

Programmierung

Zur Übersetzung eigener Funktionsblöcke und eigener Geräte ist das Robotino® View API erforderlich. Sie finden alle Version von Robotino® View und die dazu gehörigen API-Installer unter

<http://doc.openrobotino.org/download/RobotinoView/>

Weitere Hilfe erhalten Sie unter <http://forum.openrobotino.org>

Eigene Funktionsblöcke

Die hier beschriebenen Beispiele finden Sie unter:

%USERPROFILE%\Festo\RobotinoView3\units\MyFunctionBlocks.

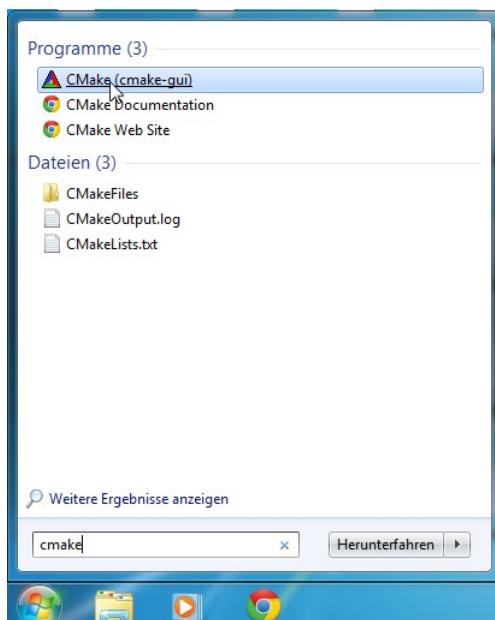
Die Umgebungsvariable %USERPROFILE% speichert den Pfad zum Profil des angemeldeten Benutzers. Die Daten befinden sich dort nach dem ersten Start von Robotino View.

Vorbereiten der Entwicklungsumgebung

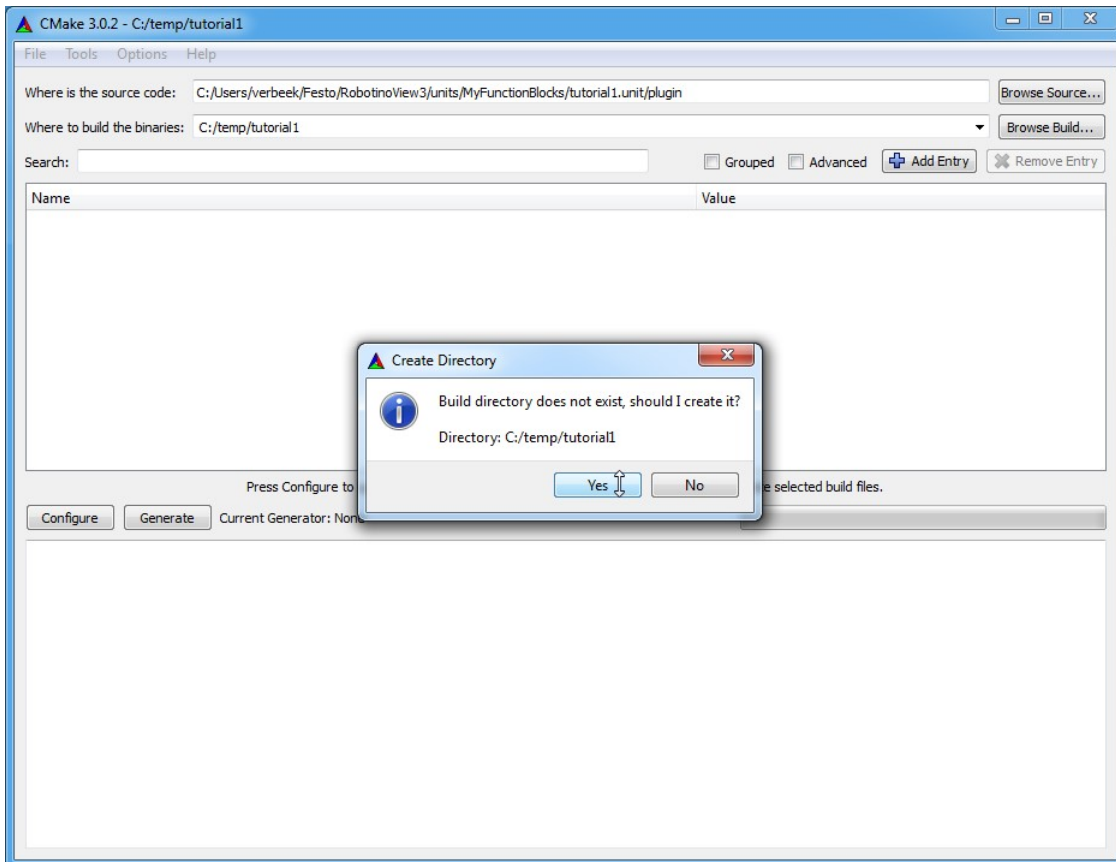
Zum Übersetzen eines Funktionsblocks kann jede neuere Version von Microsoft Visual C++ verwendet werden. Auch der Einsatz der kostenlosen Express-Variante ist ohne Probleme

Um einen Funktionsblock übersetzen zu können, muss zunächst ein Visual C++ Projekt angelegt werden. Dazu ist im Robotino® View API das Tool **CMake** enthalten, das aus zuvor a Skripten die Projektdateien erstellt. Die beiden mitgelieferten Beispiele (Übung 1 und Übung 2) enthalten bereits solche Skripte.

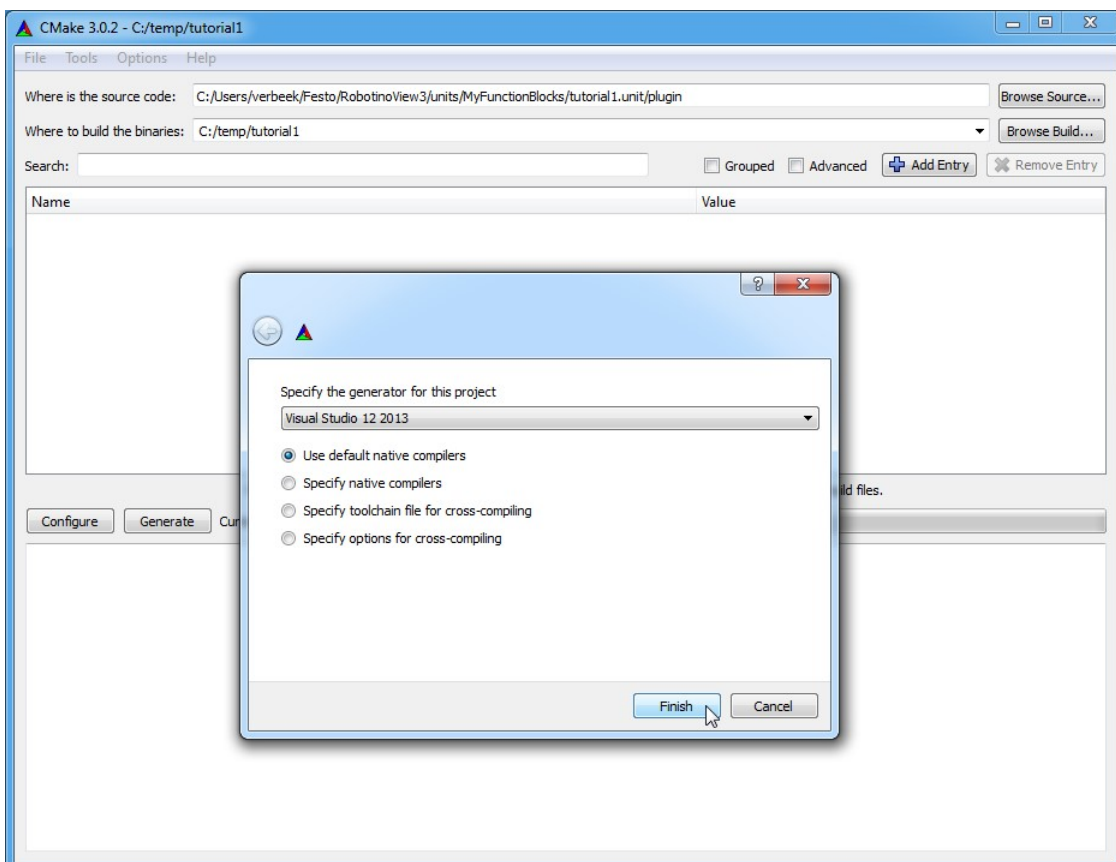
Zunächst wird CMake gestartet.



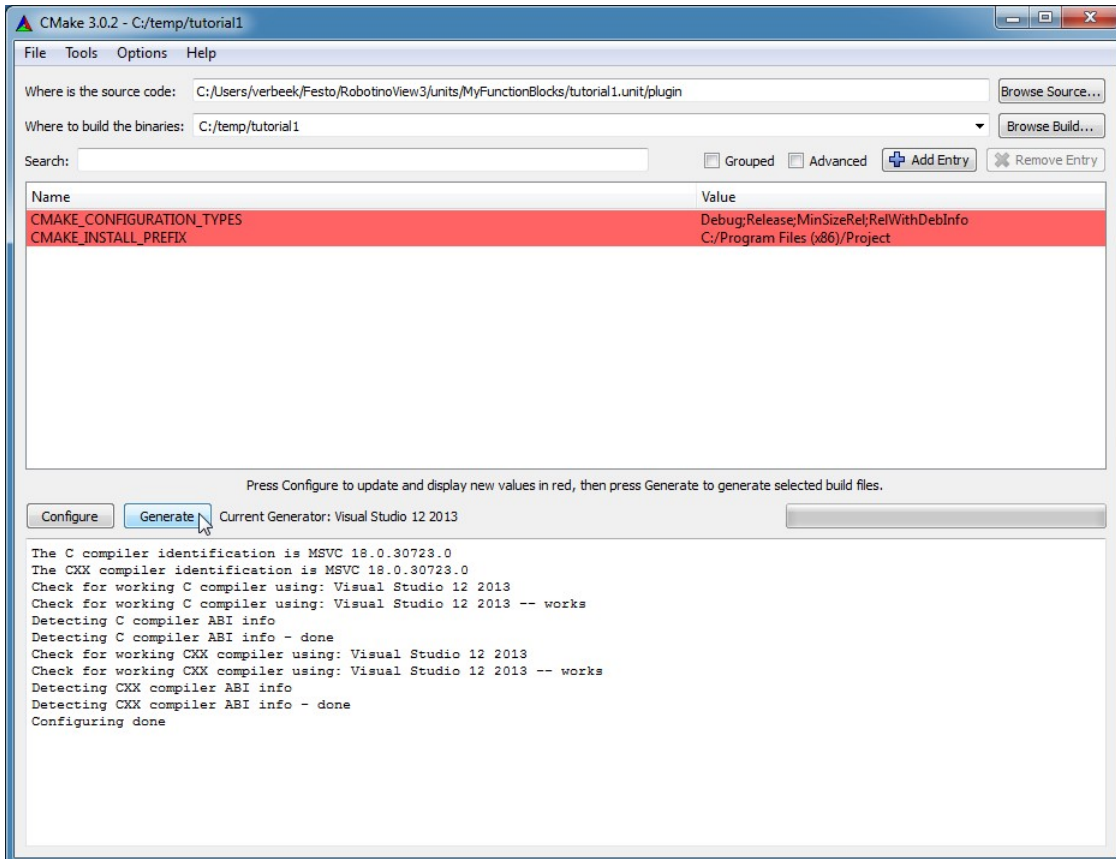
In CMake müssen anschließend der Pfad zu den Quelldateien des Funktionsblocks (im Beispiel Tutorial1 des Benutzers "verbeek") sowie der Pfad zu einem (möglichst leeren) Arbeits (im Beispiel "c:\temp\tutorial1") gesetzt werden. Durch anklicken von "Configure" wird der Erstellungsprozess gestartet.



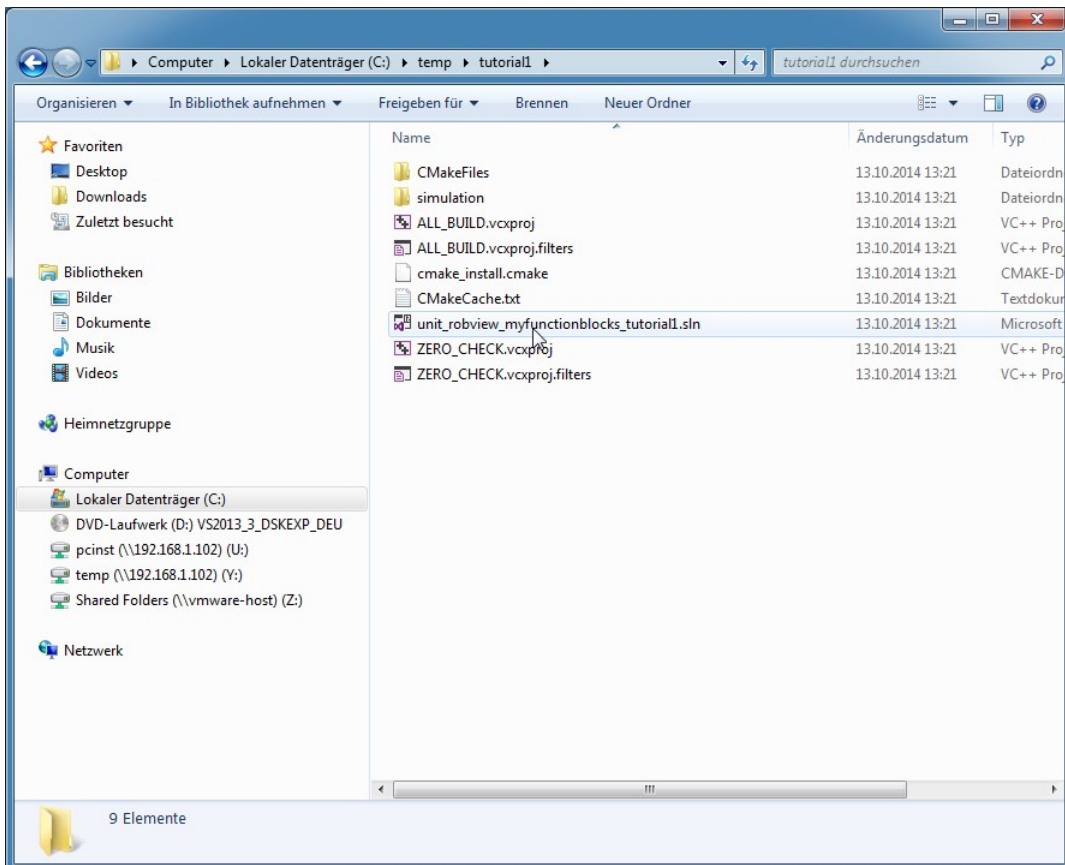
Wählen Sie nun die gewünschte Entwicklungsumgebung. Hier wird Visual Studio 2013 ausgewählt.



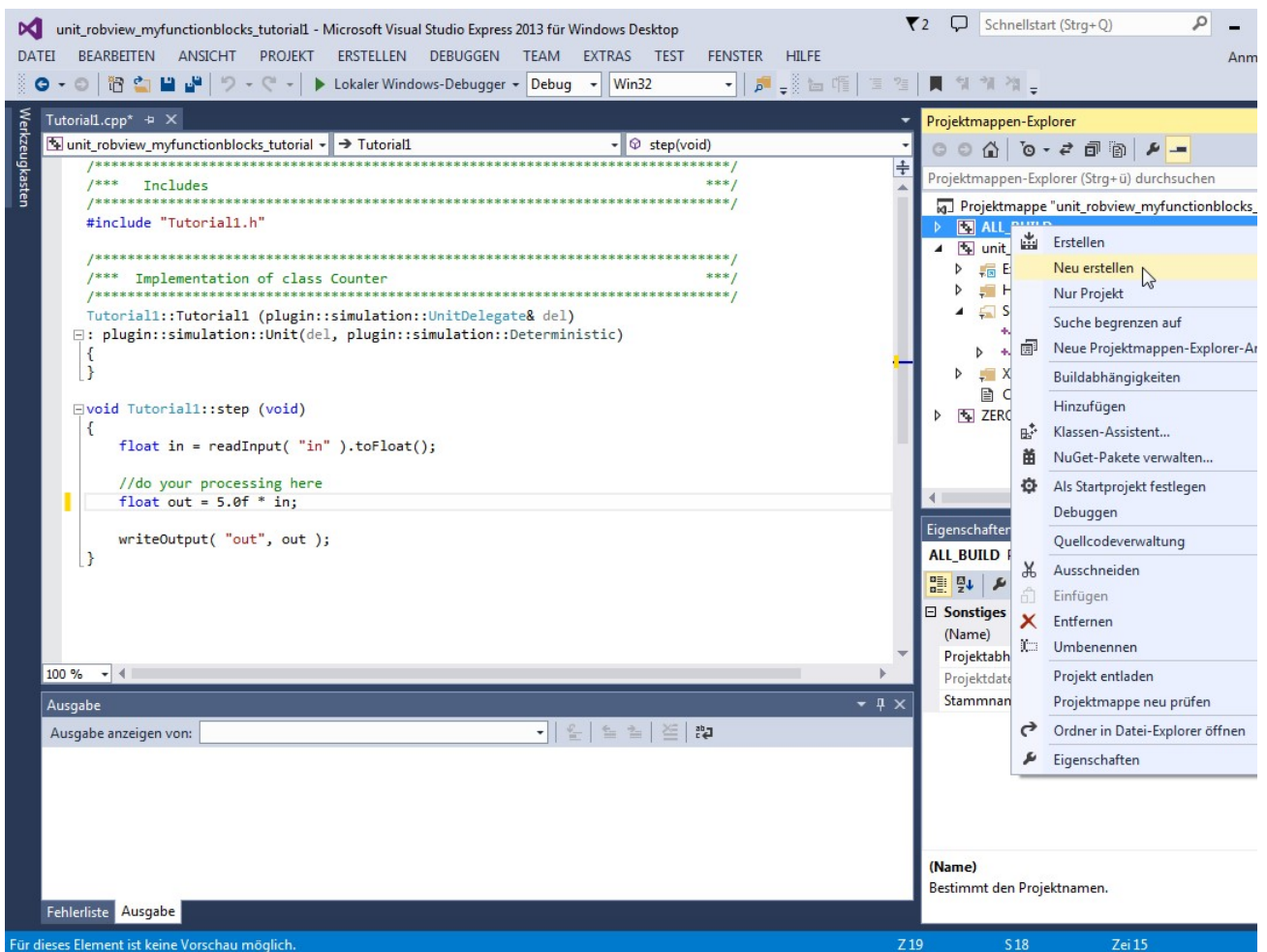
Durch anklicken von "Generate" wird schließlich die Visual Studio Solution Datei erzeugt.



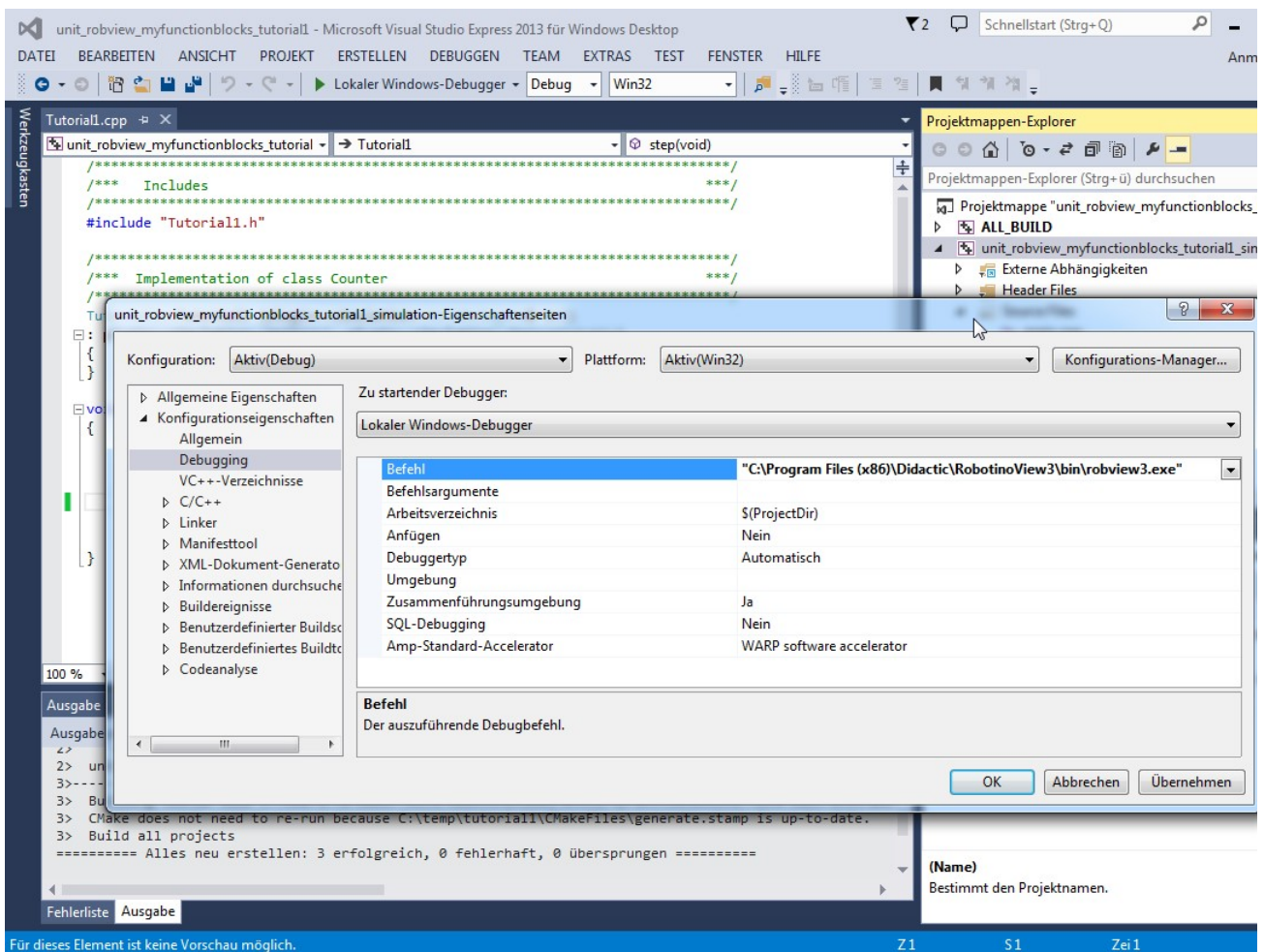
Gehen Sie nun mit dem Windows-Explorer in das Build-Verzeichnis (c:/temp/tutorial1) und doppelklicken die Solution.



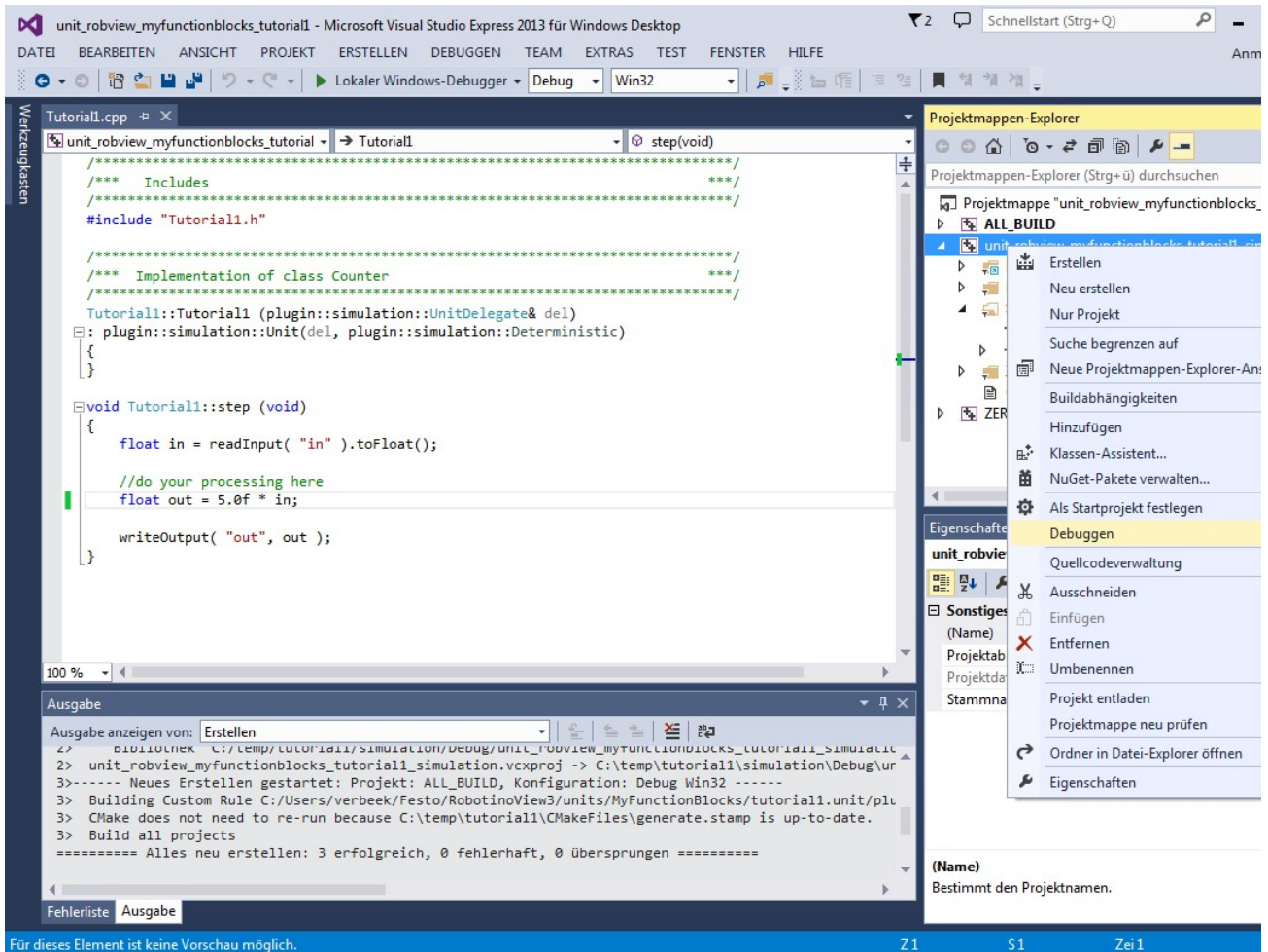
Erstellen Sie den Funktionsblock. Wie Sie sehen, haben wir vorher eine Änderung am Quellcode vorgenommen, so dass wir gleich merken, ob wirklich unser neu gebauter Funktionsblock verwendet wird.



Sie können Ihren Funktionsblock debuggen, in dem Sie Robotino View als ausführendes Programm bekannt machen.



Nun starten Sie den Debug-Prozess.



Robotino View wird gestartet. Erstellen Sie ein kleines Programm, das Ihren Funktionsblock verwendet.

The image shows two overlapping windows. The top window is 'Unbenannt* - ROBOTINO View 3.0.16'. It features a menu bar (Datei, Bearbeiten, Simulation, Robotino, Ansicht, Extras, Fenster, Hilfe) and a toolbar. The main area displays a block diagram with a block labeled 'T1' and numerical values like '2.000000' and '10.000000'. A sidebar on the right lists 'Funktionsblockbibliothek' with categories like Logik, Mathematik, Vektorrechnung, Anzeige, Bildverarbeitung, Generatoren, Filter, Navigation, Eingabegeräte, Datenaustausch, Benutzerdefiniert, and Eigene Funktionsblöcke. The bottom window is 'unit_robview_myfunctionblocks_tutorial1 (Ausführung) - Microsoft Visual Studio Express 2013 für Windows Desktop'. It shows the 'Tutorial1.cpp' file with the following code:

```

/***** Includes *****/
#include "Tutorial1.h"

/***** Implementation of class Counter *****/
Tutorial1::Tutorial1(plugin::simulation::UnitDelegate& del)
{
    plugin::simulation::Unit(del, plugin::simulation::Deterministic)
}

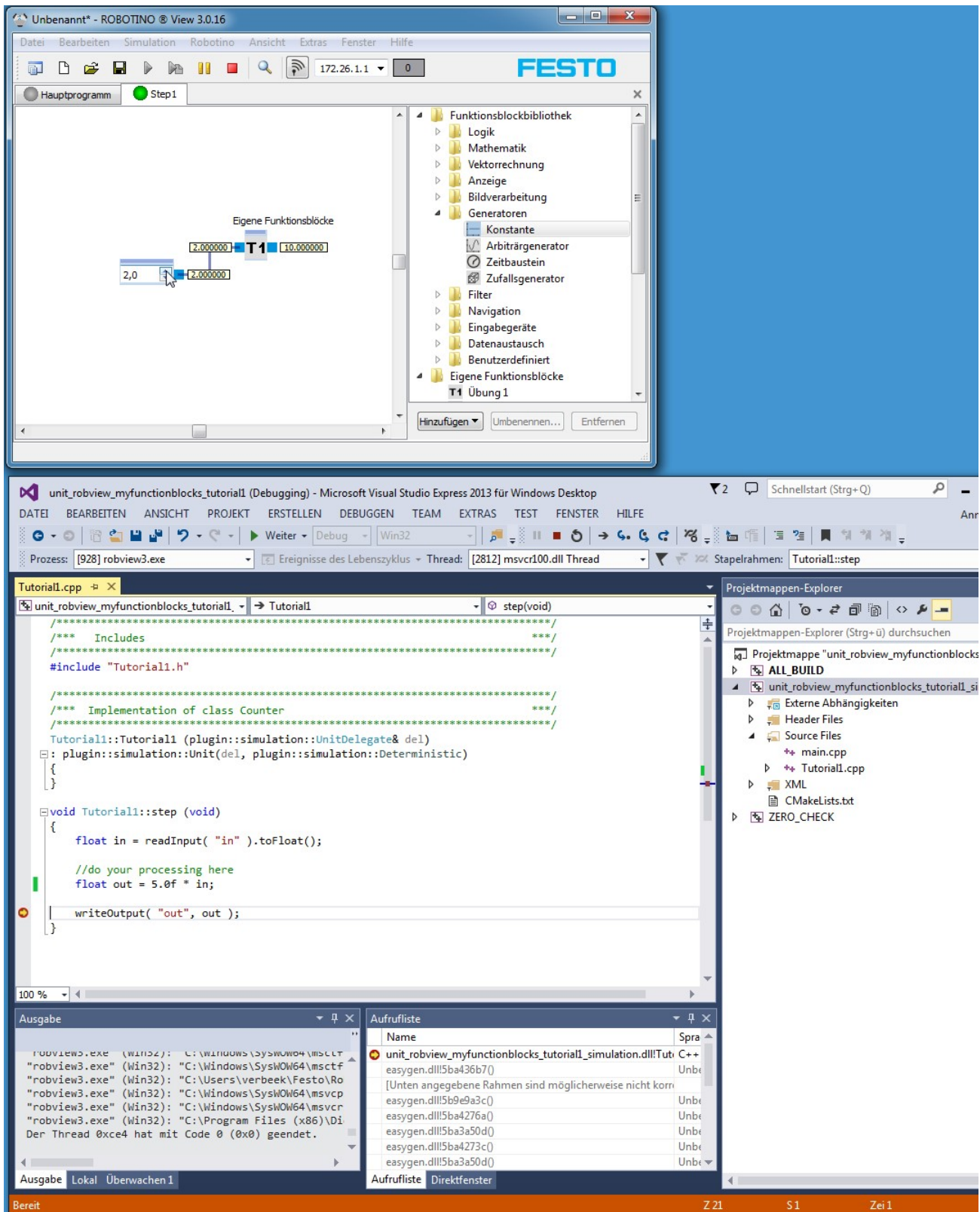
void Tutorial1::step(void)
{
    float in = readInput( "in" ).toFloat();

    //do your processing here
    float out = 5.0f * in;

    writeOutput( "out", out );
}
    
```

The right sidebar shows the 'Projektmappe-Explorer' with a tree view containing 'Projektmappe "unit_robview_myfunctionblocks_tutorial1"', 'ALL_BUILD', 'unit_robview_myfunctionblocks_tutorial1', 'Externe Abhängigkeiten', 'Header Files', 'Source Files' (main.cpp, Tutorial1.cpp), 'XML' (CMakeLists.txt), and 'ZERO_CHECK'. The bottom status bar shows 'Bereit', 'Z1', 'S1', and 'Zei 1'.

An der Ausgabe von T1 sehen Sie, dass die am Quellcode vorgenommenen Veränderungen kompiliert wurden. Setzen Sie einen Breakpoint im Quellcode. Damit die step() Funktion aufgerufen wird, muss sich der Eingangswert des Funktionsblock verändern. Klicken Sie dazu auf einen Pfeil innerhalb der Konstante, um den Wert zu erhöhen bzw. zu erniedrigen. Das Programm wird dann am Breakpoint angehalten.

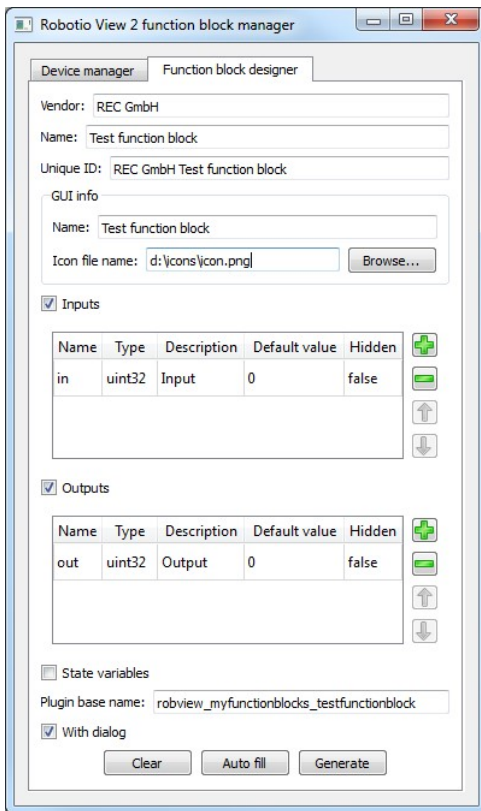


FESTO



Erstellen eines neuen Funktionsblocks

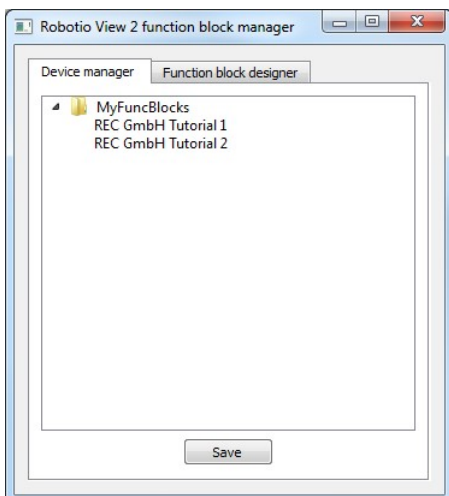
Zum Erstellen eines neuen Funktionsblocks ist im Robotino® View API das Tool "Function block manager" enthalten. Es kann im Windows-Startmenü im RobotinoView-Ordner aufgerufen werden. Zum Erstellen eines Funktionsblock muss zunächst zum Tab "Function block designer" gewechselt werden.



Im Beispiel wird der Funktionsblock "Test function block" erzeugt. Er besitzt einen Eingang und einen Ausgang, beide vom Typ uint32, keine Statusvariablen und einen Dialog. Es müssen übrigens nicht alle Daten eingegeben werden. Statt dessen genügt es, "Vendor" und "Name" anzugeben. Nach einem Klick auf "Auto fill" werden die restlichen Felder (außer "Icon file name") automatisch mit geeigneten Werten ausgefüllt. Dann müssen nur noch das Icon, Ein- und Ausgänge sowie Statusvariablen spezifiziert werden, und ob der Funktionsblock einen Dialog besitzen soll.

Mit einem Klick auf "Generate" wird der neue Funktionsblock angelegt. Im Ordner %USERPROFILE%\Festo\RobotinoView3\units\MyFunctionBlocks wird ein neues Unterverzeichnis erstellt, das die XML-Beschreibungsdatei, CMake-Skripte und bereits (funktionslose) C++-Quelldateien enthält. Damit ist es nun möglich, mit CMake ein Visual C++-Projekt zu erzeugen und damit weiterzuarbeiten.

Damit der Funktionsblock in Robotino View sichtbar ist, muss er noch ins Gerät "Eigene Funktionsblöcke" eingetragen werden. Dazu wechselt man zum Tab "Device manager".



Unterordner und Funktionsblöcke können durch entsprechende Funktionen im Kontextmenü eingefügt und gelöscht werden, Verschieben ist per Drag & Drop möglich. Unterordner können direkt in der Ansicht umbenannt werden. Nach Abschluss aller Modifikationen wird mit "Save" der neue Zustand gespeichert.



Einsatz auf Robotino

Eigene Funktionsblöcke können auch auf Robotino eingesetzt werden. Dazu müssen aber zunächst die Plugins für das auf Robotino installierte Linux-System übersetzt werden. Die neue 4GB-Speicherkarte für Robotino enthält die dazu erforderliche Software (C++-Compiler, make, CMake, RobotinoView-API), so dass das Übersetzen und Linken der Plugins direkt auf Robotino erfolgen kann. Für größere Projekte wird der Einsatz eines eigenen Entwicklungssystems mit schnellerem Prozessor und mehr Speicher empfohlen; als Betriebssystem zum Entwickeln eignet sich z.B. Ubuntu 9.04 sehr gut.

Die eigenen Funktionsblöcke liegen im Verzeichnis /home/robotino/Festo/RobotinoView3/units/MyFunctionBlocks.

Um ein Plugin zu übersetzen, muss folgendermaßen vorgegangen werden:

- Kopieren des Ordners mit den Quelldateien auf Robotino in den Ordner "/home/robotino/Festo/RobotinoView3/MyFunctionBlocks" per FTP (Benutzer: robotino, Kennwort: robotino). Der Pfad "/home/robotino/Festo/RobotinoView3/MyFunctionBlocks" wird beim ersten Start von Robotino View oder des Interpreters angelegt.
- Herstellen einer SSH-Verbindung zu Robotino (Benutzer: robotino, Kennwort: robotino).
- Erstellen eines Arbeitsverzeichnisses (z.B. für Tutorial 1) und Wechsel dorthin:

```
mkdir /home/robotino/build/tutorial1  
cd /home/robotino/build/tutorial1
```

- Erstellen von Makefiles aus den CMake-Skripten:

```
cmake /home/robotino/Festo/RobotinoView3/MyFunctionBlocks/tutorial1.unit/plugin
```

Alternativ ist es mit

```
ccmake /home/robotino/Festo/RobotinoView3/MyFunctionBlocks/tutorial1.unit/plugin
```

ähnlich wie unter Windows möglich, weitere Einstellungen wie z.B. Compiler-Flags festzulegen. Dazu muss zunächst das Projekt konfiguriert werden (Taste "C"). Dann können Einstellungen angepasst werden. Nach einem weiteren Tastendruck auf "C" und einem auf "G" ("Generate") wird das Makefile automatisch erzeugt.

- Übersetzen und Linken der Plugins für Simulation und GUI:

```
make
```

Die Bibliotheken werden nach dem Übersetzen automatisch an die richtige Stelle kopiert.

- Eintragen in die Beschreibungsdatei des "Eigene Funktionsblöcke"-Geräts. Dazu muss die Datei /home/robotino/Festo/RobotinoView3/devices/MyFunctionBlocks/device.xml bearbeitet werden. Alternativ kann die Datei unter Windows mit dem "Function block manager" bearbeitet und per FTP auf Robotino kopiert werden.