Tutorial

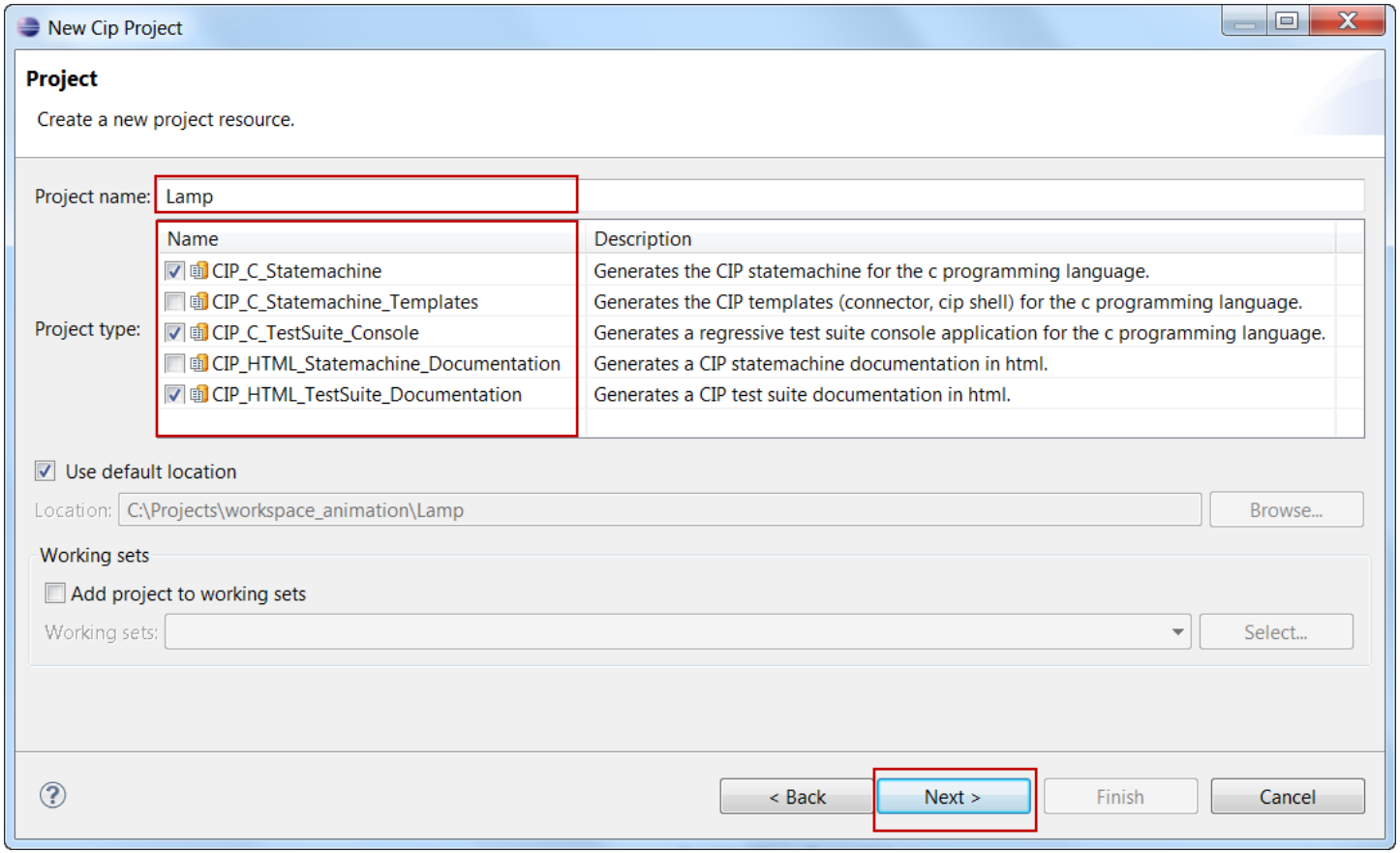
CIP Test Suite - Console

Tutorial	Actifsource Tutorial – CIP Test Suite - Console
Required Time	• 30 Minutes
Prerequisites	• Actifsource Tutorial – Installing Actifsource • Actifsource Tutorial – Simple Service • Actifsource Tutorial – CIP Statemachine - Lamp
Goal	• Creating Unit Tests for a CIP Statemachine
Topics covered	• Setting up the code options • Setting up a CIP Test Suite • Generate the CIP Test Suite • Generate the CIP Test Suite Documentation
Notation	↳ To do ⓘ Information • **Bold**: Terms from actifsource or other technologies and tools • **Bold underlined**: actifsource Resources • Underlined: User Resources • *UnderlinedItalics*: Resource Functions • `Monospaced`: User input • *Italics*: Important terms in current situation
Disclaimer	The authors do not accept any liability arising out of the application or use of any information or equipment described herein. The information contained within this document is by its very nature incomplete. Therefore the authors accept no responsibility for the precise accuracy of the documentation contained herein. It should be used rather as a guide and starting point.
Contact	**actifsource GmbH** Täfernstrasse 37 5405 Baden-Dättwil Switzerland www.actifsource.com
Trademark	**actifsource** is a registered trademark of **actifsource GmbH** in Switzerland, the EU, USA, and China. Other names appearing on the site may be trademarks of their respective owners.

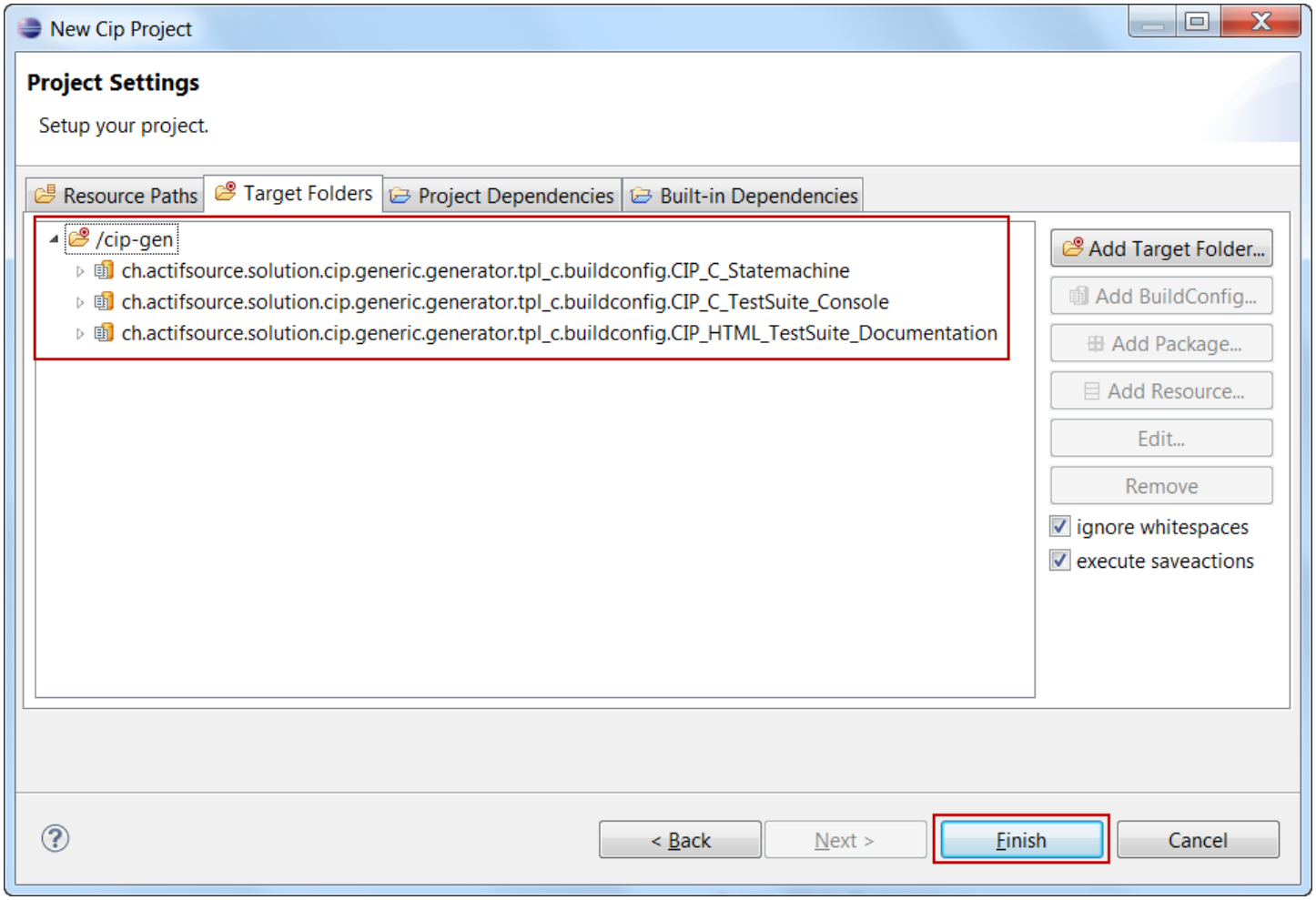
- Learn how to specify a CIPTestSuite model
  - Specify message events (input)
  - Specify message action expectations (output)
- Run the unit tests

Part I: 4
Setting up the CIP project

- Make sure to properly setup the lamp project as seen in the previous tutorial



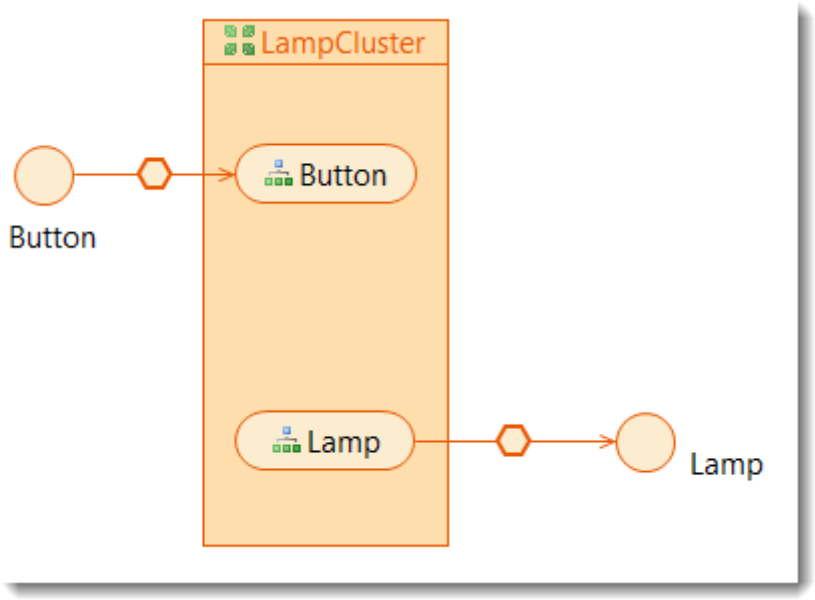
- Create a new **CIP Project** using the wizard
- Select **CIP_C_Statemachine** to generate a CIP state maschine
- Select **CIP_C_TestSuite_Console** to generate a console test suite
- Select **CIP_HTML_TestSuite_Documentation** to generate a html documentation
- Click **Next>**



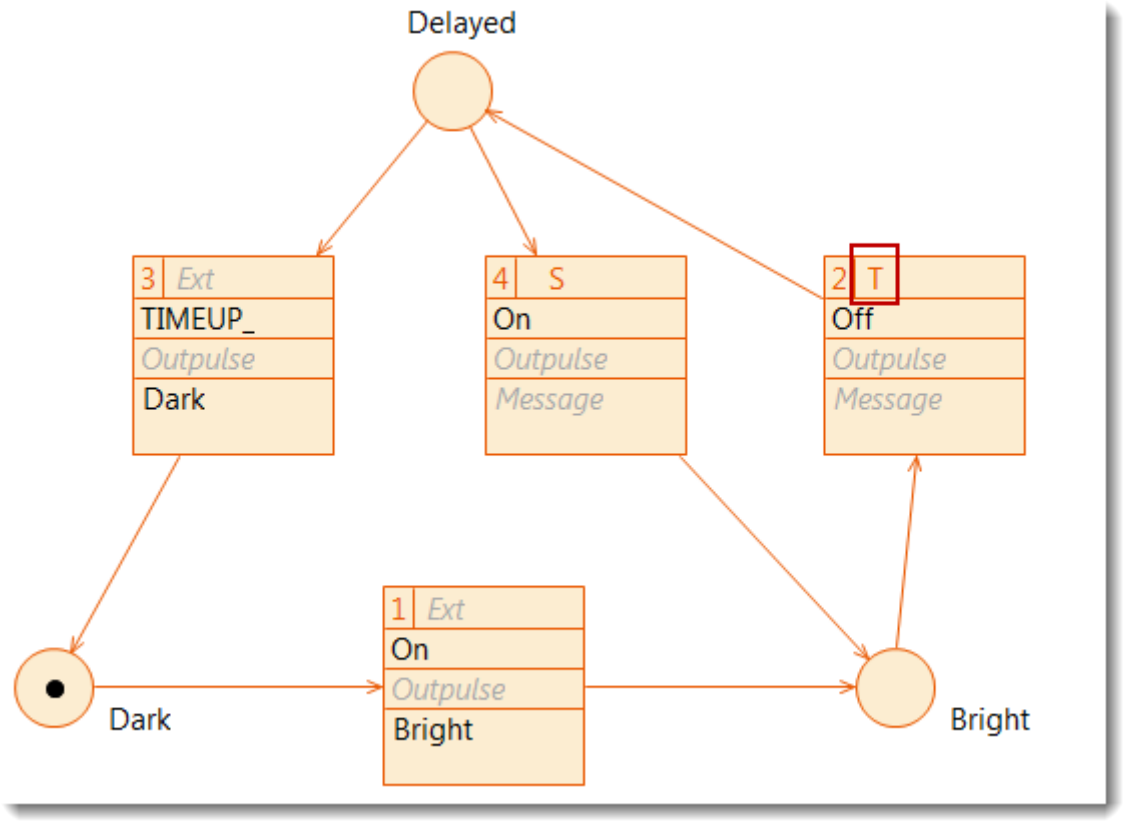
ⓘ The necessary build configs are automatically added to your target folder

ⓘ Note that you can add or remove build configs anytime to/from your target folders

↳ Click **Finish**

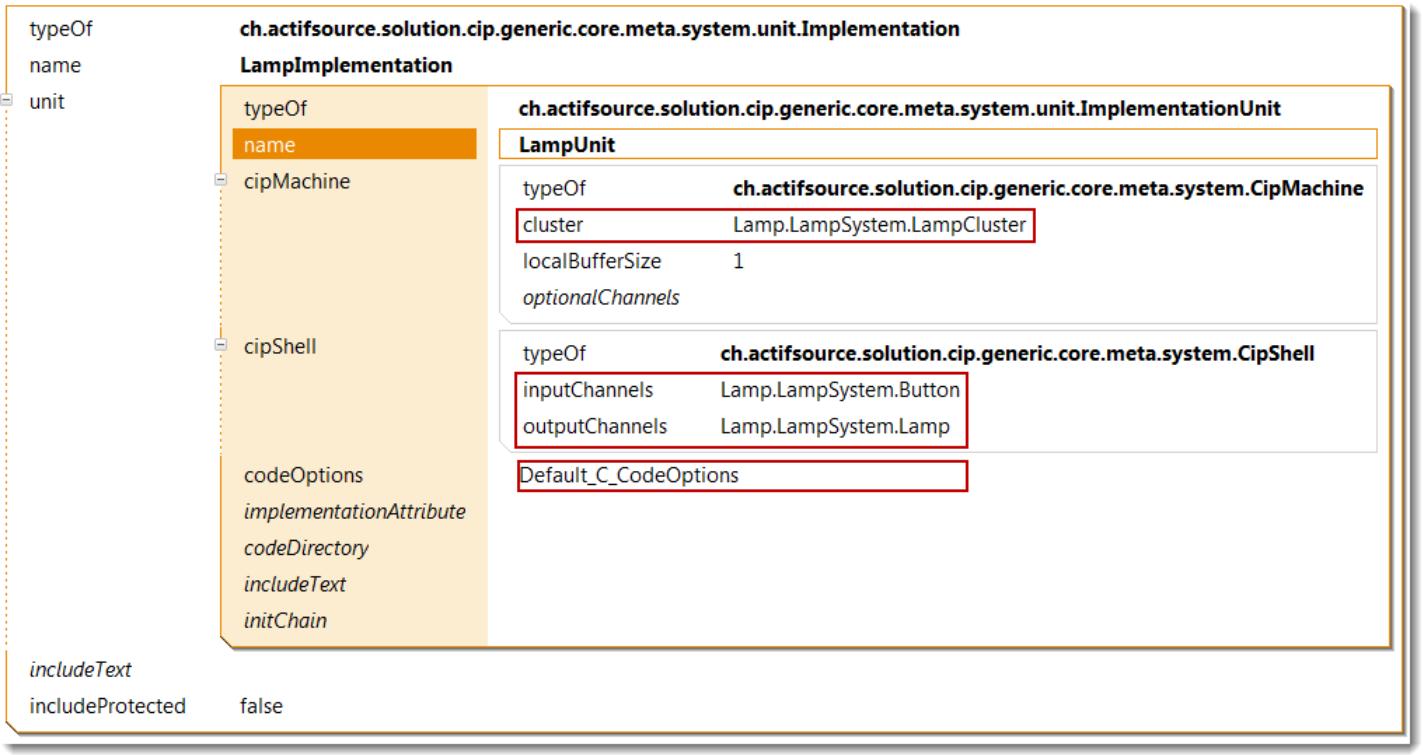


- Create the Lamp state machine as shown in the previous tutorial

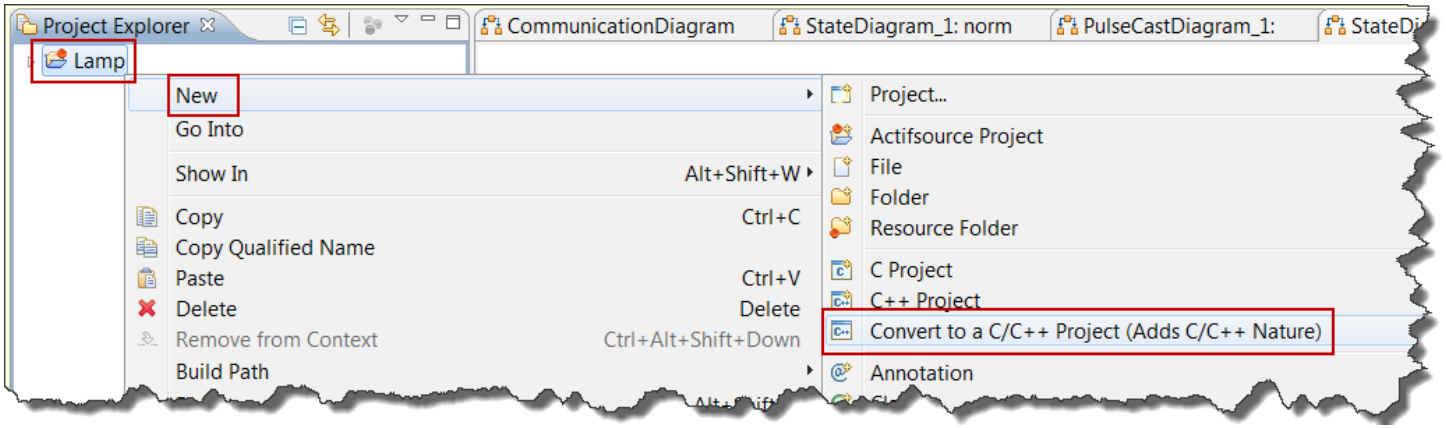


typeOf	**ch.actifsource.solution.cip.generic.core.meta.process.DelayOperation**
name	**delay**
delayText	3

↳ Set the timer delay to 3 ticks

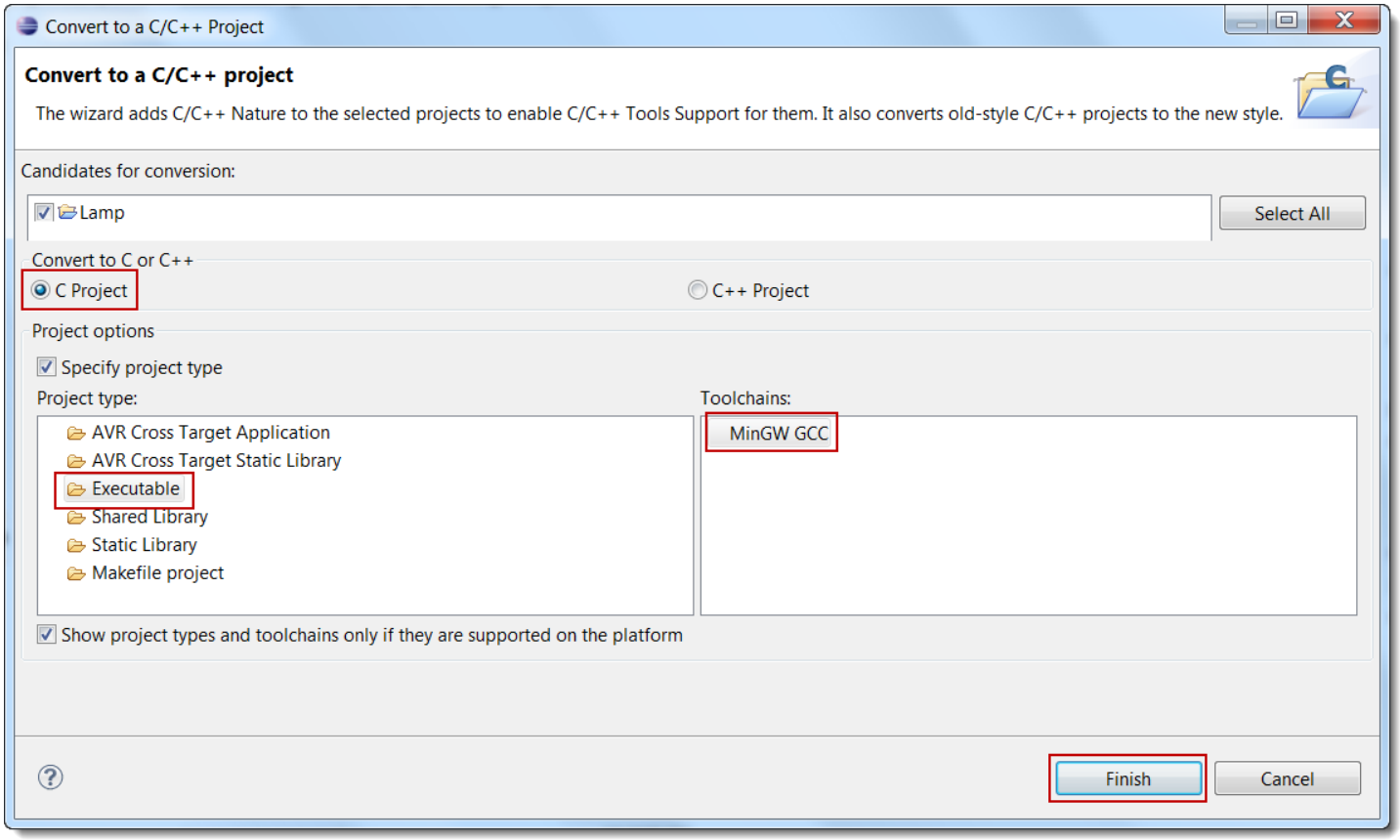


- Create a new **Implementation** in your **CIPSystem**
- Select the LampCluster in your **CipMachine**
- Select input and output channels
- Select Default_C_CodeOptions

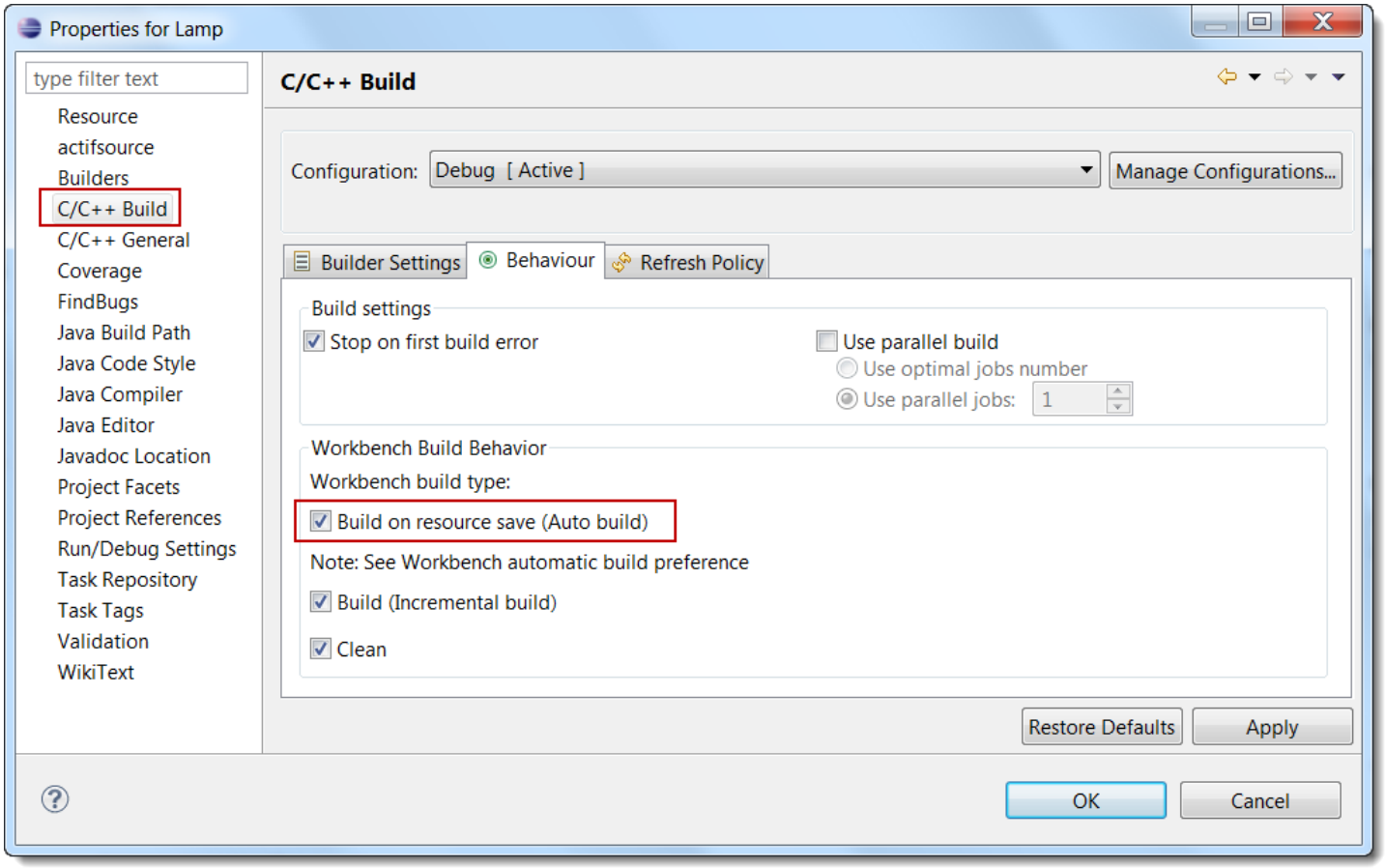


ⓘ Install the CDT (C/C++ development tool)

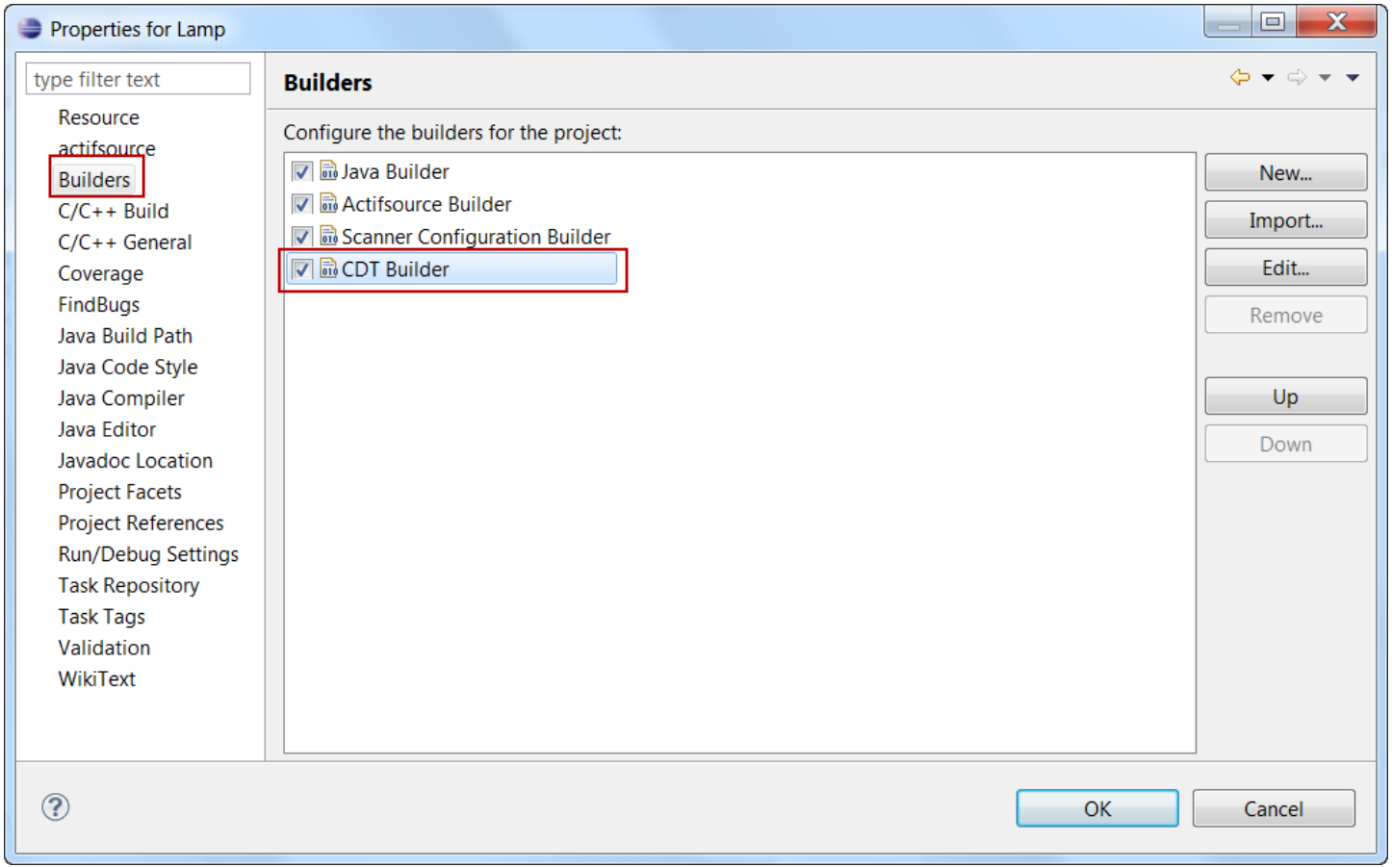
↳ Convert your project to a **C/C++ Project**



- Convert to **C Project**
- Create an **Executable**
- Click **Finish**



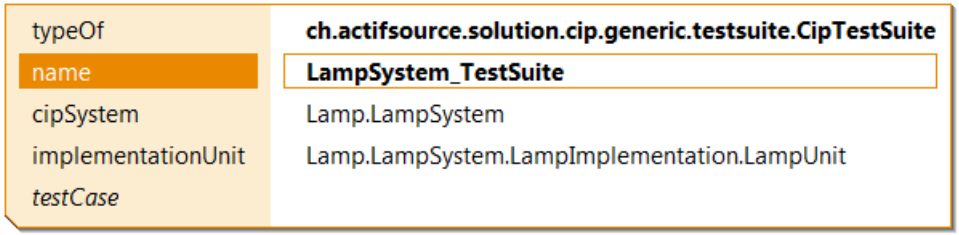
↳ To automatically compile choose *Build on resource save* in the project properties



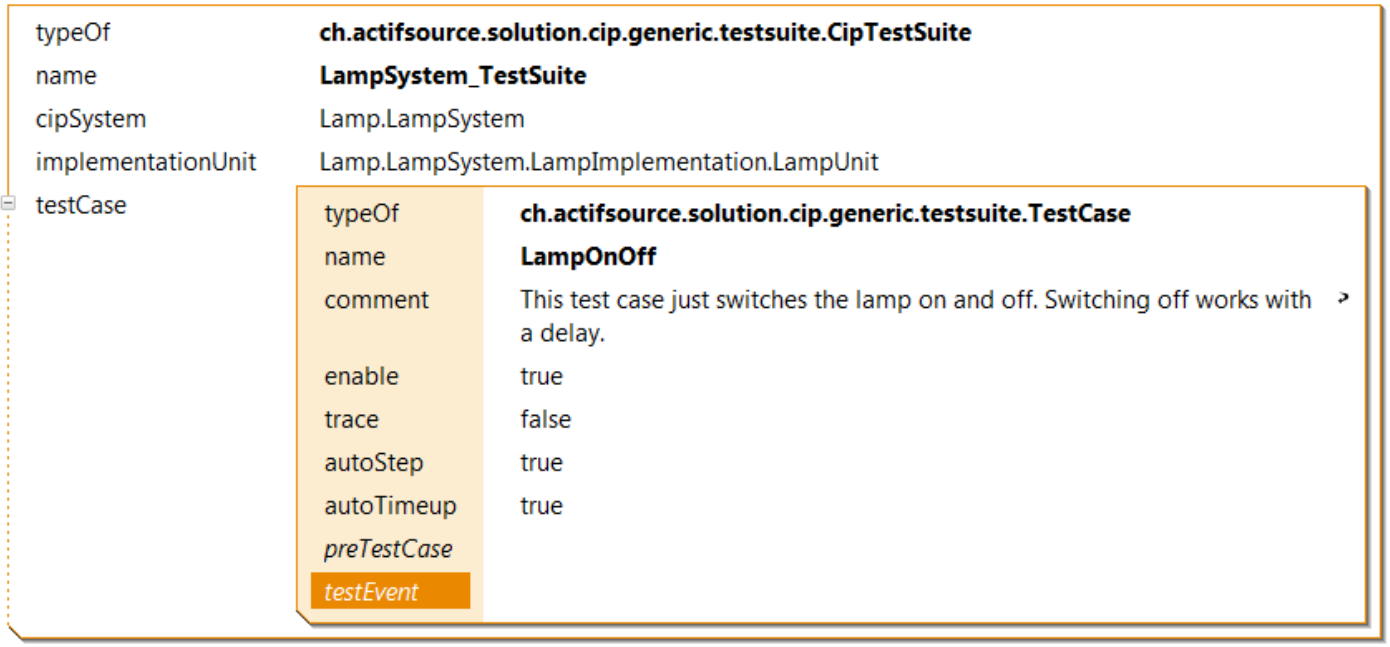
↳ Rearrange the CDT Builder to the last position to make sure that the actifsource generator runs before compiling

Part II: Setting up a CIP Test Suite

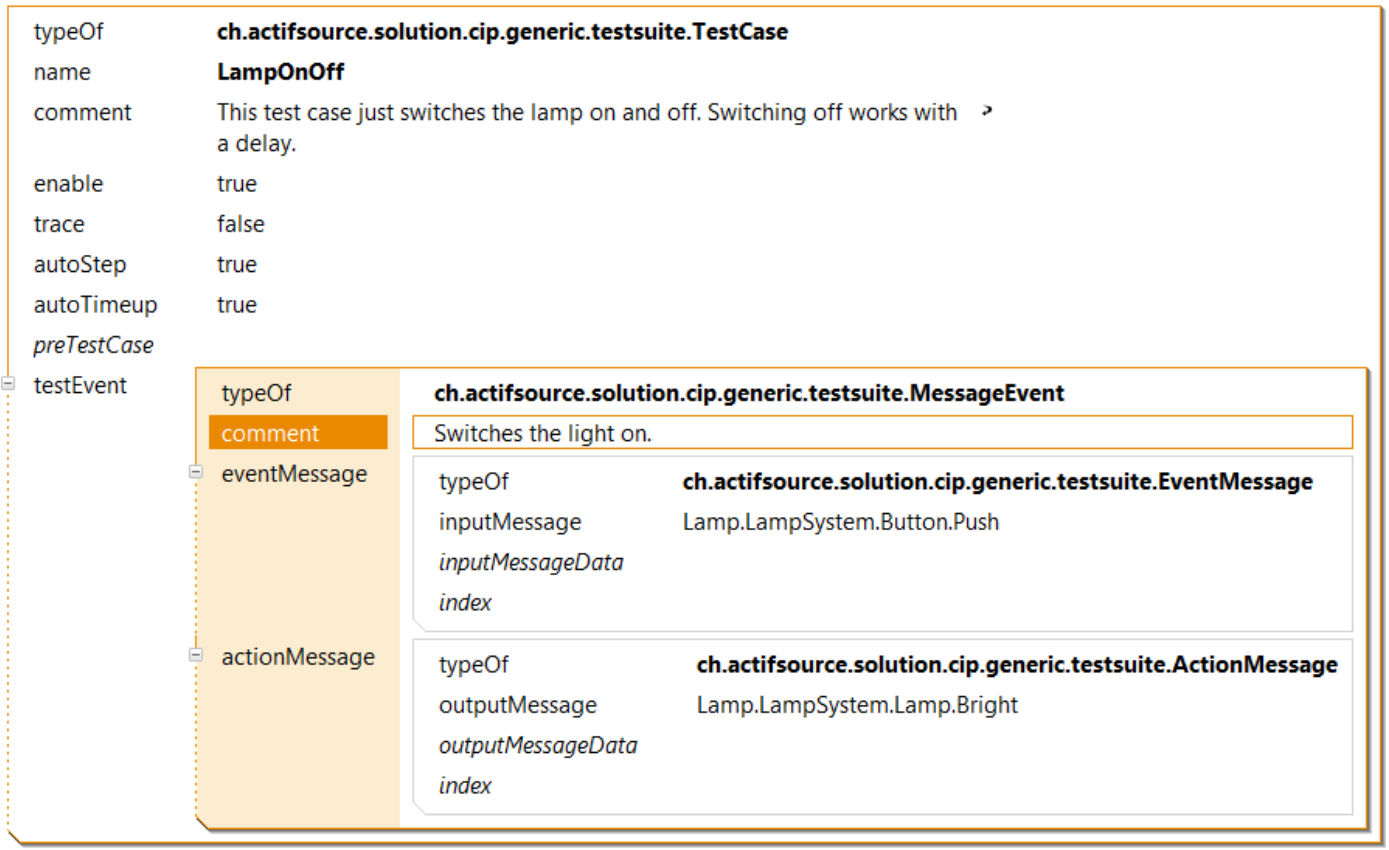
- Create a new **Resource** of type **CipTestSuite**
- Specify your test cases



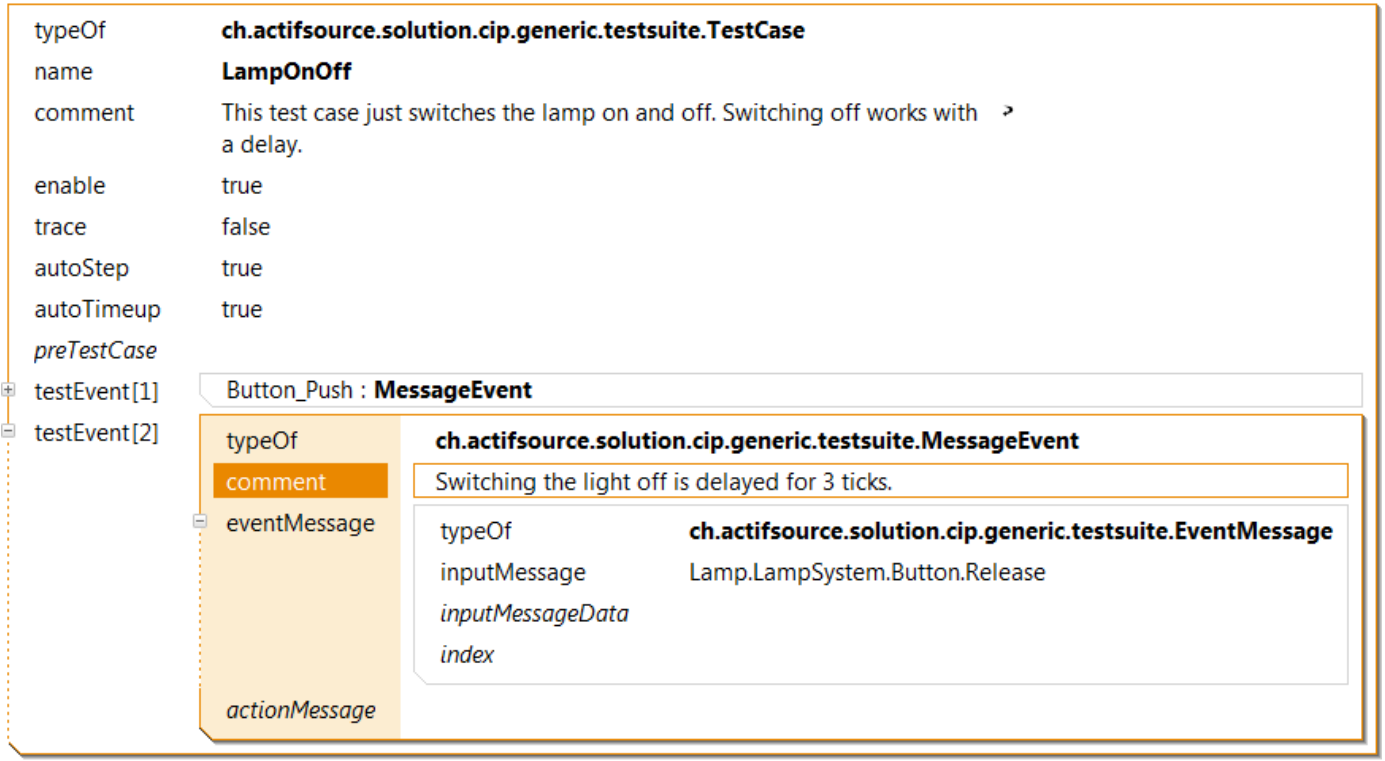
- Create a new **Resource** of type **CipTestSuite**
- Reference you CIP system and the implementation unit which shall be tested



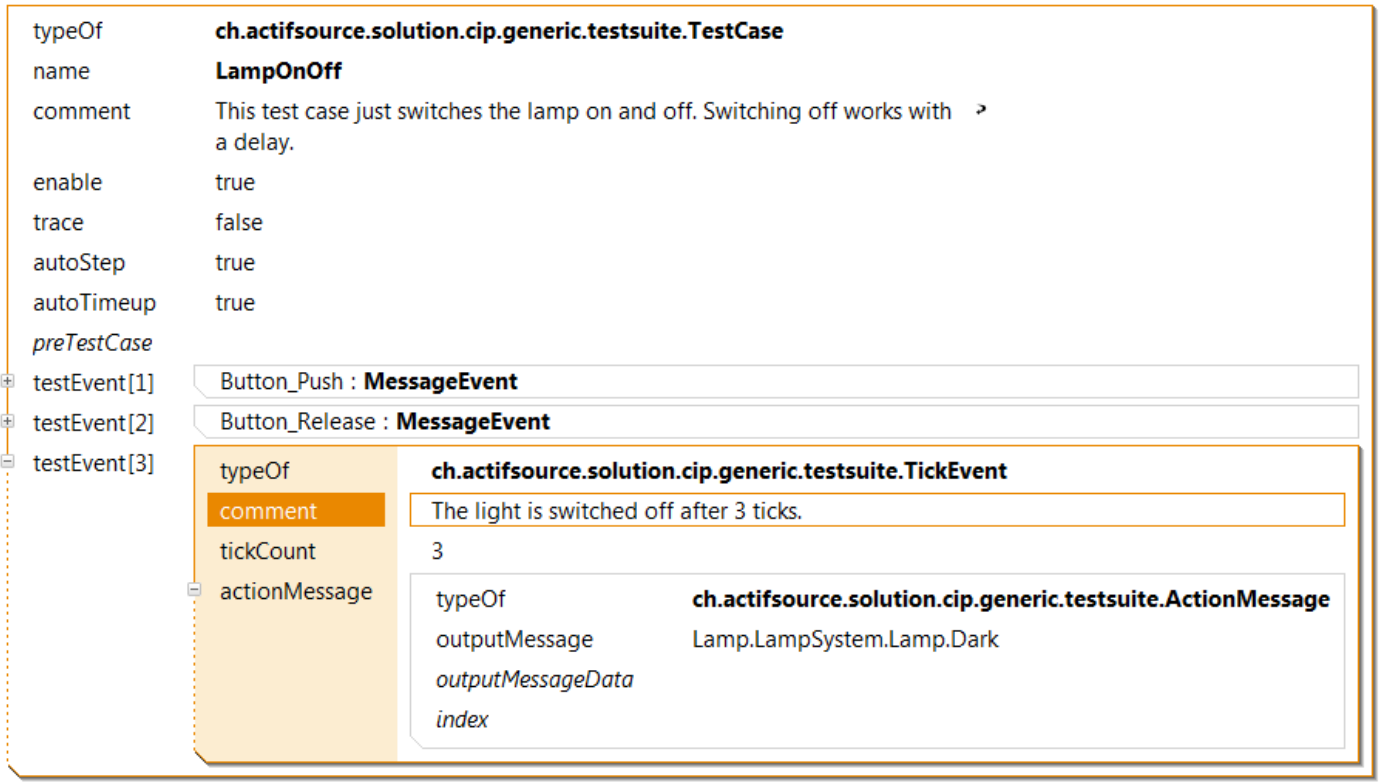
- Create a new **TestCase** called LampOnOff
- Make sure to properly comment all resources via the **comment attribute**.



- Define the first test event of type **MessageEvent**
- First define the event message which is sent to the CIP machine
- Let's switch on the light by pressing the button
  - Select event message Button.Push
  - Define the expected action message Lamp.Bright



- Define the second test event of type **MessageEvent**
- First define the event message which is sent to the CIP machine
- Let's activate the switch off delay by pressing the button
  - Select event message Button.Release
- ⓘ Since switching off is delayed, we do not expect the light to be switched off

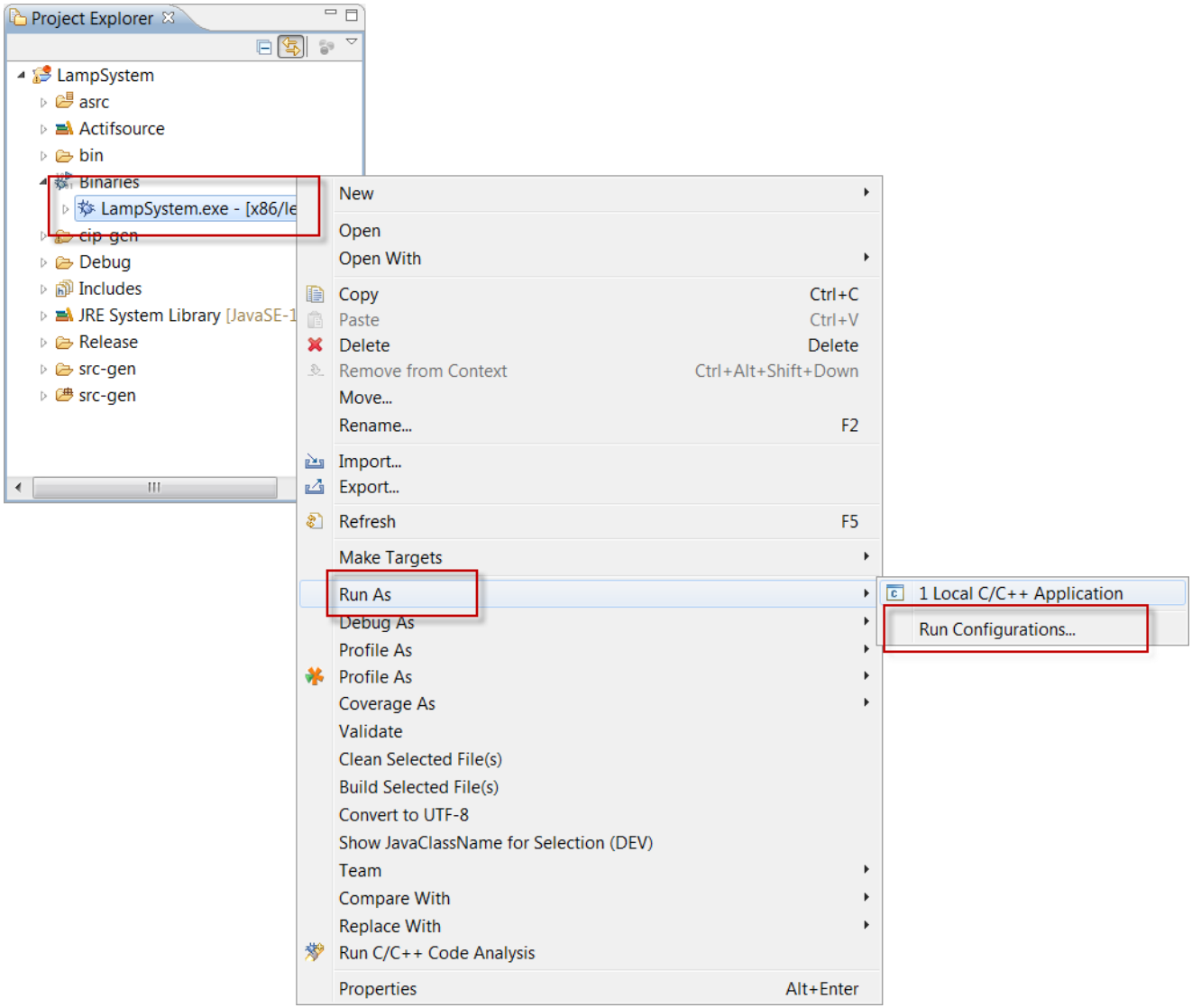


- Define the third test event of type **TickEvent**
- Define the number of ticks to time up the delay
  - 3 ticks in this example
  - Define the expected action message Lamp.Dark

ⓘ The light is expected to be switched off after 3 ticks

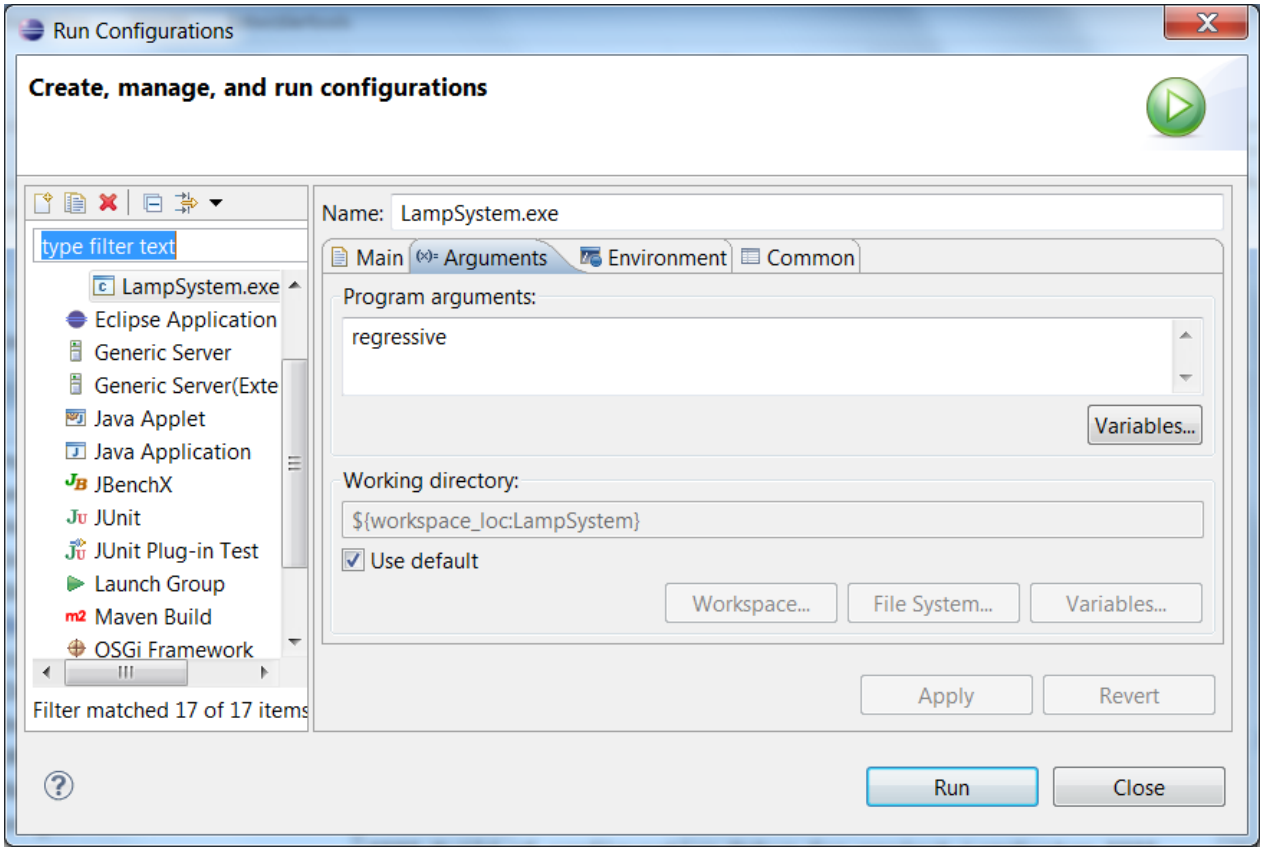
Part III: Running the CIP Test Suite

- In the second part we want to generate an executable which can run the previously defined test cases
- Make sure to install the CDT (C/C++ Development Tool) and mingw for C language support

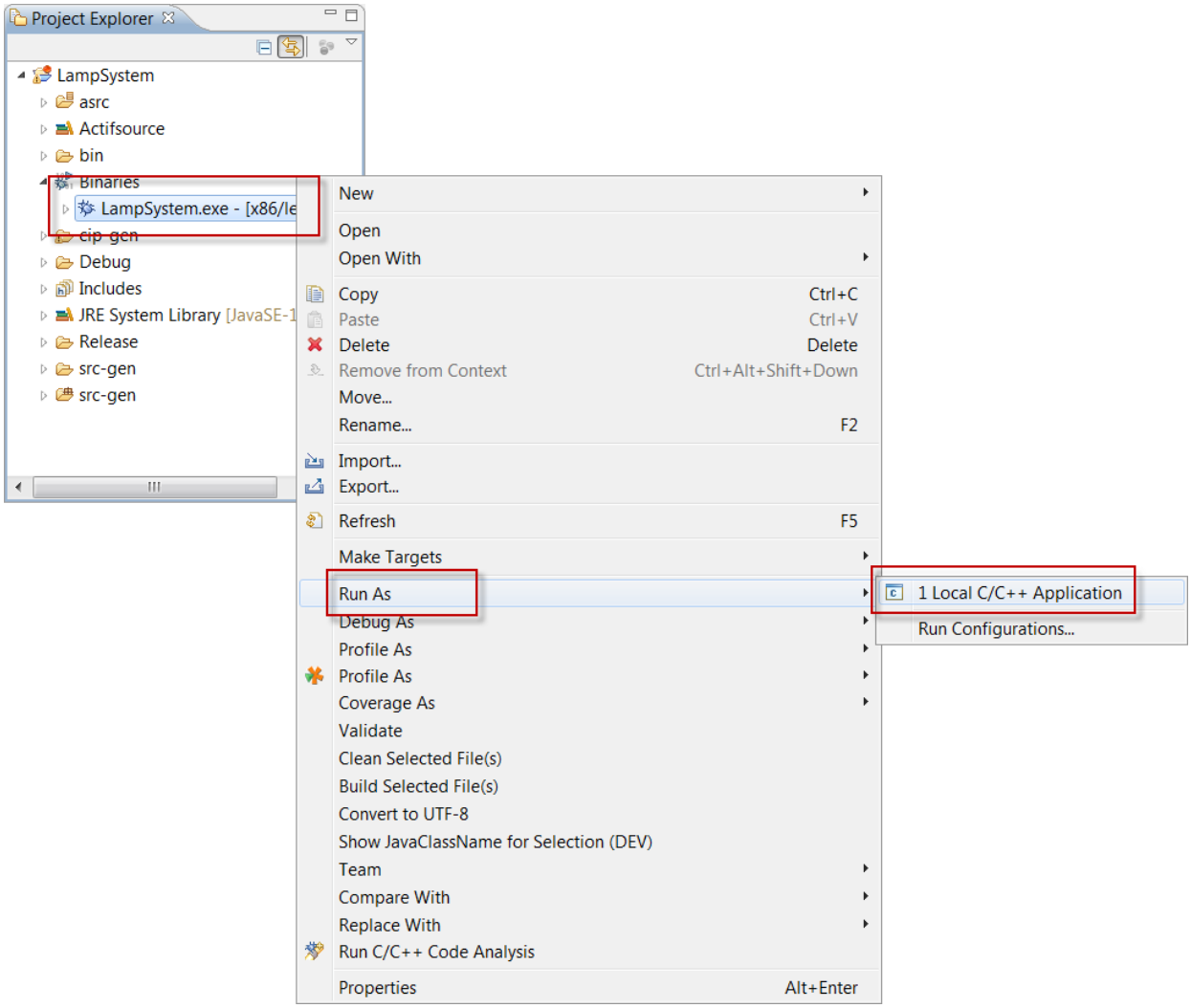


ⓘ Your lamp system should have been generated by now

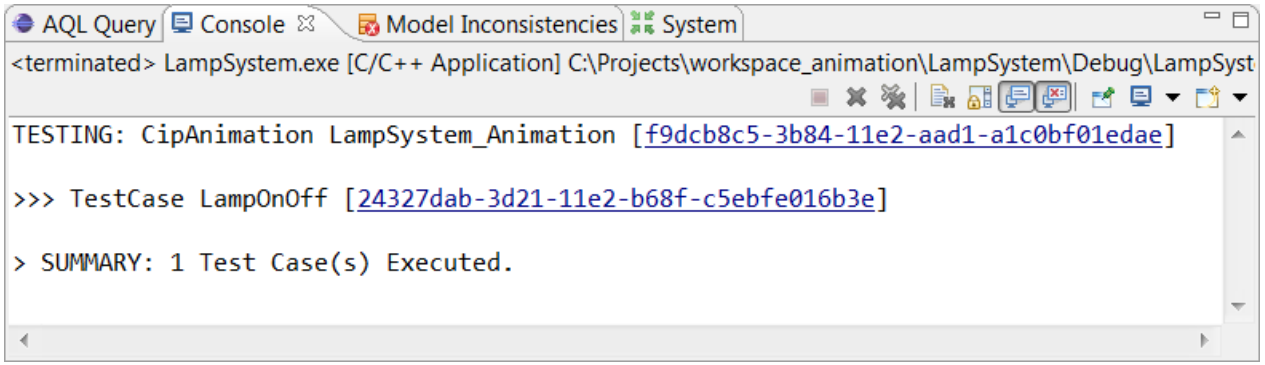
↳ Select your executable and run configuration



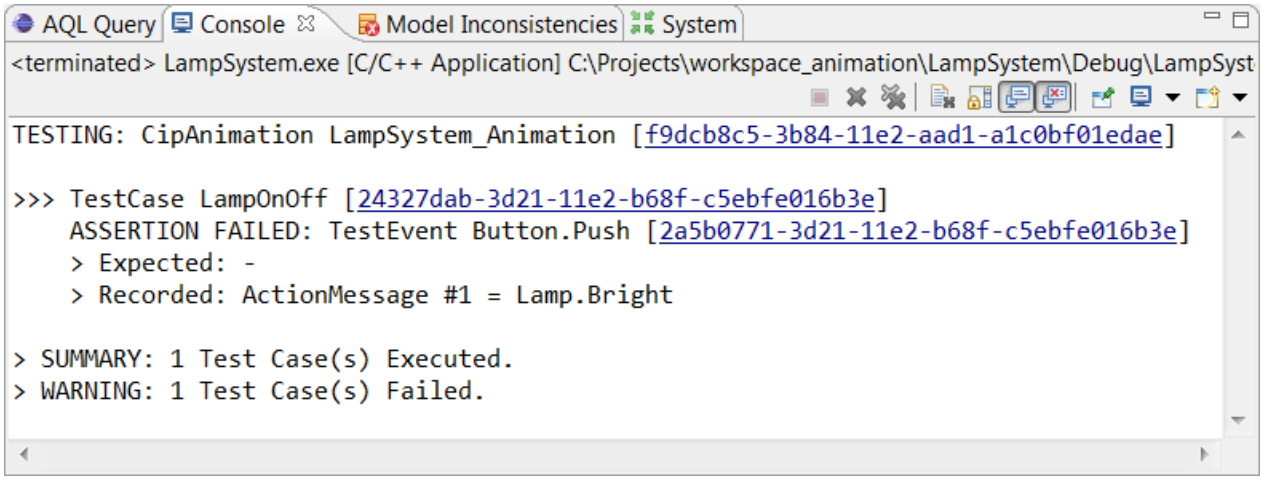
- Select the command line options
  - Select `regressive` to automatically run test cases
  - Select `manual` to manually stimulate the event



↳ Run the Local C/C++ Application



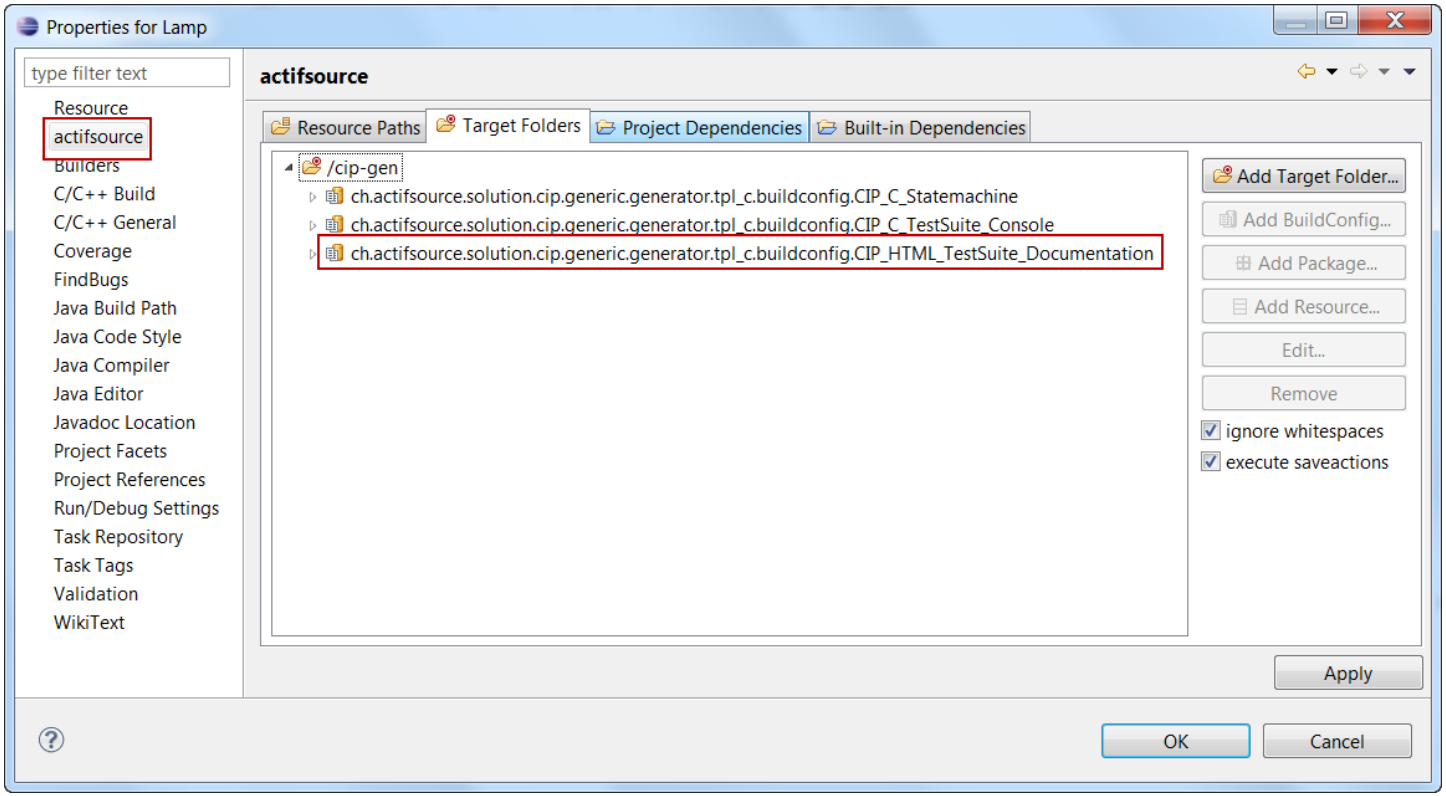
- The test out can be found in the console
- Note that you can directly link back from the console output in the corresponding model by clicking on the blue GUID links



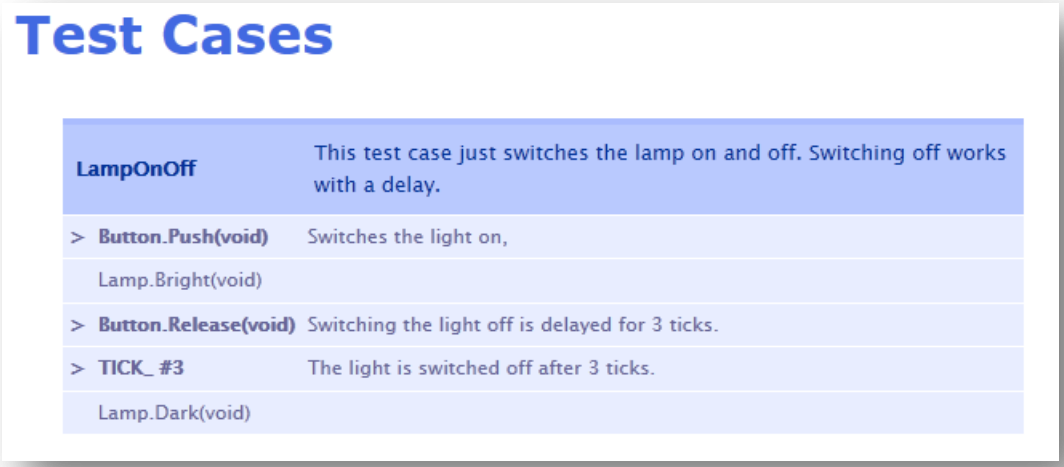
- Delete the expectation Lamp.Bright in your first test event and run the test again
- Link back to the model by clicking the GUID link next to the assertion failure
- Correct your test model

Part IX: 26
Generate the CIP Test Suite Documentation

- Last but not least we want to generate the test suite documentation
- The documentation can be used for communication and to ensure quality
- Make sure to define a new test case any time a problem in your state machine hast been corrected



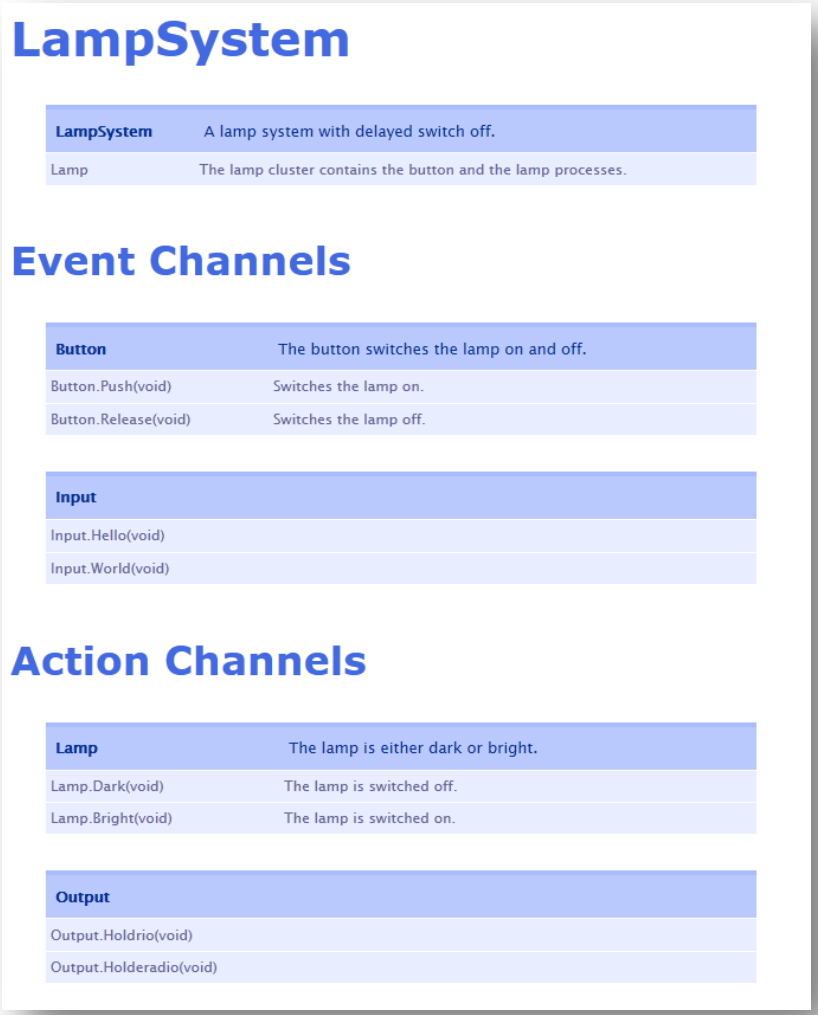
- The documentation is generated because of the **BuildConfig CIP_HTML_TestSuite_Documentation**
- A file `LampSystem_TestSuite_Documentation.html` is generated in your target folder



Test Cases

LampOnOff		This test case just switches the lamp on and off. Switching off works with a delay.
>	**Button.Push(void)**	Switches the light on,
	Lamp.Bright(void)	
>	**Button.Release(void)**	Switching the light off is delayed for 3 ticks.
>	**TICK_ #3**	The light is switched off after 3 ticks.
	Lamp.Dark(void)	

ⓘ Use the generated documentation to ensure quality and communicate with your team mates.



LampSystem

LampSystem	A lamp system with delayed switch off.
Lamp	The lamp cluster contains the button and the lamp processes.

Event Channels

Button	The button switches the lamp on and off.
Button.Push(void)	Switches the lamp on.
Button.Release(void)	Switches the lamp off.

Input
Input.Hello(void)
Input.World(void)

Action Channels

Lamp	The lamp is either dark or bright.
Lamp.Dark(void)	The lamp is switched off.
Lamp.Bright(void)	The lamp is switched on.

Output
Output.Holdrio(void)
Output.Holderadio(void)

ⓘ Klick on messages in the test cases to link back to the system information