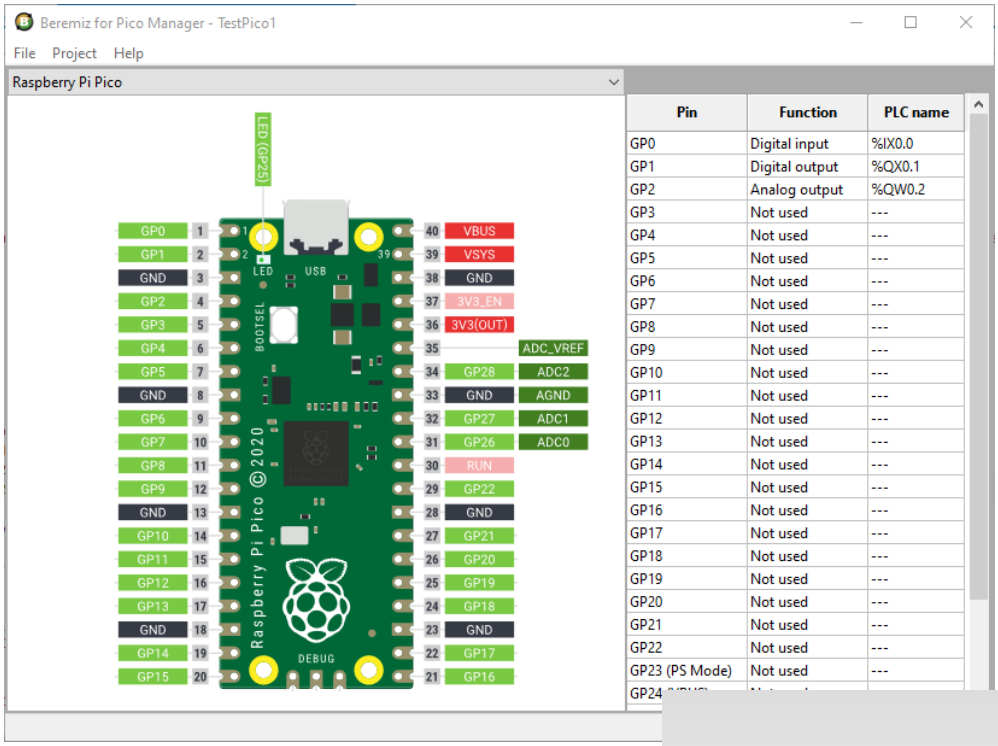
BEREMIZ4PICO
OPEN SOURCE PLC FOR
RP2040 BASED SYSTEMS
USER'S MANUAL



Beremiz
Free Software for Automation

Raspberry Pi Pico version

Contents

1 - Introduction

Beremiz4Pico is a software suite, built around Beremiz, the Open Source IDE for IEC61131-3 based machine automation. It allows you to transform RP2040-based systems (like Raspberry Pi Pico, PicoPLC, SferaLabs Iono RP) into powerful Programmable Logic Controller, programmed in any of the IEC61131-3 languages :

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

Beremiz4Pico is built on an optimized runtime, specifically written for the RP2040. This runtime has been written entirely from scratch, and not simply compiled from original Beremiz target code (written specifically processors used with multi-tasking operating systems)

The software suite is operated from Beremiz4Pico 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 send into Beremiz IDE where you can write the PLC programs.

Once IEC61131-3 code has been written, Beremiz4Pico 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, wtihout needing to write a single line of code within Arduino IDE.

1.1 - IMPORTANT REMINDER

It must be understood that Beremiz4Pico shall not be used in any safety-critical application without a full and competent review.

Beremiz4Pico suite and any code generated from it should be used only for personal applications and/or education. Beremiz4Pico 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

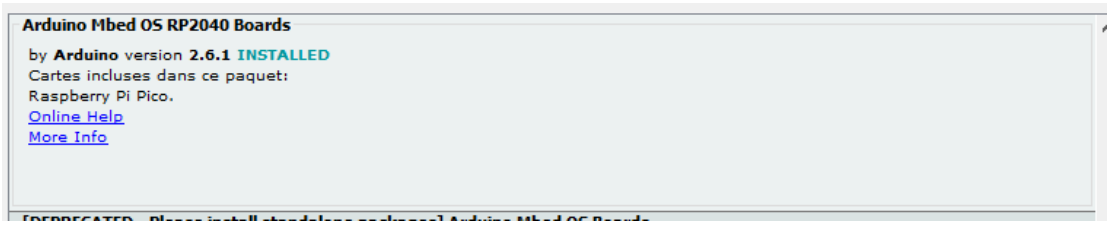
3.1 - Installation of Arduino IDE

Beremiz4Pico relies on Arduino compilation chain. Consequently, Arduino environment must be installed before a project can be compiled and downloaded in the target.

Simply download the Arduino package from : https://www.arduino.cc/en/software and install it on the computer, using the standard way described in Arduino's website help pages.

It is recommended to install Arduino 1.8.xx, as older versions may not recognize or support RP2040 targets.

Mbed OS toolchain for RP2040 must be installed manually : go into "Tools" menu, then click "Board type". Select then "Board manager" and install "Arduino Mbed OS RP2040 Boards" from the Board Manager. You should then see a window with this information in it :



3.2 - Installation of Beremiz4Pico project manager

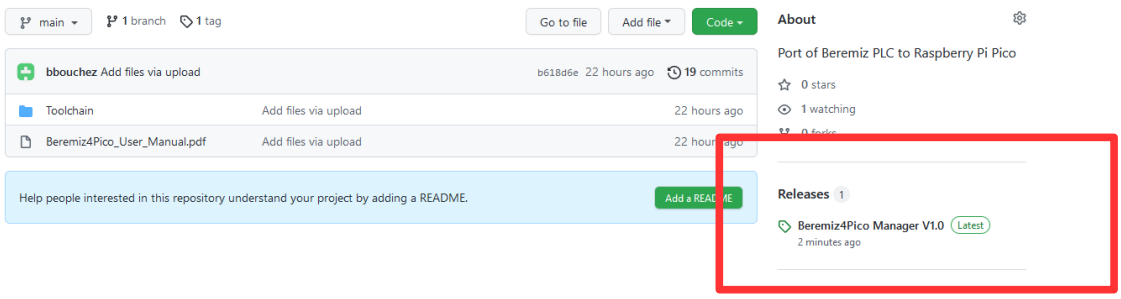
3.2.1 - Use of installer

Sorry, but we have not yet made a Windows installer for Beremiz4Pico. So please follow instructions below for manual installation...

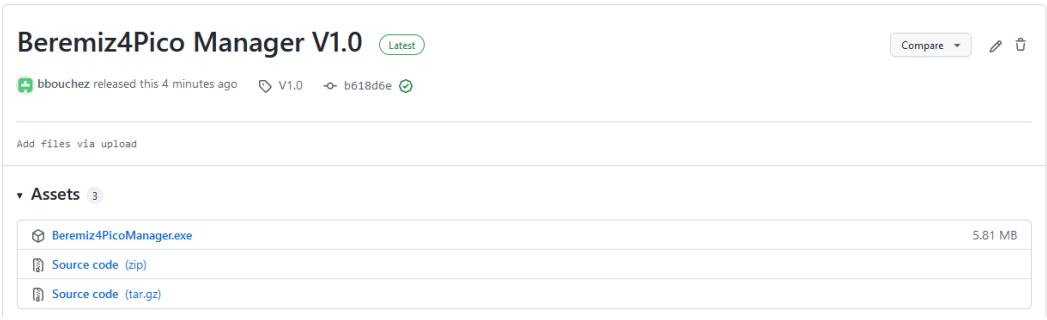
3.2.2 - Manual installation

You can install Beremiz4Pico manually, in case you don't want to use the installer (or can't use it...).

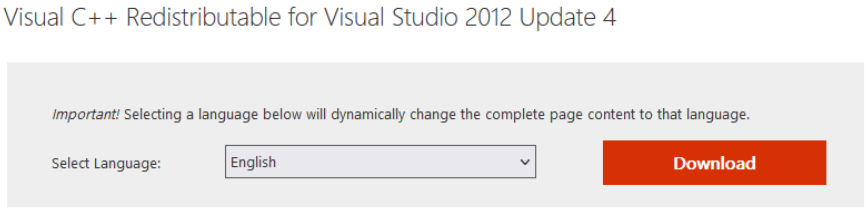
- Create a folder to receive the complete toolset. Exact name does not matter (we suggest Beremiz4Pico), but we recommend to avoid spaces and/or special characters in the name
- Using your favorite web browser, go on this page https://github.com/bbouchez/Beremiz4Pico and look for the Releases panel (see red square below) then click on "Beremiz4Pico Manager"



You will then reach this page



- Download the two following files :
  - Beremiz4PicoManager.exe
  - Source code (.zip)
- Copy the Beremiz4PicoManager.exe file in the folder previously created
- Download and install Visual C++ 11 Runtime from https://www.microsoft.com/en-us/download/details.aspx?id=30679

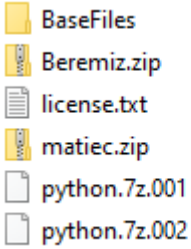


3.3 - Installation of Beremiz IDE

Stable current version of Beremiz requires Python 2 to run properly and is not yet compatible with Python 3 (port to Python 3 is still a work in progress end of 2021). As Python 2 is not supported anymore, it is not recommended to install it using the standard installer as it may interfere with an existing installation of Python 3.

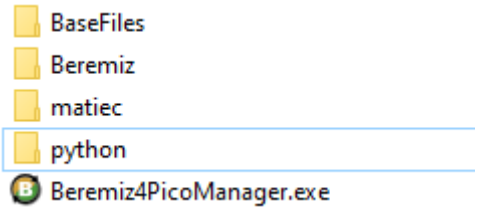
The installation processus described hereafter will install a complete copy of a pre-configured Beremiz IDE along with Python 2 binaries.

- Download and install (if not yet installed) 7-Zip utility from the main repository : https://sourceforge.net/projects/sevenzip/files/7-Zip/. We HIGHLY recommend to install 7-Zip from this place only as you may find modified 7-Zip installers which install other programs ("bloatware" or even viruses or malwares) from other places
- Open the "Source code" zip file downloaded previously and browse it until you reach the following content



- Copy "BaseFiles" folder in the folder created at first step
- Double click on beremiz.zip to open it it and copy the whole folder (not the zip file) into the folder created at first step
- Do the same thing with matiec.zip
- Open python.7z.001 with 7-zip and copy the whole "python" folder in Beremiz4Pico folder

Your Beremiz4Pico folder should now look like this :



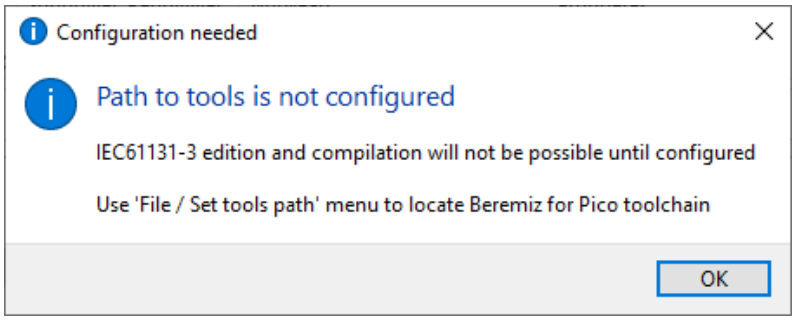
Installation is now finished, you can start to use Beremiz4Pico to create your first PLC programs.

4 - First configuration of Beremiz4Pico

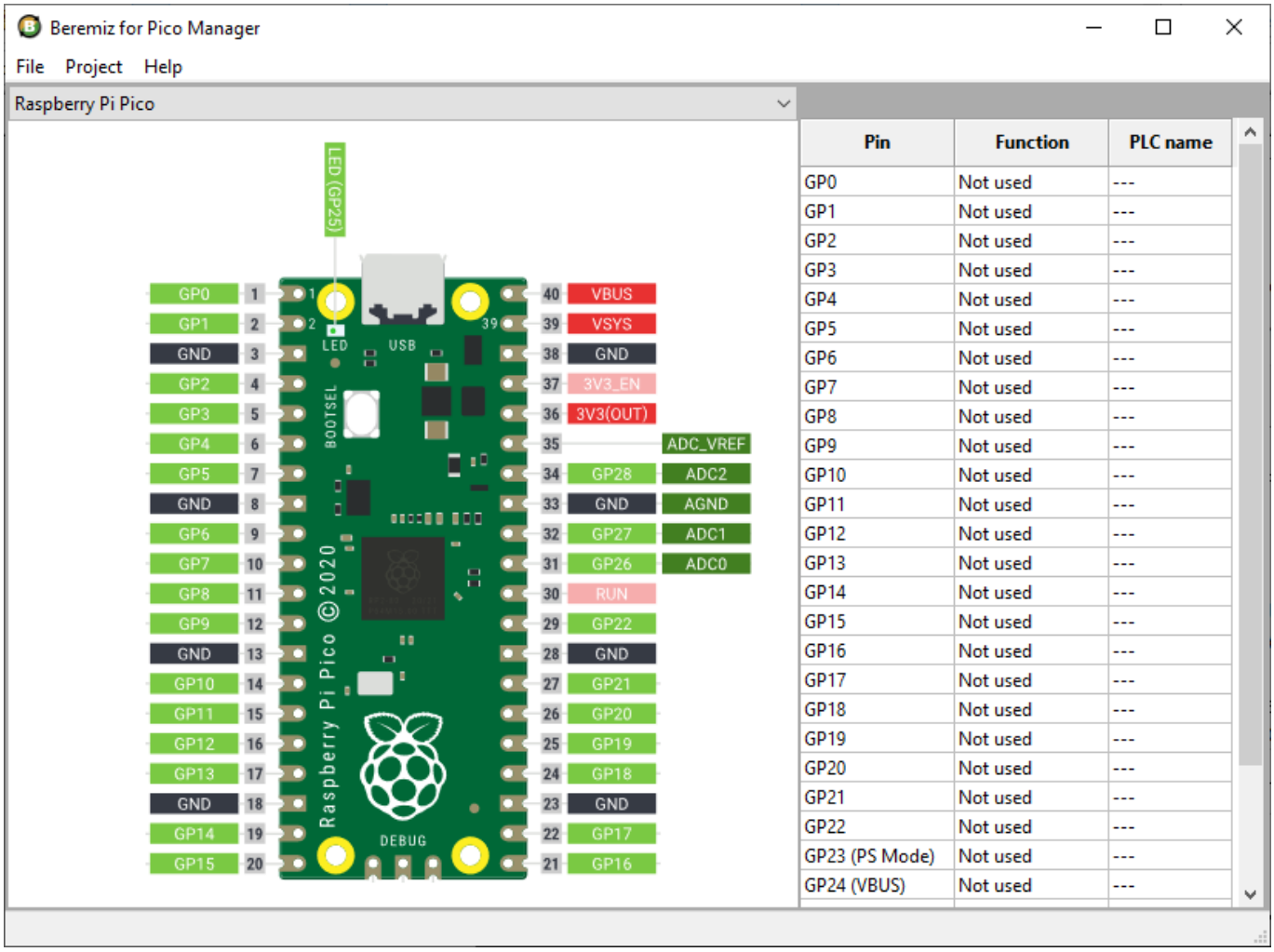
Once the needed tools have been installed, Beremiz4Pico Manager must be configured so it knows where the tools it needs are located.

Launch Beremiz4Pico Manager application. You may get a warning from Windows Defender SmartScreen complaining about a "non recognized application". In that case, click on "More info" and choose "Execute anyway".

You will see the following popup window appear

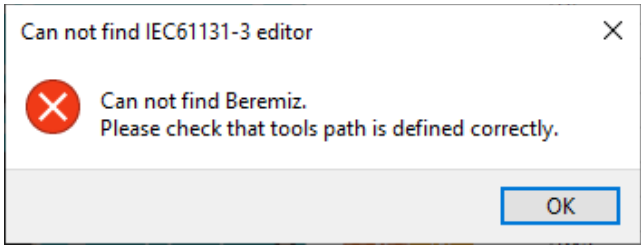


Click on "OK" button to close the popup. The main window of Beremiz4Pico Manager is then displayed.



Open File menu and select "Set tools path". Using the dialog which opens, browse your computer hard disk to locate the folder containing Beremiz tools installed previously (the folder where Beremiz4Pico Manager is installed)

If the following alert messages appears, check that you have correctly installed all files for Beremiz IDE and tools, and that you have selected the top folder containing these tools (and not one one of the folders inside)



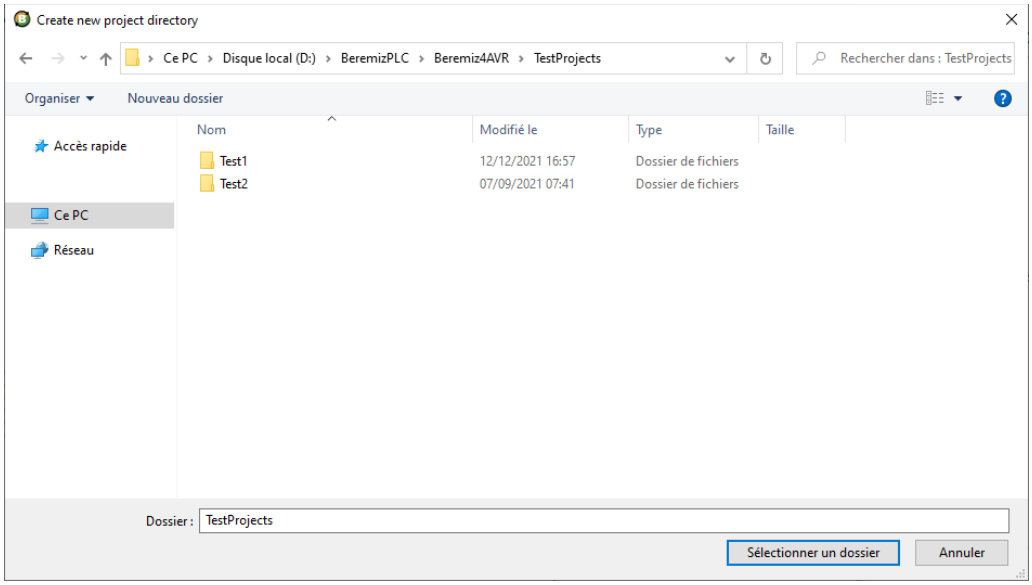
5 - Writing a PLC application

5.1 - Structure of a Beremiz4Pico project

A Beremiz4Pico project is made of a folder containing a set of files. These files should normally not be modified or moved as this may break the project and make it impossible to edit and/or compile.

5.2 - Creating a new project

As explained in previous chapter, a Beremiz project is a set of files located in a folder (folder's name being the the project name). Basically, nothing useful can be done until a project is created or loaded in the manager software.



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.

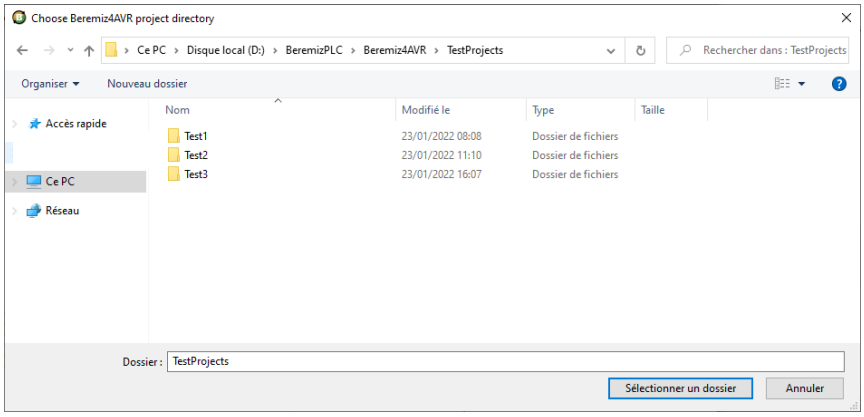
Click then on “Select folder” button (bottom right of the dialog) to confirm project creation and close the dialog.

When a project is created, Beremiz4Pico Manager copies a set of files in the newly created folder.

5.3 - Loading an existing project

To load a Beremiz4Pico project already created, 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 Beremiz4Pico projects.



Select on the project folder containing your project, then click on "Select a folder" button. This will load your project in Beremiz4Pico.

5.4 - Configuration of Physical Input and Output variables

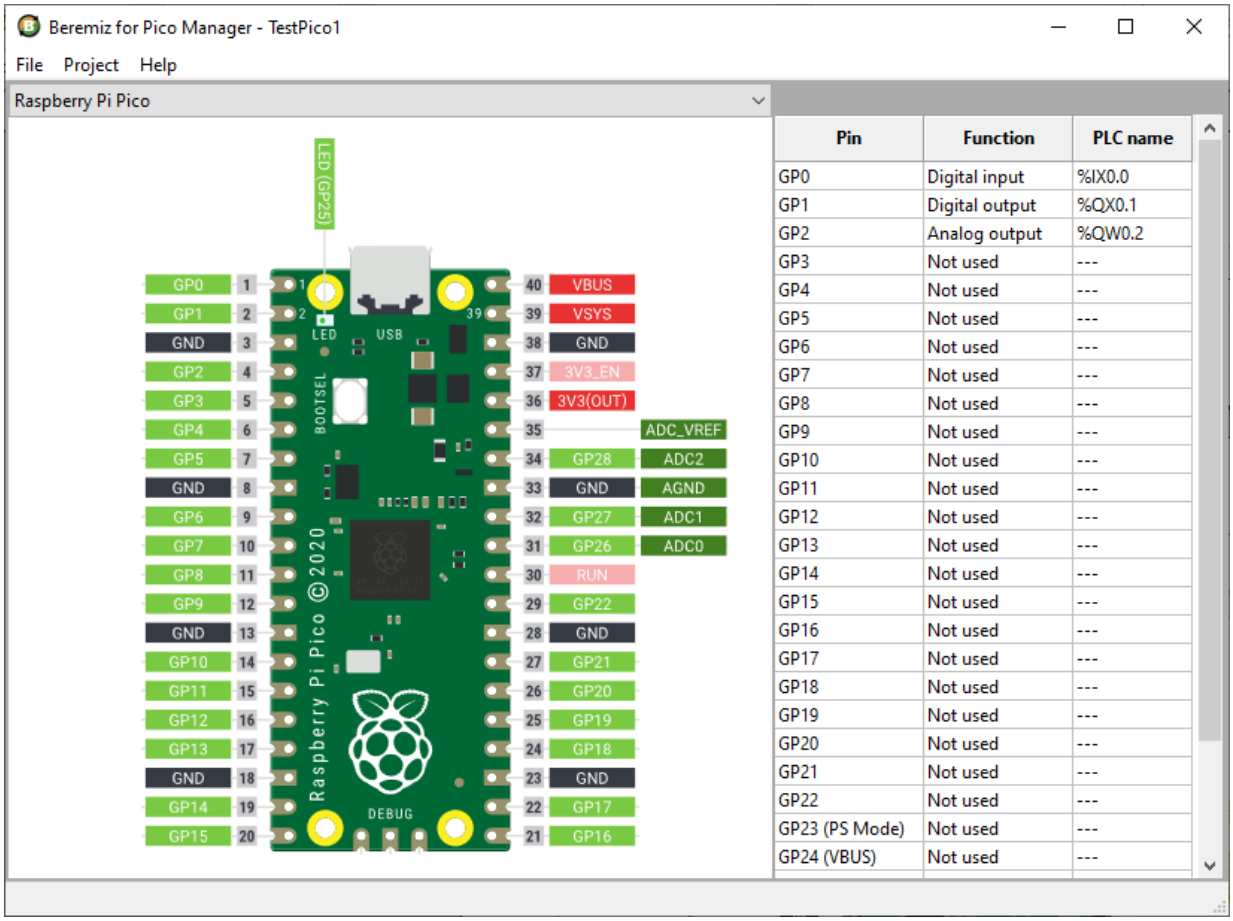
Physical input and outputs (I/O) belong to a specific category in IEC61131-3, name "Directly Represented Variables". The main difference with other variables is that I/Os are forced to be located in a specific memory area, being updated by a dedicated "driver".

The assocation between PLC program and Arduino physical I/Os is done through Beremiz4Pico Manager. This association is done in two steps :

- select the model of RP2040 based board on which the PLC program will be executed. For now, Beremiz4Pico supports three models of CPU boards : Raspbery Pi Pico, PicoPLC and SferaLabs Iono RP. Other boards will be added in future versions of Beremiz4Pico Manager
- select the I/O signal type (logic input, logic output, analog input) for each pin available on the CPU board. Other signal types (analog output, servo, etc...) will be added in coming versions of Beremiz4Pico.

To assign a function to a pin, click on the cell of the row related to a pin, in the "Function" column, then select in the list the function you want for this pin.

Once the function is assigned, you will see the PLC name associated to the pin, in the "Directly Represented Variable" notation starting by the % sign.



Pin	Function	PLC name
GP0	Digital input	%IX0.0
GP1	Digital output	%QX0.1
GP2	Analog output	%QW0.2
GP3	Not used	---
GP4	Not used	---
GP5	Not used	---
GP6	Not used	---
GP7	Not used	---
GP8	Not used	---
GP9	Not used	---
GP10	Not used	---
GP11	Not used	---
GP12	Not used	---
GP13	Not used	---
GP14	Not used	---
GP15	Not used	---
GP16	Not used	---
GP17	Not used	---
GP18	Not used	---
GP19	Not used	---
GP20	Not used	---
GP21	Not used	---
GP22	Not used	---
GP23 (PS Mode)	Not used	---
GP24 (VBUS)	Not used	---

Note that you can change I/O configuration at any moment, even after you have started to write the PLC program. The only requirement is to make sure that you re-generate project files (using "Generate Arduino files" command in "Project" menu) before you compile the project, so the latest I/O configuration is taken into account.

IMPORTANT : only declare Inputs and Outputs really needed by your PLC program. Each I/O variable uses RAM and code in order to be stored and initialized, even if the PLC program does not use it, which leads to useless memory consumption.

5.5 - Writing the PLC program

Once your PLC project is created or loaded in Beremiz4Pico Manager, you can open the Beremiz programming environment. To do that, click on "Open Beremiz Editor" command in "Project" menu. This will launch Beremiz (note that first startup of Beremiz after installation can take a few minutes).

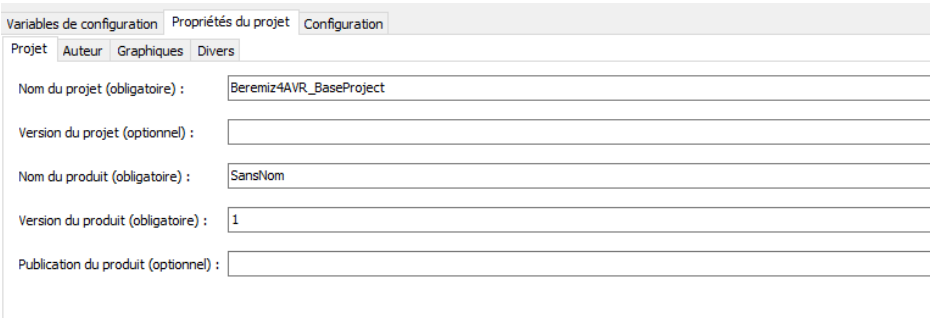
When a new project is loaded in Beremiz, you will see that it contains pre-defined elements ("Project" window in upper left corner) :

- 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

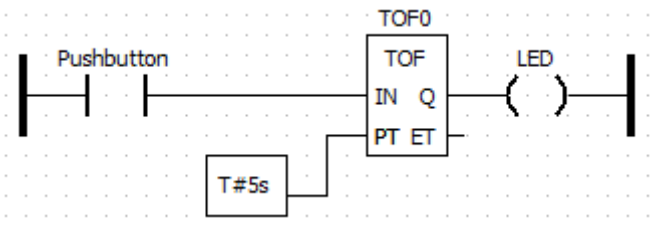


5.5.2 - Writing your first IEC61131-3 PLC program

One the best method to understand how things work is to use them. So let's see how to write a simple program using Ladder Diagram (LD), which will show how easy the task can be.

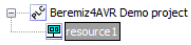
This program is simply a monostable timer which will lit a LED when a push-button is depressed. The light will stay active as long as the button is pressed, and will remain active 5 seconds after the button is released.

As you can see below, the LD program to do that is incredibly easy to understand, as this language has been created for electricians first. Any person able to read an electrical diagram is able to understand the code below.

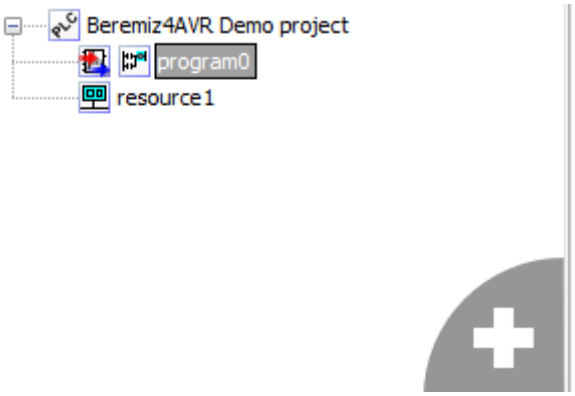


Let's see in details the process to create this program

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



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 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

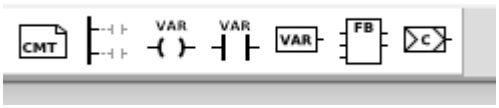
Click on the green "+" arrow in the upper right corner twice to create the lines in the variable table. This will create two default local variables.

1	LocalVar0	Locale	DINT				
2	LocalVar1	Locale	DINT				

Edit the cells in order to get these entries in the table (note that the addresses of the variables as those declared in the Beremiz4Pico Manager)

#	Nom	Classe	Type	Adresse	Valeur initiale	Option	Documentation
1	Pushbutton	Locale	BOOL	%IX0.2			
2	LED	Locale	BOOL	%QX0.8			

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



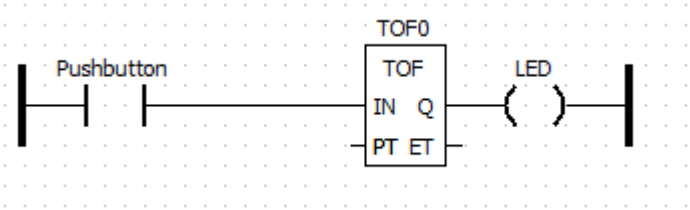
Click on the Contact button to open the Contact dialog. Select "Normal" type and choose "Pushbutton" in the

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.

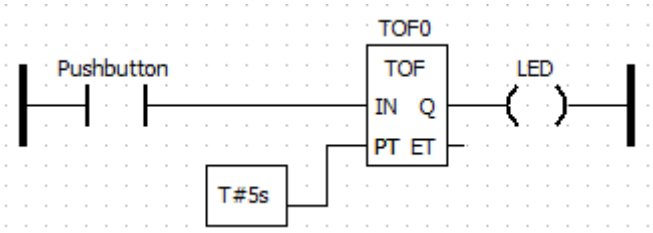
5 – Click on FB button on LD toolbar, then click on edition area where you want to put the function block. Select "Standard function block" in the dialog. Scroll down to "TOF" and drag the function between the contact and the coil. Connect IN input to the contact and Q output to coil using the mouse.



6 – To finsih 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

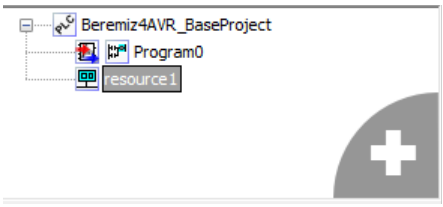
Before going further, it is very important to understand what a program is in IEC61131-3 concepts, since it easily tricks everybody (including me, even after years of IEC61131-3 developments...)

In IEC61131-3, a program **does not run by itself**. It will run only if it is triggered by a task !

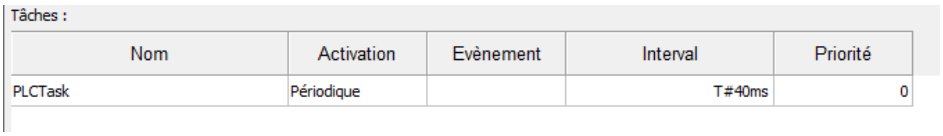
This may look strange at first sight, but this approach allows to run the same program by multiple tasks if needed, amongst other things. It also lead to the concept of POU (Program Organization Unit), which we will not

see in this presentation. 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).

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

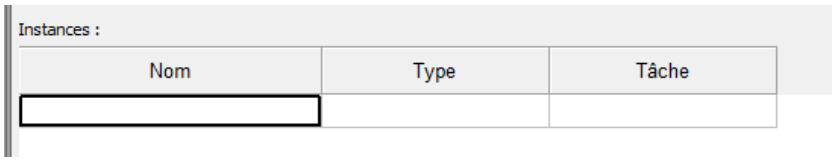


In the configuration tool, you will see that Beremiz4Pico Manager has created a default periodic task called PLCTask. Period is set to 40ms, but note that this value means nothing on Arduino target, as there is no realtime operating system to trigger the task on a regular period. In fact, the Arduino runtime from Beremiz4Pico will run PLCTask as often as possible.


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 :

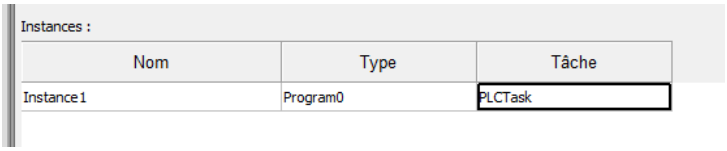
Nom	Type	Tâche

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. This step transforms your PLC program in 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 fisnihed, 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, especilly if you are a beginner in PLC and/or IEC61131-3. Most of the errors are very easy to understand if you read carefully errors 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 quite 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 dont 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 wirtten entirely in Structured Text :

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

5.6 - Target files generation

Once C files have been generated successfully ("C files generated" message in Beremiz console), it is possible to generate target files. This step is required as Beremiz knows absolutely nothing abouty the physical hardware and I/O configuration.

It is then necessary to produce a group of files in complement to the ones generated by Beremiz. These files will form the complete project for the C compiler.

Switch back to Beremiz4Pico Manager (you can keep Beremiz application opened if you want) and click on "Generate Arduino files" in Project menu. Make sure that no error is reported. You can then compile the project.

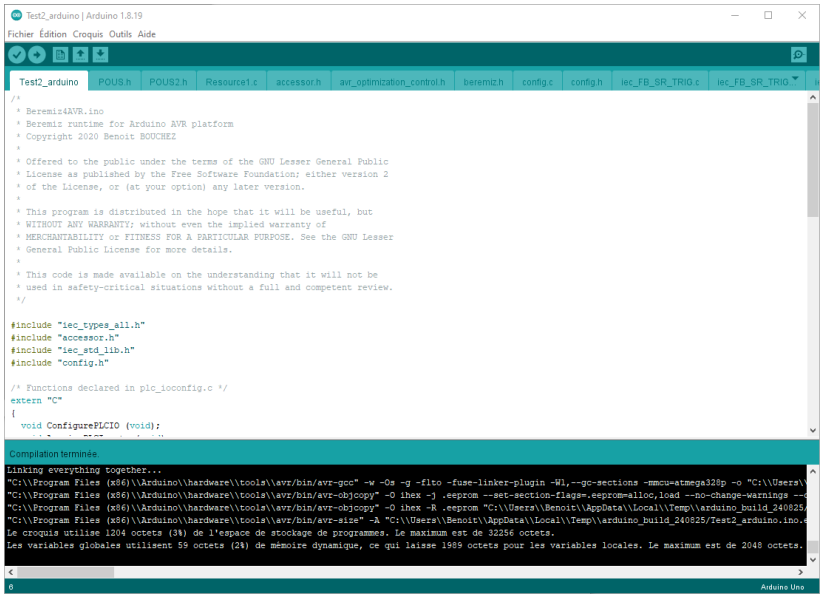
5.7 - Project compilation and target upload

When PLC project files have been generated successfully by Beremiz4Pico Manager, it is possible to compile the whole project and upload it in the target.

- Start Arduino IDE
- Go into File menu and select "Open". Using the dialog, go into your Beremiz4Pico project folder and select the .ino file which has been generated. This file has the same name as your project, followed by

"_arduino.ino". For example, if your project is called "FirstPLC", the file to load is "FirstPLC_arduino.ino"

You will then see a group of files being loaded. These files are the ones which has been generated at previous step.



IMPORTANT : **do not edit any of these files from Arduino IDE editor! These files are generated automatically by Beremiz and Beremiz4Pico.** Any change done within Arduino IDE will be lost the next you generate the project files, and will lead in most case to compilation issues.

Before going further, **make sure that the board selected in Arduino IDE is Raspberry Pi Pico (even if you have selected another hardware in the Manager)**. If this board is not selected, go to "Tools" menu and select "Arduino Mbed OS RP2040 Boards", then "Raspberry Pi Pico". A future version of Beremiz4Pico may do the change automatically, but for now, we have preferred not to interact on Arduino IDE from an external tool.

If you see a different board being selected (bottom right corner of Arduino IDE), go to Tools menu and select the correct board using "Board type" list.

Click on the "Compile sketch" icon or click on "Verify / Compile" command in Sketch menu.

Wait until compilation is completed and check that no error is being reported (you should see a final message telling the size of the in program's memory and the RAM usage by variables.

If an error is being reported, check carefully your PLC code and the configuration declared in Beremiz4Pico Manager. The three most common issues are :

- your PLC code tries to use an I/O object which has not been declared in I/O configuration (for example, PLC uses %IX0.2 while you have declared an input at %IX0.3 on the target)
- your PLC code tries to use function blocks or types which do not exist on RP2040 target (see "Limitations" chapter below)
- your PLC code uses too much code memory and/or too much RAM, and it does not fit in the RP2040

If no error is being reported by Arduino compiler, you can upload your PLC program in the target using the Upload button like any other Arduino sketch (and like always with Arduino, check that serial port used for uploading is the correct one...)

6 - Beremiz4Pico limitations

Beremiz and MatIEC fully comply with IEC61131-3 specifications. However, because of hardware limitations related to the RP2040 microcontrollers, it is not possible to use the following features in Beremiz4Pico :

- "Time of day" related function blocks

- RETAIN variables
- No “live debugging” between Beremiz and physical target

It is also strongly recommended to avoid or strictly limit the use of floating point numbers, as they seriously impact performances of the RP2040, since these microcontrollers do not provide any hardware support for floating point computations.

7 - Document revisions

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

Prepared with OpenOffice Writer software.