EUROICC

EuroICC®
jPLCPro™

User Manual

Version: 0.64
Build ID:20170224
Last change:February 23, 2017.

Table of Contents

1 Introduction to jPLCPro

In this manual you will find all the information necessary for working with jPLCPro. jPLCPro is integrated work environment at Sun Microsystem **Java**(tm) platform for program development in Functional Block (FB henceforth) language for EPLC. Definition of FB language is according to the IEC61131 standard and it is given in the separate manual.

jPLCPro represents work environment for design, development and translation of FB, and it is supported by all OS which support JAVA virtual machine (Windows XP/NT).

1.1 Features Overview

jPLCPro allows:

- Programming in FB language
- Writing special Functional Blocks in C language
- Downloading executable files to EPLC controllers
- Real-Time online monitoring
- Creation of symbols, constants and variables
- Creation of users and groups
- Creation of new data types
- BACnet objects management
- Configuration of C-Series EPLC controllers
- Configuration of BIOS parameters

1.2 System Requirements

jPLCPro requirements are:

- CPU - 1GHz
- RAM - 64 MB
- HDD - 120MB
- OS - All supporting Java Virtual Machine (JVM)
- Java - 1.8 or newer

Sun Java jre1.8 or newer version must be installed on the computer. If there is no JVM on the computer, installation of JVM will be offered during the jPLCPro installation. For all the information on JVM click here.

1.3 Website

www.euroicc.com

2 Installing the software

The installation of jPLCPro is started by running the setup file “jPLCPro X.Y – yyyymmdd-xxxx.Setup.exe”.

The installation can be stopped at any moment by left-clicking the “Cancel” button.

2.1 Starting Installation

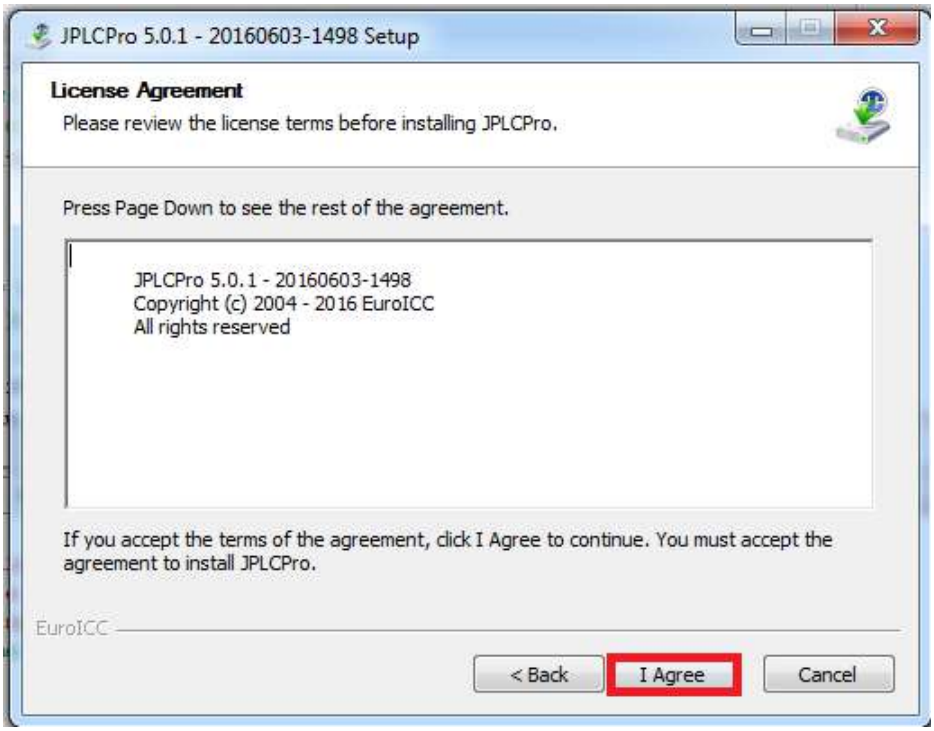
Double-click the setup file.

2.2 First Window

Left-click on “Next” button to proceed.

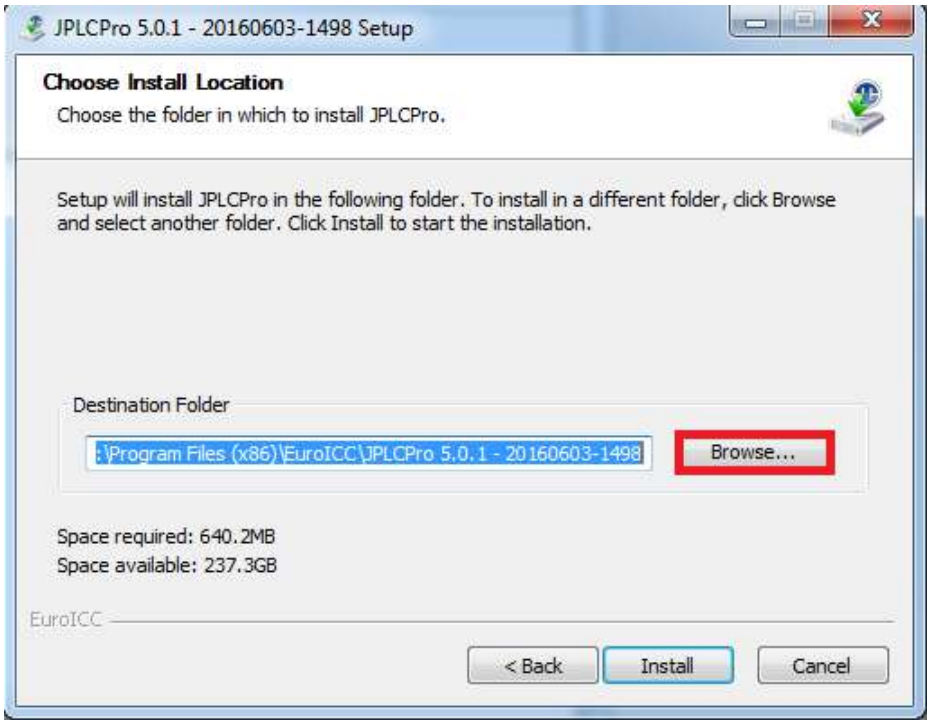
2.3 License Agreement

Review the License Agreement and left-click on “Next” button to proceed.

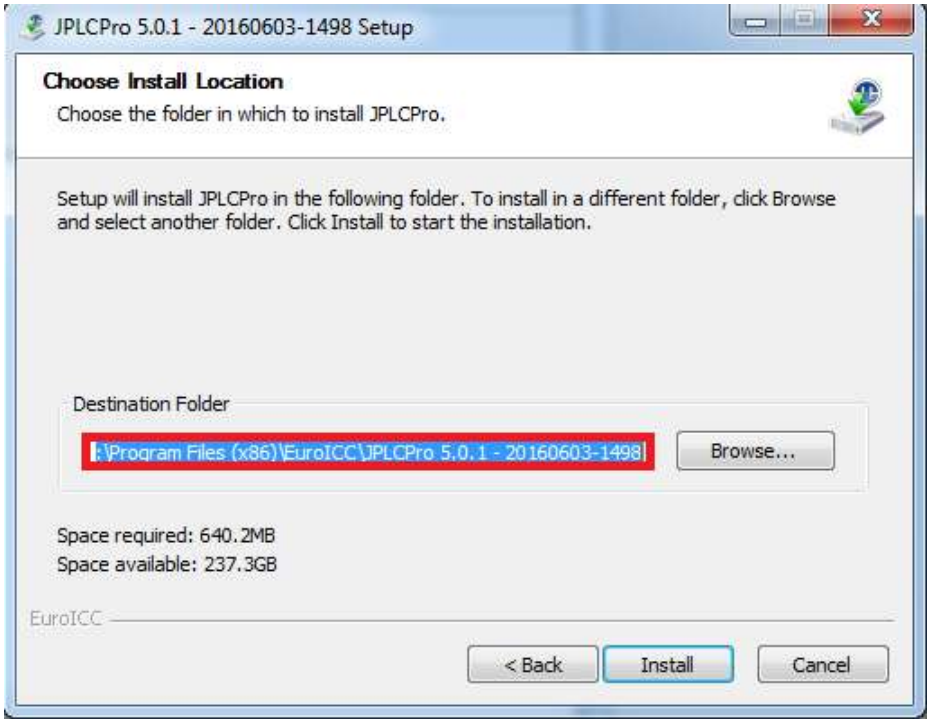


2.4 Install Location

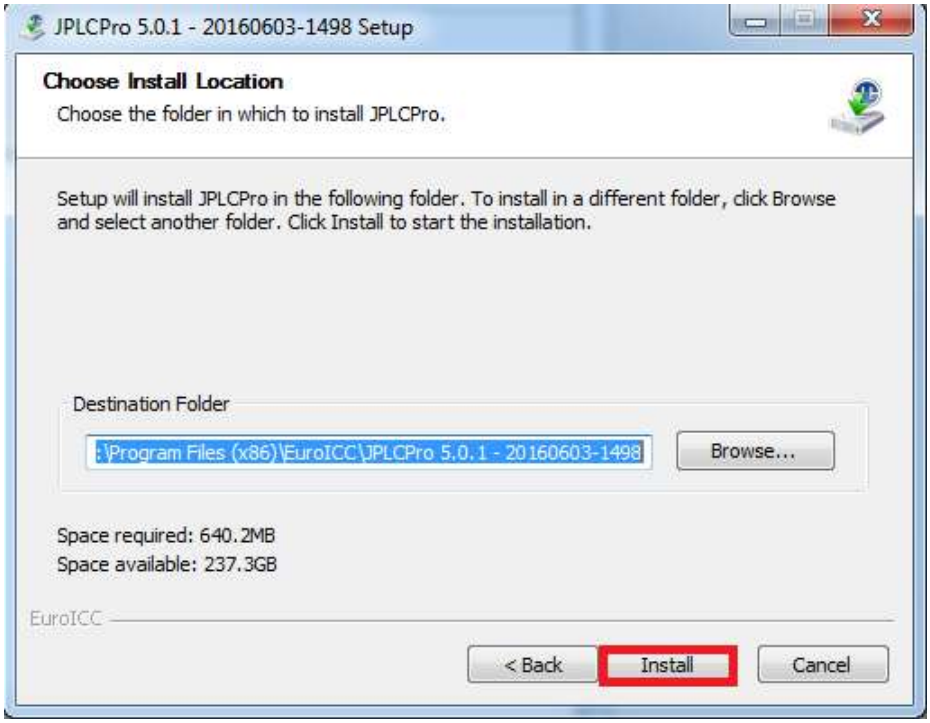
Choose the install directory for jPLCPro. This is done by left-clicking on “Browse” button and selecting the appropriate directory:



Or by typing the entire path:

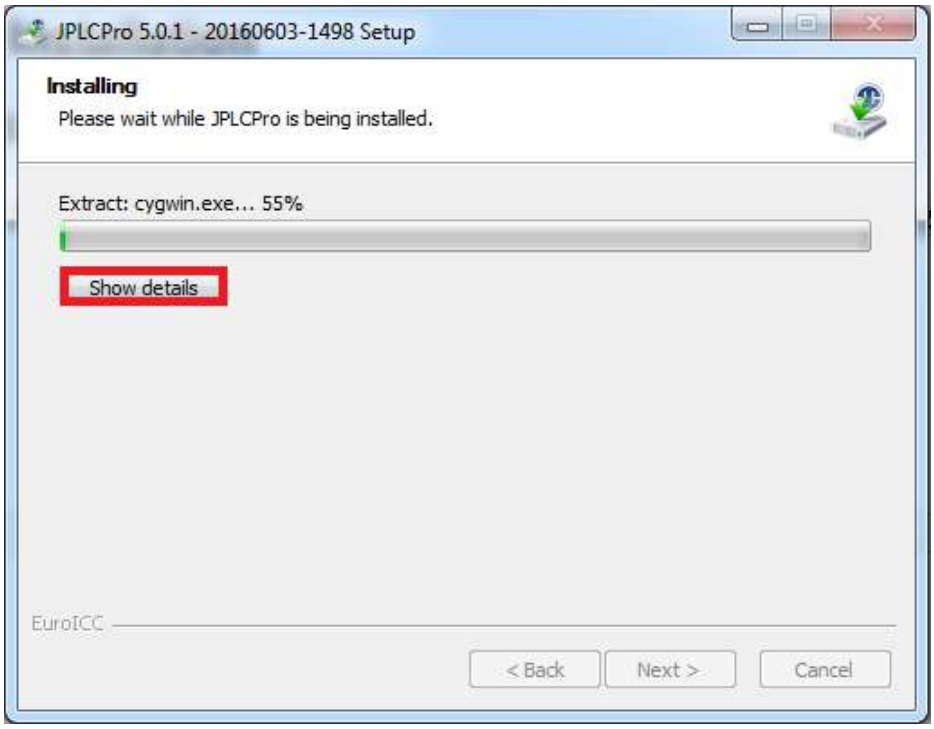


Left-click on “Next” button to proceed to installing jPLCPro



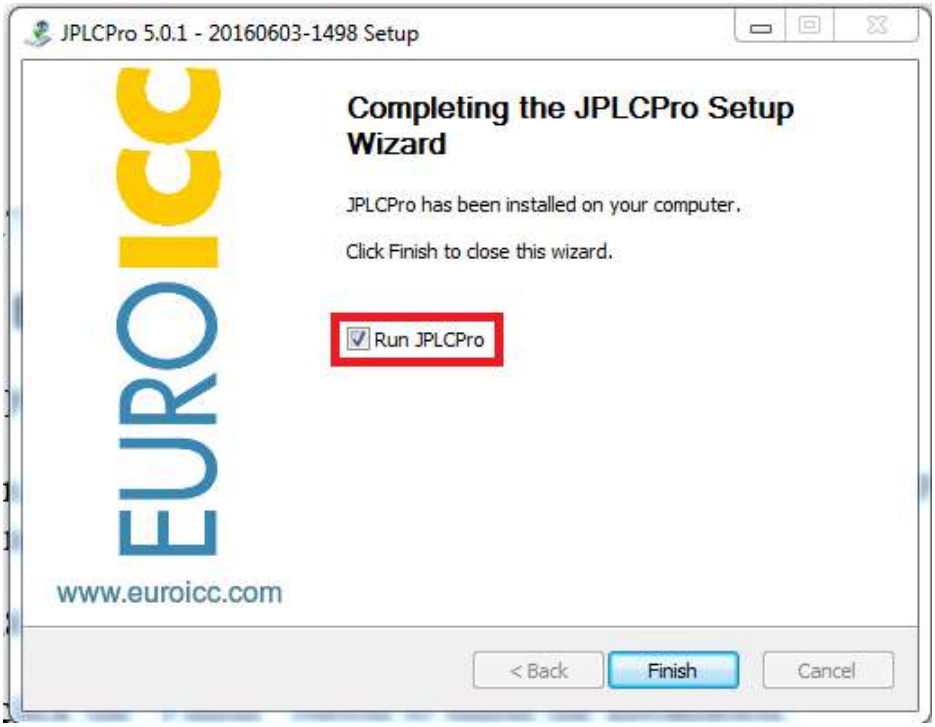
2.5 Installation

Wait for the installation to complete. Left-click on “Show Details” button to show details concerning the installation process.

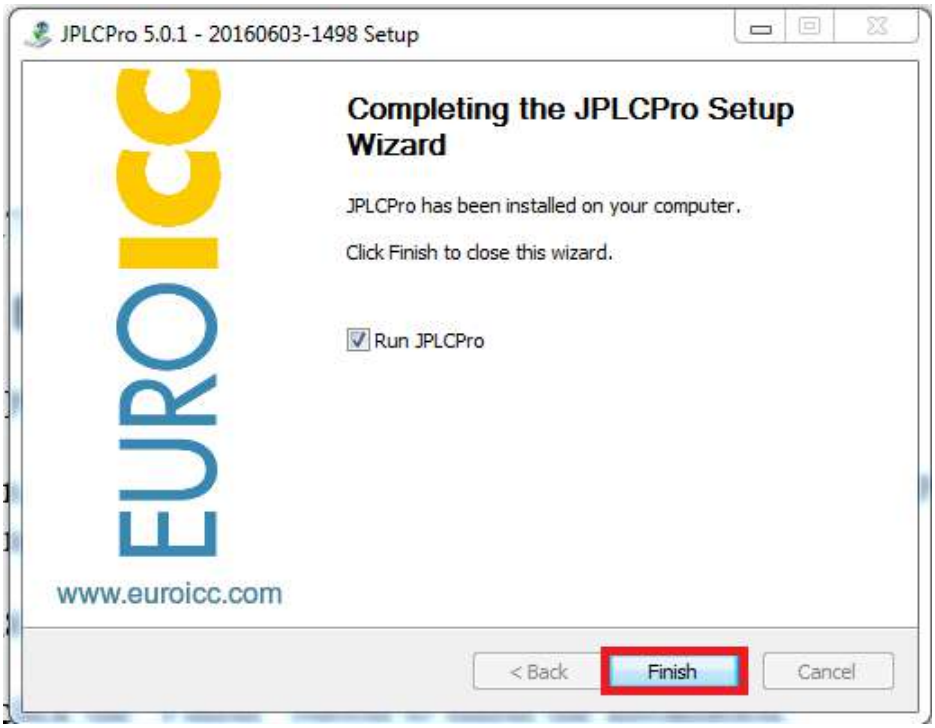


2.6 End

jPLCPro is installed. To run jPLCPro after exiting the installation check the “Run jPLCPro” checkbox by left-clicking on it:



Left-click the “Finish” button to finish the installation.



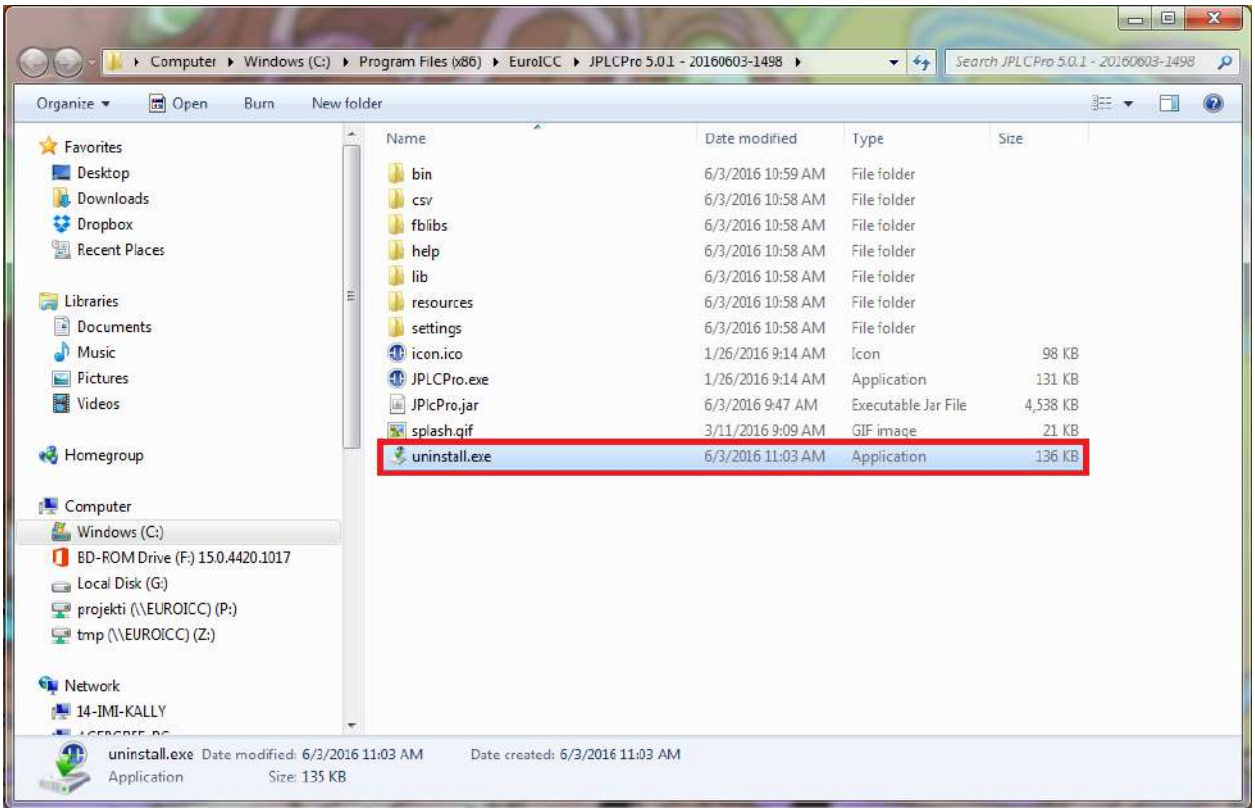
3 Uninstalling the software

Uninstalling jPLCPro is done by running the uninstall file “uninstall.exe”. This file is located in the install directory of jPLCPro.

The uninstallation process is canceled by left-clicking the “Cancel” button.

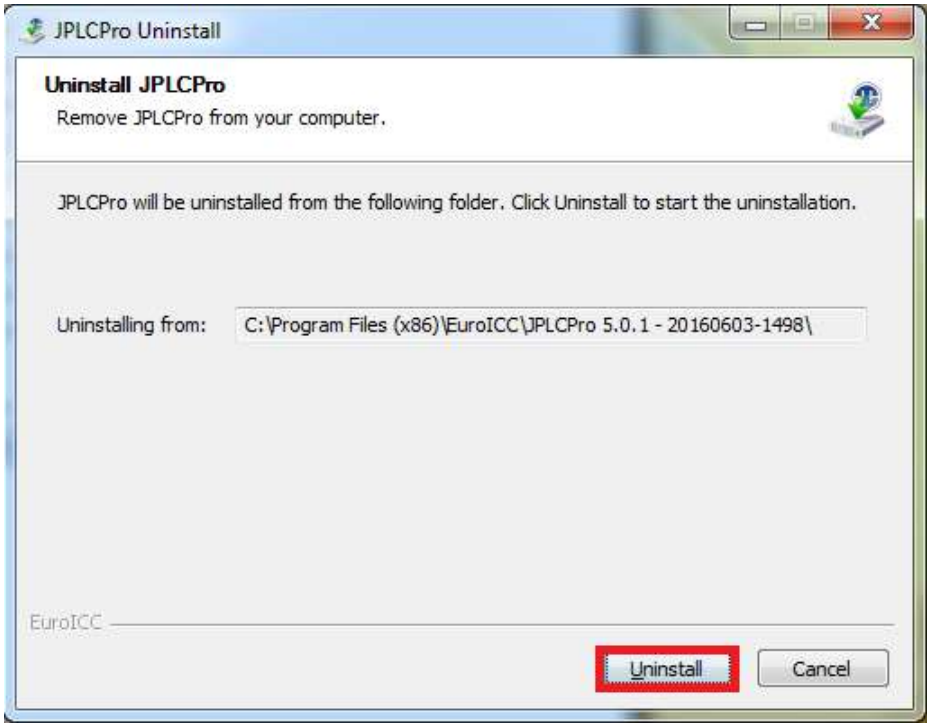
3.1 Starting Uninstallation

Double-click on the uninstallation file “uninstall.exe”;



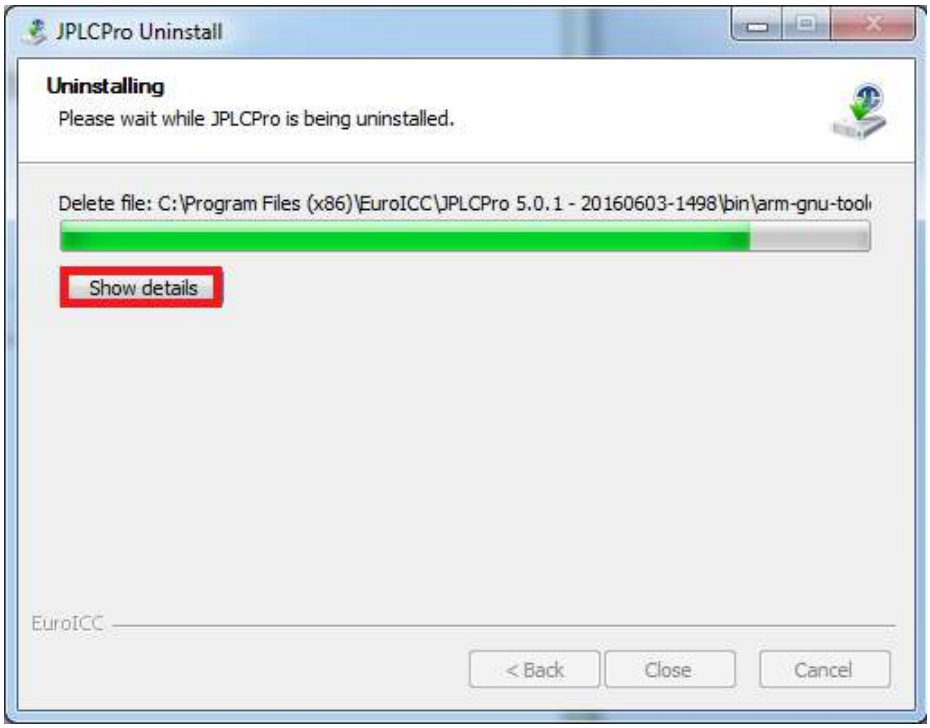
3.2 First Window

Left-click on the “Uninstall” button to start uninstallation of jPLCPro.



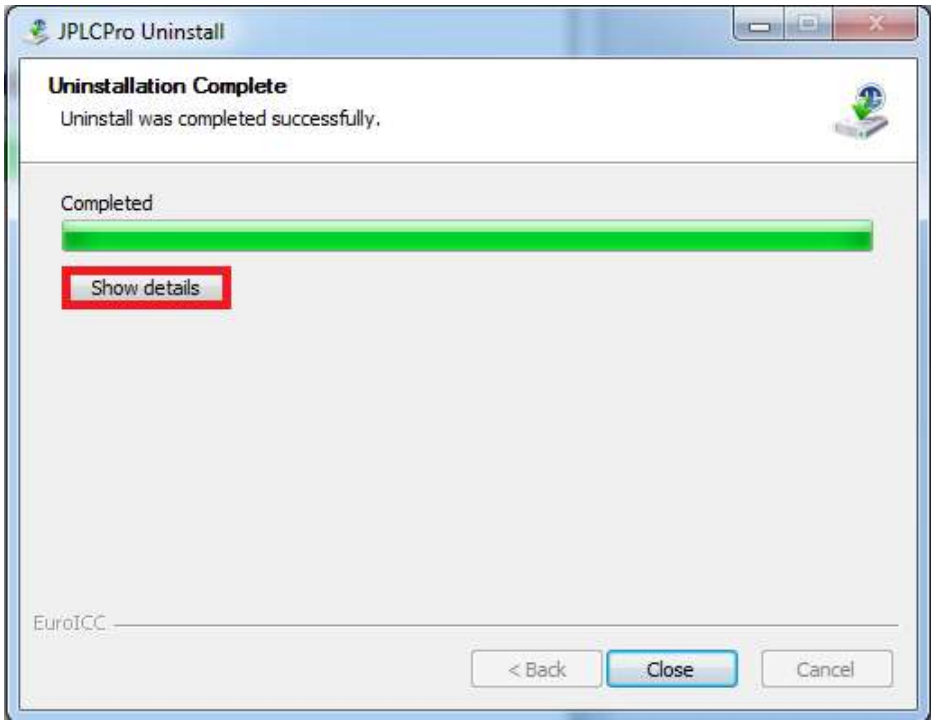
3.3 Uninstallation

Wait for the uninstallation to complete installing jPLCPro. Left-click on “Show Details” button to show details concerning the uninstallation process

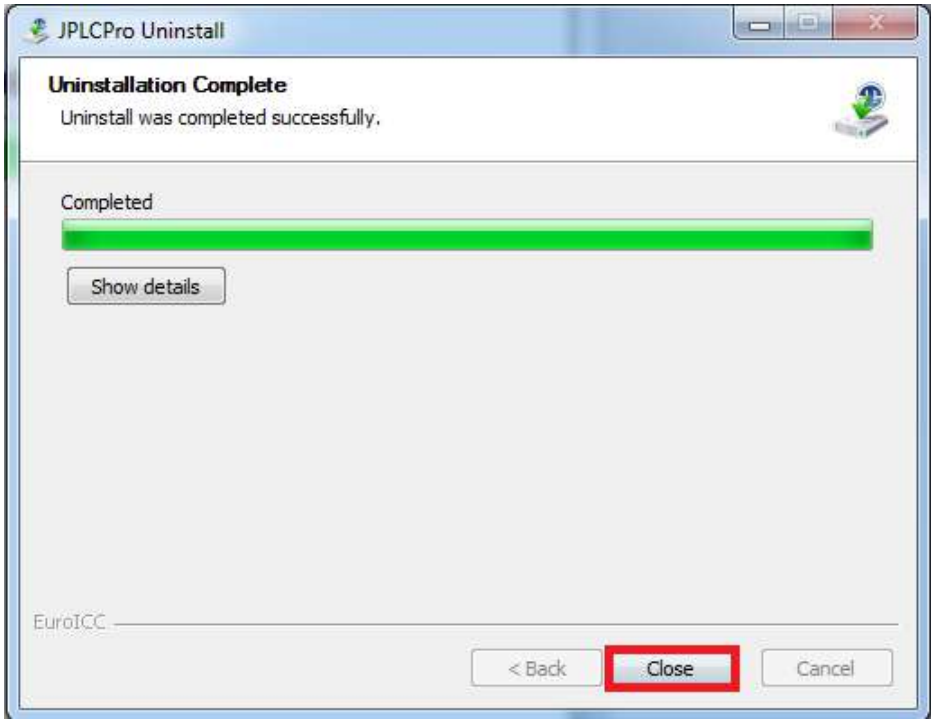


3.4 End

jPLCPro is uninstalled. To show details of the removal process left-click on the “Show Details” button.



To exit the window and finish the uninstallation left-click on the “Close” button.



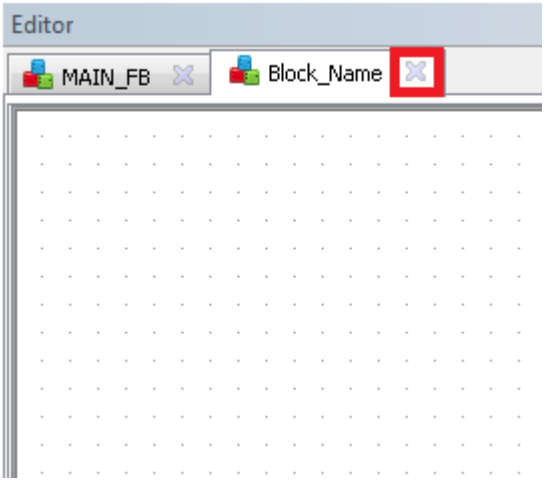
4 User Interface Features

List of User Interface features in jPLCPro:

- Closable Tabs
- Docking
- Toolbars
- Multi-desktop

4.1 Closable Tabs

jPLCPro tabs are closable. Press on “x” to close a tab.



At least one tab will always be open.

4.2 Docking

Application panels are called “Dockables”.

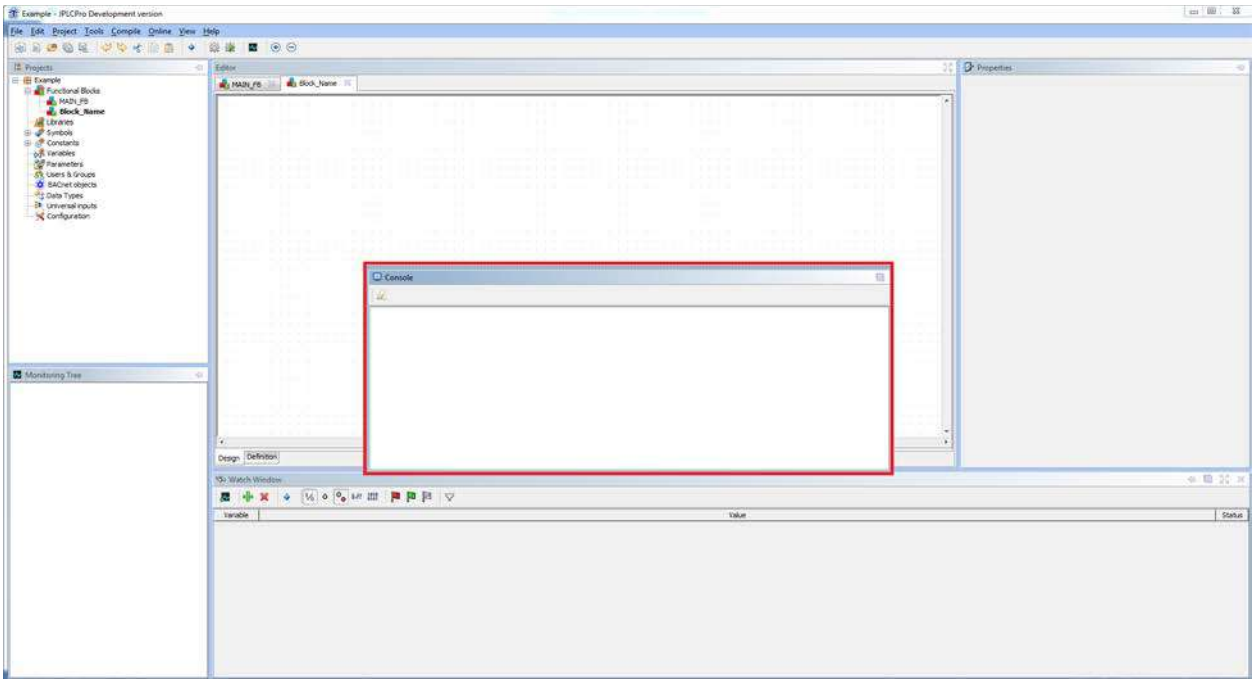
- Dockables are positioned by drag-and-drop gesture
- Dockables can be grouped into tabbed panes
- Dockables can be closed

4.2.1 Docking features

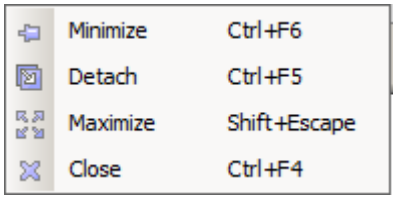
Every Dockable has a title bar with a set of common functions (and their corresponding shortcuts).

- Maximize/Restore – Target takes all the available space on the desktop
- Detach/Attach – Target is separated/added to parent window
- Close – Target is closed
- Contextual pop-up menu – Target specific options

Detach example:



Contextual pop-up menu example:



4.2.2 Docking actions

Dockables have their actions palette, depending on their type and context.

Action is performed by doing one of the following:

- Left-clicking on an icon in upper right corner of the target
- Right-clicking on targets tittle bar to open a popup menu. Left-click on an icon
- Using a shortcut

4.2.2.1 Docking Shortcuts

All dockables have the following shortcuts:

Icon	Shortcut	Action
	Ctrl+F6	Minimize
	Ctrl+F6	Restore (after Minimize)

	SHIFT+Esc	Maximize
	CTRL+Esc	Restore (after Maximize)
	CTRL+F5	Detach
	CTRL+F6	Attach (after Detach)
	CTRL+F4	Close

4.3 Toolbars

jPLCPro tools are hide-able. To toggle the visibility of a toolbar, press the appropriate icon.

Hiding a toolbar example:

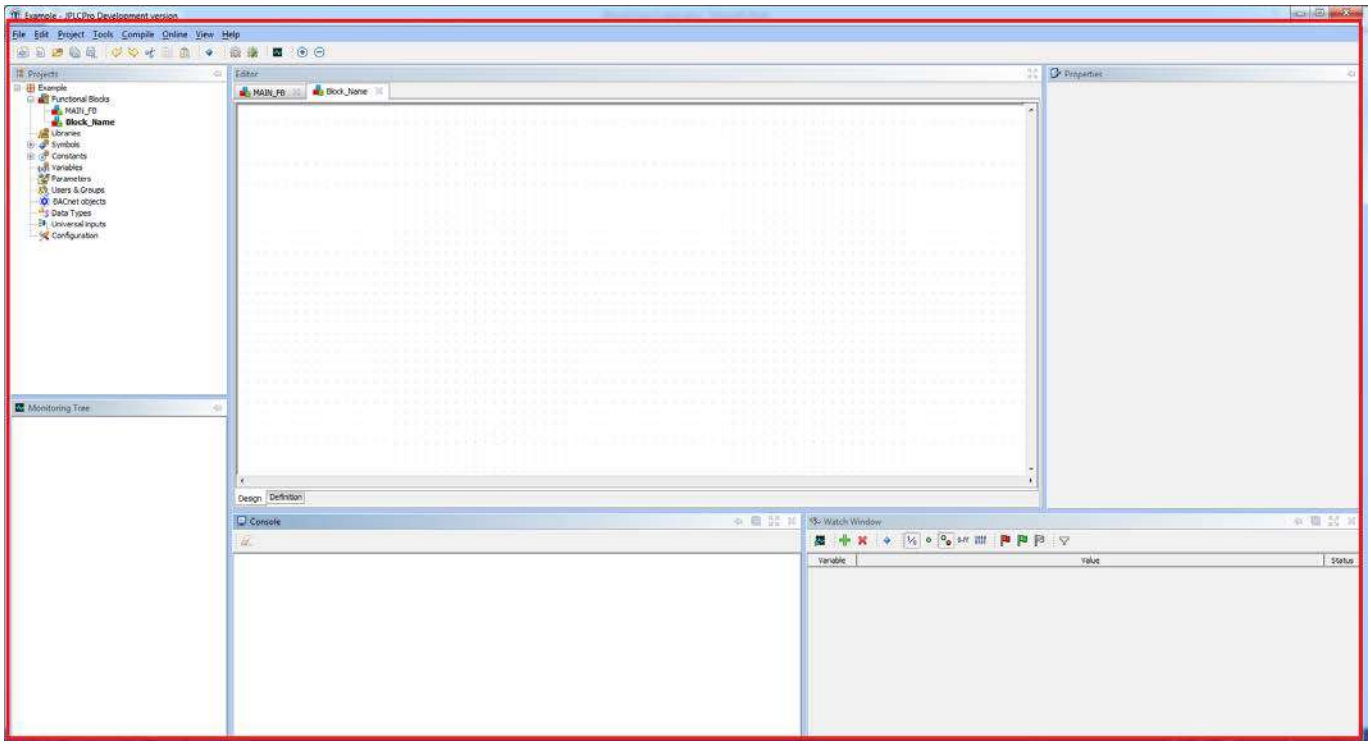
5 User Interface

jPLCPro consists of several bars and dockables (views).

- Work Environment
- Drop-down menus
- Toolbar menu
- Editor View
- Project View
- Monitoring View
- Console View
- Properties View
- Watch View

5.1 Work Environment

Work Environment is a place where all drop-down menus, bars and dockables are displayed.



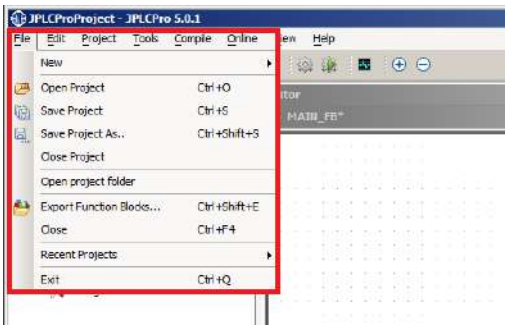
5.2 Drop-down menus

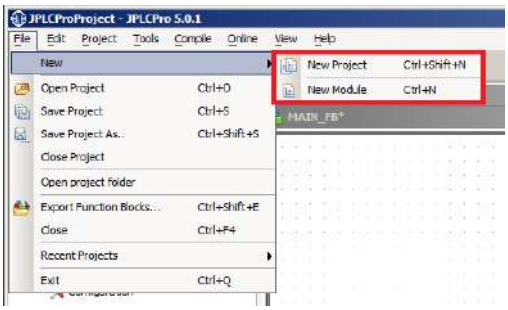
There are eight drop-down menus:

- File Menu
- Edit Menu
- Project Menu
- Tools Menu
- Compile Menu

- Online Menu
- View Menu
- Help Menu

Each drop-down menu is opened by left-clicking the appropriate button.

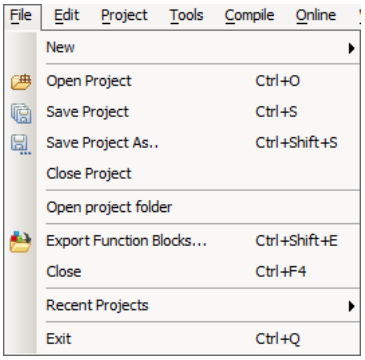




Some elements of a drop-down menu have sub-menus. These sub-menus are expanded by left-clicking on them.

5.2.1 File Menu

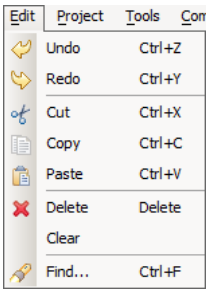
File menu allows manipulation of the project as a whole:



- New Menu – Create menus
  - New Project – Create new project
  - New Module – Create new module (FB or FB-C)
- Open Project – open a project
- Save Project – save project
- Save Project As – save project as a different project from current
- Close Project – close project
- Open Project Folder – open project location
- Export Function Blocks – export FBs
- Close – Closes the current project
- Recent Projects – Select one of the recent projects to open it
- Exit – close jPLCPro

5.2.2 Edit Menu

Edit menu allows the editing of FBs and FB-Cs:

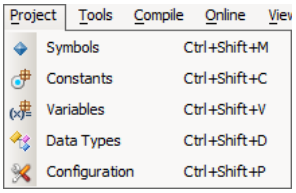


- Undo – undo last action (except Function Blocks)
- Redo – redo last action
- Cut – cut selected content to clipboard
- Copy – copy selected content to clipboard
- Paste – paste the content from clipboard
- Delete – delete selected content
- Clear – clear everything from current FB
- Find – find FB by name or address

5.2.3 Project Menu

Project menu allows the opening of various tables and configuration of project:

- Symbols Table
- Constants Table
- Variables Table
- Data Types Table
- Configuration



5.2.3.1 Symbols Table

Symbols Table contains symbols and allows editing symbols.

A symbol represents a certain part of memory. It consists of:

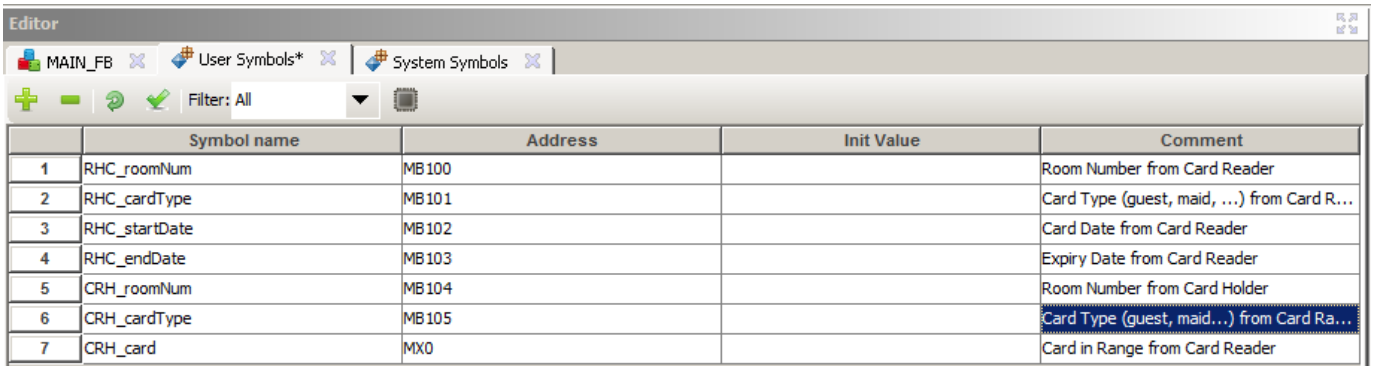
- Name – String by which the symbol is known
- Address – Memory Zone and Index in it
- Init Value – Initial Value of the symbol
- Comment

Symbols Table allows:

- Browsing symbols
- Creating new symbols
- Changing existing symbols
- Removing existing symbols

There are two types of Symbols tables, User Symbols Table and System Symbol Table.

User Symbols Table can be edited. System Symbols Table is a table defined by project settings, and cannot be edited.



	Symbol name	Address	Init Value	Comment
1	RHC_roomNum	MB100		Room Number from Card Reader
2	RHC_cardType	MB101		Card Type (guest, maid, ...) from Card R...
3	RHC_startDate	MB102		Card Date from Card Reader
4	RHC_endDate	MB103		Expiry Date from Card Reader
5	CRH_roomNum	MB104		Room Number from Card Holder
6	CRH_cardType	MB105		Card Type (guest, maid...) from Card Ra...
7	CRH_card	MX0		Card in Range from Card Reader

Note: Symbols must have unique names.

5.2.3.2 Constants Table

Constants Table contains constants and allows editing constants.

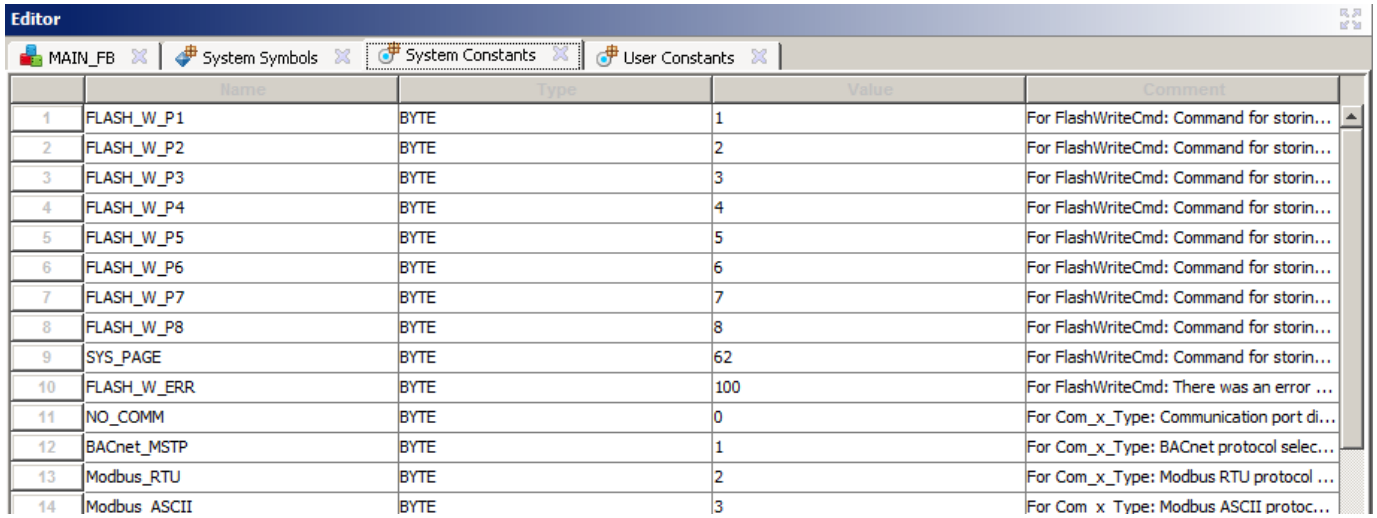
A constant is a value that can be used by calling its name. Constants value cannot be changed during runtime. It consists of:

- Name – String by which the constant is known
- Type – Type of constant
- Value – Value of constant.
- Comment

Constants Table allows:

- Browsing constants
- Creating new constants
- Changing existing constants
- Removing existing constants

Similar to Symbols Tables, there are two kinds of Constants Table: User Constants Table and System Constants Table.



	Name	Type	Value	Comment
1	FLASH_W_P1	BYTE	1	For FlashWriteCmd: Command for storin...
2	FLASH_W_P2	BYTE	2	For FlashWriteCmd: Command for storin...
3	FLASH_W_P3	BYTE	3	For FlashWriteCmd: Command for storin...
4	FLASH_W_P4	BYTE	4	For FlashWriteCmd: Command for storin...
5	FLASH_W_P5	BYTE	5	For FlashWriteCmd: Command for storin...
6	FLASH_W_P6	BYTE	6	For FlashWriteCmd: Command for storin...
7	FLASH_W_P7	BYTE	7	For FlashWriteCmd: Command for storin...
8	FLASH_W_P8	BYTE	8	For FlashWriteCmd: Command for storin...
9	SYS_PAGE	BYTE	62	For FlashWriteCmd: Command for storin...
10	FLASH_W_ERR	BYTE	100	For FlashWriteCmd: There was an error ...
11	NO_COMM	BYTE	0	For Com_x_Type: Communication port di...
12	BACnet_MSTP	BYTE	1	For Com_x_Type: BACnet protocol selec...
13	Modbus_RTU	BYTE	2	For Com_x_Type: Modbus RTU protocol ...
14	Modbus_ASCII	BYTE	3	For Com_x_Type: Modbus ASCII protoc...

User Constants Table can be edited. System Constants Table is a table defined by project settings, and cannot be edited.

Note: Constants must have unique names.

5.2.3.3 Variables Table

Variables Table contains variables and allows editing variables.

A variable is a value that has a name and type. It can be changed during runtime. Values consist of the same elements as constants:

- Name – String by which the variable is known
- Type – Type of variable
- Value – Value of variable
- Comment

Variables Table allows:

- Browsing variables
- Creating new variables
- Changing existing variables
- Removing existing variables

Note: Variables must have unique names.

5.2.3.4 Data Types Tables

Data Types Tables contain data types and allow editing variables.

Data type can be:

- Arrays – array of a certain type and length
- Enumerated – array of integers where a position is marked by a certain name
- Structures – complex structures that contain one or more other data types

For each data type there is a corresponding Data Type Table.

Data Types Tables allow:

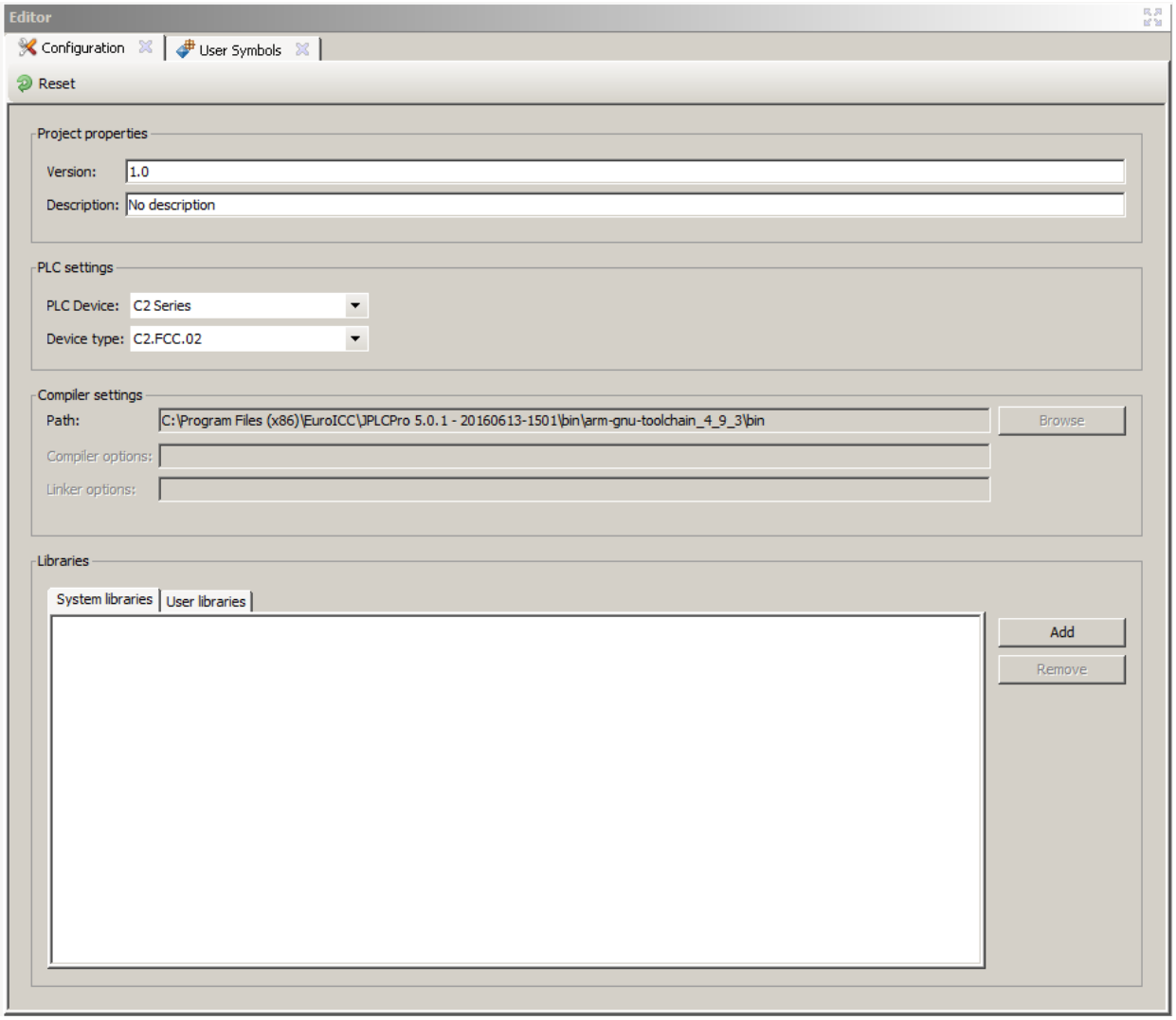
- Browsing data types
- Creating new data types
- Changing existing data types
- Removing existing data types

Note: Data types must have unique names.

5.2.3.5 Configuration

Configuration allows the configuration of the project. It has three main sections:

- Project Properties
- PLC Settings
- Libraries



5.2.3.5.1 Project Properties

Project Properties are project version and project description.

Both are defined as a text, so any value is acceptable.

5.2.3.5.2 PLC Settings

PLC Settings are PLC Device and Device Type. Both are chosen from a drop-down menu.

PLC Device is a family of EPLCs, such as “C Series”.

Device Type is an EPLC device, such as “GRU21”.

These settings define the compiler, method of communication and so on. They should be set with appropriate device type.

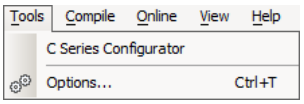
5.2.3.5.3 Libraries

This section allows management of libraries. There are “System libraries” which come with jPLCPro and “User libraries” that users create with “Export Function Block”.

Libraries contain FBs which can be used in FB programming as any other.

Libraries are added by left-clicking the “Add” button and removed by selecting the library by left-clicking on its name and left-clicking the “Remove” button.

5.2.4 Tools Menu



Tool menu contains tools for various devices and options.

5.2.4.1 C Series Configurator

Used for configuring devices from C Series, such as GRU21.

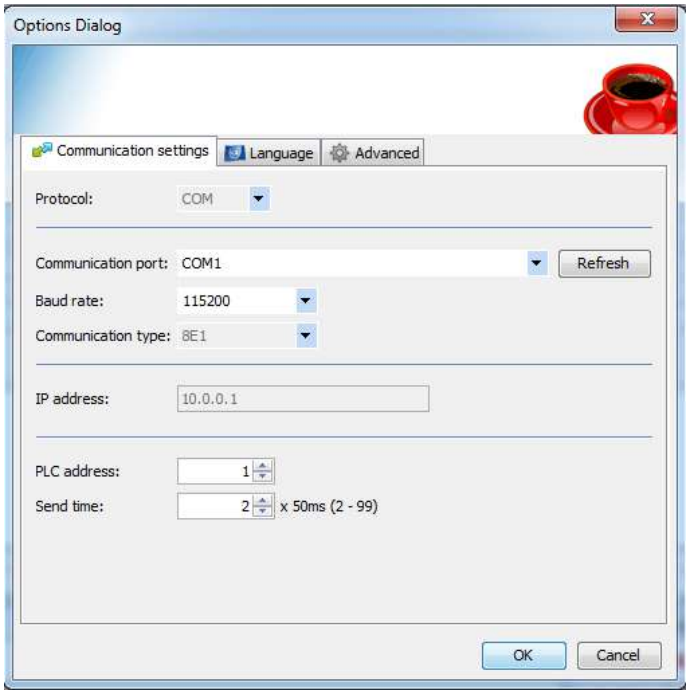
See C Series Configurator Documentation.

5.2.4.2 Options

To save changes left-click the “Ok” button. To discard changes left-click the “Cancel” button.

Options allow changing various settings of a jPLCPro and there are three main sections:

- Communication Settings
- Language Settings
- Advanced Settings



5.2.4.2.1 Communication Settings

In Communication Settings communication parameters are set.

Depending on the device some parameters will be disabled.

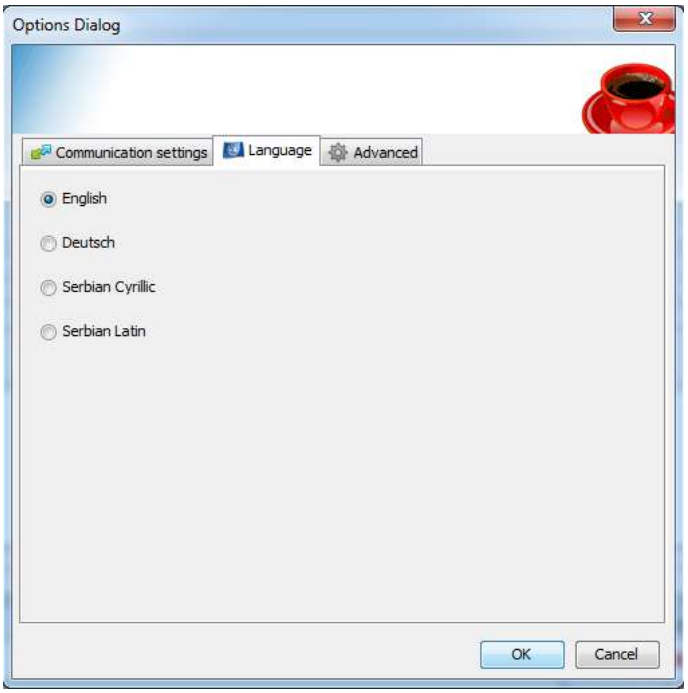
Communication Settings allow editing:

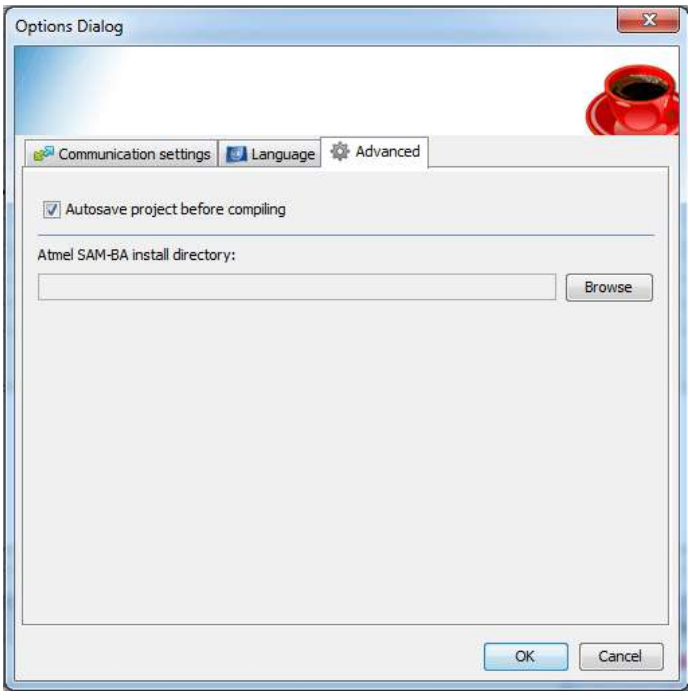
- Protocol
- Communication Port
- Baud Rate
- Communication Type
- IP Address
- PLC Address
- Send Time

5.2.4.2.2 Language Settings

Language is selected by choosing the wanted language.

The default language is English.



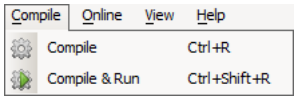


5.2.4.2.3 Advanced Settings

Advanced settings allow setting some of the advanced features of jPLCPro:

- Autosave project before compiling
- Choosing the Atmel SAM-BA directory

5.2.5 Compile Menu



Compile Menu allows compiling features:

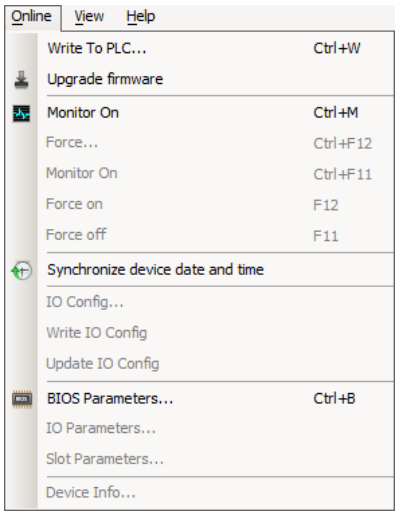
- Compile – Compiles FB program
- Compile and Run – Compiles FB program, downloads it to the device and runs it automatically

Compilation progress and status are displayed in the Console Dockable. jPLCPro needs to be connected to a device for “Compile and Run” feature to run successfully. Depending of the device type some options are unsupported and are disabled.

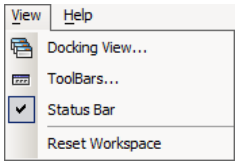
5.2.6 Online Menu

Online menu allows access to features that include some sort of communication between jPLCPro and a PLC.

- Connect to PLC – Connects to a device.
- Disconnect – Disconnects from a device
- Write to PLC – Write to PLC dialog is presented
- Upgrade Firmware
- Monitor On – Turns on monitoring
    - Device must be connected
- Force – Forces a Value
    - Monitor must be On
- Deforce – Deforces a value
    - Monitor must be On
- Force On – Forces a Bit Value
    - Monitor must be On
- Force Off – Deforces a Bit Value
    - Monitor must be On
- Synchronize device date and time
- IO Config
- Write IO Config
- Update IO Config
- BIOS Parameters
- IO Parameters
- Slot Parameters
- Device Info



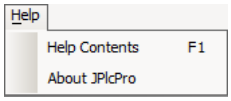
5.2.7 View Menu



View menu allows access to Workspace look features.

- Docking View opens the “Docking Configuration” window.
    - This window allows users to choose which Dockable will be shown.
- ToolBars opens the “ToolBars Configuration” window.
    - This window allows users to choose which ToolBars will be shown
- Status Bar
    - Allows users to choose whether the Status Bar should be shown. To toggle between states (showing/hidden) press this menu item.
- Reset Workspace
    - Restores default Workspace configuration (Docking Configuration, ToolBar Configuration and Status Bar)

5.2.8 Help Menu



In this menu a user can find the jPLCPro User Manual and some information considering the application.

- Help Contents
    - Opens a pdf file of jPLCPro User Manual
- About jPLCPro
    - Opens a “About jPLCPro” window that displays various technical information about the software

5.3 ToolBars

ToolBars are a way to complete certain actions faster. Instead of going through the menus and searching for a wanted action, users can simply one-click on an icon and get the desired effect.

ToolBars are separated by a separator that also functions as hide/show button. Separator on the left of a ToolBar group hides/shows that ToolBar group.

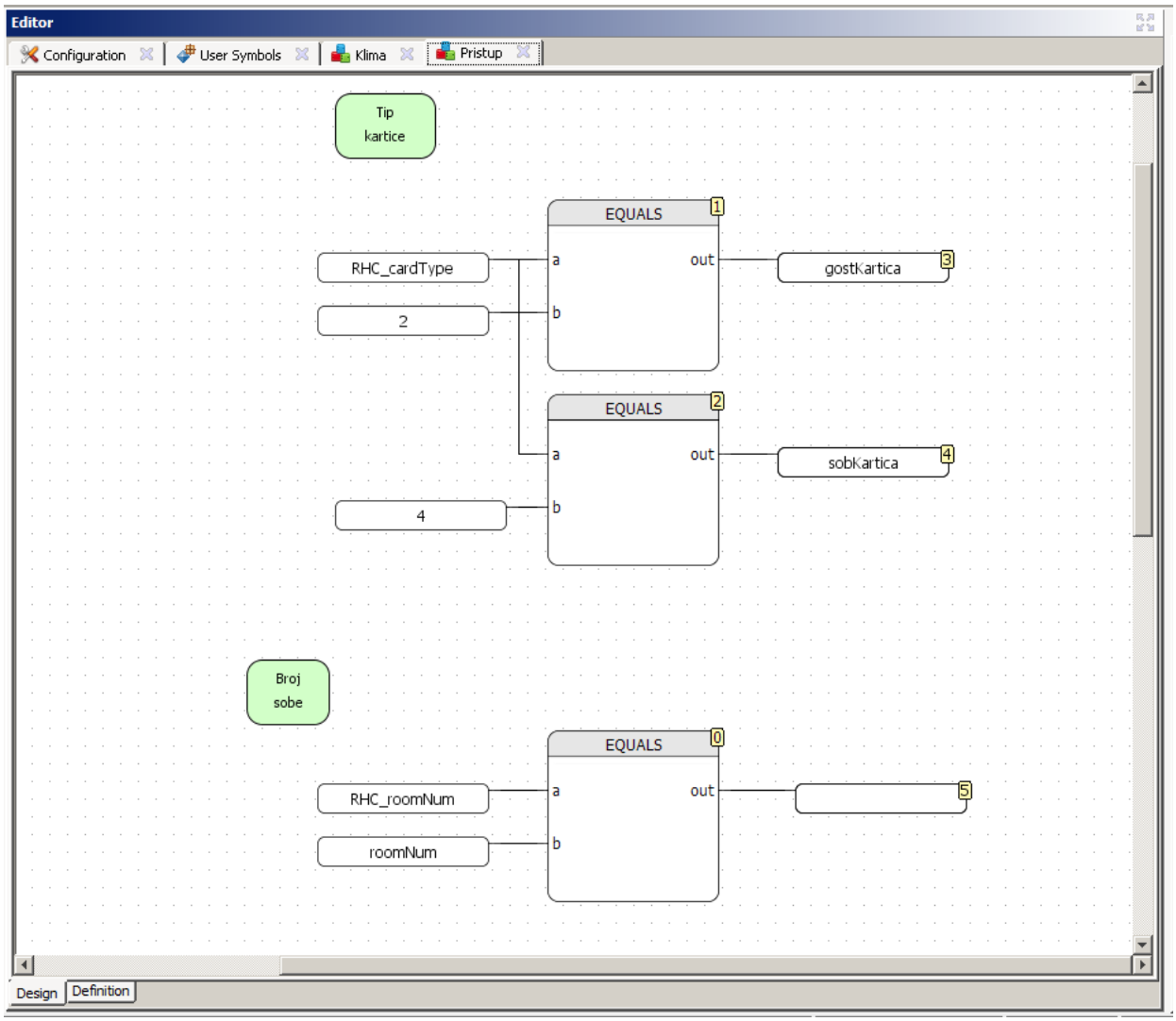
ToolBar groups are logical groups of actions that are grouped based on context.

Mouse-hovering over an icon of a ToolBar will give a brief description of that icons action.

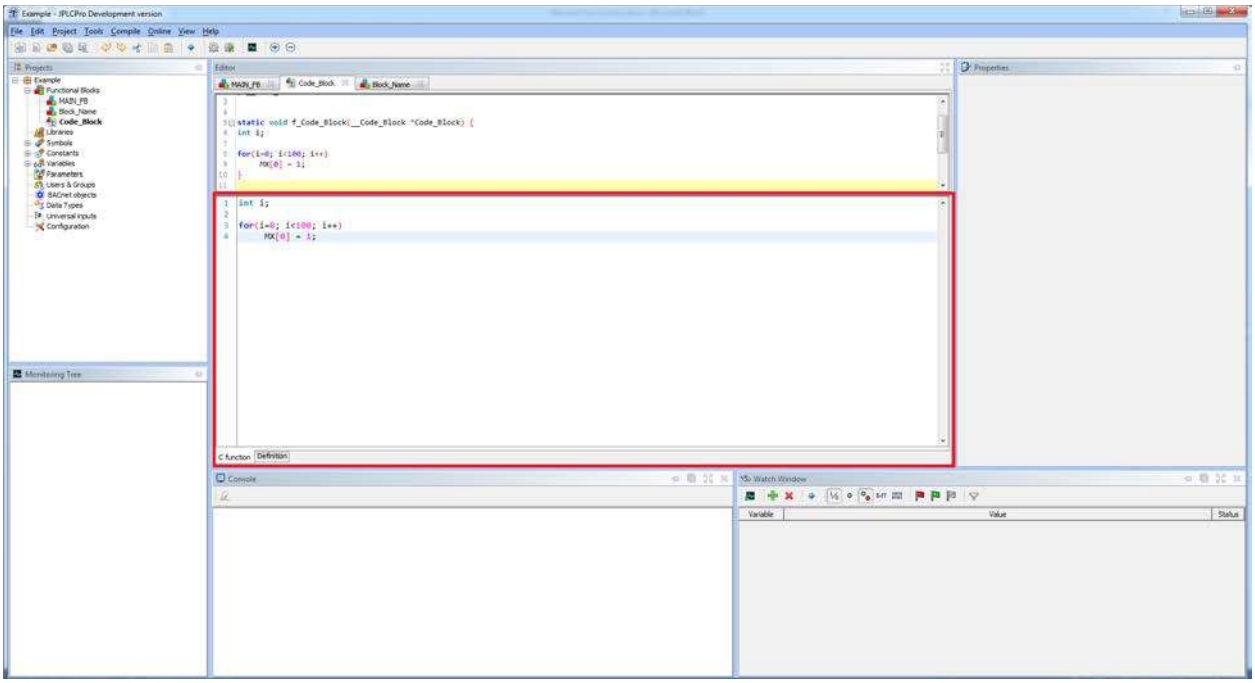
5.4 Editors View

This Dockable is reserved for FB and FB-C editors.

FB editors are in a form of a diagram



FB-C editors are in a form of a text editor (with syntax coloring)

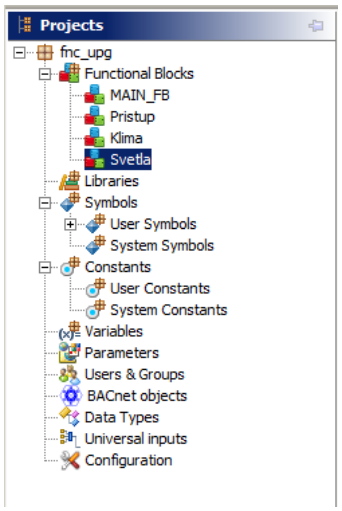


5.5 Project View

Project View is a Dockable that represents hierarchy of the project and allows editing the elements of the project.

Every element in the Project View can be right-clicked for additional actions such as adding a new FB or deleting an existing one.

Every project contains the following sections:



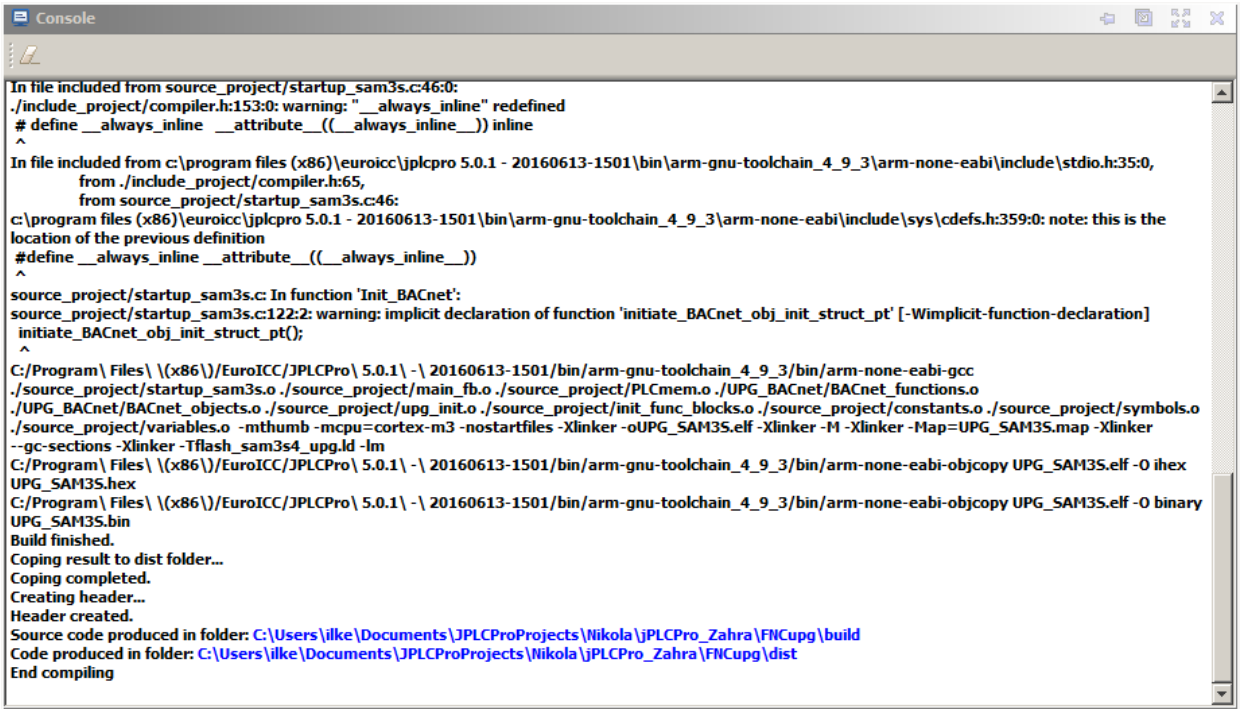
- Functional Blocks
- Symbols
- Constants
- Variables
- Parameters
- Users and Groups
- BACnet Objects
- Data Types
- Universal inputs
- Configuration

Sections are expanded with left-clicking the “+” icon and are folded by left-clicking the “-“ button.

Elements of sections are opened with double-clicking the desired element. All elements are opened in the Edit View as a new tab.

5.6 Console View

Console View displays console information. Console information is all the information that is related to compiling and distribution of the project.



To clear the console click on “Clear” icon.

5.7 Properties View

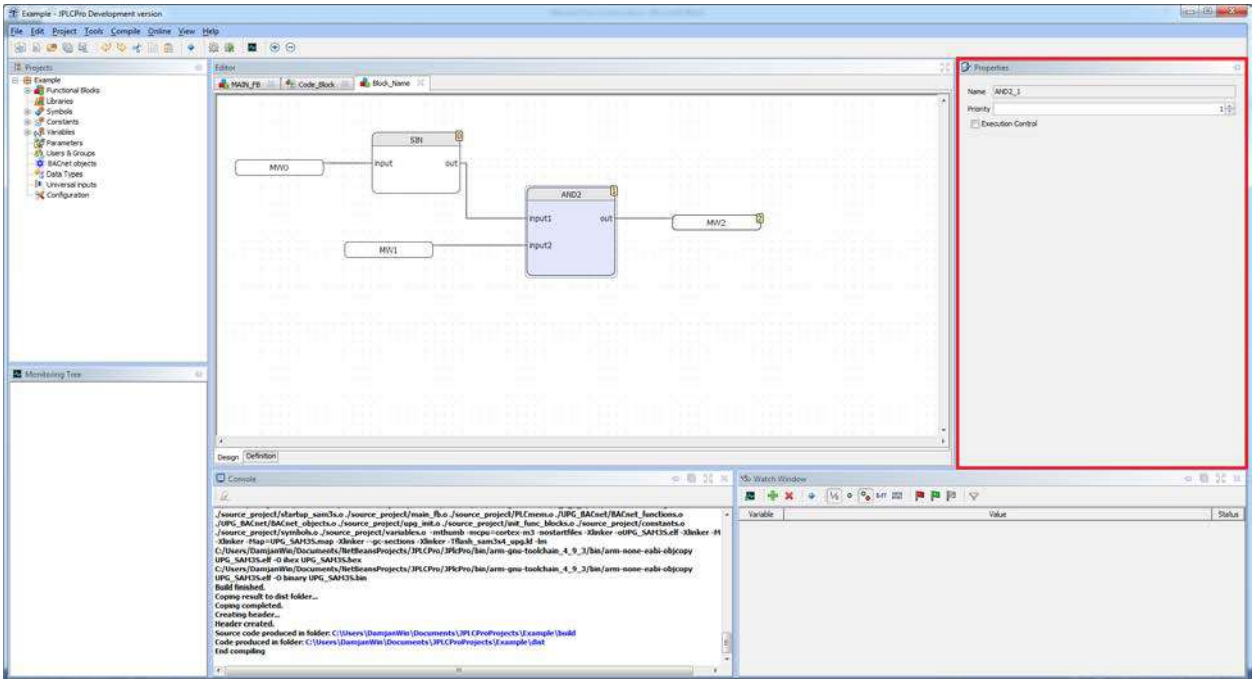
Properties View displays information about the currently selected element in the Editor View. To select an element simply left-click on desired element. As there can only be two kinds of elements, FB and IO elements, there are only two layouts of displaying information.

Additionally Properties View allows setting the priority of FB execution and display of Execution Control.

5.7.1 FB Properties Layout

Displayed FB properties are:

- Name and Instance (in form of Name_InstanceNumber)
- Priority
- Execution Control

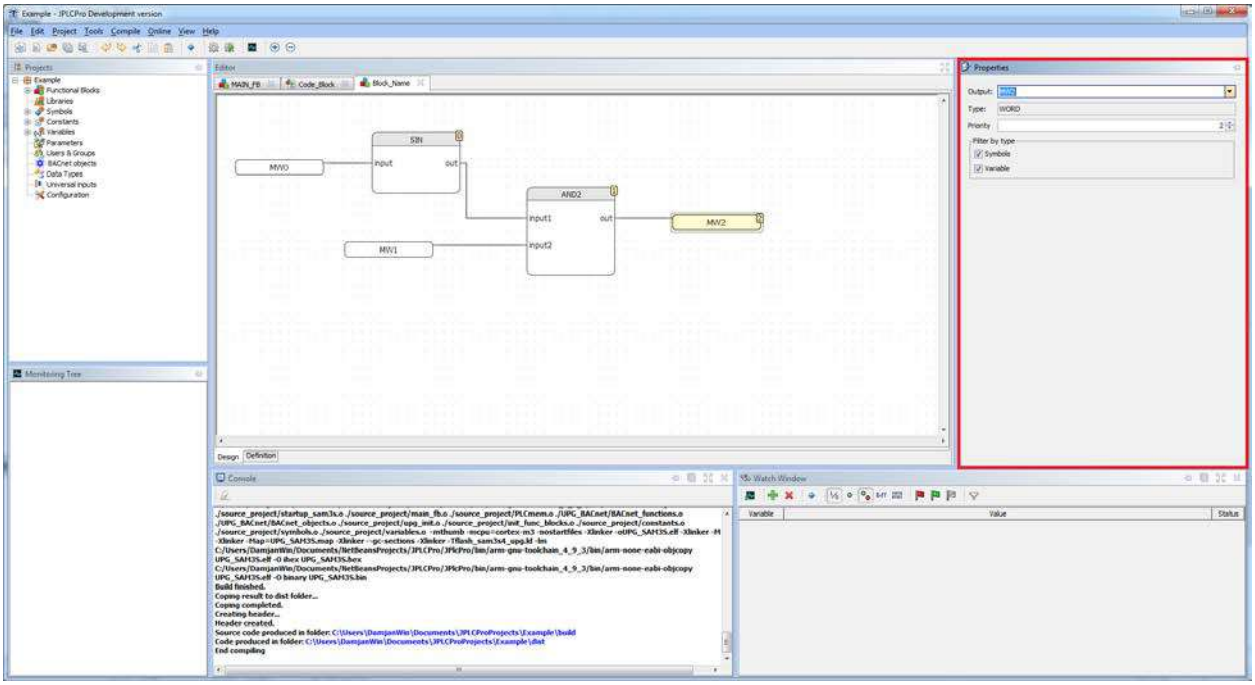


Users can change Priority and Execution Control by changing the corresponding value or checking the box.

5.7.2 IO Elements Layout

IO Elements are Input Elements and Output Elements. They are mostly the same, as the only difference is that Output Elements have the Priority field. Other properties are as follows:

- Name
- Type
- Filter By Type
- Filter By Data Type
- Priority (Outputs only)



Name can be set from Properties View. It can be written manually or selected from a drop-down list. IO Elements names follow the rules that would be explained latter.

Filters control what IO Elements would be available in the name drop-down list.

Memory that is at user programs disposal is the same on every CPLC controller. The memory locations can be of various types and memory is divided into several **zones.**

System Types

6 Memory Organization

Memory that is at user programs disposal is the same on every CPLC controller. The memory locations can be of various types and memory is divided into several zones.

6.1 System Types

System types are types that can be used for inputs, outputs, constants, symbols, variables or building blocks for complex data types:

- `BIT`
- `BYTE`
- `WORD`
- `DWORD`
- `LWORD`
- `FLOAT`
- `STRING`
- `VOID`

All types, except `STRING` and `VOID` are Numerical types. All numerical types, except `FLOAT` are Integer types (whole numbers). `BIT` type is sometimes referred as Boolean type.

6.1.1 BIT Type

`BIT` type represents one bit. Bit value can be either zero or one (0 or 1).

6.1.2 BYTE Type

`BYTE` type consists of 8 bits. It is a signed integer, which means it can have either positive, negative or zero integer value. Its range is from –128 to 127.

6.1.3 WORD Type

`WORD` type consists of 2 bytes or 16 bits. It is a signed integer. Its range is from -32768 to 32767.

6.1.4 DWORD Type

`DWORD` type consists of 4 bytes or 32 bits. It is a signed integer. Its range is from -2147438648 to 2147438647.

6.1.5 LWORD Type

`LWORD` type consists of 8 bytes or 64 bits. It is a signed integer. Its range is from -4294967296 to 4294967295.

6.1.6 FLOAT Type

`FLOAT` type represents real values. Its range is from -3.4E+38 to 3.4E+38

6.1.7 STRING Type

`STRING` type is an array of 32 bytes (or characters).

Memory that is at user programs disposal is the same on every CPLC controller. The memory locations can be of various types and memory is divided into several **zones.**

System Types

6.1.8 VOID Type

`VOID` type consists of 4 bytes or 32 bits. It is used for pointers.

6.2 Memory Zones

A memory zone is a part of memory that can be accessed through FB diagrams and FB code. It is represented as an array of a certain type. Each memory zone has a family and a type.

They are accessed by combining the shortcut name for the memory zone family and the shortcut name for the memory zone type.

Accessing the memory zones will be covered after memory zone families and memory zone types.

6.2.1 Memory Zone families

- System – device system registers - shortcut name `S`
- Input – used for device inputs - shortcut name `I`
- Output – used for device outputs - shortcut name `Q`
- Memory – used for user memory - shortcut name `M`

6.2.2 Memory Zone types:

- `BIT` - shortcut name `X`
- `BYTE` - shortcut name `B`
- `WORD` - shortcut name `W`
- `DWORD` - shortcut name `D`
- `LWORD` - shortcut name `L`

`BIT` Memory Zone is separate from others. `BYTE`, `WORD`, `DWORD` and `LWORD` Memory Zones overlap.

`WORD` type contains two `BYTE` types. For example, `WORD` type at index 0 contains `BYTE` types at indexes 0 and 1. `WORD` type at index 2 contains `BYTE` types at indexes 4 and 5.

Similarly, `DWORD` type contains two `WORD` types and four `BYTE` types.

`LWORD` contains two `DWORD` types, four `WORD` types and eight `BYTE` types.

6.2.3 Accessing a Memory Zone

Accessing a memory zone is done by writing `FT` in functional blocks and `FT[I]` in C code where:

- `F` is a memory zone family shortcut name
- `T` is a memory zone type shortcut name
- `I` is an index

For example, to access a bit output at index 25 write: `QX[25]`

6.2.4 Memory Zones List

The complete memory zone list is presented in the following table:

Name	Function	Type	Length	Note
SX	System bits	BIT	64	
SB	System byte registers	BYTE	192	Overlapped with SW, SD and SL

SW	System word registers	WORD	96	Overlapped with SB, SD and SL
SD	System double word registers	DWORD	48	Overlapped with SB, SW and SL
SL	System double word registers	LWORD	24	Overlapped with SB, SW and SD
IX	Binary inputs	BIT	64	
IB	Byte inputs	BYTE	64	Overlapped with IW, ID and IL
IW	Word inputs	WORD	32	Overlapped with IB, ID and IL
ID	Double word inputs	DWORD	16	Overlapped with IB, IW and IL
IL	Long word inputs	LWORD	8	Overlapped with IB, IW and ID
QX	Binary outputs	BIT	64	
QB	Byte outputs	BYTE	64	Overlapped with QW, QD and QL
QW	Word outputs	WORD	32	Overlapped with QB, QD and QL
QD	Double word outputs	DWORD	16	Overlapped with QB, QW and QL
QL	Long word outputs	LWORD	8	Overlapped with QB, QB and QD
MX	Memory bits	BIT	256	
MB	Memory bytes	BYTE	2048	Overlapped with MW, MD and ML
MW	Memory words	WORD	1024	Overlapped with MB, MD and ML
MD	Memory double words	DWORD	512	Overlapped with MB, MW and ML
ML	Memory long words	LWORD	256	Overlapped with MB, MW and MD

6.3 Additional Memory

Beside mentioned memory zones there is an additional zone that is primarily used for variables from C code. This zone is limited so care must be taken that any long structures or arrays are defined as variables in some user zone (in most cases M zone). One additional characteristics of this zone is that it cannot be accessed through any communication channel.

7 FB Programming

All of the programming is done by using Functional Blocks.

The default (non-removable) FB is MAIN_FB. This FB is used as a starting point of the program.

There are two types of FB, FB Diagrams (FB) and FB Code (FB-C). Each will be explained in more detail later.

Every FB has a “Definition” which defines inputs, outputs and similar elements.

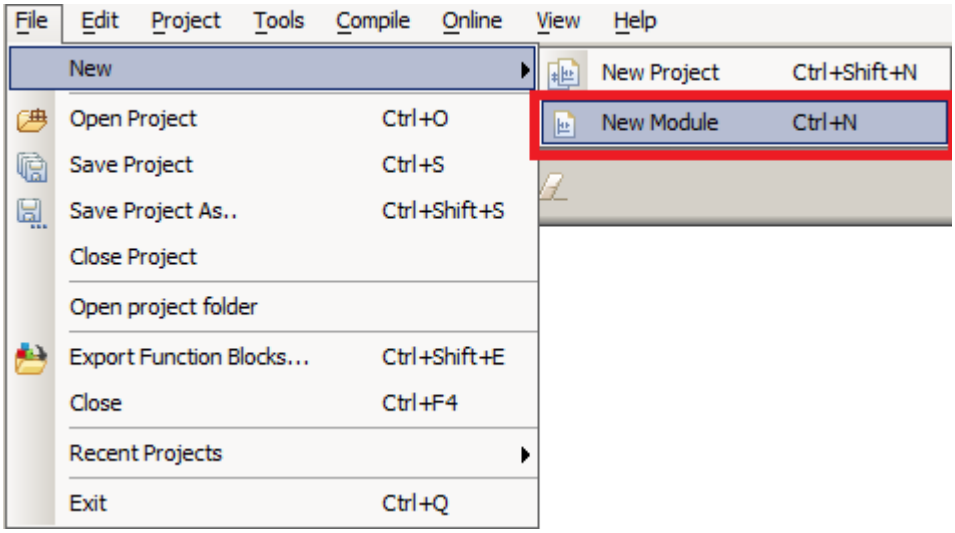
In the “Project View” are the bodies of FBs, which define how and what will FB do.

When a FB is used in another FB, an instance of FB is created. An instance of FB will follow the programming of FB, but different instances can have different values on same elements (for example on the same parameter).

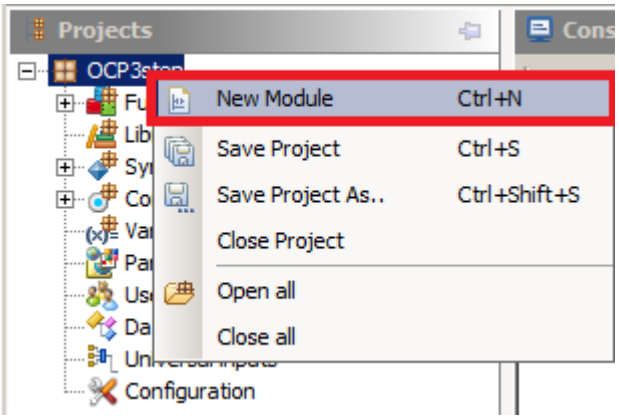
7.1 Creating FBs

Creating FBs is done from the “File Menu”, from the “Project View” or by using a shortcut.

To create a FB from “File Menu” open file menu with left-click, left-click on “New” and then left-click on “New Module”.



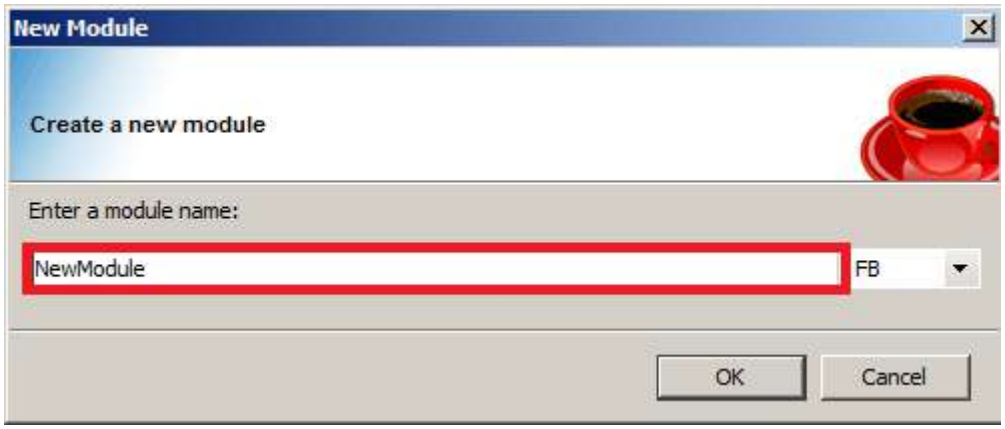
To create a FB from “Project View” right-click on “Functional Blocks” and left-click on “New Module”.



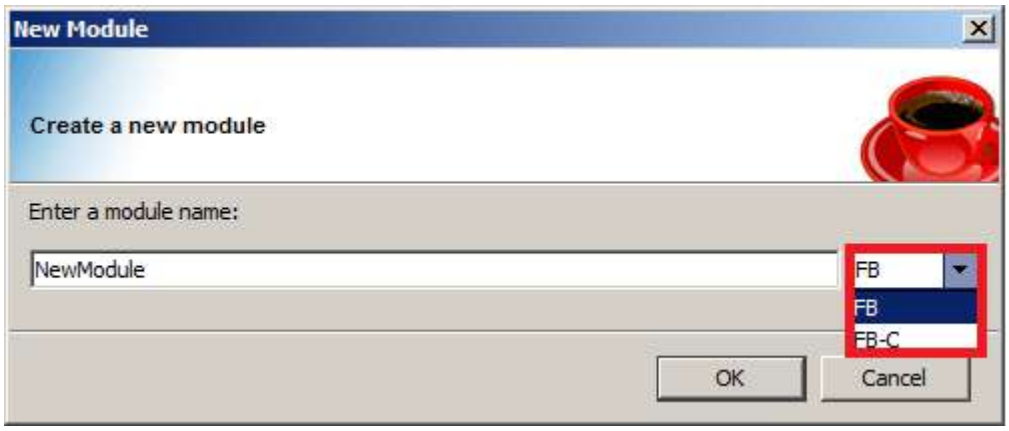
The shortcut for creating a new FB is “`CTRL+N`”.

When creating a FB a FB creation window will appear.

Type in the desired name for a new FB in the appropriate field.



Select the desired FB type from a drop-down menu.

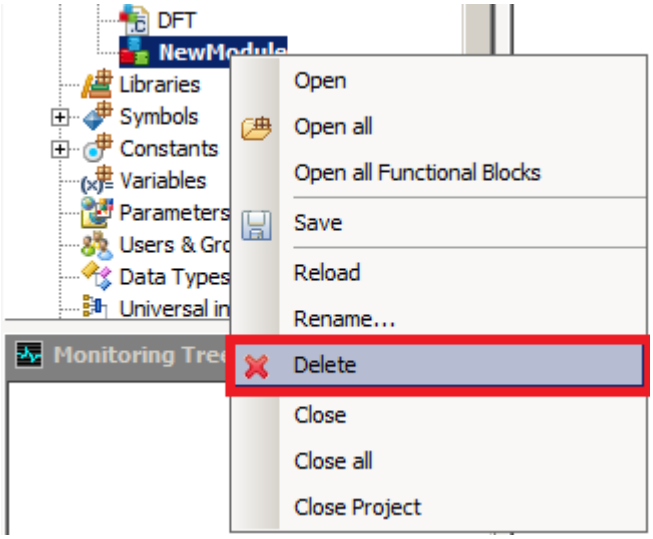


Left-click the `OK` button to create new FB or left-click the `Cancel` button to exit the FB creation window.

7.2 Deleting FBs

Deleting FBs is done from the Project View dockable.

Navigate to the desired FB in the Project View. Right-click on the FB name and left-click on the `Delete` button in the drop-down menu. Or press the `Del` button on keyboard.

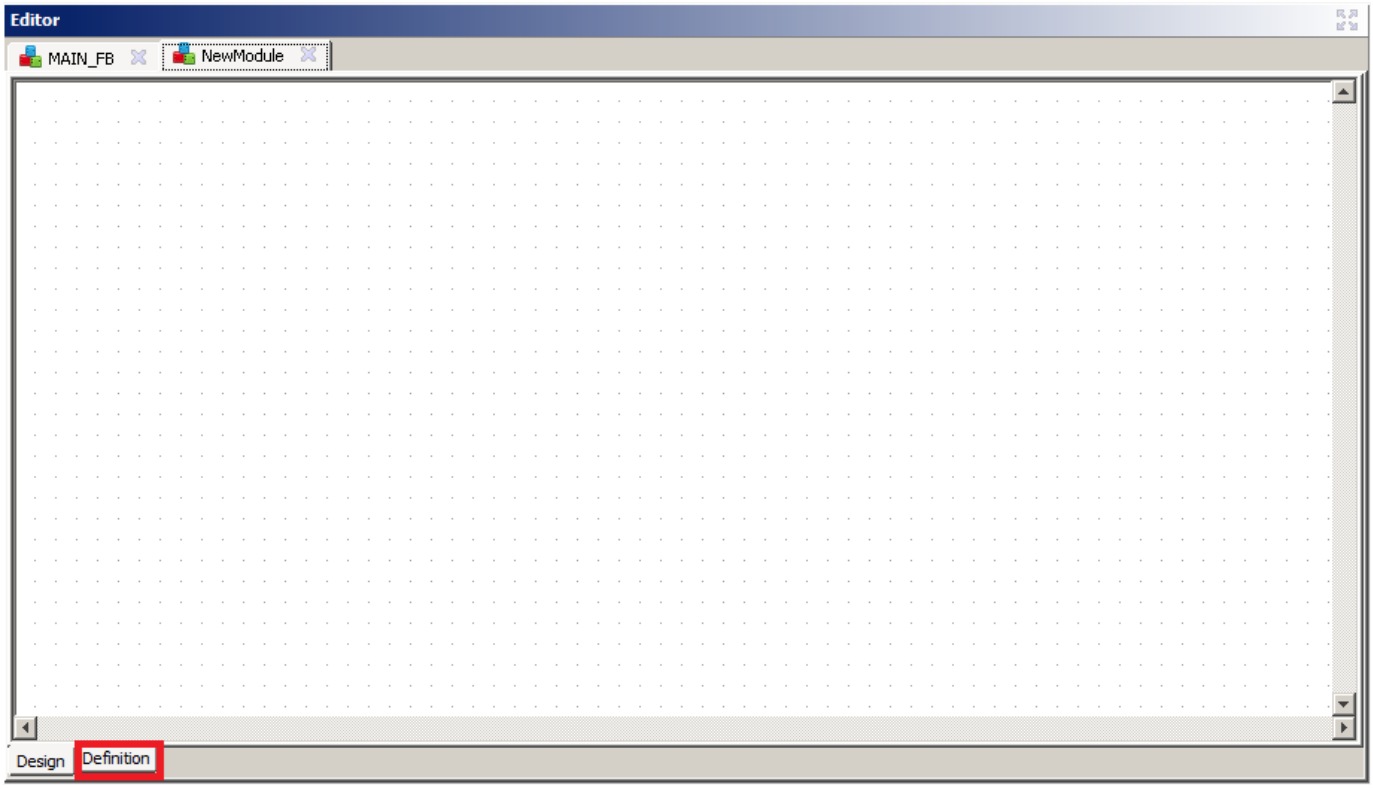


7.3 FB Definition

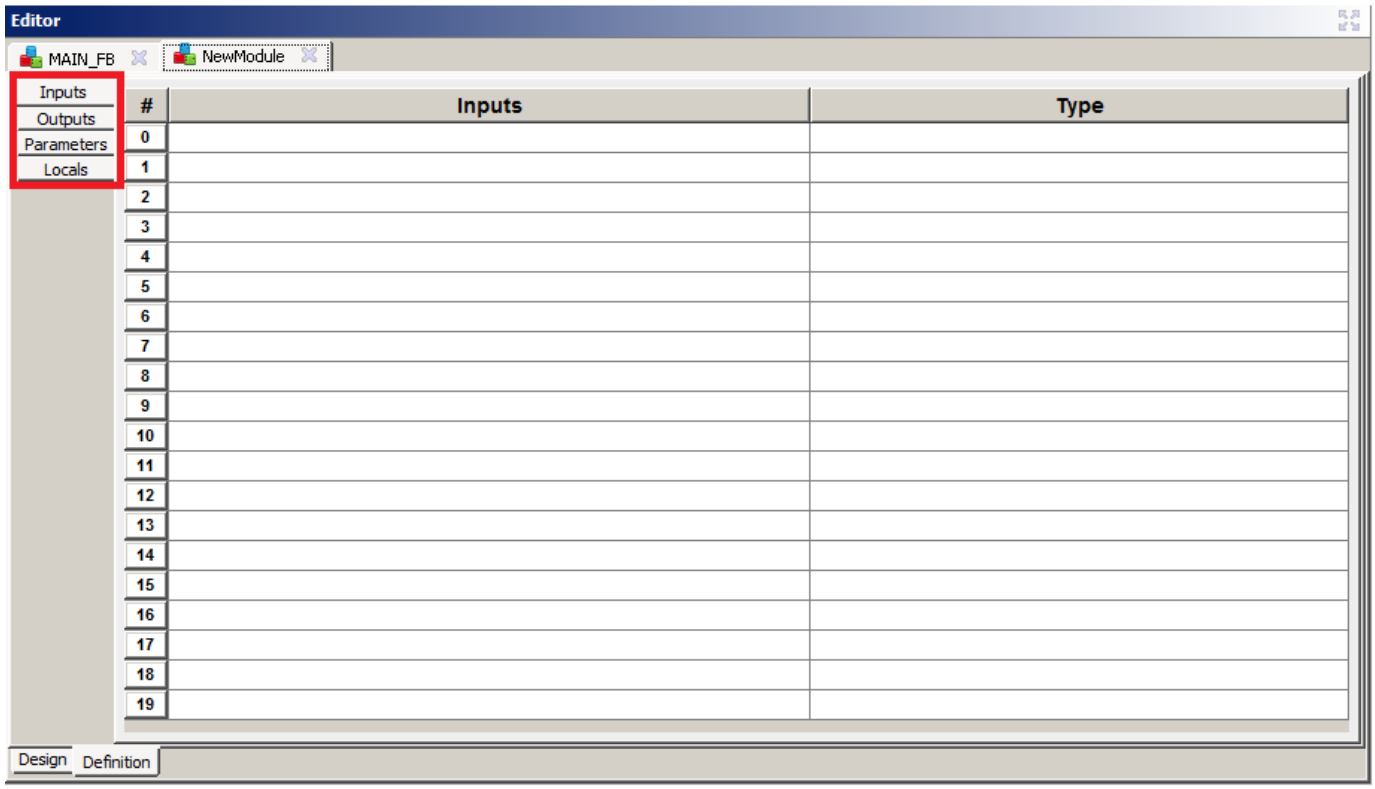
FB Definition contains:

- Inputs
- Outputs
- Parameters
- Locals

To display FB Definition that FB must be open in Editor View. To display FB definition left-click on the `Definition` tab in the bottom left corner of the FB tab.



To display different element lists left-click on the element type name.



Maximum number of any element type is twenty.

To add an element type in the first free row.

To delete an element erase the information from the appropriate row.

To change an element type in the changes in the appropriate row.

7.3.1 Inputs

FB Inputs are the entry point for data in a FB.

They appear on the left side of the graphical interpretation of the FB.

Every input has:

- Name - name of the input
- Type - type of the input

7.3.2 Outputs

FB Outputs are the exit point for data in a FB

They appear on the right side of the graphical interpretation of the FB.

Every output has:

- Name - name of the output
- Type - type of the output

7.3.3 Parameters

FB Parameters are parameters of a FB. For each instance of a FB different values of parameters can be assigned.

Every parameter has:

- Name - name of parameter
- Type - type of parameter
- Init Value - initial value
- Max - maximum value
- Min - minimum value
- Inc - increment

7.3.4 Locals

FB Locals are local variables of a FB. Each instance of a FB can have different values of locals.

Every local has:

- Name - name of local
- Type - type of local
- Init Value - initial value

7.4 FB Diagram

FB Diagrams are a graphical way to program a FB.

The three main parts of every FB diagram are:

- FB instances
- Inputs/Outputs
- Connections

FB instances and Outputs have an Execution Priority. Execution Priority defines execution order in a FB.

Connections connect FB instances, inputs and outputs.

The algorithm of every FB diagram is as follows:

- All inputs are read
- FB instances and outputs are calculated, in the order of the Execution Priority

Note: Usually FB instances will have a higher priority than the output elements.

7.4.1 Adding elements

To add an FB instance, input or output:

Right-click on the desired location and left-click on the desired element.

- System FBs are in the first group of elements.
- User defined FBs are under the “Custom” sub-menu.
- Inputs, outputs and comments are in the third group of elements.

7.4.1.1 Setting Input/Output Elements

When an input or output is added to a diagram it is empty.

Setting what is an input/output element is done from Properties View, as explained before.

Inputs can be FB inputs, FB outputs, variables, constants, symbols or a member of a memory zone.

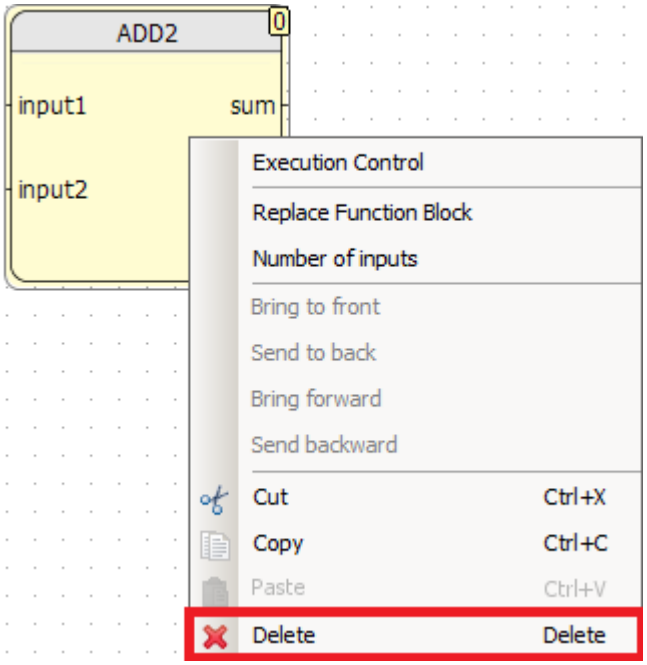
Outputs can be FB inputs, FB outputs, variables, symbols or a member of a memory zone.

Element inputs can be FB outputs, and vice versa. This seems counter-intuitive, but it is needed. Sometimes calculations depend not only from FB inputs, but from FB outputs also.

7.4.2 Removing elements

To remove an element from FB

Right-click on the element, and left-click on the “Delete” button



Or press the “Delete” button on the keyboard.

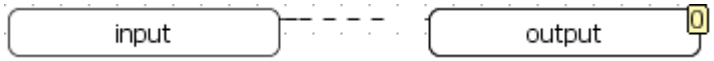
When an element is deleted its connections are also deleted.

7.4.3 Connecting elements

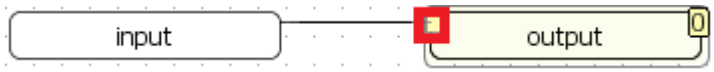
To connect elements

Left-click on the desired FB input/output or input/output

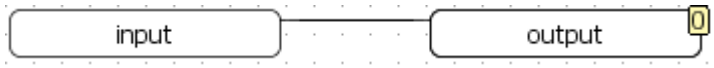
Move cursor to the other side of desired connection. Dashed line is following cursor.



Left-click on the other side of the desired connection, again on the desired FB input/output or input/output



The elements are now connected.



When output of one block is to be connected to more than one input. Each line must be connected completely. It is not possible to connect one line to the middle of another.

7.4.4 FB instances in FB diagrams

FB instances are the part of FB diagrams where all the calculating is done. As explained before, they have inputs and outputs.

In Editor View the name of the FB instance will be without the instance number. In Properties View this number will be displayed in format `FBName_Number`.

7.5 FB C Code

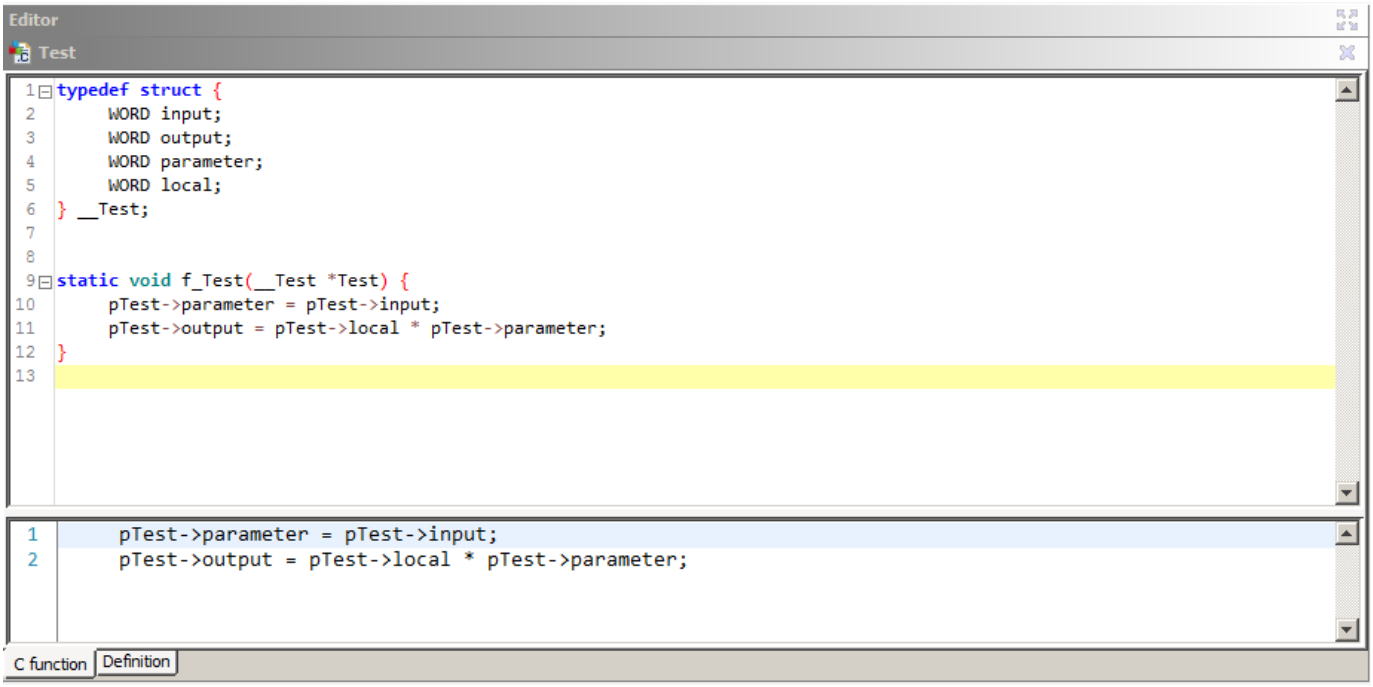
FB Code is a method of textual (standard) programming. It is done in the standard C programming language.

Writing the code is done in “Editor View” in the appropriate tab.

All of the FB definition elements are accessible through code, as are all the symbols, variables and constants.

There is package of system libraries that can be used.

There is no Execution Order, as the code itself determines it.



In the upper part of FB-C editor is code generated based on your function. In the lower part, you can enter and change your code.

7.5.1 FB-C Features

There is a number of FB-C features in jPLCPro:

- Text is colored considering the syntax
- Undo last action
- Redo last action

7.5.2 Accessing Memory Zones

Accessing a memory zone is done by writing `Z[I]` where:

- Z is the name of the memory zone

- `I` is the index in the memory zone

7.5.3 Accessing FB definition elements

When you define FB elements of FB-C element (inputs, outputs, parameters and locals), jPLCPro creates structure with FB-C element name, preceded with two underscores (__), and a function with FB-C element name, preceded with `f_`, whose only parameter is a pointer to created structure. For each instance of FB-C element, one instance of the created structure is allocated in memory, and on each scan, function is called with pointer to the allocated structure.

You can access FB-C elements in C code through passed structure:

```
pModuleName->elementName
```

7.5.4 Accessing symbols, constants and variables

To access a symbol, constant or variable in code, simply write its name.

For example, setting the value of variable `var1` of `DWORD` type to zero is done by writing:

```
var1 = 0
```

Note: Constants can NEVER be changed

7.5.5 Accessing complex data types

Accessing a value in a structure is done by writing `structureName.elementName` where:

- `structureName` is the variable name
- `elementName` is the value name

Accessing a member of array is done by writing `arrayName[index]` where:

- `arrayName` is the variable name
- `index` is the index in array

Accessing a member of array that is a member of structure is done by writing `structureName.arrayName[index]` where:

- `structureName` is the name of structure
- `arrayName` is the array field name
- index `index` the index in array

7.5.6 System Libraries

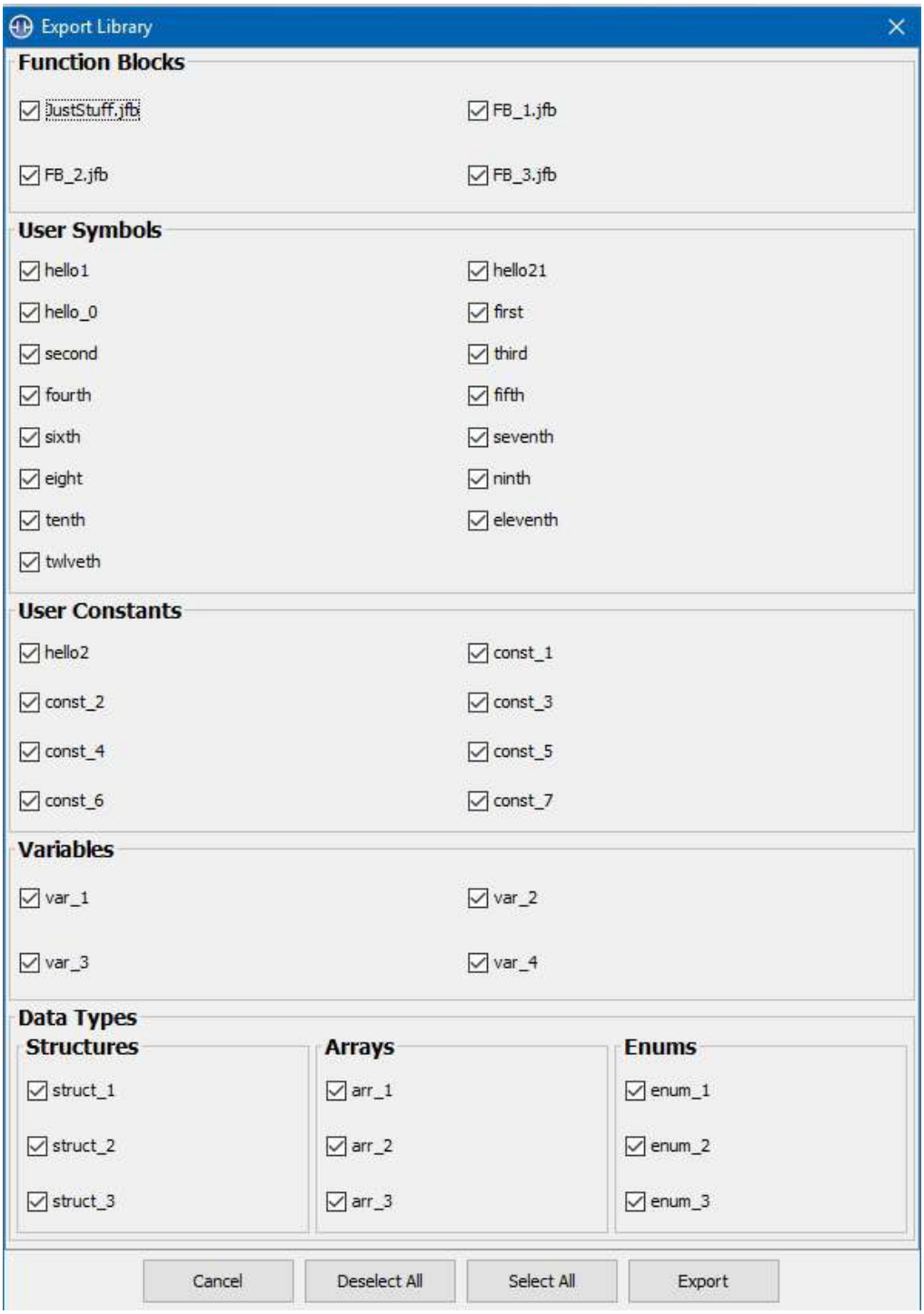
There are number of system libraries that can be used in FB-C code. All of them are standard libraries.

- `stdint.h`
- `stdbool.h`
- `string.h`
- `math.h`
- `stdlib.h`

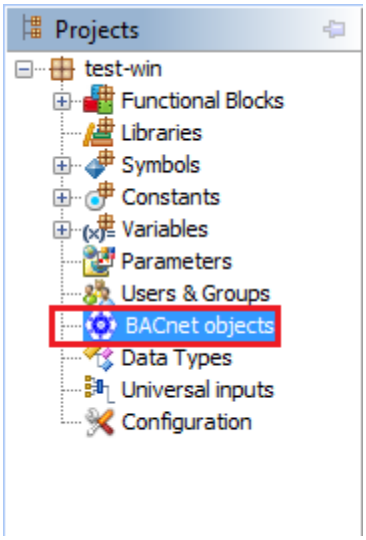
Since users of FB-C code have to have at least basic knowledge of C Programming, these libraries are not discussed in this document.

7.6 Exporting FBs

To export Function Blocks and all other variables inside your project (User Symbols, User Constants, Variables or Data Types) go to File>Export Function Blocks… or press Ctrl+Shift+E. You will be presented with a dialog showing the selection of things to export. Once you select the content you want your library to contain simply press export and and select the location for the library file. After that the file will be created.



By clicking on node BACnet Object, you open the table **for setting the list of BACnet objects.**

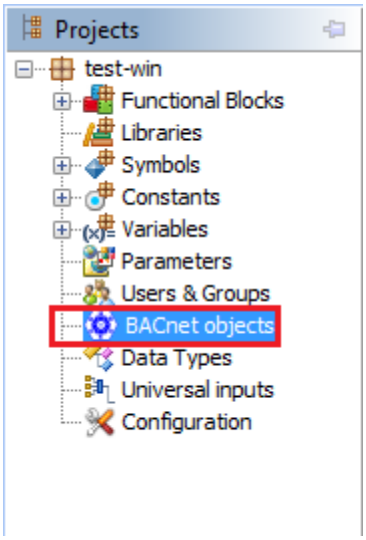


First **you choose the type of object you want to define:**

Analog values
Analog inputs
Binary values
Binary inputs
Multistate values

8 BACnet Objects

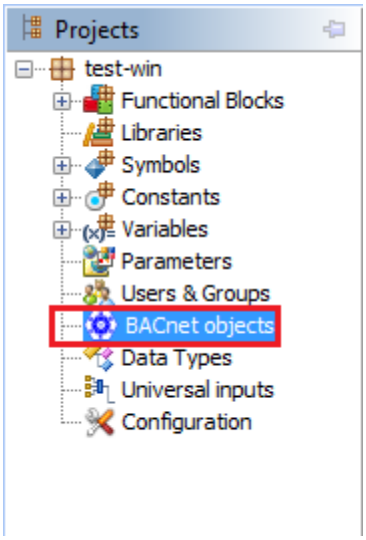
By clicking on node BACnet Object, you open the table for setting the list of BACnet objects.



First you choose the type of object you want to define:

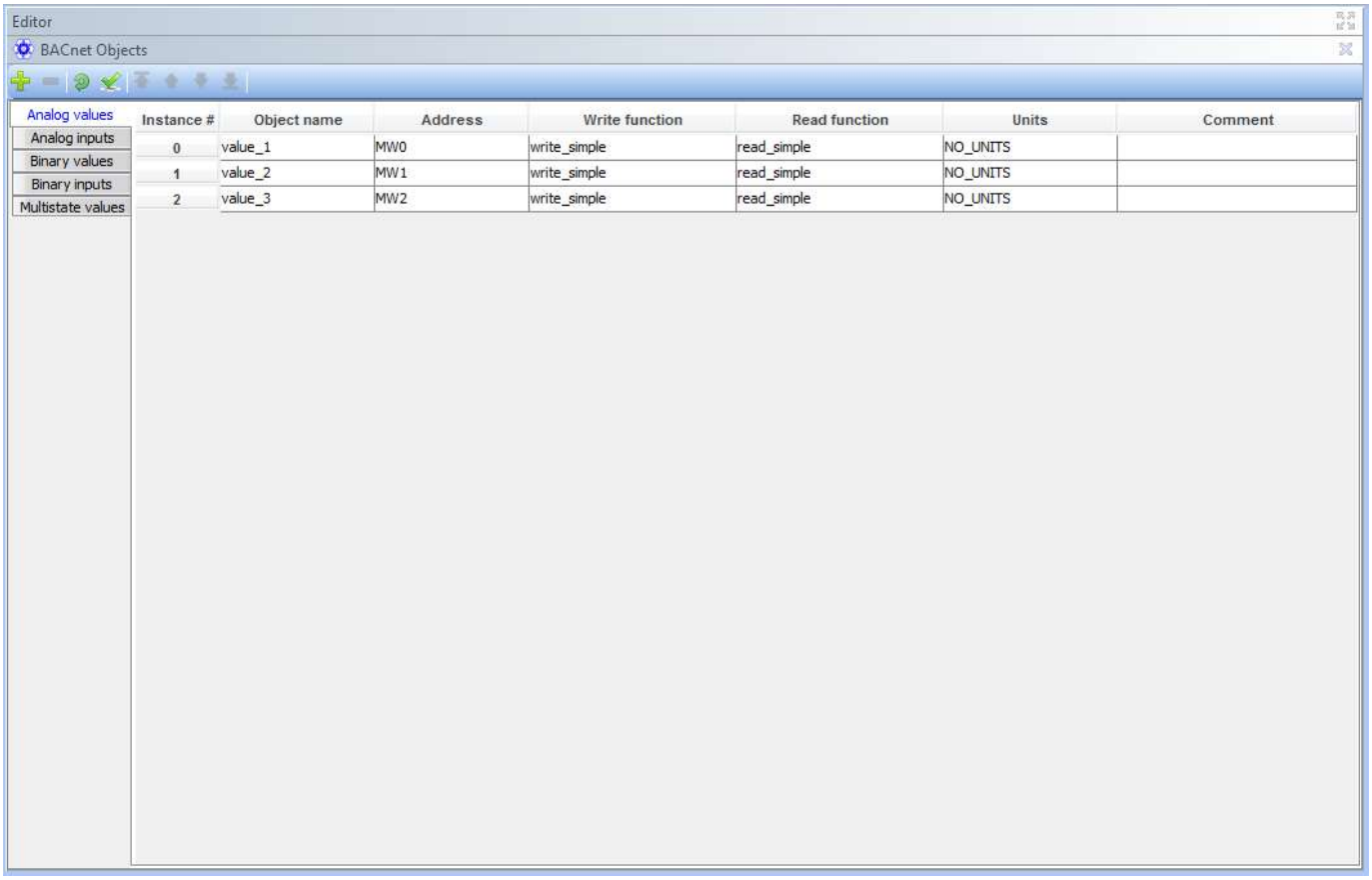
- Analog values
- Analog inputs
- Binary values
- Binary inputs
- Multistate values

By clicking on node BACnet Object, you open the table **for setting the list of BACnet objects.**

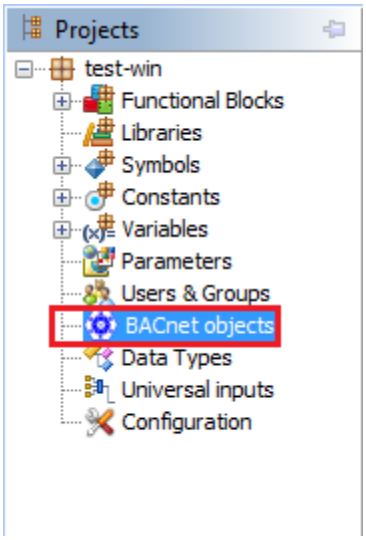


First **you choose the type of object you want to define:**

Analog values
Analog inputs
Binary values
Binary inputs
Multistate values



By clicking on node BACnet Object, you open the table **for setting the list of BACnet objects.**



First **you choose the type of object you want to define:**

Analog values
Analog inputs
Binary values
Binary inputs
Multistate values

9 Compiling

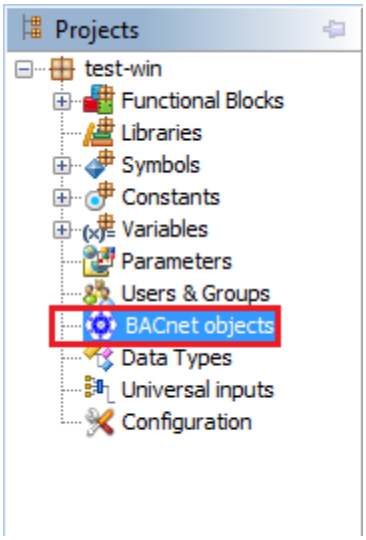
Compiling the program is done through the Compile Menu or a shortcut.

Compilation process is syntax analysis of all of FB (both diagrams and code) and translating the program into target code.

After successful compilation a data file named after the project is created. This file is used to write the program to the PLC. Also, all of the “.c”, “.h” and “.o” files are created. These files can be explored through a standard text editor.

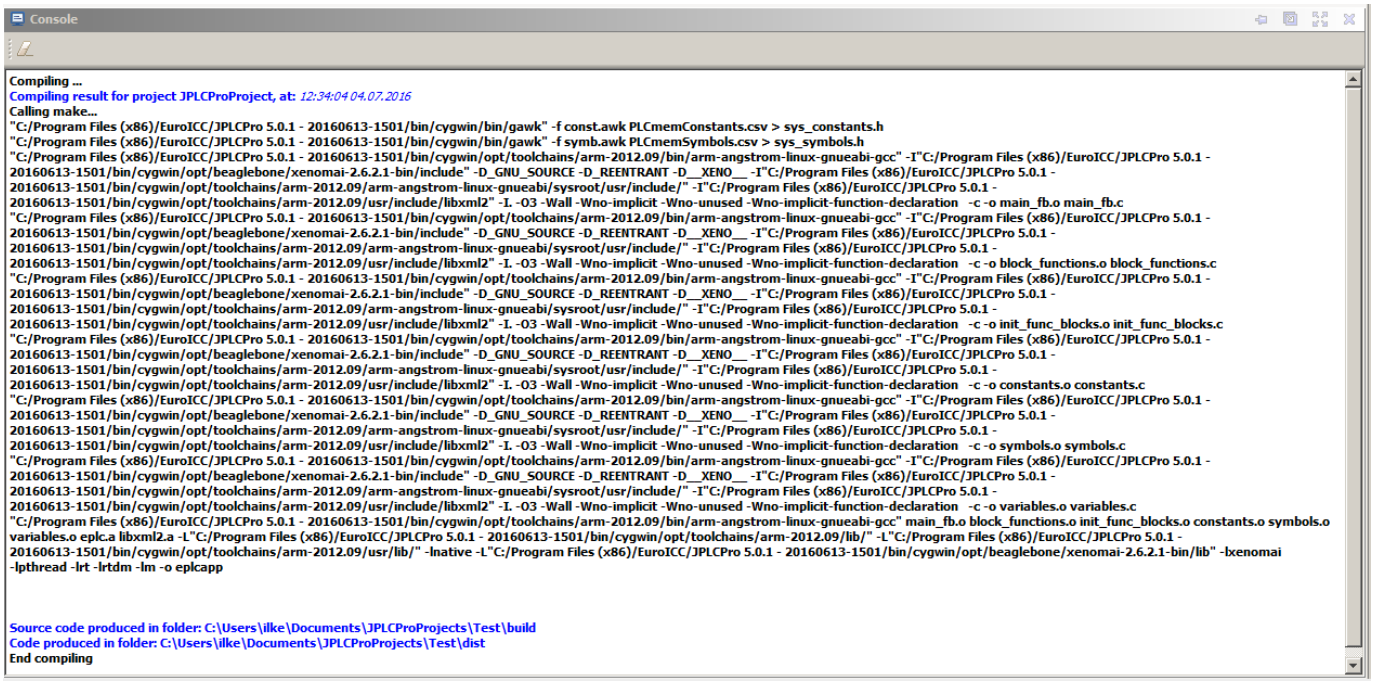
Successful compilation example:

By clicking on node BACnet Object, you open the table **for setting the list of BACnet objects.**

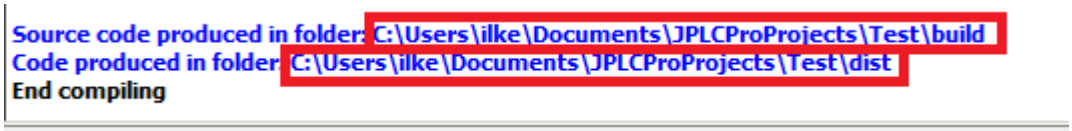


First **you choose the type of object you want to define:**

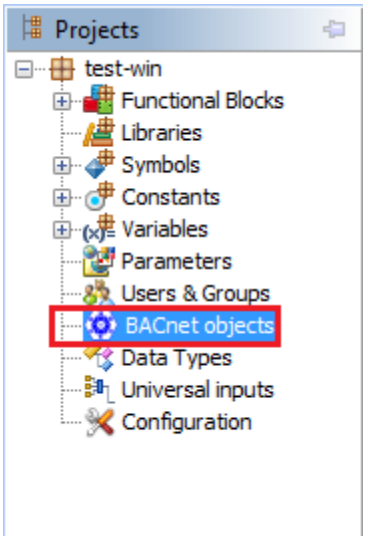
Analog values
Analog inputs
Binary values
Binary inputs
Multistate values



To open the `build` or `distr` (distribution) folder left-click on the folder path in Console View.



By clicking on node BACnet Object, you open the table **for setting the list of BACnet objects.**

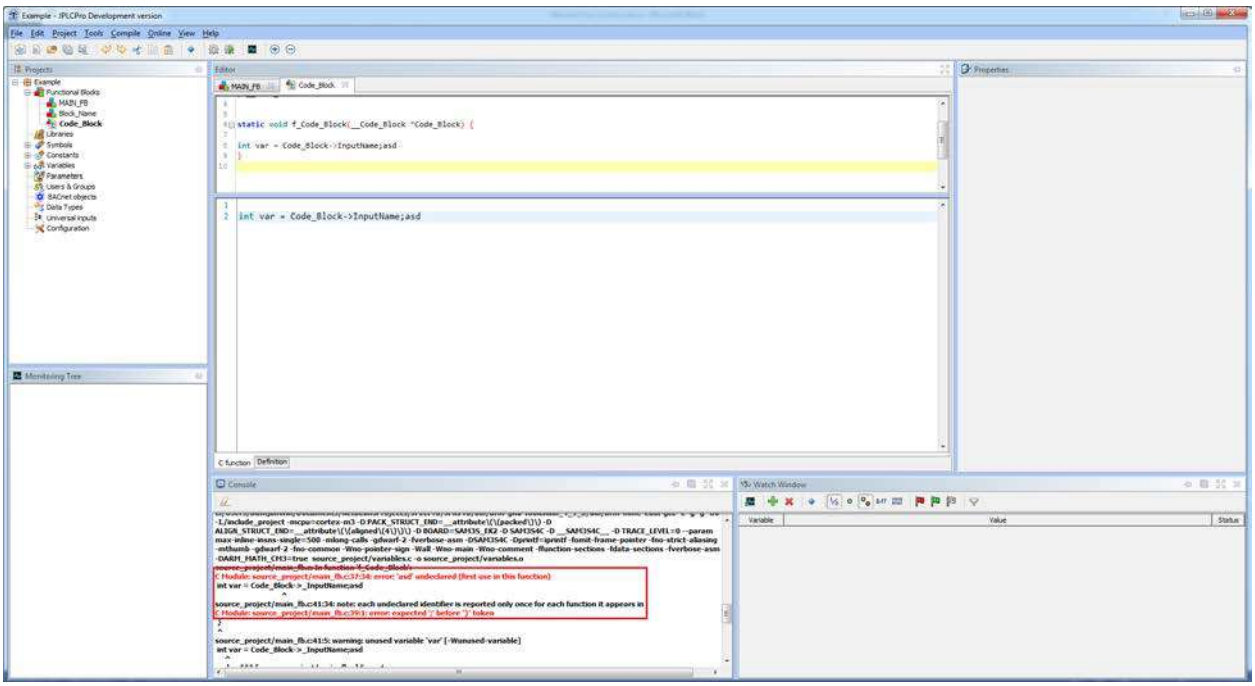


First **you choose the type of object you want to define:**

Analog values
Analog inputs
Binary values
Binary inputs
Multistate values

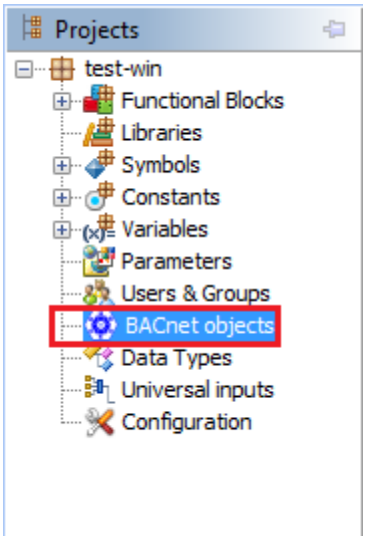
All of the information about the compilation process and files locations can be found in Console View.

If an error occurs during the compilation user is informed. More information about the error can be found in the Console View.



An error is displayed in Console View, painted red, in the following format:

By clicking on node BACnet Object, you open the table **for setting the list of BACnet objects.**



First **you choose the type of object you want to define:**

Analog values
Analog inputs
Binary values
Binary inputs
Multistate values

```
module:file_location:error_row:error_position:error_info
```

where:

- `module` is the module where error occurred
- `file_location` is the file where error occurred
- `error_row` is the row where error occurred
- `error_position` is the position of the error in a row
- `error_info` is the information generated by compiler about what caused the error

10 Communication

10.1 Downloading the program

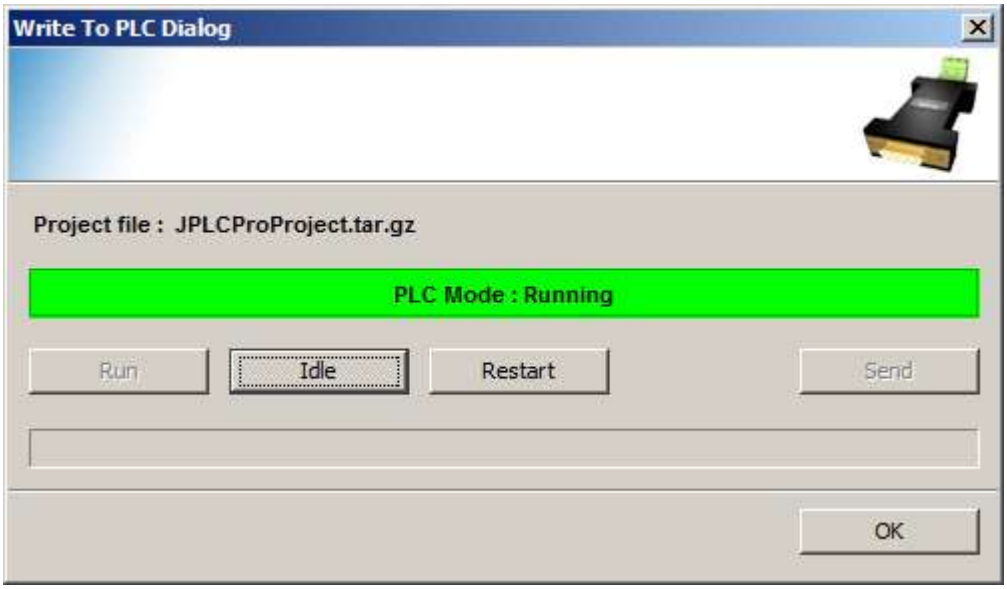
To download the program to EPLC the device needs to be connected to jPLCPro. The program must compile without error or download will fail.

There are two ways to download the program, via the `Write To PLC` menu item or from the tool bar.

To compile and download a program left-click the `Compile and Run` icon in the toolbar. After compiling `Write to PLC` dialog will appear, download already in progress. Wait for download to complete and left-click the `OK` button.

10.1.1 Write To PLC dialog

`Write to PLC` dialog is accessed through the `Write To PLC` menu item in `Online Menu`



It features:

- Project file name
- Status bar
- Run button
- Idle button
- Send button
- Ok button
- Download status bar

Project File name is the name of the project file.

Status bar displays information about the current state of the program (running, idle or sending). It is green when the program is running, yellow when the program is stopped and blue while the program is downloaded to the PLC.

`Run` button is used to start the program on PLC.

`Idle` button is used to stop the program on PLC.

`Send` button is used to download the program to PLC.

OK button is used to close the Write To PLC dialog.

Download status bar displays information about the download of the program.

To be able to start or download the program the PLC status must be Idle.

10.1.2 Monitoring

Monitoring is a way to observe information about the FBs and various values of the PLC in real-time.

Monitoring needs for jPLCPro and EPLC to be connected and the program on EPLC to be running.

Monitoring is started through the Monitor On menu item in Online Menu, or via Monitor On icon from toolbar, or via shortcut CTRL+M

Monitoring is stopped through the Monitor Off menu item that replaces the Monitor On menu item in Online Menu during monitoring or via Monitor Off icon from toolbar or via shortcut CTRL+M.

When monitoring is started all of the FB tabs (except the MAIN_FB tab) are temporarily closed. This happens because monitoring is done on instances and editing is done on the bodies of FB.

During monitoring, FB instances can be entered with double-clicking the appropriate FB. When a FB instance is entered during monitoring a new tab is opened. The name of this tab is Parent/Name_Index where:

- Parent is the parent FB
- Name is the name of the FB instance
- Index is the index of the FB instance

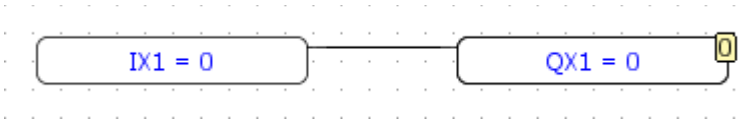
FB-C cannot be entered as there is nothing to display in them.

10.1.3 FB Monitoring

FBs look the same as in edit mode with addition of the current values of inputs and outputs. Values of inputs and outputs are displayed below their names.

10.1.4 Input/Output Monitoring

Inputs/Outputs look the same as in edit mode, with addition of their current values. These values are displayed next to the name of input/output.



10.1.5 Forcing Values

Forcing (values) is a feature that can be used only during monitoring. It allows a user to change the current value of element (input or output) i.e. to force a value.

Forcing is done via Force… or Force on menu items in Online menu. Force on is used for forcing values of Bit type, and Force… is used for forcing all the others values.

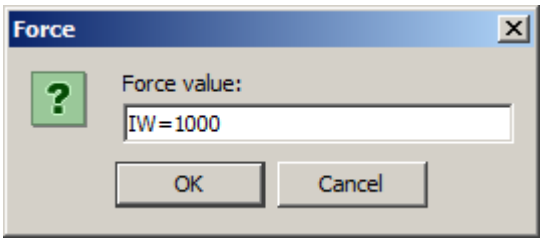
To use Force on an input/output of type bit must be selected. Force on forces variable value

to 1. To force value of 0 to variable of bit type, select Force off. To turn forcing off, select Force off again.

Force… displays a dialog where user enters what and how should be forced in the format of N=V where:

- N is the name of variable, symbol or memory zone
- V is the value

To force left-click the OK button. To exit the dialog left-click the Cancel button.



Force dialog can be invoked by pressing F11 shortcut.

10.1.6 Deforcing Values

De-forcing values is stopping the forcing of values. It can only be done on forced values.

To de-force any value left-click the “Deforce…” menu item in the “Online” menu.

A deforce dialog will appear. Its layout and button functions are the same as in the “Force…” dialog.

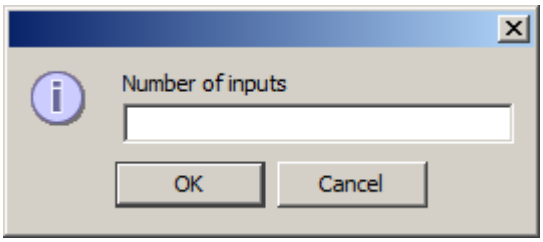
To deforce a value write its name.

Deforce dialog can be invoked by pressing F12 shortcut.

11 FB Overview

As mentioned before, there are a number of already defined FBs. Some of them have predetermined number of inputs, and some allow user to choose the number of inputs.

For those FBs with variable number of inputs a “Number of Inputs” dialog will appear

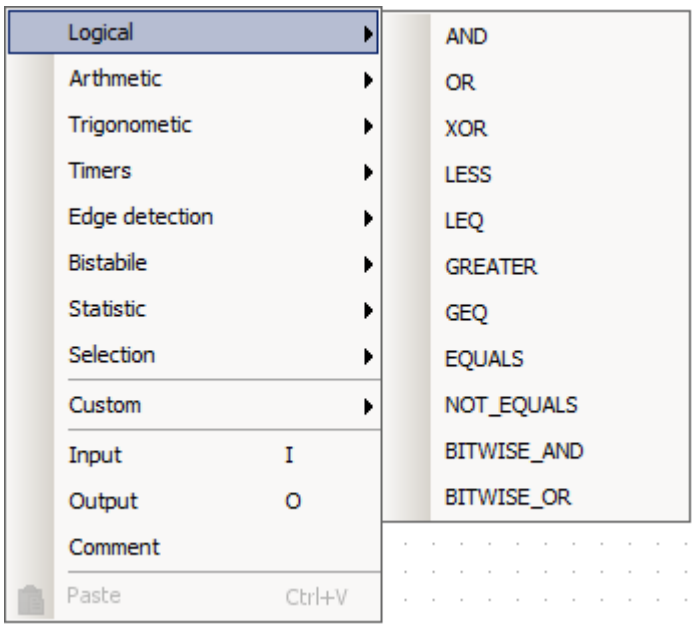


Type in the desired number of inputs from the keyboard and left-click the OK button. Left-click the `Cancel` exit the dialog.

When number of inputs is selected, the element will have that number appended to its name.

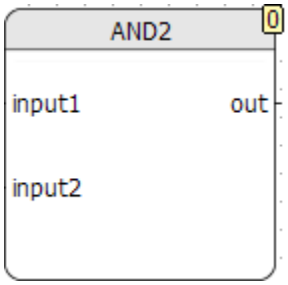
11.1 Logical

Logical is a group of FBs that are operating with bits or bits in bytes or words etc.

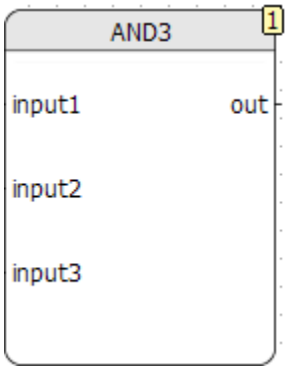


11.1.1 AND

AND FB is a FB that performs the and operation on inputs.



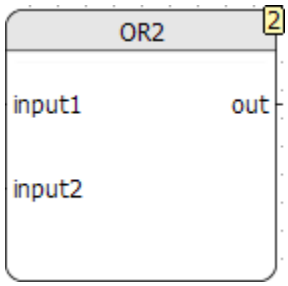
input2	input1	out
numeric	numeric	BIT
0	0	0
0	1	0
1	0	0
1	1	1



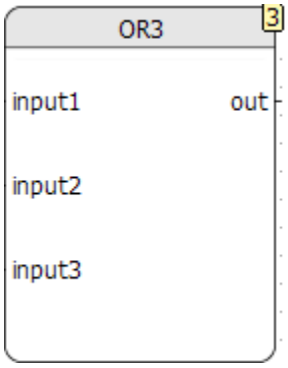
input3	input2	input1	out
numeric	numeric	numeric	BIT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	1	1

11.1.2 OR

OR FB is a FB that performs the `or` operation on inputs.



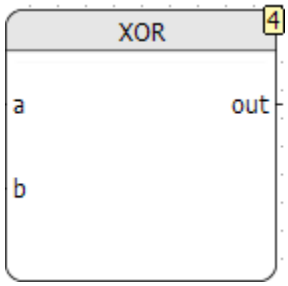
input2	input1	out
numeric	numeric	BIT
0	0	0
0	1	1
1	0	1
1	1	1



input3	input2	input1	out
numeric	numeric	numeric	BIT
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	1	1

11.1.3 XOR

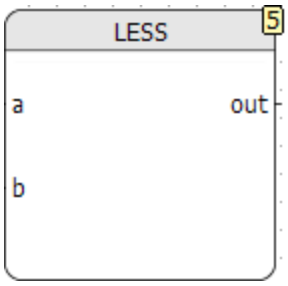
XOR FB is a FB that performs the `xor` (exclusive `or`) operation.



a	b	out
BIT	BIT	BIT
0	0	0
0	1	1
1	0	1
1	1	0

11.1.4 LESS

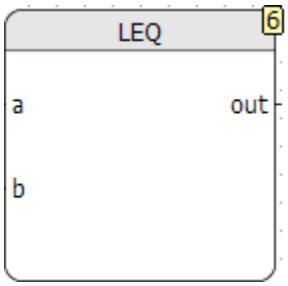
LESS FB compares a and b. Output is `1` if a `<` b and is `0` if a `>=` b.



a, b	out
numeric	BIT
a >= b	0
a < b	1

11.1.5 LEQ

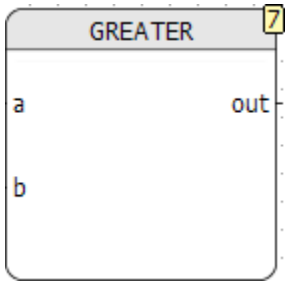
LEQ FB compares a and b. Output is `1` if a `<=` b and is `0` if a `>` b.



a, b	out
numeric	BIT
a > b	0
a <= b	1

11.1.6 GREATER

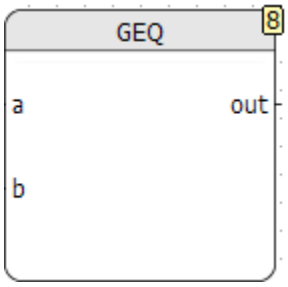
GREATER FB compares a and b. Output is `1` if a `>` b and is `0` if a `<=` b.



a, b	out
numeric	BIT
a <= b	0
a > b	1

11.1.7 GEQ

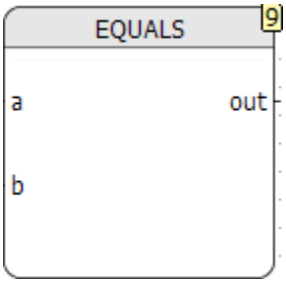
GEQ FB compares a and b. Output is `1` if a `>=` b and is `0` if a `<` b.



a, b	out
numeric	BIT
a < b	0
a >= b	1

11.1.8 EQUALS

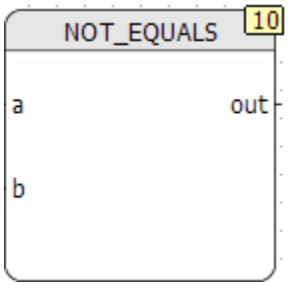
EQUALS FB compares a and b. Output is `1` if a `=` b and is `0` if a `!=` b.



a, b	out
numeric	BIT
a != b	0
a = b	1

11.1.9 NOT_EQUALS

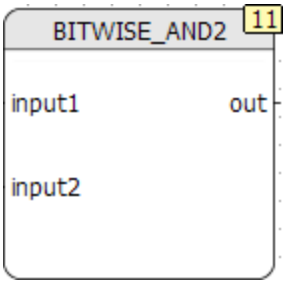
NOT_EQUALS FB compares a and b. Output is 1 if a != b and is 0 if a = b.



a, b	out
numeric	BIT
a = b	0
a != b	1

11.1.10 BITWISE_AND

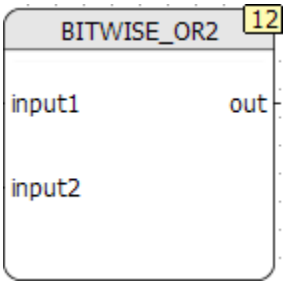
BITWISE_AND performs bitwise and function on corresponding bits of a and b.



input1, input2	out
integral	integral
11000011, 01010101	01000001

11.1.11 BITWISE_OR

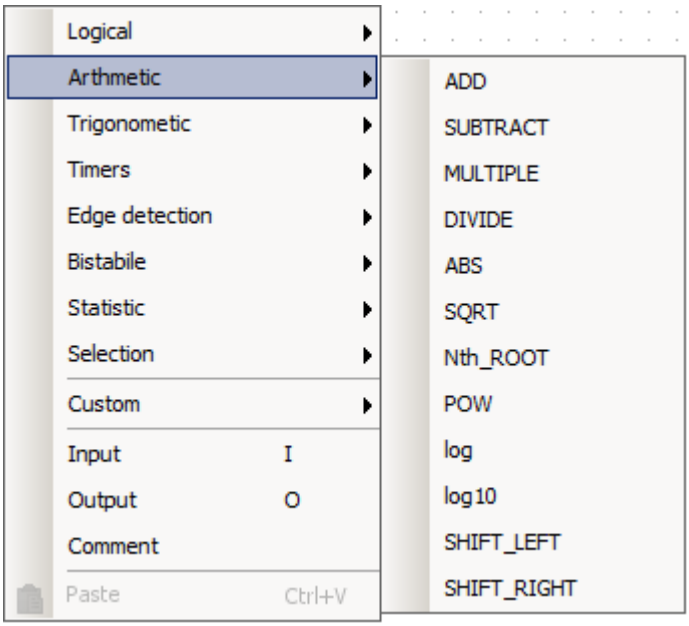
BITWISE_OR performs bitwise or function on corresponding bits of a and b.



input1, input2	out
integral	integral
11000011, 01010101	11010111

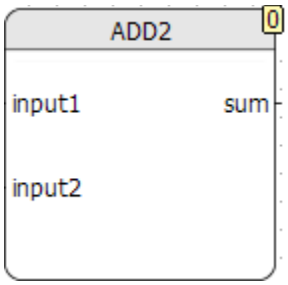
11.2 Arithmetic

Arithmetic group is a group of arithmetic FBs.



11.2.1 ADD

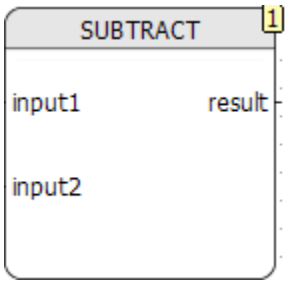
ADD FB is a FB that performs addition.



input1	input2	sum
numeric	numeric	numeric
a	b	a + b

11.2.2 SUBTRACT

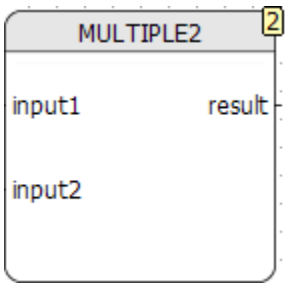
SUBTRACT FB is a FB that performs subtraction.



input1	input2	result
numeric	numeric	numeric
a	b	a - b

11.2.3 MULTIPLY

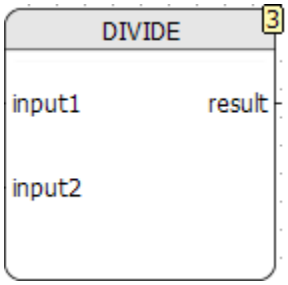
MULTIPLY FB is a FB that performs multiplication.



input1	input2	result
numeric	numeric	numeric
a	b	a * b

11.2.4 DIVIDE

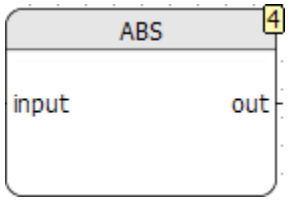
DIVIDE FB is a FB that performs division.



input1	input2	result
numeric	numeric	numeric
a	b	a / b

11.2.5 ABS

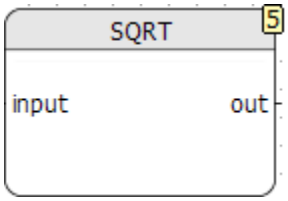
ABS FB is a FB that finds absolute value of `input`.



input	out
numeric	numeric
x	|x|

11.2.6 SQRT

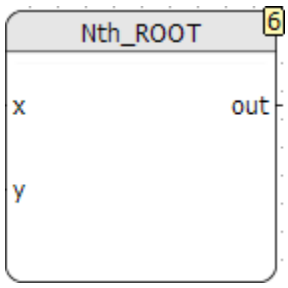
SQRT FB is a FB that finds square root of `input`.



input	out
numeric	FLOAT
x	$\sqrt{x}$

11.2.7 NTH_SQRT

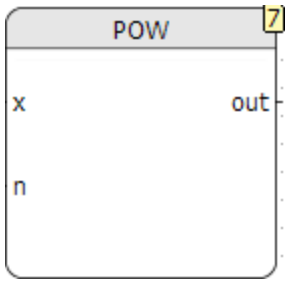
NTH_SQRT FB is a FB that finds n^{th} root of input x.



x	y	out
numeric	**numeric**	**FLOAT**
x	y	$\sqrt[y]{x}$

11.2.8 POW

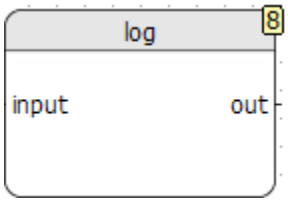
POW FB is a FB that finds the nth power of input.



x	n	out
numeric	**numeric**	**FLOAT**
x	n	x^n

11.2.9 LOG

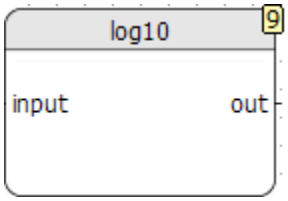
LOG FB is a FB that finds natural logarithm of `input` value.



input	out
numeric	**FLOAT**
x	ln(x)

11.2.10 LOG10

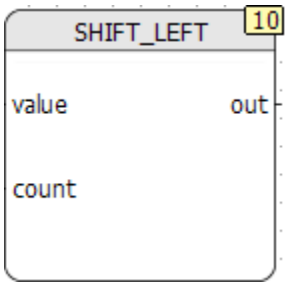
LOG10 is a FB that finds logarithm with base 10 of `input` value.



input	out
numeric	**FLOAT**
x	log(x)

11.2.11 SHIFT_LEFT

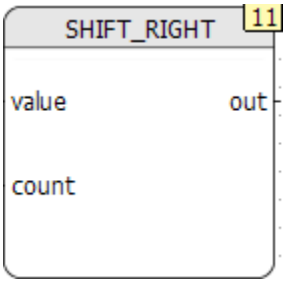
SHIFT_LEFT is a FB that shifts bits of `value` to the left.



value	count	out
integral	**integral**	**integral**
10101010	3	00010101

11.2.12 SHIFT_RIGHT

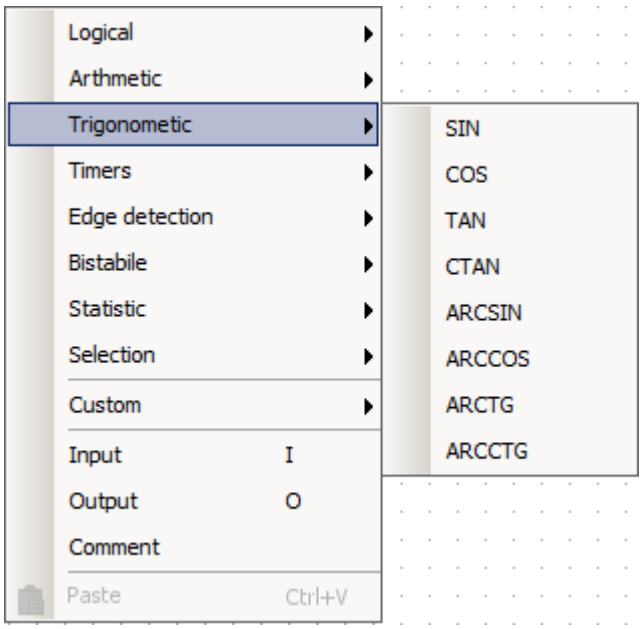
SHIFT_RIGHT is a FB that shifts bits of `value` to the right



value	count	out
integral	integral	integral
10101010	3	01010000

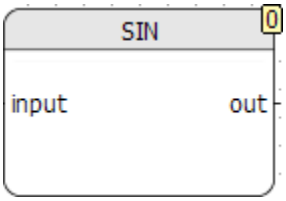
11.3 Trigonometric

Trigonometric group is a group of FBs that perform trigonometric calculations.



11.3.1 SIN

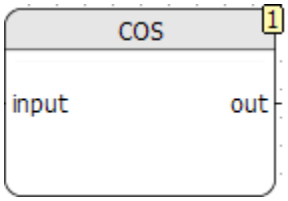
SIN FB is a FB that finds sinus of a value.



input	out
numeric	FLOAT
x	sin(x)

11.3.2 COS

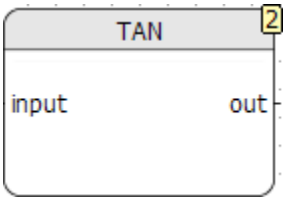
COS FB is a FB that finds cosines of a value.



input	out
numeric	FLOAT
x	cos(x)

11.3.3 TAN

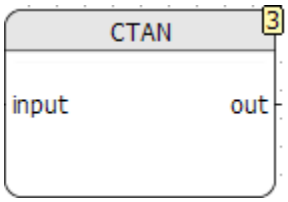
TAN FB is a FB that finds tangents of a value.



input	out
numeric	FLOAT
x	tan(x)

11.3.4 CTAN

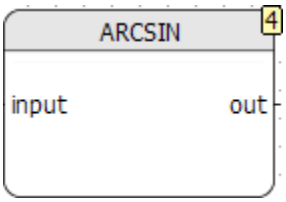
CTAN FB is a FB that finds cotangents of a value



input	out
numeric	FLOAT
x	ctg(x)

11.3.5 ARCSIN

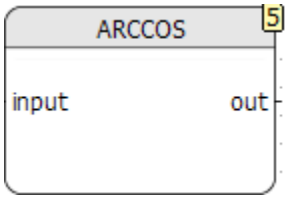
ARCSIN FB is a FB that finds arcus sinus of a value



input	out
numeric	FLOAT
x	arcsin(x)

11.3.6 ARCCOS

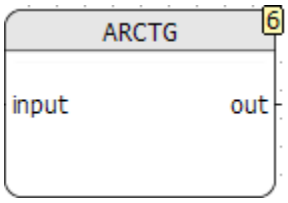
ARCCOS FB is a FB that finds arcus cosines of a values



input	out
numeric	FLOAT
x	arccos(x)

11.3.7 ARCTG

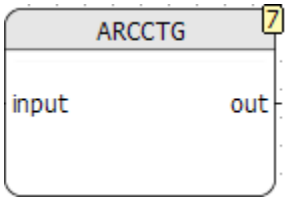
ARCTG FB is a FB that finds arcus tangents of a value



input	out
numeric	FLOAT
x	arctan(x)

11.3.8 ARCCTG

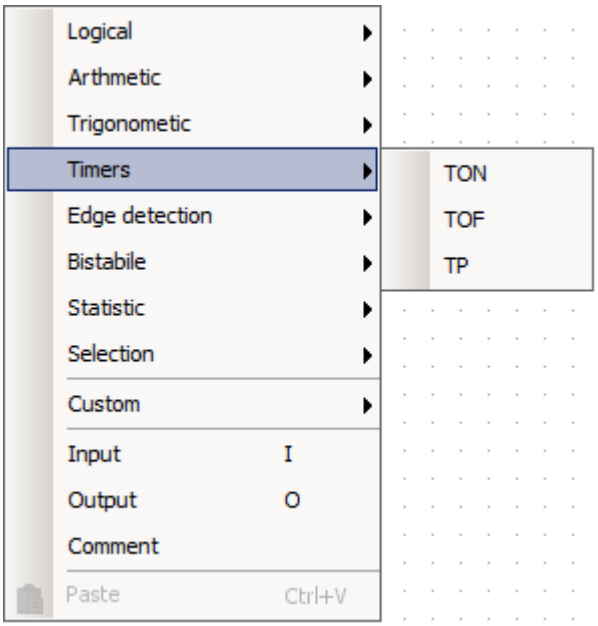
ARCCTG FB is a FB that finds arcus cotangents of a values



input	out
numeric	FLOAT
x	arcctan(x)

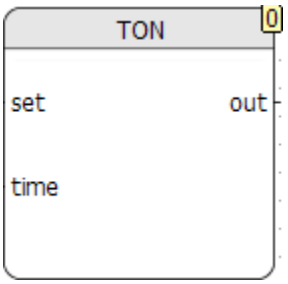
11.4 Timers

Timers group is a group of FB that deal with time.

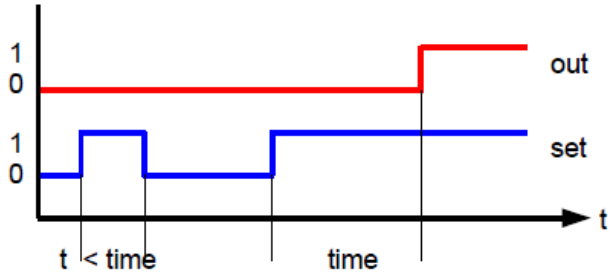


11.4.1 TON

TON FB activates the output after it is triggered, and `time` period expires.

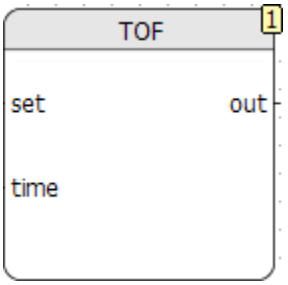


set	time	out
BIT	integral	BIT

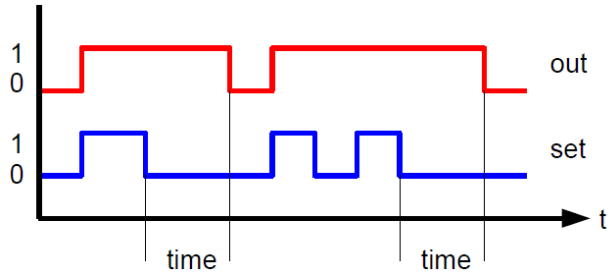


11.4.2 TOF

TOF FB activates the output upon triggering and deactivates the output after a period of time.

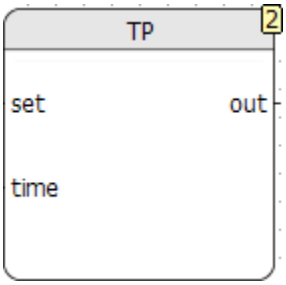


set	time	out
BIT	integral	BIT

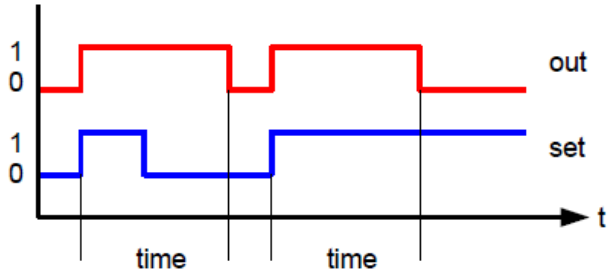


11.4.3 TP

TP FB hold the output active for a period of time.

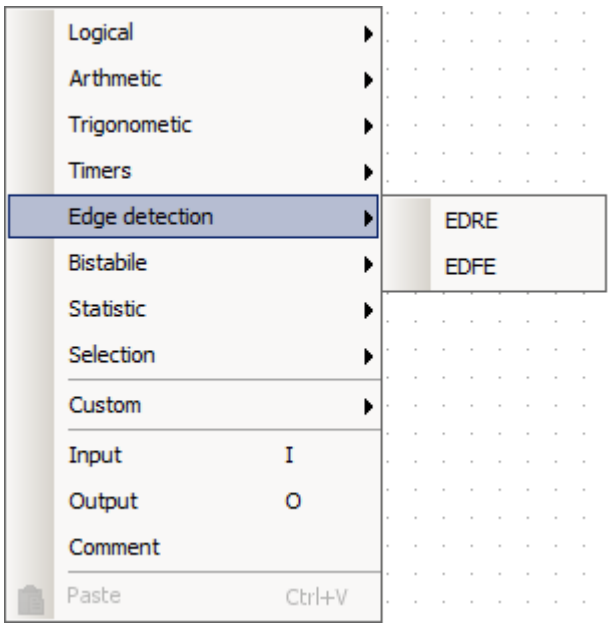


set	time	out
BIT	integral	BIT



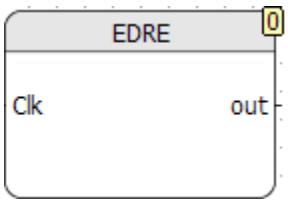
11.5 Edge Detection

Edge Detection group is a group of FBs that are triggered on signal level change.

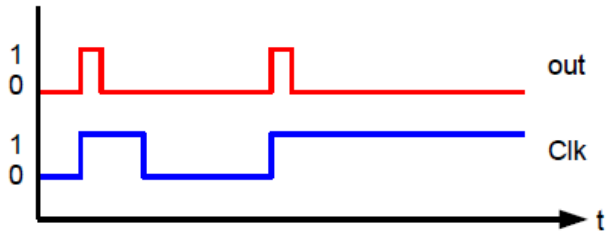


11.5.1 EDRE

EDRE FB detects signal rising edge.

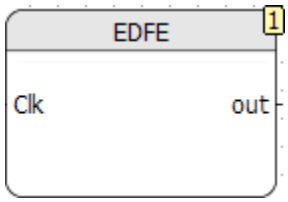


Clk	out
BIT	BIT

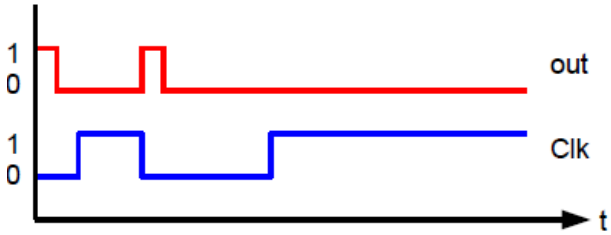


11.5.2 EDFE

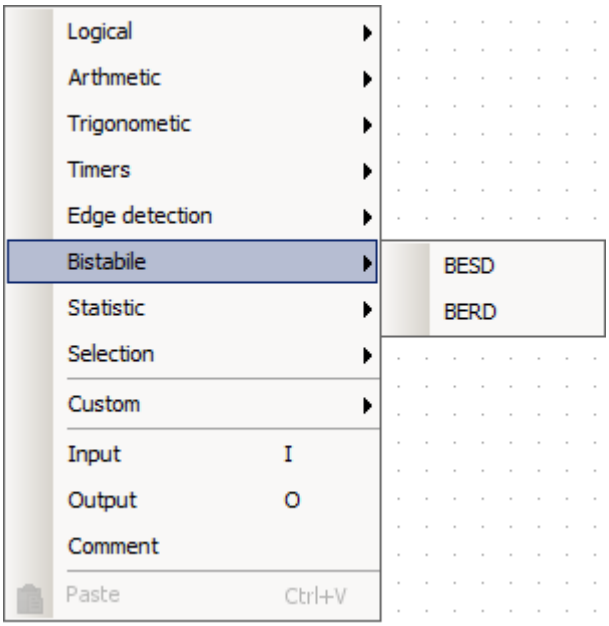
EDFE FB detects signal falling edge.



Clk	out
BIT	BIT



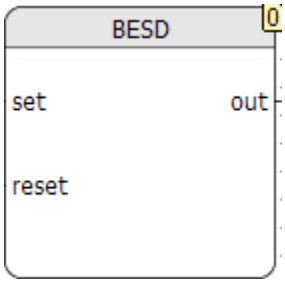
11.6 Bistable



11.6.1 BESD

BESD FB (Bistable Edge Set Dominant) detects status of its inputs, and sets its output

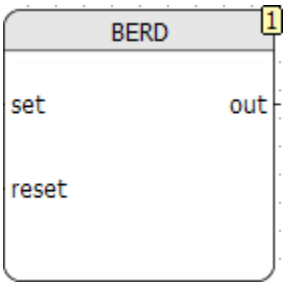
according to table.



set	reset	prev. out	out
BIT	BIT	BIT	BIT
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

11.6.2 BERD

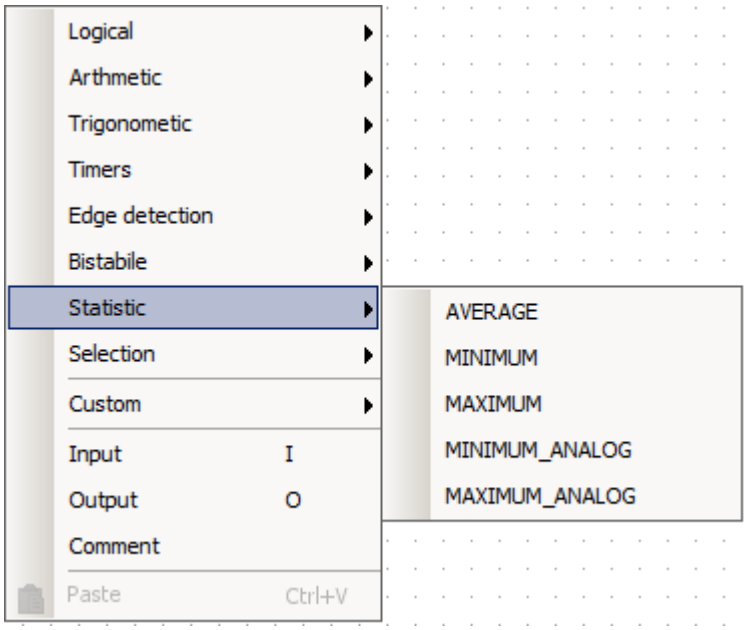
BESD FB (Bistable Edge Reset Dominant) detects status of its inputs, and sets its output according to table.



set	reset	prev. out	out
BIT	BIT	BIT	BIT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

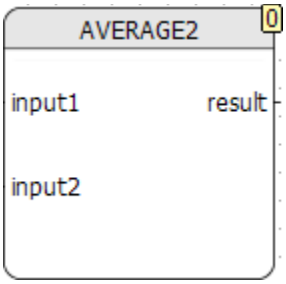
11.7 Statistic

Static group is a FBs groups that find various statistical information.



11.7.1 AVERAGE

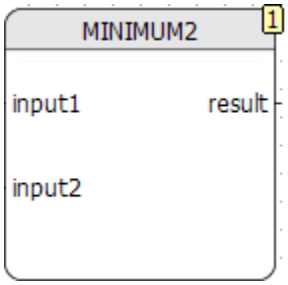
AVERAGE FB finds the average value of inputs.



inputs	out
numeric	numeric
input1, input2, … inputN	$\frac{\sum_{m=1-N} inputM}{N}$

11.7.2 MINIMUM

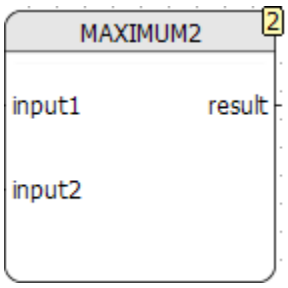
MINIMUM FB finds the minimum value of inputs.



input1, input2	out
numeric	numeric
a, b	min(a, b)

11.7.3 MAXIMUM

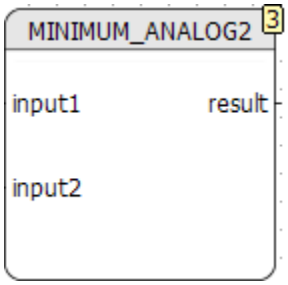
MAXIMUM FB finds the maximum value of inputs.



input1, input2	out
numeric	numeric
a, b	max(a, b)

11.7.4 MINIMUM_ANALOG

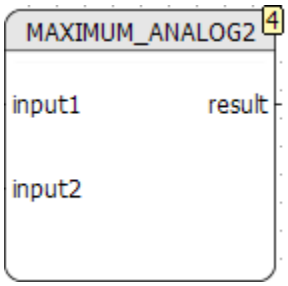
MINIMUM_ANALOG FB finds the minimum value of inputs.



input1, input2	out
numeric	numeric
a, b	min(a, b)

11.7.5 MAXIMUM_ANALOG

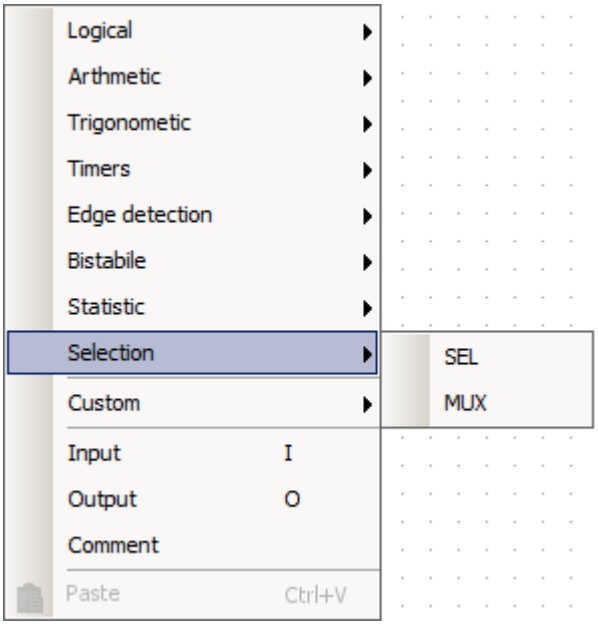
MAXIMUM_ANALOG FB finds the maximum value of outputs.



input1, input2	out
numeric	numeric
a, b	max(a, b)

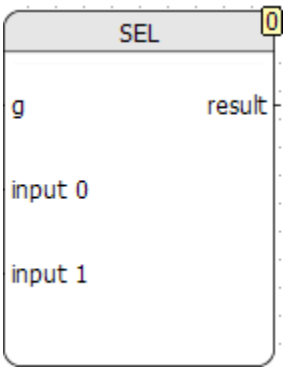
11.8 Selection

Selection group is a FBs group selects one of the inputs and passes it to output.



11.8.1 SEL

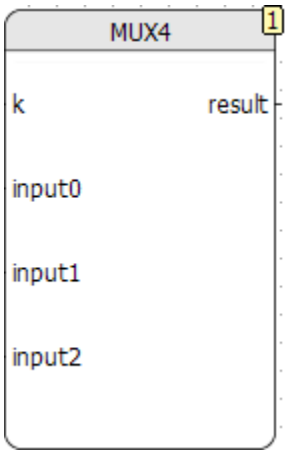
SEL FB passes the first input if “g” is inactive or the second input if “g” is active.



g	result
integral	numeric
0	input0
1	input1

11.8.2 MUX

MUX FB passes the input indicated by the “k” input.



k	result
integral	numeric
0	input0
1	input1
2	input2

12 FB Programming Examples

12.1 Binary Input to Binary Output

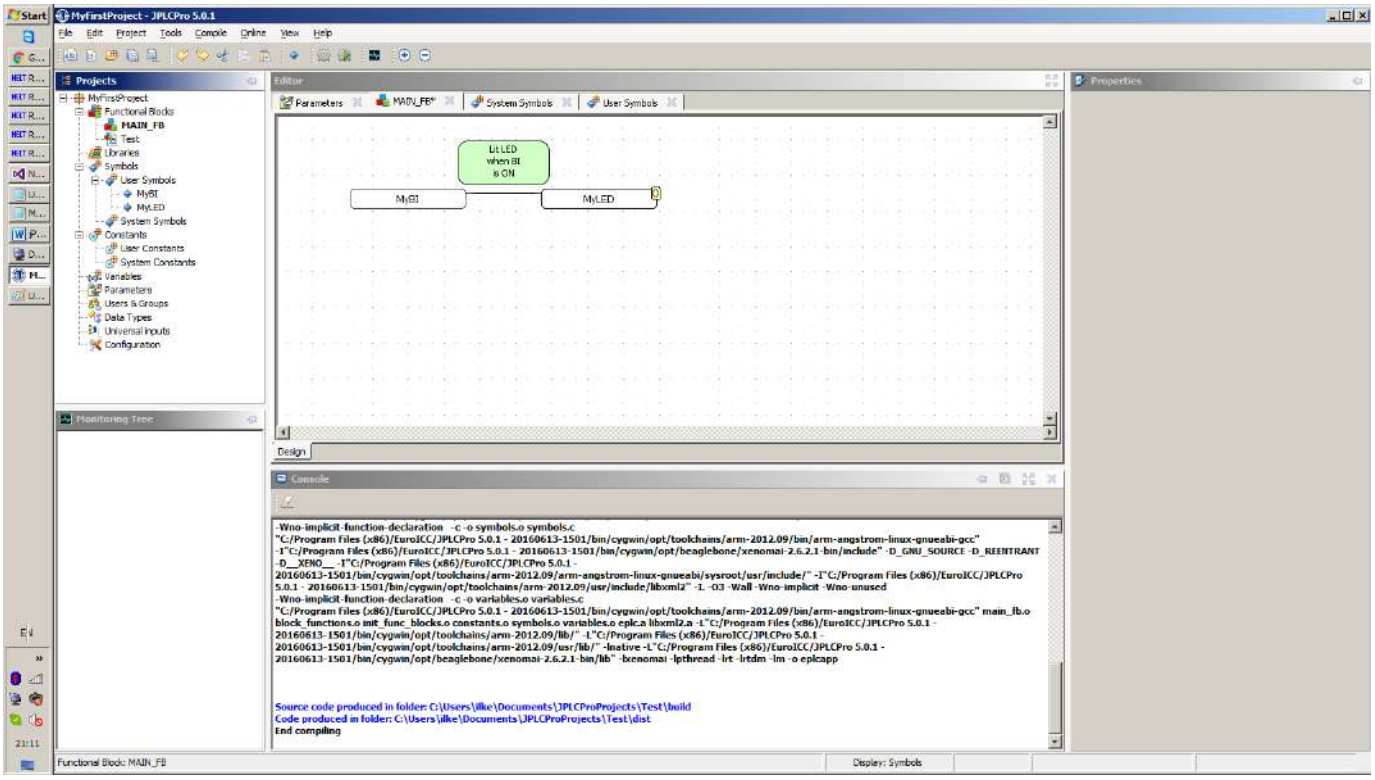
In FB programming, there is not much to do with strings there is no use of `Hello World` application (unless you have a device with Display). So our first example will be to just pass Binary Input (BI) to Binary Output (BO). This example works on all devices, which have at least one BI, and at least one BO. It does not have much sense if yours BO is relay output, but you may want to lit LED (which is also BO), when your BI is On.

1. First, we have to create new jPLCPro Project:
    a. Click `File->New->NewProject` and in dialog that opens, type `MyFirstProject`, and then press `OK` button.
        - Select where you want your Project to be saved.
    b. jPLCPro creates new Project, and displays empty `Main FB` in `Editor View`.
2. Next, we want to give meaningful names to our BI and BO:
    a. In `Project View`, open `MyFirstProject->Symbols->User Symbols`
        - Empty `User Symbols` table is opened in `Editor View`.
    b. Click on green + icon, and type `MyBI`, `IX0`, `0`, `My Binary Input` in `Symbol name`, `Address`, `Init Value` and `Comment` fields respectively.
        - This will create User Symbol `MyBI`, which corresponds to first BI on your Device.
    c. Click on green + icon, again, and type `MyLED`, `QX0`, `0`, `My LED` in `Symbol name`, `Address`, `Init Value` and `Comment` fields respectively.
        - This will create User Symbol `MyLED`, which corresponds to first BO on your Device (we assume that it is LED).
    d. Click on `File->Save Project` to make `MyBI` and `MyLED` visible to the rest of the Project.
3. Next, we want to connect `MyBI` with `MyLED`
    a. Go to `Editor View`, `Main FB`
    b. Right click on a grid of `Main FB`, and select `Input` (or you can just type I on a keyboard).
        - New Input is created.
    c. Click on newly created Input (to make it selected), and in `Properties View`, from Drop Down Menu select `MyBI`.
    d. Right click on the grid of `Main FB`, and select `Output` (or you can just type O on a keyboard).
        - New Output is created.
    e. Click on newly created Output (to make it selected), and in `Properties View`, from Drop Down Menu select `MyLED`.
    f. Hover with a mouse over output pin of `MyBI`, left click on a yellow square that appears, then hover over input pin of `MyLED` and left click on a yellow square that appears.
        - `MyBI` and `MyLED` are connected with a line.
4. You may want to add a comment, to explain your intent, to somebody else who is looking into your FB Diagram, or for yourself, if you open Diagram after number of years:
    a. Right click on the grid of Main FB, and select Comment.
        - Green Comment is displayed in Diagram.

b. Type text into Comment Text Box of Properties View:
   - `Lit LED when BI is ON`

c. If you wish, you can change a color of Comment, by clicking on `Color` button, and selecting a color from a pallet.

5. Now, we are ready to compile our project.
   a. Select `Compile->Compile` from `Main Menu`.
      - Project is saved and compiled. It is now ready to be downloaded to your Device.
6. If you have your Device ready and connected, you can download program to Device.
   a. Select `Online->Write to PLC` (or press `Ctrl+W`)
      - `Write to PLC` dialog opens.
   b. Stop progam on the Device by click on `Idle` button.
      - Program status indicates that Device is in Idle state.
   c. Download program to Device with click on `Send` button.
   d. After download has finished, start your program with click on `Run` button.
      - When you apply voltage to first BI on the Device, LED is lit.



12.1.1 Lessons Learned

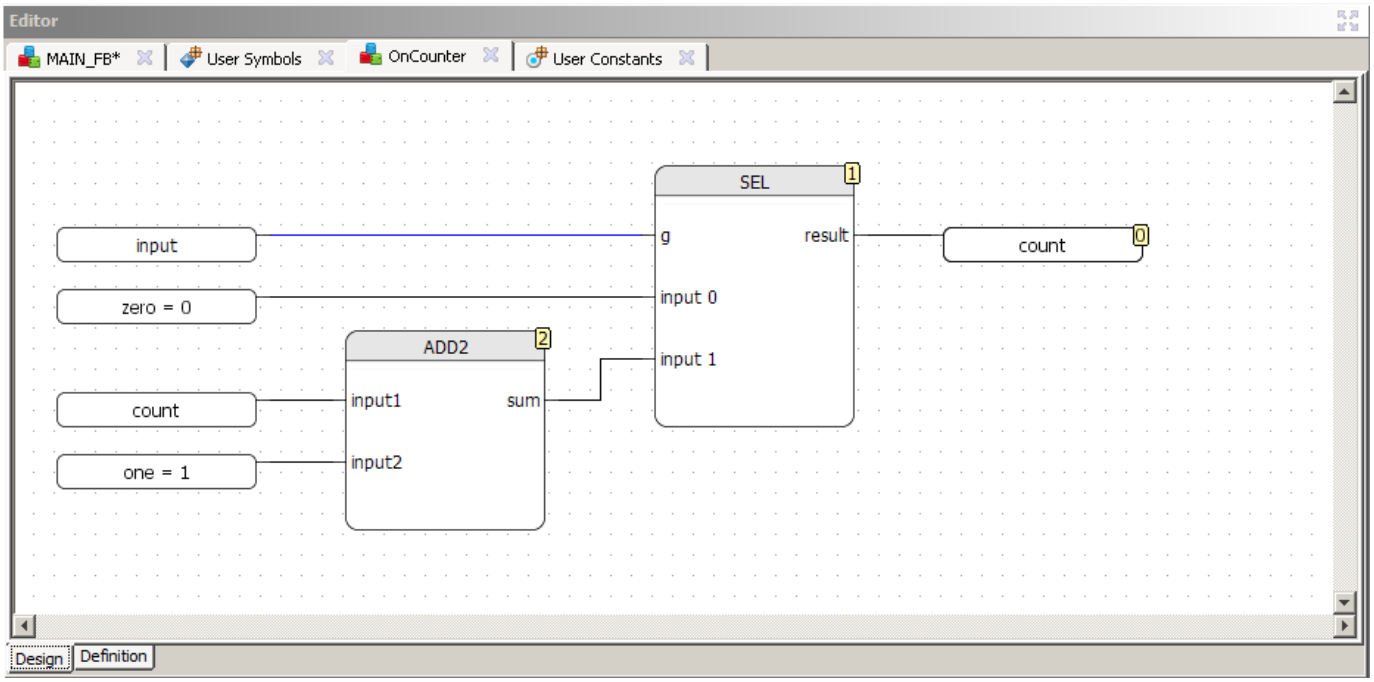
In this example, we learned:

- How to create new Project
- How to give meaningful names to Device inputs and outputs
- How to connect FB inputs and outputs
- How to add Comments and change their color
- How to compile Project
- How to download program to Device and start it

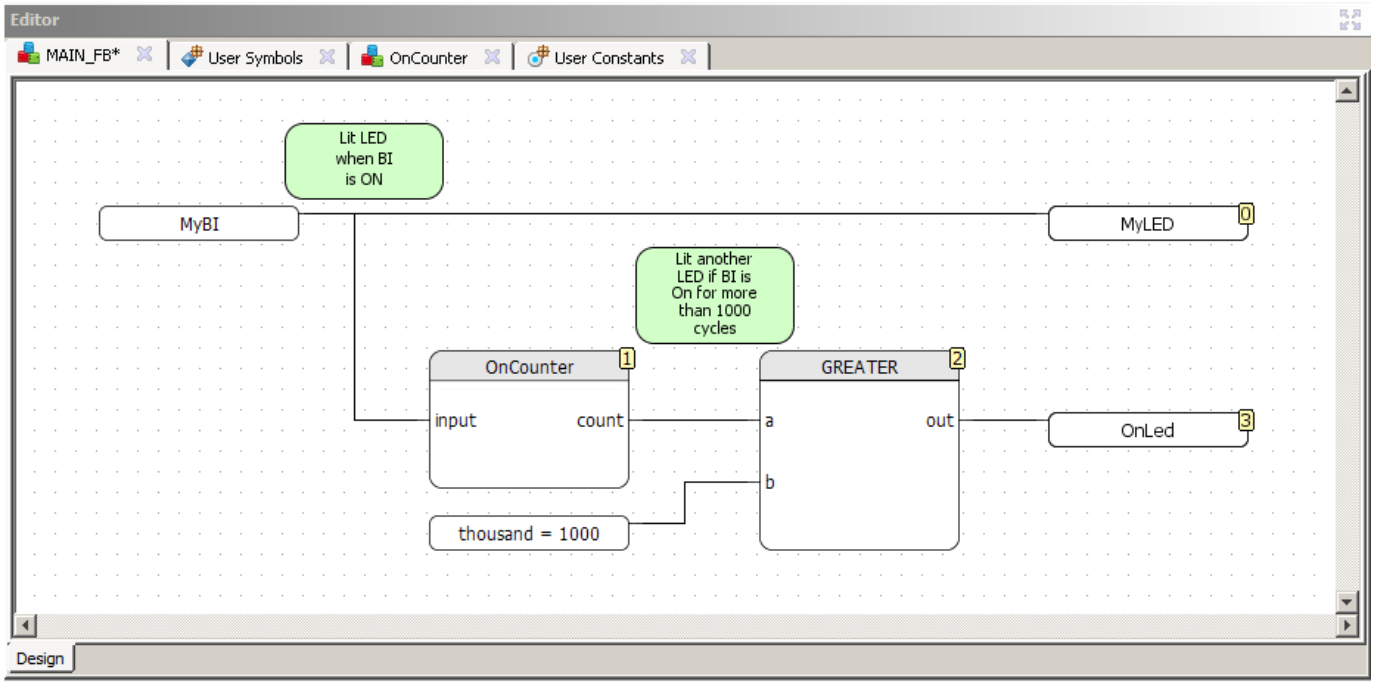
12.2 OnCounter FB

Sometimes, you may want to count number of cycles while your BI is On. In the next example we will create `OnCounter` FB, which does exactly that.

1. We can reuse Project from previous example, by saving it under new name:
   a. Select `File->Save Project As…` from `Main Menu` (or press `Ctrl-W`), and type `MySecondProject` in opened dialog.
   b. You may select the same location as for the first Project.
2. For this example, we will need three constants 0, 1 and 1000:
   a. Select `Constants->UserConstants` from `Project View`.
   b. Create `zero` constant of type `DWORD` with value 0
   c. Create `one` constant of type `DWORD` with value 1
   d. Create `thousand` constant of type `DWORD` with value 1000
3. Now, we can create new FB
   a. In `Project View` right click on `Functional Blocks`, and select `New Module…`
   b. In opened dialog type `OnCounter`, and leave `FB` as selection in Drop Down Box.
      - Click `OK` button, and new, empty, FB will be crated.
4. Next, we create input and output for our new FB:
   a. Select Definition tab
   b. In Inputs list, create one input (with name `input`) of type `BIT`.
   c. In Outputs list, create one output (with name `count`) of type `DWORD`.
   d. Save Project to share changes with the rest of the program.
5. Next, we create FB Diagram for our `OnCounter` FB:
   a. Select `Design` tab of `OnCounter`
   b. Create four inputs, and in Project View name them input, zero, count and one.
   c. Create one output and in Project View name it count
   d. Right click on the grid of OnCounter and add one Arithmetic->ADD block, with 2 inputs
   e. Right click on the grid of OnCounter and add one Selection->SEL block
   f. Connect inputs, outputs and blocks as in following picture:



6. Now, we can use our newly created FB:
    a. Go to `Main FB` in `Editor View`
    b. Right click on `Main FB` grid, and select `Custom->OnCounter` (our local FBs are located there).
        - Our new FB is displayed, with one input and one output.
7. To make some use of our new FB, let's add one more LED output
    a. Let's suppose that it is our second output `QX2`, and name it `OnLED`
8. Finish our Main FB, by adding one input with thousand value, and one Logical->Greater block.
    a. Connect diagram as in following picture



    b. One input can be connected to multiple outputs, as `MyBI` is connected to `MyLED` and `OnCounter->input`.
    c. Any output, can have only one line in.
9. Select Compile->Compile & Run from Main Menu, download program to your Device and run it.

12.2.1 Lessons Learned

In this example we learned:

- How to save Project under new name
- How to create new FB
- How to define inputs and outputs of FB
- How to use output of FB as input to contained FB
- How to create Project constants
- How to use our new FB
- How to compile and download program to Device in one step

NOTE: You may have noticed that functionality that we implemented in this example could be achieved with standard Timers->TON FB.

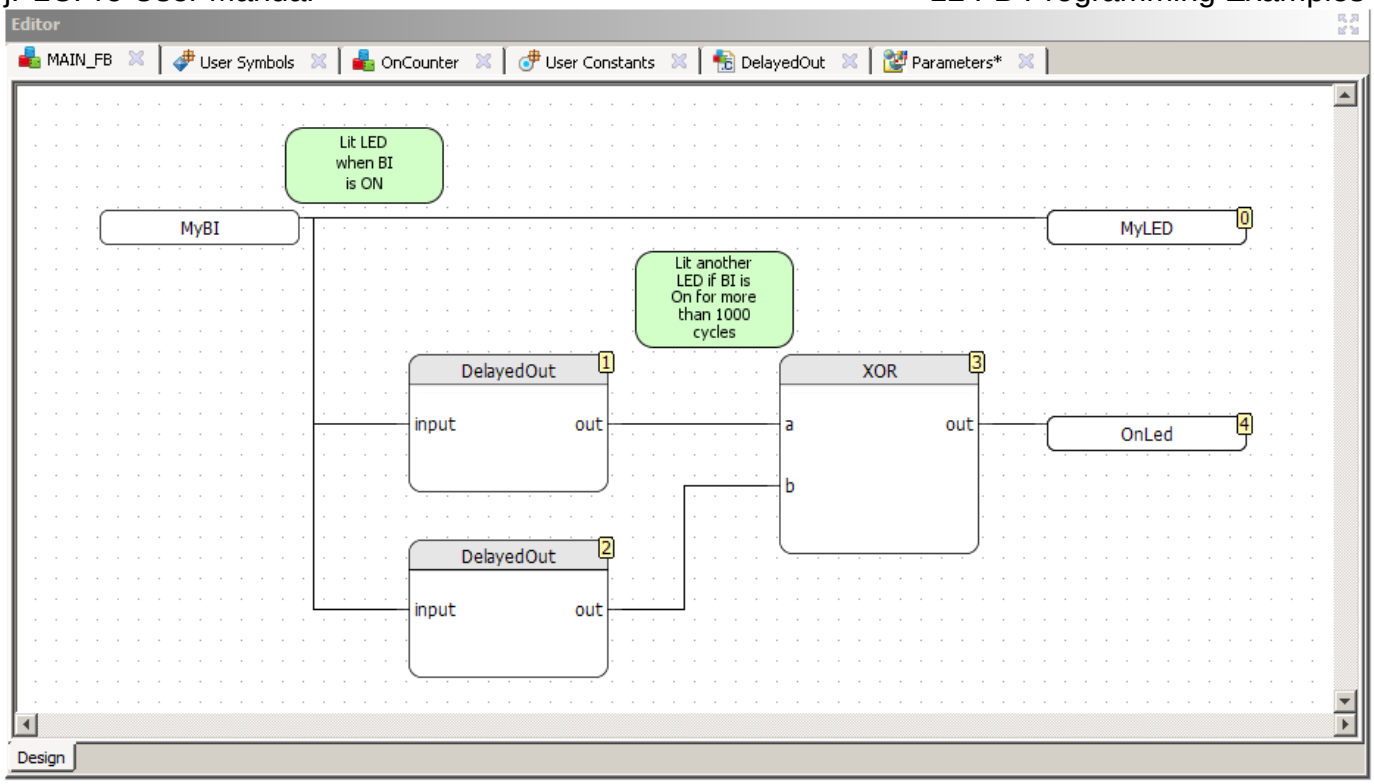
12.3 DelayedOut FB

If you have experience with C programming, you may notice that it would be very easy to combine `OnCounter` and `GREATER` FBs in one simple C function. Let's see how we can do that in jPLCPro.

1. Start with saving our previous example as `MyThirdProject`
2. Now, we can create new FB
   a. In `Project View` right click on `Functional Blocks`, and select `New Module…`
   b. In opened dialog type `DelayedOut`, and change selection Drop Down Box to `FB-C`.
      - Click `OK` button, and new, FB will be crated, with empty structure `__DelayedOut` and empty function `f_DelayedOut`.
3. Next, we create input and output for our new FB:
   a. Select Definition tab
   b. In Inputs list, create one input (with name `input`) of type `BIT`.
   c. In Outputs list, create one output (with name `out`) of type `DWORD`.
4. Let us create one Parameter:
5. In Parameters tab, create parameter `count` of type `DWORD`, initial value `1000`, max value `100000`, min value `0` and increment of `1`.
6. Save Project to publish new values.
7. Go to C function tab of DelayedOut FB.
   a. You can notice that __DelayedOut structure is filled with created variables.
   b. Paste following code in lower part of DelaydOut editor:

```
if (DelayedOut->input) {
     DelayedOut->count++;
     if (DelayedOut->count > DelayedOut->time) {
          DelayedOut->out = 1;
     }
}
else {
     DelayedOut->count = 0;
     DelayedOut->out = 0;
}
```

   c. Save project
8. Replace `OnCounter` and `GREATER` FBs with newly created FB:
   a. Drag a square around `OnCounter` and `GREATER` FBs, and `thousand` input, and press `Del` key on keyboard
      - Selected elements will be deleted
   b. Place new `Custom->DelayedOut` FB instead of deleted elements
   c. Connect `MyBI` with `DelayedOut->input` and `DelayedOut->out` with `OnLed`.
   d. Select `DelayedOut` FB, and in `Properties View` change its priority to `1`
      - Priority of `DelayedOut` will change to `1`, and priority of `OnLed` will change to 2
9. Create another DelayedOut FB, change its priority to 2 and add Logical->XOR element, as infollowing picture.
   a. Save Project



10. Change parameters for DelayedOut_2 FB
    a. In Project View double click Parameters
        - Parameters window will open
    b. Expand Default Group and MAIN_FB
    c. If DelayedOut_1 and DelayedOut_2 Parameter groups are not displayed, right click on Default Group and select Reset.
        - DelayedOut_1 and DelayedOut_2 parameter groups will appear.
    d. Select DelayedOut_2->time parameter's Init value and change it to 5000
11. Compile and download project to Device

12.3.1 Lessons Learned

In this example we learned:

- How to create new FB-C
- How to select group of FBs
- How to delete FBs
- How to crate FB parameters, and change it for different instances

13 Shortcuts Overview