

Click on the empty cell in "Type" column and choose the program to associate with this instance (in our example, it is "Program0")

Click on the empty cell in "Task" column and choose "PLCTask" (the task created by default)

You should now see this on the configuration window :

Instances :		
Nom	Type	Tâche
Instance1	Program0	PLCTask

5.5.4 - Generation of PLC code

Once you have written a program and associated it with a task, you can generate the target code (do not generate the PLC code before the tasks are created as described before, otherwise Beremiz will report an error). This step transforms your PLC program into a set of C source code files.

To generate the PLC code, click on the "Build" button on top toolbar :



Note that Beremiz generates PLC code in two steps :

- All programs in the project are translated into Structured Text language, whatever the language used (Ladder Diagram, Function Block Diagram, etc...)

- The global program obtained at previous step is translated into C code using MatIEC tool

When code generation is finished, make sure that no error is being reported by Beremiz. You must see "C code generated successfully" in the Beremiz console before you can compile the PLC program. If you see the message "**PLC code generation failed**" in the console, you have to check your program syntax against IEC61131-3 rules.

Do not hesitate to use the "Build" button regularly to check the syntax of your program, especially if you are a beginner in PLC and/or IEC61131-3. Most of the errors are very easy to understand if you read carefully error messages from Beremiz.

Keep in mind that IEC61131-3 languages are less permissive than Python or even C. One of most common problem when you begin in PLC programming is to think that the code generator will understand what you want to do because it is clear in your mind.

IEC61131-3 uses strict syntax to be sure that what is written describes precisely what it is intended to be done. For example : don't try to copy a BYTE variable inside a SINT variable. The code generator will report an error (not a warning), as the two variables don't have the same size and the system does not know what it has to do to generate the missing part (in that case, it does not assume it should use a null field. You have to write that YOU want to do it, using a typecast operator for example)

The "Show IEC code" button allows to see your whole PLC program written entirely in Structured Text :



The "Clean target" button erases all files created previously (needed only if you want to force a rebuild) :



5.6 - Cleaning project files

Under some special circumstances, it can happen that outdated project files remain in the project directory. When the files are copied into the target compilation project directory (used by Arduino environment, see next chapter), they will be compiled with all other project files, which can lead to compilation errors, like duplicate symbols or missing functions.

This happens typically when C extensions are used and renamed (see "Using C with Beremiz" document). For example, if you create a C extension named "0.x" in Beremiz and generate PLC code a first time, Beremiz will create a file called CFile_0.c in the build directory. If you rename later the C extension (for example, "3.x"), Beremiz will now generate a file called CFile_3.c, but it will not delete the previously created CFile_0.c.

CFile_0.c will then be copied into project compilation directory and compiled automatically by Arduino compiler.

The **Cleanup** command in the Build menu erases all files both in PLC code directory and compilation directory. This command is then typically used when you have renamed some C extensions within Beremiz after generating the project files a first time.

IMPORTANT : when this command has been executed in the Manager, you will need to re-generate project files completely :

- Open Beremiz Editor (if it has been closed) and click on "**Build**" button in Beremiz

5.7 - Project compilation

The last step to generate a PLC program is the "Project compilation". This will generate a PLC program which can be executed on the target.

The compilation process is always controlled by the target plugins, the Beremiz Project Manager "only" sending a command to the plugin to start the compilation and reporting the messages from the compiler if needed. So please report to plugin user's manuals for details about compilation process details for the different targets.

However, whatever the target selected for your project, you must check the three following points before you can compile your project :

- the PLC framework must have been generated successfully (see §5.4)
- the C code must have been generated successfully by Beremiz (see §5.5.4)
- the I/O variables names used in the PLC code must have been declared in the target configuration (see §5.3)

This last point is the most common error source : all I/O variables used in the PLC code must have been declared in the target configuration **and** they must have **exactly** the same name. For example, if your PLC code declares and uses an output called %QX0.0, you must be sure this variable exists in the target configuration. Be aware that %QX0.0 is not the same thing than %QX0.00 for example.

Once the three points have been checked, you can start the compilation process. To do that, you can either click on the "Build" button in the Beremiz Project Manager or use the "Build" command in the "Build" menu.

As explained, this will transmit the compilation command to the target plugin. Depending on the plugin, the compilation process can be followed in the log window or in a plugin dedicated window. Please refer to the plugins manuals for detailed explanation about the compilation process depending on the target.

6 - Document revisions

Date	Auteur	Version	Description
05/12/2021	B.Bouchez	1.0	First version

Prepared with OpenOffice Writer software.