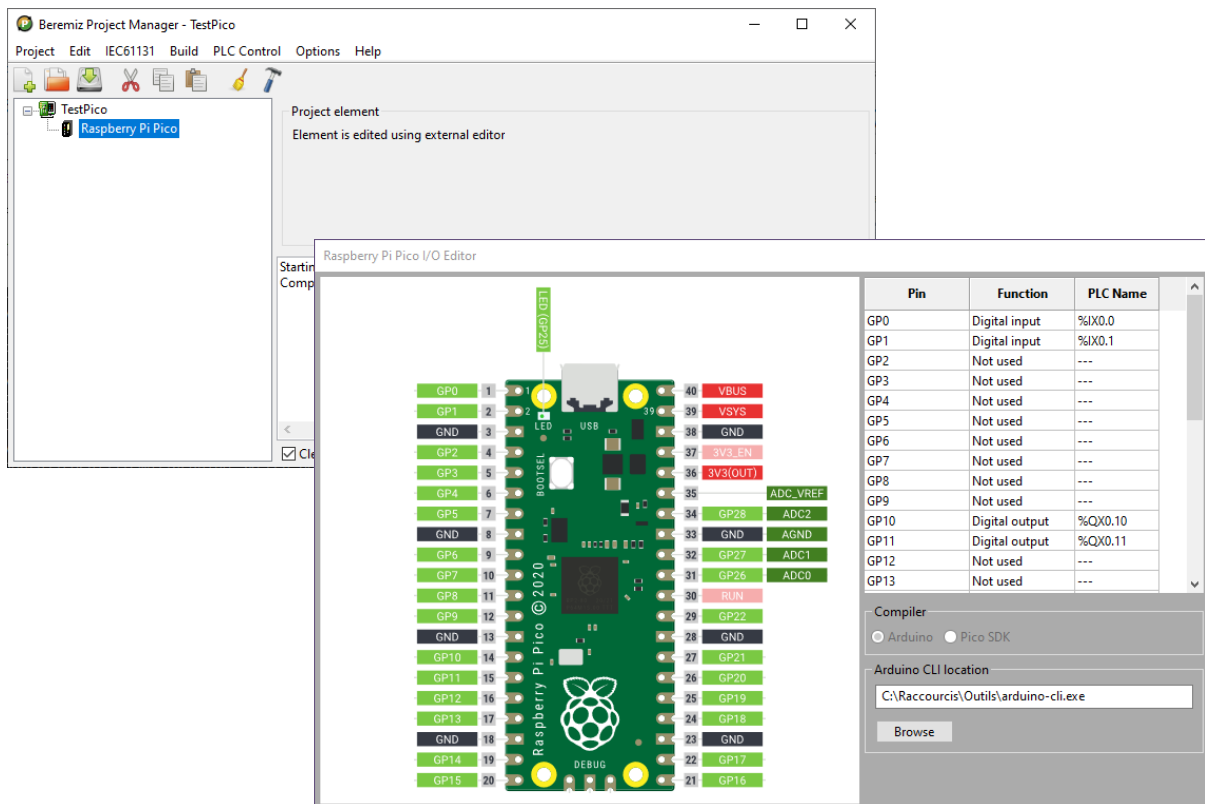


# BEREMIZ PROJECT MANAGER

## RASPBERRY PI PICO TARGET

### CONFIGURATION / INSTALLATION MANUAL



# Contents

---

<b>1 - Introduction.....</b>	<b><u>3</u></b>
1.1 - IMPORTANT REMINDER.....	<u>3</u>
<b>2 - IEC61131-3 terminology.....</b>	<b><u>3</u></b>
<b>3 - Software installation.....</b>	<b><u>4</u></b>
3.1 - Arduino compiler installation.....	<u>4</u>
3.2 - Raspberry Pi Pico SDK toolchain.....	<u>5</u>
<b>4 - PLC project creation for Pico target.....</b>	<b><u>5</u></b>
4.1 - Creating a new project.....	<u>5</u>
4.2 - PLC Framework generation.....	<u>6</u>
4.3 - Inputs/Outputs configuration.....	<u>7</u>
<b>4.3.1 - Arduino CLI location.....</b>	<b><u>8</u></b>
4.4 - Writing the PLC program.....	<u>8</u>
<b>4.4.1 - Generation of PLC application code.....</b>	<b><u>8</u></b>
<b>5 - Building project.....</b>	<b><u>9</u></b>
5.1 - Using Arduino IDE to build PLC project.....	<u>10</u>
5.2 - Location of compiled PLC binary (using Arduino / Arduino CLI).....	<u>11</u>
<b>6 - Raspberry Pi Pico target limitations.....</b>	<b><u>11</u></b>
<b>7 - Document revisions.....</b>	<b><u>12</u></b>

# 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 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

The software suite is operated from Beremiz Project Manager 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 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

Three set of files are needed in order to build Beremiz PLC application on the Raspberry Pi Pico :

- The target files, which are source files and project files needed along the PLC application code. These files will be compiled and linked automatically under control of Beremiz Project Manager
- The target plugin, which is loaded automatically in Beremiz Project Manager when the Raspberry Pi Pico target is selected when creating a new project
- The compiler tools and libraries

The two first sets of files (target files and target plugin) are part of the Beremiz Project Manager package and they don't need to be installed separately, as they are copied automatically when you install Beremiz Project Manager. Please refer to Beremiz Project Manager documentation for details about the installation processus.

The compiler tools must be installed separately from the Beremiz Project Manager. Two different compilers are supported by the Raspberry Pi Pico plugin :

- The Arduino RP2040 compiler
- The Raspberry Pi Pico SDK compiler (will be supported in a coming version)

### 3.1 - Arduino compiler installation

The RPi Pico target plugin can generate source code for the Arduino IDE. However, the plugin also has the capability to launch directly the Arduino compilation chain, so you don't need to open Arduino environment.

The compilation is performed in that case by Arduino CLI tool, which must be installed following steps :

- Download arduino-cli.exe from Arduino website <https://arduino.github.io/arduino-cli> (file is located in the "Installation" page)
- Copy arduino-cli.exe on your system. The file location can be the Beremiz Project Manager folder or another one. The exact location does not matter, but avoid paths with special characters. It is also better to use a path without spaces in it. Note the location of the file as you will need it to configure the plugin
- Open a Windows command interpreter window (Win key + 'R' then enter "cmd")
- Navigate in the folder containing arduino-cli.exe and enter the following commands :

```
arduino-cli core update-index  
  
arduino-cli core install arduino:mbed_rp2040  
arduino-cli lib install servo
```

For each line, wait until the completion of the command. To check that everything has been downloaded and installed properly, enter the following command :

```
arduino-cli core list
```

You must then see something like that (the important line is arduino:mbed\_rp2040)

ID	Installé	Latest	Name
arduino:avr	1.8.4	1.8.5	Arduino AVR Boards
arduino:mbed_nano	2.6.1	3.3.0	Arduino Mbed OS Nano Boards
arduino:mbed_rp2040	3.3.0	3.3.0	Arduino Mbed OS RP2040 Boards

You can also check that **servo** library has been installed correctly using this command

```
arduino-cli lib list
```

Once you have checked everything, you can close the command interpreter window (enter **exit** command or click on the close icon)

## 3.2 - Raspberry Pi Pico SDK toolchain

It is planned in the future to support the Raspberry Pi Pico SDK compiler. However, this solution is not yet working properly and forces user to make complex manipulations to compile generated code especially on Windows platform.

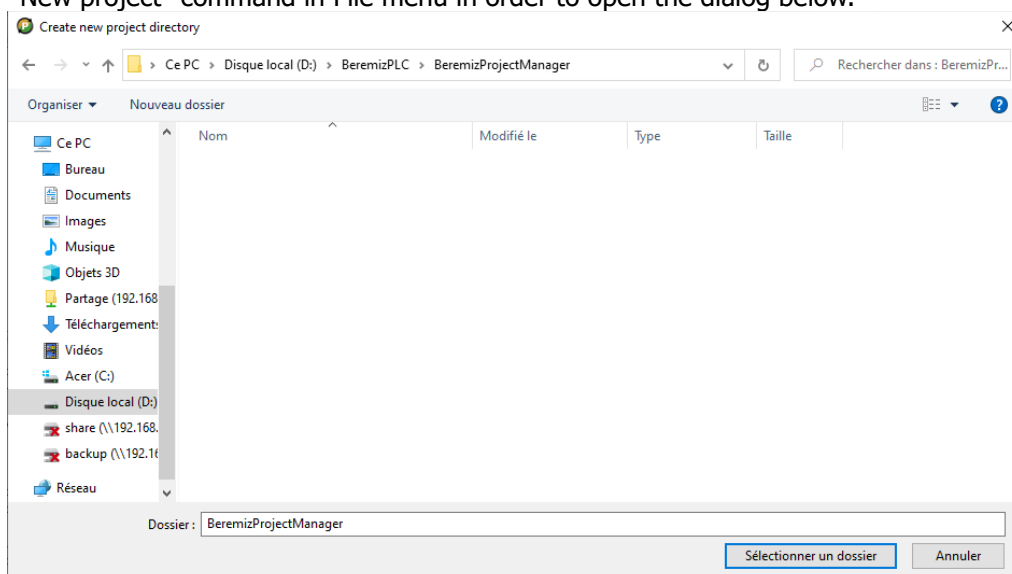
This option is currently disabled, so there is no need for now to install the Pico SDK in order to use it with Beremiz Project Manager.

# 4 - PLC project creation for Pico target

## 4.1 - Creating a new project

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.

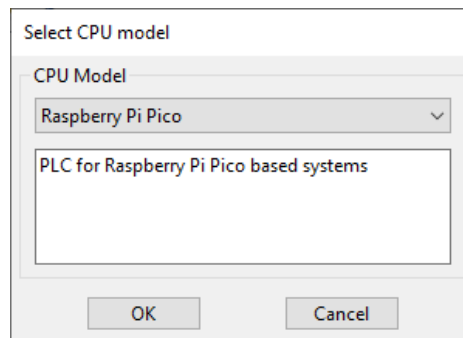
Click on "New project" command in File menu in order to open the dialog below.



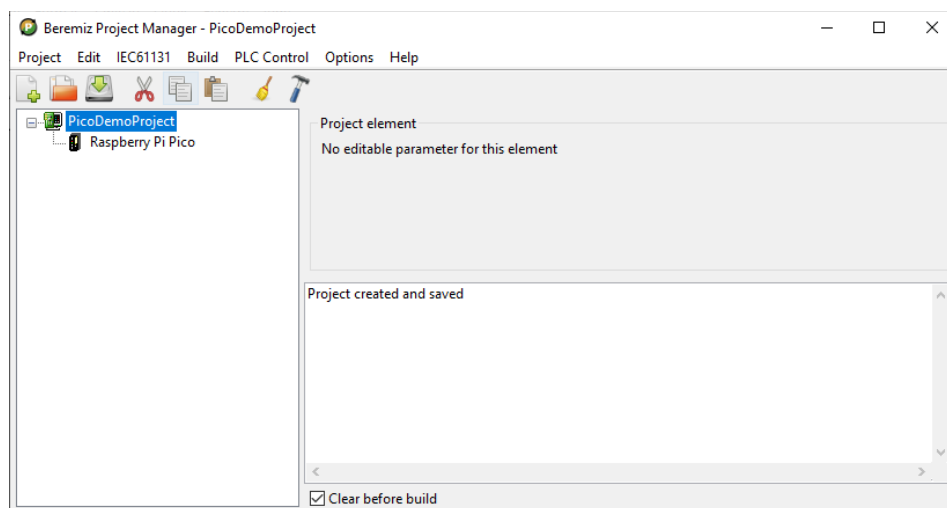
Projects can be created anywhere on the hard disk (you don't have to use the same location as on the previous screenshot). Using this dialog, select the folder in which you want to create the project. Click then on "New folder" button and enter the desired name for the project. Make sure that selection 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.

Once the project is created, the following dialog appears. Select "Raspberry Pi Pico" for the CPU model.



Click on OK to confirm the CPU model. The project will then appear in the main tree of Beremiz Project Manager.



## 4.2 - PLC Framework generation

The PLC framework for Raspberry Pi Pico is provided by default with Beremiz Project Manager and you have nothing special to do to install it on your machine.

However, the framework must be copied along with your PLC application, since this last one requires the PLC framework to run properly. Moreover, the framework may have to be modified depending on options related to the target itself (these options are defined by the plugin editor).

The biggest difference between the PLC framework and the PLC application is that the framework is not supposed to be modified once the target properties have been defined. That's why the framework is copied independantly of the PLC application itself : it is normally copied only one time in your project.

The PLC framework is generated by calling the "Generate framework" from the "IEC61131" menu.

**Don't forget to call this command at least one time before the first compilation, otherwise it will be impossible to compile your PLC application !**

The PLC framework can be generated at any moment, as long as it has been generated BEFORE the first build. The best practice is to generate it before starting to work on the PLC code in Beremiz editor (as the PLC project may need informations from the framework), just after the project has been created.

You can call this command as many time as you want during project creation and edition, as it will never

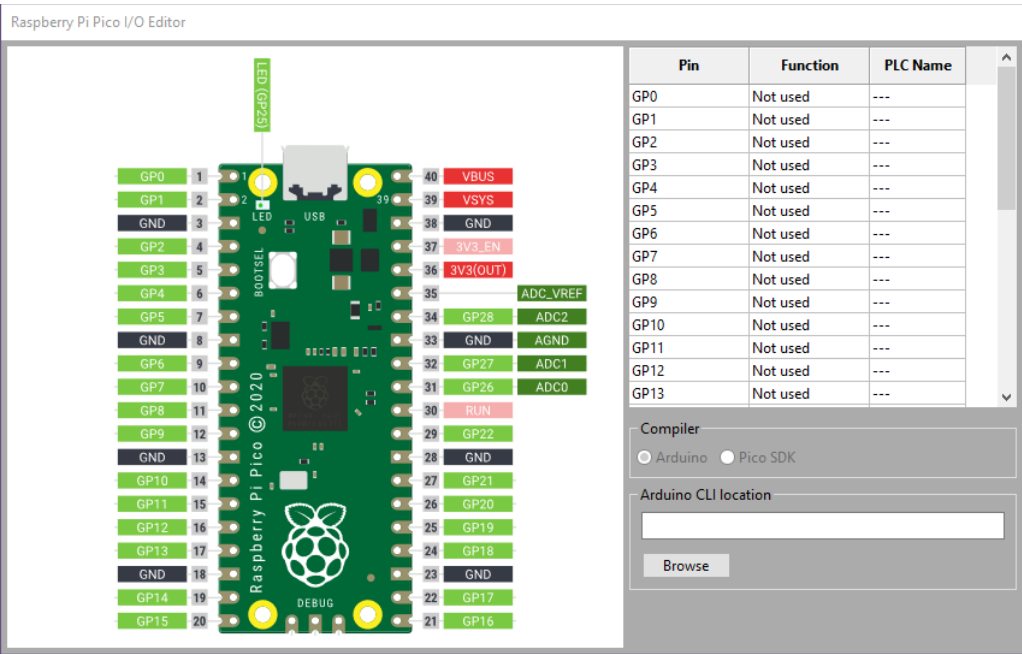
affect the PLC source code itself.

### 4.3 - Inputs/Outputs configuration

The Raspberry Pi Pico has 28 General Purpose Input Output lines, which can be configured in multiple various ways.

These I/O lines must be configured so they receive a "name" which will be used in the PLC program.

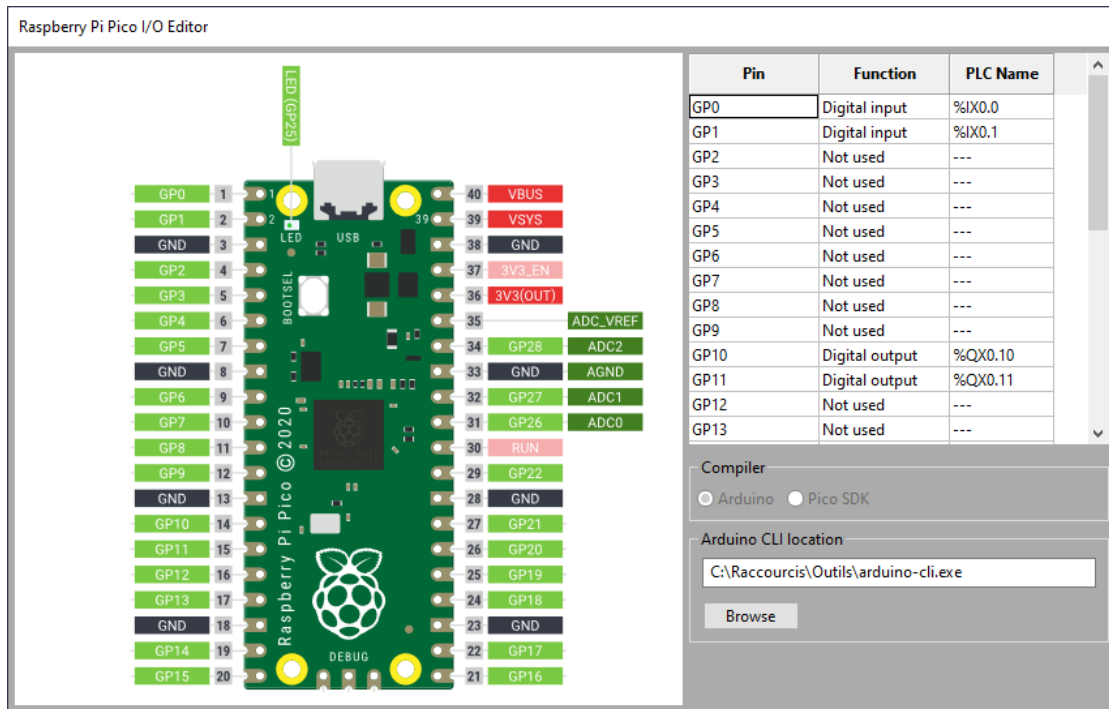
Configuration is performed using a dedicated editor window, which is opened by a double-click on the Raspberry Pi Pico line in the project tree :



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".

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.



Note that you can change I/O configuration at any moment, even after you have started to write the PLC program.

### 4.3.1 - Arduino CLI location

Note that the target editor contains a dedicated edition box which is used to specify where Arduino CLI tool is located. You can enter the path directly or use the Browse button.

Note that if you don't specify the Arduino CLI location using the editor, you will not be able to launch compiler from Beremiz Project Manager (you can always compile your project manually within Arduino IDE in that case)

## 4.4 - Writing the PLC program

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).

Please refer to Beremiz Project Manager manual for details about the project configuration and writing the PLC code.

### 4.4.1 - Generation of PLC application code

Once you have written your PLC program, you will need to generate the target code. This step transforms your PLC program in a set of C source code files.

**THIS STEP MUST BE EXECUTED EACH TIME THE PLC PROGRAM HAS BEEN MODIFIED, BEFORE IT CAN BE COMPILED !!**

If you do not generate the PLC code, it will either be impossible to compile your PLC project or the changes done in the PLC code will be ignored.

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 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 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 IEC61131-3 requires a typecast.

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 - Building project

Once the PLC application code and framework have been generated, it becomes possible to build the PLC application.

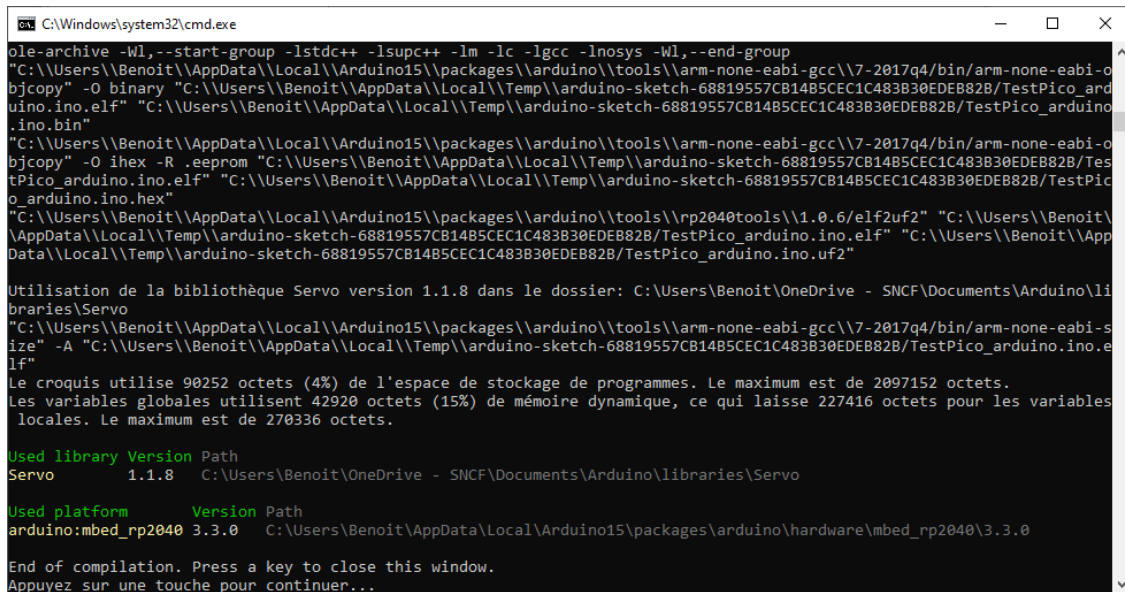
The "Build" command performs the following operations :

- Generation of I/O configuration files (based on the configuration declared in the I/O configuration editor)
- Copy of C generated files generated by Beremiz editor
- Copy of C "extensions" (Function Blocks written by user in C)
- Generation of makefile

Before calling the Build command, it is **MANDATORY** to be sure that C files have been generated correctly (no errors reported) by Beremiz, and that PLC framework has been generated successfully.

The Build command in the Build menu will perform automatically the tasks described herebefore. In the case of the Raspberry Pi Pico plugin, all the files are copied in a dedicated folder with \_arduino added to the project name.

The plugin will then launch automatically the Arduino compiler toolchain. You will then see a window open, showing the progress of the compilation



```
C:\Windows\system32\cmd.exe
ole-archive -Wl,--start-group -lstdc++ -lsupc++ -lm -lc -lgcc -lnosys -Wl,--end-group
"C:\Users\Benoit\AppData\Local\Arduino15\packages\arduino\tools\arm-none-eabi-gcc\7-2017q4/bin/arm-none-eabi-objcopy" -O binary "C:\Users\Benoit\AppData\Local\Temp\arduino-sketch-68819557CB14B5CEC1C483B30EDEB82B\TestPico_arduino.ino.elf" "C:\Users\Benoit\AppData\Local\Temp\arduino-sketch-68819557CB14B5CEC1C483B30EDEB82B\TestPico_arduino.ino.bin"
"C:\Users\Benoit\AppData\Local\Arduino15\packages\arduino\tools\arm-none-eabi-gcc\7-2017q4/bin/arm-none-eabi-objcopy" -O ihex -R .eeprom "C:\Users\Benoit\AppData\Local\Temp\arduino-sketch-68819557CB14B5CEC1C483B30EDEB82B\TestPico_arduino.ino.elf" "C:\Users\Benoit\AppData\Local\Temp\arduino-sketch-68819557CB14B5CEC1C483B30EDEB82B\TestPico_arduino.ino.hex"
"C:\Users\Benoit\AppData\Local\Arduino15\packages\arduino\tools\arm-none-eabi-gcc\7-2017q4/bin/arm-none-eabi-gcc" -x c -c "C:\Users\Benoit\AppData\Local\Temp\arduino-sketch-68819557CB14B5CEC1C483B30EDEB82B\TestPico_arduino.ino.elf" "C:\Users\Benoit\AppData\Local\Temp\arduino-sketch-68819557CB14B5CEC1C483B30EDEB82B\TestPico_arduino.ino.uf2"

Utilisation de la bibliothèque Servo version 1.1.8 dans le dossier: C:\Users\Benoit\OneDrive - SNCF\Documents\Arduino\libraries\Servo
"C:\Users\Benoit\AppData\Local\Arduino15\packages\arduino\tools\arm-none-eabi-gcc\7-2017q4/bin/arm-none-eabi-gcc" -x c -c "C:\Users\Benoit\AppData\Local\Temp\arduino-sketch-68819557CB14B5CEC1C483B30EDEB82B\TestPico_arduino.ino.elf" "C:\Users\Benoit\AppData\Local\Temp\arduino-sketch-68819557CB14B5CEC1C483B30EDEB82B\TestPico_arduino.ino.uf2"

Le croquis utilise 90252 octets (4%) de l'espace de stockage de programmes. Le maximum est de 2097152 octets.
Les variables globales utilisent 42920 octets (15%) de mémoire dynamique, ce qui laisse 227416 octets pour les variables locales. Le maximum est de 270336 octets.

Used library Version Path
Servo 1.1.8 C:\Users\Benoit\OneDrive - SNCF\Documents\Arduino\libraries\Servo

Used platform Version Path
arduino:mbed_rp2040 3.3.0 C:\Users\Benoit\AppData\Local\Arduino15\packages\arduino\hardware\mbed_rp2040\3.3.0

End of compilation. Press a key to close this window.
Appuyez sur une touche pour continuer...
```

The compilation shall end with the lines you see on the screenshot. The compiler shall report the memory used by the PLC application, without any error messages displayed at the end of list.

If you see any messages written in red at the end of the compilation, it means that your project contain an error and must be corrected before it can be compiled successfully.

Once the compilation process is finished, hit any key to close the window.

## 5.1 - Using Arduino IDE to build PLC project

Even if Arduino CLI is the best option to build the project (as you remain in the Project Manager during the whole process), it is also possible to use the Arduino IDE to compile the PLC project if you want to. This can also be a solution if you experience any problem with Arduino CLI.

Note : this method can only be used if the selected target in the configurator is Arduino. **It will not work if Pico SDK is selected.**

In order to compile the PLC project within Arduino IDE, follow these steps :

- start Arduino application
- load the .ino file located in the folder called xxxxx\_arduino located in the project folder (for example, if your project is called PicoDemo, the folder containing the Arduino project is called PicoDemo\_arduino)
- The whole project will then load into the IDE (don't be afraid by the number of files...)



Once the project is loaded, click on the “Verify sketch” button (or use “Sketch verify/compile” menu).

As with Arduino CLI described in the previous chapter, you must check that no error is reported at the end of the compilation process.

**VERY IMPORTANT : never edit ANY file of your PLC project using the Arduino IDE editor.** Don't forget that the files are generated automatically by Beremiz and Beremiz Project Manager, so the changes will be erased automatically the next time you call the Generate Framework, Generate project or Build commands from Beremiz Project Manager.

Moreover, any change in these files may render your project impossible to compile or not compatible with the physical target.

## 5.2 - Location of compiled PLC binary (using Arduino / Arduino CLI)

Once the PLC program has been compiled successfully, you will find the binary file to copy on the RP2040 / Pico Flash memory in the **/plc\_binary** folder of the project directory (folder you created when the project was created as a first step)

The compiled PLC program is the file called *project\_name\_arduino.ino.uf2* in the plc\_binary folder.

To load this file on the Pico (or any RP2040 based system) :

- switch off the system
- connect USB cable to your computer
- activate BOOTSEL signal (on the Pico, push the BOOTSEL button) and hold it while the target system is powered
- the PC will now see a USB drive being mounted (typically containing two files : INDEX.HTM and INFO\_UF2.TXT)
- Copy the UF2 file from /plc\_binary on the Pico USB drive which has been mounted
- Once the file is transferred, the PLC program will start automatically (this unmounts the USB drive)

## 6 - Raspberry Pi Pico target limitations

The Beremiz Project Manager suite fully complies with IEC61131-3 specifications. However, because of hardware limitations related to the RP2040 microcontrollers, it is not possible to use the following features in PLC code written using the Raspberry Pi Pico target :

- "Time of day" related function blocks
- RETAIN variables
- No "live debugging" between Beremiz and physical target (for now...)

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

## 7 - Document revisions

Date	Author	Version	Description
05/10/2022	B.Bouchez	1.0	First version
13/11/2022	B.Bouchez	1.1	Added location of compiled PLC binary (with plugin version > 0.1.0.0)

Prepared with OpenOffice Writer software.