Table of contents

CFC Editor in Online Mode

CFC Editor in Online Mode

In online mode, you can monitor and change variable values of the control. In addition, debugging features are provided such as breakpoints and stepping.

Monitoring

As usual, you can monitor values in the declaration part as well as in the implementation part (with inline monitoring).

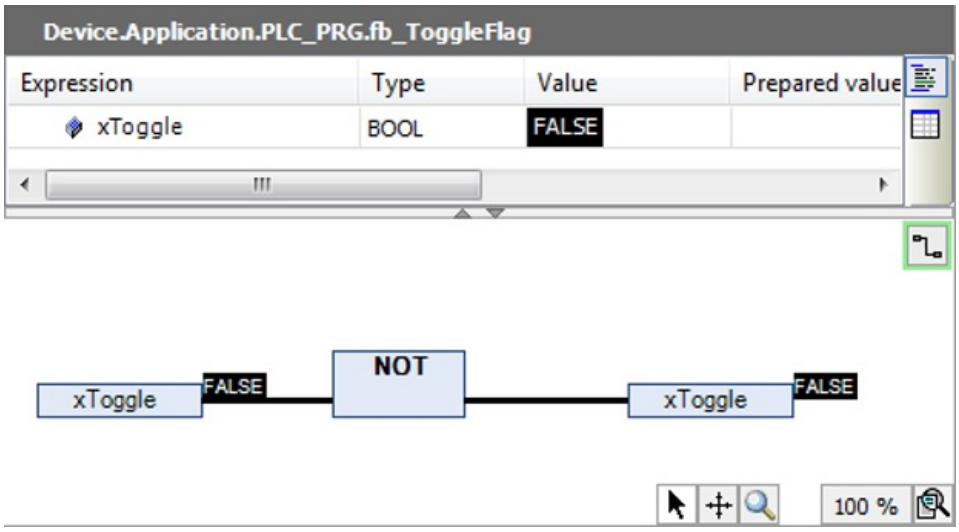
Inline monitoring of a function block is possible only when an instance of the function block is open. No values are displayed in the basic implementation view.

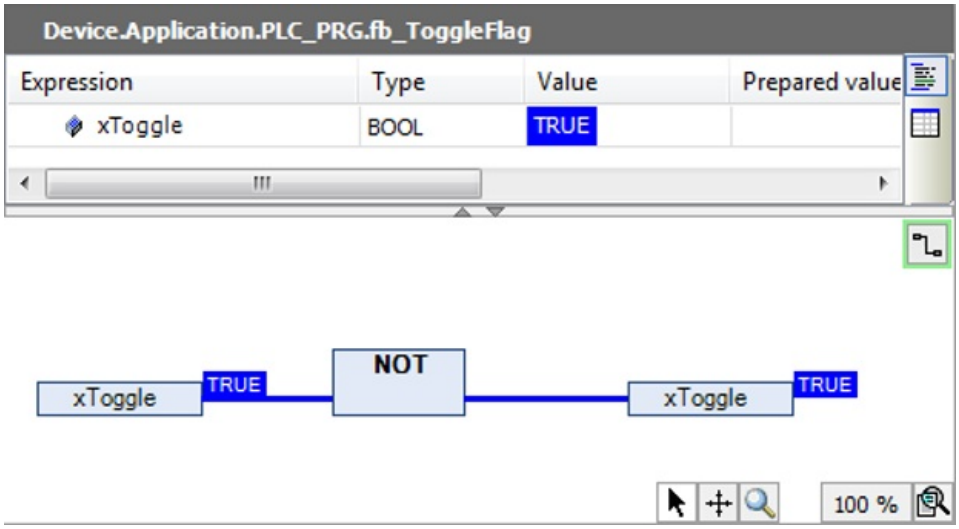
Monitoring a Boolean variable

The connections between Boolean variables are displayed in color according to their actual value: TRUE in blue and FALSE in black. The element pins are decorated with the actual value.

Example

An application contains a CFC POU. An internal Boolean variable is switched there. The `iToggle` variable changes its state from TRUE to FALSE with each bus cycle.

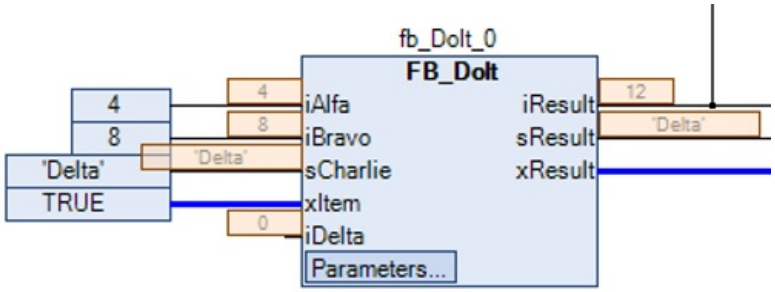




Monitoring a scalar variable

In the case of scalar variables, the element pins are decorated with the actual values.

Example



Forcing and writing of variables

In online mode in the declaration editor, you can prepare a value for forcing or writing a monitored variable.

When you select the “Prepare values in implementation part” check box in the “CFC Editor” category of the PLC Engineering options, you can also prepare values in the implementation part.

To open the “Prepare value” dialog, double-click on an element or on the monitoring box next to an element. No dialog appears for Boolean variables. However, with each mouse click on the value displayed next to the variable, the values TRUE and FALSE are toggled.

Prepared values are displayed in angle brackets. After executing a write or a force, a red "F" is shown in the monitoring box.

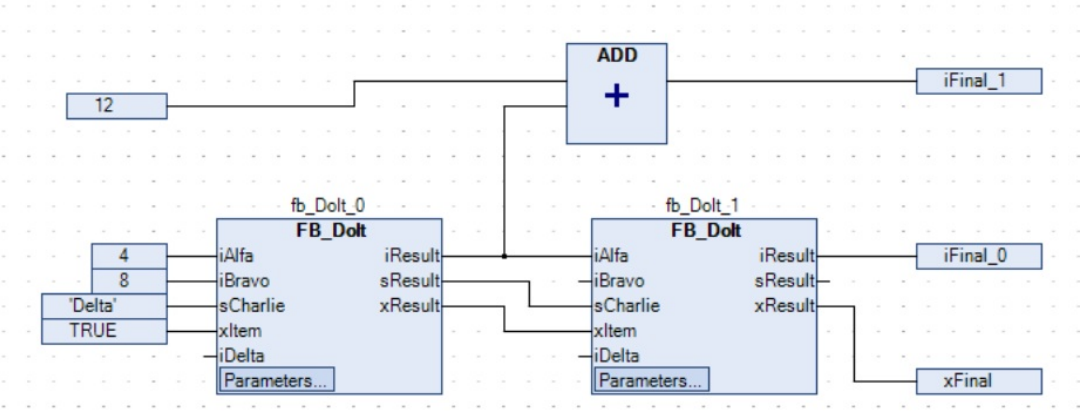
Changing of constant input parameters of function block instances

You can write input parameters of function block instances of type `VAR_INPUT CONSTANT` in online mode and modify the parameters in this way. To save these parameters after logout, click on “Save prepared parameters to project”.

Prerequisite: A CFC editor is active. An instantiated function block has `VAR_INPUT CONSTANT` variables in its declaration.

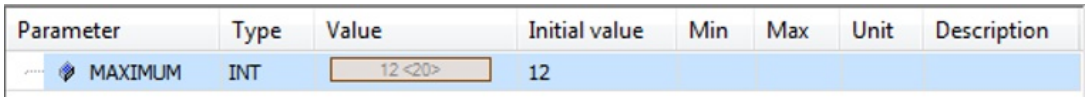
1. In the editor, open the POU by calling the function block instance.
   ⇒ The declaration of `FB_DOIt` has been supplemented by the constant `MAXIMUM`.

```
FUNCTION_BLOCK FB_DoIt
VAR_INPUT
    iAlfa : INT;
    iBravo: INT;
    sCharlie : STRING := 'Charlie';
    xItem : BOOL;
    iDelta : INT;
END_VAR
VAR_INPUT CONSTANT
    MAXIMUM : INT := 12;
END_VAR
VAR_OUTPUT
    iResult : INT;
    sResult : STRING;
    xResult : BOOL;
END_VAR
```

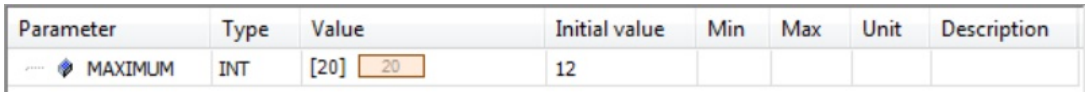


The graphical representation of the function block instances contains the “Parameters” button.

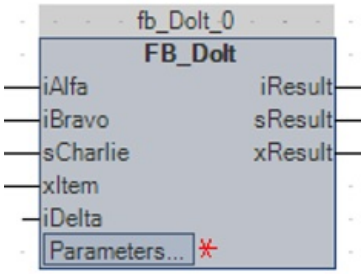
2. Log in to the controller.
3. Click the “Parameters” button of the function block instance.
   ⇒ The “Edit Parameters” dialog opens.
4. Click the “Value” column in an inline monitoring field of a parameter.
   ⇒ The “Prepare Value” dialog opens.
5. Type `20` in the “Prepare a new value for the next write or force operation” field.
6. Click “OK” to confirm the entry.
   ⇒ The prepared value is shown in angle brackets next to the current value (e.g. `<20>`).



7. Click Debug ➔ Write Values.
   ⇒ The prepared value is written. The parameter is changed and is displayed in the project in brackets after the value.



The difference between both values is shown by a red cross next to the parameter field of the function block instance.



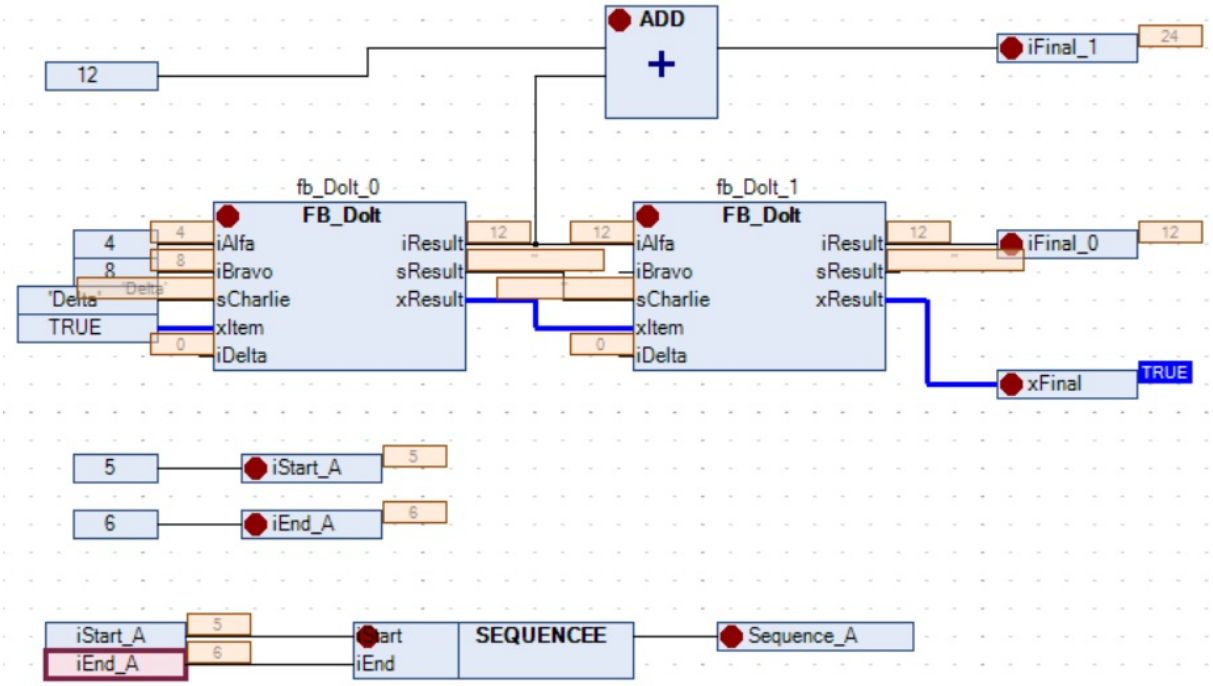
8. Click on “Edit parameters” to close the dialog. Logout.
9. Click CFC ➔ Save Prepared Parameters to Project.
   ⇒ The changed parameter values are applied to the project. The asterisk next to the parameter field disappears.

Breakpoint locations

Possible position of a breakpoint

- Element “Output”
  Variables are described.
- Element “Box”
  POUs are called.
- Element “RETURN”
  The program flow branches.
- Element “Selector”
  Structure elements are described.

Click Debug ➔ Toggle Breakpoint to set a new breakpoint or delete an existing breakpoint. A red circle in the block diagram represents an active breakpoint.



NOTICE!

A breakpoint is automatically set in all methods that can be called.

Thus, the following applies: If a method is called that is defined via an interface, breakpoints are set in all methods of function blocks that implement this interface. This also applies to all derived function blocks defining the method.

Stepping into a POU

You can process a POU in steps in debug mode. A called POU is supplemented internally by a `RETURN` at the beginning before the element with the number `0` and at the end after the last element. In case of step-by-step processing, they are automatically started.

Commands in online mode

Also refer to

- ↘ “Command 'Force Function Block Input'”
- ↘ “Command 'Prepare Box for Forcing'”
- ↘ “Command 'Edit Parameters'”
- ↘ “Command 'Save Prepared Parameters to Project'”

Also refer to

- ↘ “Forcing and writing of variables”
- ↘ “Forcing a function block input in CFC”
- ↘ “Using Breakpoints”
- ↘ “Stepping Through a Program”