# B&R Automation Studio Target for Simulink® TM140







## Requirements

Training modules: TM210 - The Basics of Automation Studio 3

Software:

Automation Studio 3 (Version 3.0.80 and higher)

Automation Studio Target for Simulink $^{\ensuremath{\mathbb{R}}}$  (V3.0 and higher)

MATLAB® (R2008a and higher)

Simulink<sup>®</sup> (R2008a and higher)

Real-Time Workshop® (R2008a and higher)

Hardware:

None

# Trademarks

MATLAB, Simulink, Stateflow, Handle Graphics, Real-Time Workshop, SimBiology, SimHydraulics, SimEvents, and xPC TargetBox are registered trademarks and MathWorks, the L-shaped membrane logo, Embedded MATLAB, and PolySpace are trademarks of MathWorks, Inc.

Other product or brand names are trademarks or registered trademarks of their respective holders.

# Patents

MathWorks products are protected by one or more U.S. patents. Please see <u>www.mathworks.com/patents</u> for more information.

1

# Table of contents

<ul><li>1.1 Objectives</li><li>1.2 Definition</li><li>2. PREPARATIONS</li><li>2.1 Installation</li></ul>	7 8 9 10 11
2. PREPARATIONS	9 10
	10
2.1 Installation	
	11
2.2 Advanced software requirements	
3. FUNCTION BLOCKS OF B&R AUTOMATION STUDIO TOOLBOX	13
3.1 B&R Automation Studio Toolbox	13
3.2 B&R Config block	14
3.3 B&R Input / Output blocks	15
3.4 B&R Parameter block	19
3.5 B&R Extended Input / Output blocks	21
3.6 B&R Structure Input / Output blocks	27
4. CONFIGURATION SETTINGS	31
5. WORKING WITH B&R AUTOMATION STUDIO TARGET FOR	
SIMULINK	36
5.1 Basic example	36
5.2 Function block generation	46
5.3 Structure interface blocks	49
5.4 Automatic transfer	53
5.5 External Mode	54
6. EXAMPLES	56
6.1 PID controller	56
6.2 Temperature model	58
6.3 Hydraulics applications	61
6.4 Inverted Pendulum on a Cart Model	62
7. APPENDIX	72
7.1 Real-Time Workshop Embedded Coder	72
7.2 Simulink block support	72
7.3 Suggestions	73
7.4 Additional links	74

#### INTRODUCTION

For years the MATLAB<sup>®</sup> program package from MathWorks (www.mathworks.com) has served as a powerful tool in solving technical, mathematical and economic problems and has been used extensively in the industrial world. MATLAB<sup>®</sup> is a numerical computing environment and programming language. The biggest strength of the program lies in its handling of large matrices, as its name MATrix LABoratory suggests. MATLAB<sup>®</sup> can be expanded using various add-on packages, as Simulink<sup>®</sup> for instance. This program package allows graphic creation of simulation models used to adjust complex technical processes under realistic conditions.



Fig. 1: MATLAB<sup>®</sup> and Simulink<sup>®</sup> by MathWorks, Inc.

Automatic implementation of Simulink models in C-Code, specially optimized for use in B&R target systems, offers the developer new possibilities for designing sophisticated simulation models and control structures that would otherwise be impossible or very time-intensive to implement.

Δ

The biggest advantage of **Automatic Code Generation** affects those developers who already use MATLAB<sup>®</sup> and Simulink<sup>®</sup> for simulation and solutions design and to developers who used to tediously rework implemented structures in a language supported by Automation Studio in the past. In the procedures listed below the Automatic Code Generation tool provided by B&R represents an innovation with endless possibilities that help to productively reform the development of control systems.

The basic principle is simple: The module created in Simulink<sup>®</sup> is automatically translated using **Real-Time Workshop**<sup>®</sup> and **Real-Time Workshop**<sup>®</sup> **Embedded Coder** (optional) into the optimal language for the B&R target system guaranteeing maximum performance of the generated source code. Seamless integration into an **Automation Studio** project makes the development process perfect.



Fig. 2: Workflow of the Automatic Code Generation

The elimination of extensive reengineering in Automation Studio allows simple transfer of complex and sophisticated Simulink models to the PLC (Hardware-in-the-Loop). Closed-loop controllers can also be easily tested and optimized on the target system without requiring the user to adjust large amounts of code and run the risk of creating coding errors (On-Target Rapid Prototyping). **On-Target Rapid prototyping:** Automatic Code Generation makes it possible to quickly and easily transform sophisticated Simulink based control systems into source code and integrate them into an Automation Studio project.

Many potentially successful ideas have been immediately rejected due to the large amount of time required for conversion into executable machine code and the risk of developing a dead end solution. The 'Rapid Prototyping' concept brings an end to this. Using Simulink<sup>®</sup> and the Automatic Code Generation tool provided by B&R, any system, no matter how complex, can be intuitively built, compiled and tested in a short amount of time. This practically eliminates implementation errors as the Automatic Code Generation tool has been well-proven over several years in critical fields like aviation or automotive industry.

In the case of 'On-Target Rapid Protoyting' the prototyping process is directly performed on the industrial target system which makes test results even more realistic and significant.

Hardware-in-the-Loop: Every modification of the closed-loop controller bears the risk of damaging the controlled system during commissioning. 'Hardware-in-the-Loop' is the key word that stands for simple transfer of sophisticated system models developed in Simulink<sup>®</sup> to a B&R target system. The prepared PLC assumes the roll of the actual system for the duration of the first function test. This allows easy and riskless testing of new controller concepts without risking the damage of costly machine parts. In some cases the controller and system simulation can even run on the same target system.

Although there are numerous applications for using B&R's Automatic Code Generation, they have one thing in common: the possibility to generate source code for B&R target systems **at the push of one single button**.

#### 1.1 Objectives

After completing the installation described in section 1.3, simple access to professional application can be learned with the help of an example worked out in section 3. More detailed examples are located in section 5. In section 6 there is a short introduction to MATLAB and Simulink functions as well as an overview of more detailed links. A description of all B&R Automation Studio blocks for Simulink is located in section 2.



Fig. 3: Objectives

After successful completion of the training module, the user should be able to prepare existing Simulink models for the Automatic Code Generation using *B&R* Automation Studio Target for Simulink.

An additional part of this training module deals with the integration of automatically generated tasks in existing Automation Studio projects as well as recognition of numerous options for error diagnosis.

An introduction to the products of MathWorks is not included in the course of this module, but must be found in the documentation accompanying the respective products.

## 1.2 Definition

#### 1.2.1 Rapid Prototyping

As mentioned above, 'Rapid Prototyping' offers numerous possibilities for easy and flexible implementation of sophisticated control systems solutions. Innovative ideas that in the past would have been rejected because of time and resource restraints can now be smoothly developed using Simulink and transferred to the PLC using *B&R Automation Studio Target for Simulink*. Tedious manual reimplementation of source code, which always bears the risk of coding errors, is a thing of the past.



Fig. 4: Rapid Prototyping

The procedure is quite simple - the task created in Simulink and transferred to the controller via *B&R Automation Studio Target for Simulink* is ready for application in a matter of a few steps.

#### 1.2.2 Hardware-in-the-Loop

In order to avoid damaging the real machine system when applying newly developed algorithms, it is recommended to implement critical system parts into an emulation system.

For this purpose, a second B&R target system is used. The emulation task runs on this system, which mimics the behaviour of the real plant as accurately as possible. New developments are thus tested on the emulation target system without putting the system operator at risk of experiencing damage to hardware components.



Fig. 5: Hardware-in-the-Loop on two separate B&R target systems

As there is enough free processing power available on the controller in most of the cases, both tasks can be run on the same B&R controller, thanks to the task structure of B&R Automation Studio.

If the behaviour of the physical inputs and outputs must not be neglected it is essential to use two separate, hard-wired B&R target systems, however.



Fig. 6: Hardware-in-the-Loop on one single B&R target system Preparations

### 1.3 Installation

The components required for using B&R Automation Studio Target for Simulink can be installed using the enclosed setup script 'install.p'.

📄 Name 🛆			2
國 data.bin			
🚯 install.p			
	Run	F9	-
	Create Zip File		
	Rename	F2	1
يستعمموني ا	Delete	Entf	1

Fig. 7: B&R Automation Studio Target for Simulink setup

The B&R Automation Studio Target for Simulink components will be installed into a directory of your choice (e.g. C:\Program Files\B&R Automation Studio Target for Simulink – the directory 'B&R Automation Studio Target for Simulink' will be added automatically) and registered in MATLAB. After the installation please restart MATLAB in order to guarantee smooth functionality.

## IMPORTANT

The *B&R Automation Studio Target for Simulink* components must not be installed into the MATLAB program path (e.g. 'C:\Program Files\MATLAB\R2010a').

For removing *B&R Automation Studio Target for Simulink* from your system please use the enclosed uninstall script '**uninstall.p**'.



Fig. 8: B&R Automation Studio Target for Simulink uninstall

## IMPORTANT

Only the 32bit version of MATLAB is supported (also on 64bit systems). For installation on Windows Vista or Windows 7 the installation of B&RAutomation Studio Target for Simulink has to be performed as administrator (Right-click  $\rightarrow$  'Run as administrator').

	Open
MATL	Troubleshoot compatibility
R2010b	Open file location
	😯 Run as administrator
	7-Zip 🕨
	Pin to Taskbar
	Pin to Start Menu
	Restore previous versions
	Send to 🔶 🕨
	Cut
	Сору
	Create shortcut
	Delete
	Rename
	Properties

## 1.4 Advanced software requirements

For use of Automatic Code Generation with *B&R Automation Studio Target for Simulink*, the following software components are required:

- Automation Studio 3 (Version 3.0.71 and higher)<sup>i</sup>
- Automation Studio Target for Simulink<sup>®</sup> (Version 3.0 and higher)
- MathWorks Release 2007b and higher<sup>ii</sup>
  - MATLAB<sup>®</sup> (Version 7.5 and higher)
  - Simulink<sup>®</sup> (Version 7.0 and higher)
  - Real-Time Workshop<sup>®</sup> (Version 7.0 and higher)

For optimal code efficiency Real-Time Workshop<sup>®</sup> Embedded Coder<sup>™</sup> is suggested to be used for automatic code generation.

• Real-Time Workshop<sup>®</sup> Embedded Coder<sup>™</sup> (Version 5.0 and higher) Furthermore in case of use of Stateflow objects the following products are also required for code generation:

<sup>i</sup> **Recommended:** 3.0.80 or higher

<sup>&</sup>lt;sup>ii</sup> **Recommended:** Release 2008a or higher

- Stateflow<sup>®</sup> (Version 7.0 or higher)
- Stateflow<sup>®</sup> Coder<sup>™</sup> (Version 7.0 or higher)

Most toolboxes provided by MathWorks are also fully compatible with B&R Automation Studio Target for Simulink.

## 2. FUNCTION BLOCKS OF B&R AUTOMATION STUDIO TOOLBOX

In this section, the individual components of *B&R Automation Studio Target* for *Simulink* are described and explained step by step.

- B&R Automation Studio Toolbox
- B&R Config block
- B&R Input block
- B&R Output block
- B&R Parameter block
- B&R Extended Input block
- B&R Extended Output block
- B&R Structure Input block
- B&R Structure Output block

## 2.1 B&R Automation Studio Toolbox

The 'B&R Automation Studio Toolbox' is automatically installed during the setup of *B&R Automation Studio Target for Simulink*. It contains several B&R specific interface and configuration blocks that are described in the following sections.

File Edit View Help	😽 Simulink Library Browser			
Libraries       Libraries       Most Prequently Used Blocks         Image: Simulink       B&R Automation Studio Toolbox       Search Results: (none)       Most Prequently Used Blocks         Image: Simulink System Toolbox       B&R CONFIG       B&R EXT_IN       >       B&R EXT_OUT         Image: Simulink Strass       Simulink Extras       Config       Extended Input       Extended Output         Image: Six Stateflow       B&R IN       >       B&R OUT       Extended Output         Imput       Output       Parameter         Imput       Structure Input       Structure Output	File Edit View Help			
Image: Simulink       Bark Automation Studio Toolbox         Image: Simulink Studio Toolbox       Bark Automation Studio Toolbox         Image: Simulink Studio Toolbox       Bark Real-Time Workshop         Image: Simulink States       Bark CONFIG         Image: States       Bark CONFIG         Ima	📗 🗅 😅 \Rightarrow 📕 Enter search term 🔽	<b>A</b> 📺		
B&R Automation Studio Toolbox   Real-Time Workshop   Real-Time Workshop   Simulink SD Animation   Simulink Extras   Stateflow     B&R RIN   B&R OUT     B&R PARAM ETER     Input   Output     Parameter     Structure Input     Structure Input     Structure Input     Structure Input     Structure Input     Structure Input     Structure Output	Libraries	Library: B&R Automation Studio Toolb	0× Search Results: (none)	Most Frequently Used Blocks
Input Output Parameter  B&R STRUCT_IN B&R STRUCT_OUT Structure Input Structure Output	Bar Automation Studio Toolbox     Solution Toolbox     General System			
B&R STRUCT_IN     >     >     B&R STRUCT_OUT       Structure Input     Structure Output		B&R IN	B&R OUT	B&R PARAMETER
Structure Input Structure Output		Input	Output	Parameter
Channing DOD Automatics Church Tealling				
Snowing, Bar Automation Studio toolbox	Showing: B&R Automation Studio Toolbox	·		

Fig. 9: B&R Automation Studio Toolbox

B&R Automation Studio Target for Simulink®

#### 2.2 B&R Config block

The 'B&R Config block' is used to switch between three modes of operation, 'Simulation', 'Code Generation (ERT based)' and 'Code Generation (GRT based)'. Once the block is inserted to an existing Simulink model the current configuration set is renamed to 'Simulation' and the 'Code Generation (ERT based)' and 'Code Generation (GRT based)' configuration sets are added.

#### IMPORTANT

Only one instance of the B&R Config block can be added to a Simulink model. The B&R Config block has to be located in the root of your Simulink model.

당 Block Para	ameters: Config			×	
B&R - Config	uration Block (mask) (li	nk)			
Configuration	n module for B&R Auto	mation Studio Tar	get for Simulink.		
-Parameters-					
Config Set:	Code Ceneration (EP)	T baced)			
Coning bec.	Code Generation (ERT based)				
	Code Generation (GRT based)				
	Simulation				
	ОК	Cancel	Help	Apply	

Fig. 10: B&R Config block

Selecting the configuration set 'Code Generation (ERT based)' automatically invokes the system target file 'bur\_ert.tlc' for the current Simulink model. If 'Code Generation (GRT based)' is selected the system target file 'bur\_grt.tlc' will be activated. Alternatively the corresponding system target file can also be selected manually by experienced users.

#### INFO

The configuration set 'Code Generation (ERT based)' will only be available if Real-Time Workshop® Embedded Coder™ is installed on the current system.

## 2.3 B&R Input / Output blocks

The 'B&R Input block' serves as the interface between the automatically generated task or function block based on the Simulink model and the other parts of the project. For each 'B&R Input block' a variable is created on the target system.

Source Block Parameters: Input
B&R - Input Block (mask) (link)
Input interface for B&R Automation Studio process variable.
Parameters
Variable Name
Input
Variable Description
Variable Scope LOCAL
Variable Data Type LREAL
Initial Value
0
Array Dimension (0Scalar)
0
Memory standard
<u>OK</u> <u>C</u> ancel <u>H</u> elp <u>Apply</u>

Fig. 11: B&R Input block

Variable Name: Specifies the Automation Studio variable name on the target system.

Variable Description: Description of the Automation Studio variable.

**Variable Scope:** Specifies the scope ('GLOBAL' or 'LOCAL') of the variable created on the target system.

#### INFO

Local variables are created automatically. Global variables have to be declared manually in Automation Studio or can be generated automatically by setting the "create global\*.var file" feature in the B&R Advanced Settings. The automatically generated file 'global.txt' is intended as support for the user, however. (see chapter 4.1.8)

**Variable Data Type:** The data type of the created variable can be selected from all data types available in Automation Studio and Simulink:

Antennation Chudie	<b>C</b> :	
Automation Studio	Simulink	Value range
BOOL	boolean	FALSE, TRUE
DOOL	Doolean	TAESE, TROE
DINT	int32	-2.147.483.648 2.147.483.647
INT	int16	-32768 32767
	ما م <b>ا</b> م ا	4 75 . 200 4 75 . 200
LREAL (default)	double	-1.7E+308 1.7E+308
REAL	single	-3.4E+38 3.4E+38
	Single	
SINT	int8	-128 127
UDINT	uint32	0 4.294.967.295
UINT	uint16	0 65535
USINT	uint8	0 255
USINI	unito	0 233

#### IMPORTANT

When manually declaring global variables in Automation Studio, the user must make sure that the data type of the variable in the project matches the selected data type in the dialog field.

**Initial Value:** The start value of the variable is defined in this entry field. The variable created on the B&R target will be initialized with this value.

#### INFO

All elements of an array are initialized with the same value.

Array Dimension: If the value of this field exceeds zero, an array is created instead of a scalar variable.

Memory: It can be choosen if the variable is retain or standard.

**Parameter (only for functionblock):** If this feature is active the variable is treated as a internal variable and not as input.

The 'B&R Output block' serves as the interface between the automatically generated task or function block based on the Simulink model and the other parts of the project. For each 'B&R Output block' a variable is created on the target system.

🙀 Sink Block Parameters: Output	×
B&R - Output Block (mask) (link)	
Output interface for B&R Automation Studio process variable.	
Parameters	
Variable Name	
Output	
Variable Description	
Variable Scope LOCAL	
Variable Data Type LREAL	1
Initial Value	
0	
Array Dimension (0Scalar)	
0	
Memory standard	1
OK Cancel Help Apply	

Fig. 12: B&R Output block

**Variable Name**: Specifies the Automation Studio variable name on the target system.

Variable Description: Description of the Automation Studio variable.

**Variable Scope:** Specifies the scope ('GLOBAL' or 'LOCAL') of the variable created on the target system.

#### INFO

Local variables are created automatically. Global variables have to be declared manually in Automation Studio or can be generated automatically by setting the "create global\*.var file" feature in the B&R Advanced Settings. The automatically generated file 'global.txt' is intended as support for the user, however. (see chapter 4.1.8)

**Variable Data Type:** The data type of the created variable can be selected from all data types available in Automation Studio and Simulink:

Automation Studio	Simulink	Value range
BOOL	boolean	FALSE, TRUE
DINT	int32	-2.147.483.648 2.147.483.647
INT	int16	-32768 32767
LREAL (default)	double	-1.7E+308 1.7E+308
REAL	single	-3.4E+38 3.4E+38
SINT	int8	-128 127
UDINT	uint32	0 4.294.967.295
UINT	uint16	0 65535
USINT	uint8	0 255

#### IMPORTANT

When manually declaring global variables in Automation Studio, the user must make sure that the data type of the variable in the project matches the selected data type in the dialog field.

**Initial Value**: The start value of the variable is defined in this entry field. The variable created on the B&R target will be initialized with this value.

#### INFO

All elements of an array are initialized with the same value.

**Array Dimension:** If the value of this field exceeds zero, an array is created instead of a scalar variable.

**Memory:** It can be choosen if the variable is retain or standard.

### 2.4 B&R Parameter block

The 'B&R Parameter block' is used to make internal parameters of individual blocks accessible during operation on the target system. For each 'B&R Input block' a variable is created on the target system.

Block Parameters: Parameter1	
B&R - Parameter Block (mask) (link)	
Parameter interface for B&R Automation Studio process variable.	
Parameters	
Variable Name	
Parameter	
Variable Description	)
Variable Scope LOCAL	
Variable Data Type LREAL	
Initial Value	
0	
Array Dimension (0Scalar)	
0	
Memory standard	
<u>O</u> K <u>C</u> ancel <u>H</u> elp <u>A</u> pply	

Fig. 13: B&R Parameter block

**Variable Name:** Specifies the Automation Studio variable name on the target system.

Variable Description: Description of the Automation Studio variable.

**Variable Scope**: Specifies the scope ('GLOBAL' or 'LOCAL') of the variable created on the target system.

#### INFO

Local variables are created automatically. Global variables have to be declared manually in Automation Studio or can be generated automatically by setting the "create global\*.var file" feature in the B&R Advanced Settings. The automatically generated file 'global.txt' is intended as support for the user, however. (see chapter 4.1.8)

**Variable Data Type:** The data type of the created variable can be selected from all data types available in Automation Studio and Simulink:

Automation Studio	Simulink	Value range
Automation Studio	Sindink	value runge
BOOL	boolean	FALSE, TRUE
DINT	int32	-2.147.483.648 2.147.483.647
INT	int16	-32768 32767
LREAL (default)	double	-1.7E+308 1.7E+308
REAL	single	-3.4E+38 3.4E+38
SINT	int8	-128 127
UDINT	uint32	0 4.294.967.295
UINT	uint16	0 65535
USINT	uint8	0 255

#### IMPORTANT

When manually declaring global variables in Automation Studio, the user must make sure that the data type of the variable in the project matches the selected data type in the dialog field.

**Initial Value**: The start value of the variable is defined in this entry field. The variable created on the B&R target will be initialized with this value.

#### INFO

All elements of an array are initialized with the same value.

Array Dimension: If the value of this field exceeds zero, an array is created instead of a scalar variable.

**Memory:** It can be choosen if the variable is retain or standard.

#### 2.5 B&R Extended Input / Output blocks

The 'B&R Extended Input block' serves as the interface between the automatically generated task or function block based on the Simulink model and the other parts of the project. For each 'B&R Extended Input block' a variable is created on the target system.

The 'Extended B&R blocks' provide an easy to use means to convert hardware inputs or outputs (usually INT) to floating point values (REAL or LREAL) for powerful calculations in the control algorithm and vice versa. The conversion and casting is done automatically by the library block.

Source Block Parameters: Extended Input1	×
B&R - Extended Input Block (mask) (link)	
Extended input interface for B&R Automation Studio process varia	ble.
Parameters	
Variable Name	
Input	
Variable Description	
Variable Scope LOCAL	•
Array Dimension (0Scalar)	
0	
Automation Studio Data Type INT	◄
Automation Studio Minimum Value	
-32768	-
, Automation Studio Maximum Value	
32767	
Automation Studio Initial Value	
0	
Simulink Data Type LREAL	<b>-</b>
Simulink Minimum Value	
-100	- 1
Simulink Maximum Value	
100	-
Simulink Simulation Value	
0	
Memory standard	┓║
,,	
<u>OK</u> <u>C</u> ancel <u>H</u> elp <u>A</u> p	oly

Fig. 14: B&R Extended Input block

B&R Automation Studio Target for Simulink®

Variable Name: Specifies the Automation Studio variable name on the target system.

Variable Description: Description of the Automation Studio variable.

Variable Scope: Specifies the scope ('GLOBAL' or 'LOCAL') of the variable created on the target system.

#### INFO

Local variables are created automatically. Global variables have to be declared manually in Automation Studio or can be generated automatically by setting the "create global\*.var file" feature in the B&R Advanced Settings. The automatically generated file 'global.txt' is intended as support for the user, however. (see chapter 4.1.8)

**Array Dimension:** If the value of this field exceeds zero, an array is created instead of a scalar variable.

Automation Studio Data Type: The data type of the created variable can be selected from all data types available in Automation Studio and Simulink:

Automation Studio	Simulink	Value range
BOOL	boolean	FALSE, TRUE
DINT	int32	-2.147.483.648 2.147.483.647
INT	int16	-32768 32767
LREAL (default)	double	-1.7E+308 1.7E+308
REAL	single	-3.4E+38 3.4E+38
SINT	int8	-128 127
UDINT	uint32	0 4.294.967.295
UINT	uint16	0 65535
USINT	uint8	0 255

#### IMPORTANT

When manually declaring global variables in Automation Studio, the user must make sure that the data type of the variable in the project matches the selected data type in the dialog field.

Automation Studio Minimum Value: Minimum value for Automation Studio input corresponding to the minimum Simulink value used for calculations.

Automation Studio Maximum Value: Maximum value for Automation Studio input corresponding to the maximum Simulink value used for calculations.

Automation Studio Initial Value: The start value of the variable is defined here. The variable created on the B&R target will be initialized with this value.

#### INFO

All elements of an array are initialized with the same value.

**Simulink Data Type**: The data type of the variable used for calculations can be selected from all data types available in Automation Studio and Simulink.

The conversion operation from Simulink calculation value to Automation Studio value is:

$$Sl\_value = (Sl\_DataType) \left[ Sl\_Min + \frac{(Sl\_Max - Sl\_Min)}{(AS\_Max - AS\_Min)} \cdot (AS\_value - AS\_Min) \right]$$

**Simulink Minimum Value**: Minimum Simulink value corresponding to the minimum value for Automation Studio input. The calculated value is limited automatically.

**Simulink Maximum Value**: Maximum Simulink value corresponding to the maximum value for Automation Studio input. The calculated value is limited automatically.

**Simulink Simulation Value:** During Simulink simulations the output of the block will be set to the value defined here.

Memory: It can be choosen if the variable is retain or standard.

**Parameter (only for functionblock):** If this feature is active the variable is treated as a internal variable and not as input.

The 'B&R Extended Output block' serves as the interface between the automatically generated task or function block based on the Simulink model and the other parts of the project. For each 'B&R Extended Output block' a variable is created on the target system.

The 'Extended B&R blocks' provide an easy to use means to convert hardware inputs or outputs (usually INT) to floating point values (REAL or LREAL) for powerful calculations in the control algorithm and vice versa. The conversion and casting is done automatically by the library block.

🖬 Sink Block Parameters: Extended Output	x
B&R - Extended Output Block (mask) (link)	
Extended output interface for $\ensuremath{B\&R}$ Automation Studio process variable.	
Parameters	
Variable Name	
Output	1
Variable Description	
	1
Variable Scope LOCAL	I
Array Dimension (0Scalar)	
0	-
Simulink Data Type LREAL	1
Simulink Minimum Value	1
-100	-
Simulink Maximum Value	
100	-
Simulink Simulation Value	Í
0	-
Automation Studio Data Type INT	1
Automation Studio Data Type Jun	1
-32768	-
Automation Studio Maximum Value	
32767	-
Automation Studio Initial Value	
	-
	1
Memory standard	
OK Cancel Help Apply	

Fig. 15: B&R Extended Output block

Variable Name: Specifies the Automation Studio variable name on the target system.

Variable Description: Description of the Automation Studio variable.

Variable Scope: Specifies the scope ('GLOBAL' or 'LOCAL') of the variable created on the target system.

#### INFO

Local variables are created automatically. Global variables have to be declared manually in Automation Studio or can be generated automatically by setting the "create global\*.var file" feature in the B&R Advanced Settings. The automatically generated file 'global.txt' is intended as support for the user, however. (see chapter 4.1.8)

**Array Dimension:** If the value of this field exceeds zero, an array is created instead of a scalar variable.

**Simulink Data Type**: The data type of the variable used for calculations can be selected from all data types available in Automation Studio and Simulink:

Automation Studio	Simulink	Value range
BOOL	boolean	FALSE, TRUE
DINT	int32	-2.147.483.648 2.147.483.647
INT	int16	-32768 32767
LREAL (default)	double	-1.7E+308 1.7E+308
REAL	single	-3.4E+38 3.4E+38
SINT	int8	-128 127
UDINT	uint32	0 4.294.967.295
UINT	uint16	0 65535
USINT	uint8	0 255

**Simulink Minimum Value**: Minimum Simulink value corresponding to the minimum value for Automation Studio output.

**Simulink Maximum Value:** Maximum Simulink value corresponding to the maximum value for Automation Studio input. The calculated value is limited automatically.

**Simulink Simulation Value**: During Simulink simulations the output of the block will be set to the value defined here.

Automation Studio Data Type: The data type of the variable used for calculations can be selected from all data types available in Automation Studio and Simulink.

The conversion operation from Simulink calculation value to Automation Studio value is:

 $AS\_value = (AS\_DataType) \left[ AS\_Min + \frac{(AS\_Max - AS\_Min)}{(Sl\_Max - Sl\_Min)} \cdot (Sl\_value - Sl\_Min) \right]$ 

#### IMPORTANT

When manually declaring global variables in Automation Studio, the user must make sure that the data type of the variable in the project matches the selected data type in the dialog field.

Automation Studio Minimum Value: Minimum value for Automation Studio input corresponding to the minimum Simulink value used for calculations. The calculated value is limited automatically.

Automation Studio Maximum Value: Maximum value for Automation Studio input corresponding to the maximum Simulink value used for calculations. The calculated value is limited automatically.

Automation Studio Initial Value: The start value of the variable is defined here. The variable created on the B&R target will be initialized with this value.

#### INFO

All elements of an array are initialized with the same value.

Memory: It can be choosen if the variable is retain or standard.

#### **B&R Structure Input / Output blocks**

The 'B&R Structure Input block' enables the use of structure elements defined in Automation Studio for Automatic Code Generation. Structures that are defined in the type files assigned to the current Simulink model can be used as an interface for the automatically generated task or function block.

Variable Name	
Input	
Variable Scope	
LOCAL	
Type File	
<empty></empty>	
Structure Name	(
<empty></empty>	
Structure Element <empty></empty>	>
Variable Data Type	
Array Dimension (0Scalar)	
p	
Ok Cancel	

Fig. 16: B&R Structure Input block

**Variable Name:** Specifies the Automation Studio variable name on the target system.

**Variable Scope:** Specifies the scope ('GLOBAL' or 'LOCAL') of the variable created on the target system.

#### INFO

Local variables are created automatically. Global variables have to be declared manually in Automation Studio or can be generated automatically by setting the "create global\*.var file" feature in the B&R Advanced Settings. The automatically generated file 'global.txt' is intended as support for the user, however. (see chapter 4.1.8)

**Type File:** Lists all available type files (\*.TYP) for the current model (see chapter 4.3). After selecting a type file the according structure names defined in the file are displayed in 'Structure Name'.

**Structure Name:** Lists the available structures in the currently selected type file. After selecting a structure name the according structure elements are displayed in 'Structure Element'.

**Variable Data Type:** Displays the data type of the currently selected structure element. The data type is defined in the type file (see chapter 4.3) and cannot be modified in this mask.

**Array Dimension:** Displays the array dimension of the currently selected structure element. The array dimension is defined in the type file (see chapter 4.3) and cannot be modified in this mask.

The 'B&R Structure Output block' enables the use of structure elements defined in Automation Studio for Automatic Code Generation. Structures that are defined in the type files assigned to the current Simulink model can be used as an interface for the automatically generated task or function block.

📣 B&R Structure Block	
Variable Name	
Output	
Variable Scope	
LOCAL	<b>*</b>
Type File	
<empty></empty>	*
,	_
Structure Name	
<empty></empty>	<b>*</b>
Structure Element	
<empty></empty>	<b>*</b>
Variable Data Type	
Array Dimension (0Scalar)	
,	
	1
Ok	Cancel

Fig. 17: B&R Structure Input block

**Variable Name:** Specifies the Automation Studio variable name on the target system.

**Variable Scope:** Specifies the scope ('GLOBAL' or 'LOCAL') of the variable created on the target system.

## INFO

Local variables are created automatically. Global variables have to be declared manually in Automation Studio or can be generated automatically by setting the "create global\*.var file" feature in the B&R Advanced Settings. The automatically generated file 'global.txt' is intended as support for the user, however. (see chapter 4.1.8) **Type File:** Lists all available type files (\*.TYP) for the current model (see chapter 4.3). After selecting a type file the according structure names defined in the file are displayed in 'Structure Name'.

**Structure Name:** Lists the available structures in the currently selected type file. After selecting a structure name the according structure elements are displayed in 'Structure Element'.

Variable Data Type: Displays the data type of the currently selected structure element. The data type is defined in the type file (see chapter 4.3) and cannot be modified in this mask.

**Array Dimension:** Displays the array dimension of the currently selected structure element. The array dimension is defined in the type file (see chapter 4.3) and cannot be modified in this mask.

## 3. CONFIGURATION SETTINGS

For the configuration of the interface between Simulink and Automation Studio the following project settings options are available.

🍇 Configuration Parameters: sa	mple/Code Generation ERT based (Active)		×
Select:	-Simulation time		<b>_</b>
Solver Data Import/Export	Start time: 0.0	Stop time: inf	
… Optimization ⊕. Diagnostics … Hardware Implementation	Solver options		
Model Referencing	Type: Fixed-step	Solver: discrete (no continuous states)	<u></u>
in Simulation Target in Real-Time Workshop in Report	Fixed-step size (fundamental sample time): 0.1		
Comments	Tasking and sample time options		
Symbols	Periodic sample time constraint:	Unconstrained	-
Custom Code Debug	Tasking mode for periodic sample times:	, SingleTasking	
Interface		angotoang	
Code Style	Automatically handle rate transition for data transfer		
···· Templates ···· Data Placement	Higher priority value indicates higher task priority		
Data Type Replacement			
Memory Sections			
bax in omidion			
			<b>-</b>
0		OK Cancel Help	Apply

Fig. 18: Fundamental sample time

**Fundamental sample time:** The fundamental sample time of the Simulink model must be equal to the selected task class cycle of the PLC. The sample time is entered in seconds.

## IMPORTANT

If the fundamental sample time specified in Simulink does not match the duration of the cyclic task class for the automatically generated task in Automation Studio, the task will be suspended and an entry in the PLC's logbook will be created.

elect:	
Solver	Automation Studio project path c:\projects\project_1
"Data Import/Export	Browse
- Optimization	
Diagnostics	Automation Studio task or function block name sample
Hardware Implementation Model Referencing	☐ Create zip-file
Simulation Target	
Code Generation	Zip-file path C:\projects
Report	Browse
Comments	Drowse
Symbols	✓ Add task or library to hardware configuration
Custom Code	Automation Studio configuration name Simulation
- Debug	Automation Studio conliguration name Simulation
Interface	Automation Studio PLC name PLC1
-SIL and PIL Verification	
-Code Style	Taskdass 1.0
Templates	Change taskclass timing settings
- Code Placement - Data Type Replacement	
Memory Sections	Cycletime [s] 0.01
B&R Basic Settings	Taskdass tolerance [s] 0.01
B&R Information	Systemtick [s] 0.001
	Create function block
	1 Create function block
	QK Cancel Help Apply

Fig. 19: B&R Basic Settings

Automation Studio Project Path: Base directory (absolute or relative path) of the Automation Studio project containing the automatically generated task

Automation Studio Task Name: Name of the automatically generated task

**Create zip-file:** If the generated source code should not be integrated into an existing Automation Studio on the developer's computer, there is also the possibility to export the generated source code into a zip-file and easily import the zipped task into Automation Studio (File  $\rightarrow$  Import...).

**Zip-file path:** Destination directory (absolute or relative path) for the automatically generated zip-file

Add task to hardware configuration: Automatically add the generated task to the current hardware configuration of the Automation Studio project

Automation Studio configuration name: Name of the current Automation Studio configuration

Automation Studio PLC name: Name of the current Automation Studio PLC

Taskclass: Number of taskclass where the task should be assigned

**Change taskclass timing settings:** Taskclass timing settings, of selected taskclass, are adepted automatically

Cycletime: Taksclass cycletime in seconds

Systemtick: Systemtick in seconds

INFO

For more information regarding taskclass timing settings, see the chapter "Real-time Operating System" in the Automation Studio help.

**Create function block:** Create an Automation Studio compatible function block instead of a task (see chapter 4.2)

#### IMPORTANT

The creation of function blocks using B&R Automation Studio Target for Simulink is intended for discrete models only.

For continous time models, please use the Simulink® Model Discretizer or contact the B&R support.

elect:	
Solver	Automation Studio package name
Data Import/Export	Add TYP files
Optimization     Diagnostics	
Hardware Implementation	Add source files
Model Referencing	Add include directories
Simulation Target	
Code Generation	Additional compiler switches
Report	Version
- Comments Symbols	
Custom Code	Encyption password
Debug	Finable continous time support
Interface	Enable expert mode
-SIL and PIL Verification	
- Code Style	Enable logging
- Templates	F Enable automatic transfer
Data Type Replacement	F Enable External Mode
Memory Sections B&R Basic Settings	External Mode buffer memory size 1000000
B&R Advanced Settings	External Mode IP address 127.0.0.1
box mornation	Enable model specific post processing
	Post processing file name
	☐ Create global *.var file
	OK Cancel Help Appl

Fig. 20: B&R Advanced Settings

Automation Studio package name: Name of the target Automation Studio project package (optional)

#### INFO

Only one single level of Automation Studio packages is supported by the Automatic Code Generation. Control Packages are not supported neither.

Add source files: Include additional source and header files (\*.c, \*.h) to the generated task (optional)

Add include directories: Include additional include directories (optional)

#### INFO

If the generated task is moved to a different development system, please make sure to adapt all include directories .

Additional compiler switches: Define additional compiler switches for the Automation Studio compilation (optional)

Version: Declare a version number for the generated task (optional)

**Encryption password:** Enter an encryption password for source files (\*.c) in Automation Studio (optional)

**Enable continuous time support:** Allow continuous time blocks in your Simulink model (not recommended for production use)

**Enable expert mode:** Enable all options to be set manually (only recommended for experienced users)

**Enable logging:** Create a log file of all warnings and errors during code generation (Simulink model name + '.log').

**Enable automatic transfer:** Automatically compile and transfer the generated task or function block to the target system (see chapter 4.4).

#### INFO

Before using the 'Automatic transfer' feature make sure that the target Automation Studio project can be compiled and transferred without any errors or warnings in Automation Studio and that the connection to the target system is established.

**Enable External Mode:** Enable the External Mode feature described in chapter 4.5.

**External Mode buffer memory size:** Buffer size for use of External Mode (see chapter 4.5). Default value: 1000000

**External Mode IP address:** IP address for External Mode (see chapter 4.5). Default value: 127.0.0.1

**Enable model specific post processing:** After finishing the code generation, a specific \*.m file could be called for a post processing routine.

**Post processing file name:** Name of the model specific post processing file. The file has to be in a MATLAB known path.

**Create global \*.var file:** Automatically generates a model specific global.var file.

## Configuration Settings

Sonfiguration Parameters:	untitled/Code Generation ERT based (Active)	×	
Select:			
Solver	Getting started		
Data Import/Export	Info		×
Optimization	1110		
Diagnostics	B&R Homepage		
Hardware Implementation			
Model Referencing			
🗄 Simulation Target			
È-Real-Time Workshop			
Report			
Comments			
Symbols			
Custom Code			
Debug			
Interface			
SIL and PIL Verification			
Code Style			
Templates			
Code Placement			
Data Type Replacement			
Memory Sections			
-B&R Basic Settings			
B&R Advanced Settings			
B&R Information			
		Ţ	
1		1	
0	QK <u>Cancel</u> <u>H</u> elp <u>A</u> pply		
-		_	1

Fig. 21: B&R Information

Getting started: Open this documentation in your pdf viewer

Info: Show version information

B&R Homepage: Link to B&R Homepage (<u>http://www.br-automation.com/</u>)

## 4. WORKING WITH B&R AUTOMATION STUDIO TARGET FOR SIMULINK

#### 4.1 Basic example

The following example clearly explains the use of the blocks introduced above and gives an introduction about the first steps in connection with *B&R Automation Studio Target for Simulink*.

#### Example:

The following introductory example illustrates, in simple steps, how an existing Simulink model is prepared for Automatic Code Generation with *B&R Automation Studio Target for Simulink*.

- Basic Simulink model
- Interface and parameter blocks
- Project and target configuration
- Debugging

An introduction to Automation Studio, MATLAB and Simulink is not included in this training module and is a prerequisite for working with the following excerpts.

#### 4.1.1 The model: A simple algebraic system

The algebraic system displayed in Fig. 22 serves as basic structure for the following implementation example. The two inputs a and b are added, multiplied by a constant factor k and copied to variable c.

c = k \* (a + b)

Because basic knowledge of the use of MATLAB and Simulink is prerequisite, the implementation of the basic model will not be discussed here.


Fig. 22: Basic Simulink model

4.1.2 Configuration settings: B&R Config block

Inserting the B&R Config block completes the first step towards Automatic Code Generation. By choosing the 'Code Generation (ERT based)' or 'Code Generation (GRT based)' config set all relevant basic settings are prepared automatically. This means that for instance a fixed step discrete solver is chosen and the simulation time is set to infinite.



Fig. 23: Adding the B&R Config block

B&R Automation Studio Target for Simulink®

TM140 TM140

# 4.1.3 Interfaces: B&R Input and Output block

In order to make the process variables accessible in Automation Studio and to allow communication with other processes in the system application the corresponding external interfaces must be defined. In the course of the Automatic Code Generation a variable is created in the target system for every B&R Input and Output block.





As the interface blocks should be accessible for other Automation Studio tasks in our example we decide to use global variables. Furthermore the process variables created on the target do not have scalar values but represent arrays of three float values (0..2) in our example.

Fig. 25: Adding a B&R Output block

In our example, variables a, b and c will have the following settings:

# Working with B&R Automation Studio Target for Simulink

a:	Variable Name:	a
	Variable Description:	first input
	Variable Scope:	LOCAL
	Variable Data Type:	LREAL
	Initial Value :	1
	Array Index :	2
b:	Variable Name:	b
	Variable Description:	second input
	Variable Scope:	LOCAL
	Variable Data Type:	LREAL
	Initial Value :	2
	Array Index :	2
c:	Variable Name:	c
	Variable Description:	first output
	Variable Scope:	LOCAL
	Variable Data Type:	LREAL
	Initial Value :	0
	Array Index :	2
1.1		

# 4.1.4 Parameter configuration: B&R Parameter block

To make factor k accessible during operation, a B&R Parameter block must be inserted.

<u>Scample</u> File Edit View Simulation Format Tools Help □   <b>彦 밑 曇</b>   美 略 噫   マーラ 介   ユ 二   ▶ =   mf    Normal    又 器 語 ② 愛 語   異 國 圖 參		
B&R CONFIG B&R CONFIG LOCAL REAL k B&R IN _OLOBAL REAL 4(0.2) B&R IN _OLOBAL REAL 4(0.2) B&R IN _OLOBAL REAL 4(0.2)	Block Parameters: Parameter         PBR - Parameter Block (mask) (mk)         Parameter interface for BRR Automation Studio process variable.         Parameter interface for BRR Automation Studio process variable.         Parameters         Variable Description         parameter         Q         OK         Cancel         Help         Apply	
Ready	125% FixedStepDiscrete	1

Fig. 26: Adding a B&R Parameter block

As described for the B&R Input and Output block, the settings Variable Description, Variable Scope, Variable Data Type, Initial Value and Array Index also apply for the B&R Parameter block:

k:	Variable Name:	k
	Variable Description:	parameter
	Variable Scope:	LOCAL
	Variable Data Type:	LREAL
	Initial Value :	10
	Array Index :	2

4.1.5 Model settings: Automation Studio project path and sample time

To allow automatic integration of code produced from the model in an existing Automation Studio project, the according project settings have to be done.

First of all the path to the corresponding Automation Studio project has to be entered as well as the Automation Studio task name and package name (optional).

# INFO

Only one single level of Automation Studio packages is supported by the Automatic Code Generation. Control Packages are not supported.

If the generated Automation Studio task should also be added to a certain hardware configuration, the option 'Add task to hardware configuration' has to be selected and the Automation Studio configuration name and PLC name have to be entered correctly.

In order to be able to handle also time continuous Simulink models (should not be used for generation of production code) the option 'Enable continuous time support' must be activated. As in this basic model there are no time continuous Simulink blocks this option is left disabled.

Enabling the expert mode allows the user to modify various additional settings and should only be activated by users who are familiar with Real-Time Workshop Embedded Coder.

The log file option can be used to record warnings and errors during code generation.

🍇 Configuration Parameters:	untitled/Code Generation ERT based (Active)
Select: Solver	Automation Studio project path C:\projects\project_1
Data Import/Export Optimization ⊕Diagnostics	Browse Automation Studio task or function block name sample
Hardware Implementation Model Referencing ⊕Simulation Target	Create zip-file
G-Real-Time Workshop	Zip-file path C:\projects
Comments	Browse
Symbols Custom Code	Add task or library to hardware configuration
Debug Interface	Automation Studio configuration name Config1
SIL and PIL Verification Code Style	Automation Studio PLC name PLC1
Templates	Create function block
Data Type Replacement	
Memory Sections 88R Basic Settings	
0	OK Cancel Help Apply

Fig. 27: B&R Automation Studio settings

# Info

The configuration name and PLC name used in the current Automation Studio Project can be easily checked as seen below.

Configura	ition		
Ξ 🕅	Simula	tion [Active]	
	🛍 Ha	rdware.hc	. t
<u> </u>	🗗 PL	C1	
	2	Cpu.sw	
	÷ 🖻	Cpu.per	
	÷ 🝻	loMap.iom	
	🖬	sysconf.br	
	🖬	sysconf.syc	
	i 😭	ArConfig.rtc	1
	<u> </u>		

B&R Automation Studio Target for Simulink®

Before starting the Automatic Code Generation routine the **correct sample time** for the Simulink model has to be set.

Configuration Parameters:	ample/Code Generation ERT based (Active)		×
ielect:	Simulation time		<b>_</b>
Solver	Start time: 0.0	Stop time: inf	
Data Import/Export	Start time: 10.0	Stop time: Tim	
Optimization	Solver options		
-Diagnostics Hardware Implementation			
Mardware Implementation Model Referencing	Type: Fixed-step	Solver: discrete (no continuous states)	7
-Simulation Target			
-Real-Time Workshop	Fixed-step size (fundamental sample time): 0.1		
Report	Tables and secols the astisse		
Comments	Tasking and sample time options		
Symbols	Periodic sample time constraint:	Unconstrained	<b>V</b>
Custom Code	Tasking mode for periodic sample times:	SingleTasking	
Debug Interface	lasking mode for periodic sample times.	Understand	
Code Style	Automatically handle rate transition for data transfer		
- Templates	Higher priority value indicates higher task priority		
Data Placement	I righer priority value indicates higher task priority		
Data Type Replacement			
Memory Sections			
			_
			<b>•</b>
			1 1
1		OK Cancel Hel	P Apply

Fig. 28: Setting the sample time

#### IMPORTANT

If the fundamental sample time specified in Simulink does not match the duration of the cyclic task class for the automatically generated task in Automation Studio, the task will be suspended.

# 4.1.6 Preparations: B&R Automation Studio libraries

In order to be able to run the automatically generated source code on the B&R target two B&R Automation Studio libraries are required in the project: **'brsystem'** and **'sys\_lib'**. If these libraries are not yet part of the project they have to be added.

Project Explorer *					
Object Name	Description				
🖃 🗁 project_1	Default project				
📄 🗄 🕂 🔁 🖬 🖬 🕂	Global data types				
🗄 🗝 🧭 Global var 🛛 Global variables					
🖃 🗁 🔁 Libraries	Global libraries				
🗼 👘 🙀 Convert	IEC 61131-3 type conversion declarations				
🗄 🛶 🥪 Runtime	Internal support library				
🗄 🙀 Operator	Built-in IEC 61131-3 standard functions				
🔅 🥪 brsystem	The BRSystem library provides the user with a number of system functions for e.g. handling permanent memory, accessing exception information, etc.				
🗼 🛷 sys_lib	The SYS_LIB library contains functions for memory management and operating system manipulation as well as hardware-specific functions.				
🗧 📲 Logical View 🏼 🐁 Configuratio	n View 🛛 🖓 Physical View				

Fig. 29: Required B&R Automation Studio libraries

At this point all preparations are complete and Automatic Code Generation can be started.

4.1.7 Integration: Automatic code generation and project download

Once the above preparations have been successfully completed, you can start the Automatic Code Generation by using the menu item **Tools**  $\rightarrow$  **Real**-**Time Workshop**  $\rightarrow$  **Build Model**... (Ctrl+B) or using the corresponding button on the toolbar.

🗋 😂 🖬 🚳 👗 🖻 💼 😓 수 수 🗠 으로 🕨	inf Normal 💌 🔛	📇 🕑 🧇 🔛 🔰 🖾 📠 🛞 👘
----------------------------	----------------	-------------------

A message will appear in the MATLAB command window indicating that the code generation was successful. Then the automatically created source code can be compiled in Automation Studio and transferred to the target system.

Object Name       Description         Diject Name       Description         Diject Name       Version         Table Lype       Global data types         Global var       For QL (EC 61131-3)         Global var       The BRSystem litres         Global var       Global var         G	roject Explorer	* X	🝠 🛝 🖾 💽 🥥 🔂 🐼 💊 🔊 🤇	*				
■ joreC1       Default project         ■ joreC1					Transfer	Size (butes)	Source	Source File Description
Bibbal var       Bibbal var       Bibbal var       Cyclic #1 · (100 ms)       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary         Bibbal var       Bibbal var       Bibbal var       Cyclic #1 · (100 ms)       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3 yee control in the Bisbal bibrary       Image: Convert Lic C i113 / 3				VOIDIOIT	Transfor	0120 (0)(00)	00000	Course the Description
Image: Single		Global data types						
Image: Source in the constraint of the cons		Clubel Bergins		1.00.0	U		l	VC
Brack       Runkme       Internal support borat         Brack       Operator       Builtin IEC 6113135         Brack       Dispator       Dispator         Brack       The SYS_LIB library				1.00.0	USEINUM	u	sample	\cpu.sw
•								
Brown Brystem     The BRSystem libray       Brown Brystem     The SRSystem libray       Brown Brystem     The SYS_LB libray       Brown Brystem     The SYS_LB libray       Brown Brystem     Cyclic #5 (2000 me)       Brown Brystem     Data Objects       Brown Brystem     Brown Brystem       Brown Brystem     Bro		Built-in IEC 61131-3 s						
Image: Stands       Image: Constraint of the stands       Image: Constraint of the stands         Image: Stands       Image: Constraint of the stands       Image: Constraint of the stands         Image: Stands       Image: Constraint of the stands       Image: Constraint of the stands         Image: Stands       Image: Constraint of the stands       Image: Constraint of the stands         Image: Stands       Image: Constraint of the stands       Image: Constraint of the stands         Image: Stands       Image: Constraint of the stands       Image: Constraint of the stands         Image: Stands       Image: Constraint of the stands       Image: Constraint of the stands         Image: Stands       Image: Constraint of the stands       Image: Constraint of the stands         Image: Stands       Image: Constraint of the stands       Image: Constraint of the stands         Image: Stands       Image: Constraint of the stands       Image: Constraint of the stands         Image: Stands       Image: Constraint of the stands       Image: Constraint of the stands         Image: Constraint of the stands       Image: Constraint of the stands       Image: Constraint of the stands         Image: Constraint of the stands       Image: Constraint of the stands       Image: Constraint of the stands         Image: Constraint of the stands       Image: Constraint of the stands       Image: Constraint of the stands								
• a j global txt               - C Cyclic #7 - [4000 ms]                 - a j global txt               - C Cyclic #7 - [4000 ms]                 - a j global txt               - C Cyclic #7 - [4000 ms]                 - a j global txt               - C Data Dipicts                 - a jh sample_private h               - G Data Dipicts                 - a jh sample_private h               - G Binay Übjects                 - a jh sample_private h               - G Binay Übjects                 - a jh sample_private h               - G Binay Übjects			📃 🖂 💭 Cyclic #6 · [3000 ms]					
global.txt     - C Cyclic H8 (5000 ms)       a h bur, globalvar.h     - D Data Objects       b h, wolypes.h     - W Nc Data Objects       a h) sample_private.h     - W Sinay Objects       - a h sample_private.h     - W Sinay Objects       - a h sample_private.h     - W Sinay Objects       - a h sample_private.h     - W Sinay Objects			🛛 🗠 🙄 Cyclic #7 · [4000 ms]					
Image: Determinant of the sample private. h     Image: Determinant of the sample private. h     Image: Determinant of the sample private. h       Image: Determinant of the sample private. h     Image: Determinant of the sample private. h     Image: Determinant of the sample private. h       Image: Determinant of the sample private. h     Image: Determinant of the sample private. h     Image: Determinant of the sample private. h       Image: Determinant of the sample private. h     Image: Determinant of the sample private. h     Image: Determinant of the sample private. h       Image: Determinant of the sample private. h     Image: Determinant of the sample private. h     Image: Determinant of the sample private. h	global.txt		📕 🛏 😂 Cyclic #8 · [5000 ms]					
In twopesh     - @ No Data Dijects       In sample, private h     - @ Wisuaisation       In sample, private h     - @ Binany Objects       In sample, private h     - @ Binany Objects       In sample, private h     - @ Binany Objects	h bur_globalvar.h		Data Objects					
In Sample_private.h     Image: Sample_private.h       Image: Sample_priste.h     Image: Sample_private.h	🛶 h) rtwtypes.h		- 🎡 Nc Data Objects					
an h sample_types h ⊕	h sample.h		🚜 Visualisation					
The sample_types h the sample_types h the sample sa	<b>h</b> sample_private.h		Binary Objects					
	sample_types.h							
+ a C sample.c	main.c							
🚽 🗃 global bur	sample.c							
		→ .						4
	have a second	Channell V	Hanna Maria Maria	N. and			hand the second	and the second

Fig. 30: Automatically integrated program code from the example program

# 4.1.8 Variables: Global and local variable files

As mentioned before local variables are automatically registered in the corresponding variable file 'local.var'. Global variable have to be registered manually. As support the automatically generated file 'global.txt' can be

used in order to declare global variables needed by the automatically generated task. The file contains all needed declarations and can be copied to the source directory of the Automation Studio project and renamed to 'global.var' if no other global variables exist in the project.



Fig. 31: Adding declaration for global variables

### IMPORTANT

When manually declaring global variables in Automation Studio, the user must make sure that the data type of the variable in the project matches the selected data type in the dialog field.

By opening the variable files for global and local process variables the correct declaration can be checked.

• a          LREAL[0.2]               1.0             first input                 • b          LREAL[0.2]               2.0             second input                 • c          LREAL[0.2]               0.0             first output	Name	Туре	& Reference	🔒 Constant	Value	Description [1]
c      LREAL[0.2]      D      0.0      first output	🤣 a	LREAL[02]			1.0	first input
	🔗 Ь	LREAL[02]			2.0	second input
e e e e e e e e e e e e e e e e e e e						
The second se					and the state	

Fig. 32: Declaration of global variables



Fig. 33: Declaration of local variables

# 4.1.9 Physical view: Hardware assignment

The assignment of the generated task to your hardware configuration can be done automatically by selecting the option 'Add task to hardware configuration'.

Automation Studio project Path C:\projects\project_1	
	Browse
Automation Studio Task Name sample	
🔽 Create zip-file	
Zip-file path	
	Browse
🔽 Add task to hardware configuration	
Automation Studio configuration name Simulation	
Automation Studio PLC name PLC1	

Fig. 34: Adding task to hardware configuration

#### INFO

The generated task is added to the first cyclic task class on the target system by default. If the task should run in a different task class, it has to be moved by drag-and-drop.

If you choose to manually add the generated task to a cyclic task class of the hardware, it is important to add the needed 'Additional build options' and the needed 'Additional include directories'. All necessary options can be found in the automatically generated text file 'task\_properties.txt'.

6	🗟 sample - Properties	<u>?</u> ×
	General Compiler Linker Dependencies Transfer Memory	
	<b>1</b> sample	
	Translate ANSI (Windows) to ASCII (DDS) characters Debugging	
	Additional build options: Im -D MODEL=sample -D NUMST=1 -D NCSTATES=0 -D HAVE	ST
	ANSI C default build options:	_
	-fPIC -00 -g -nostartfiles -Wall -x c -ansi -D _DEFAULT_INCLUDES	4
	Additional include directories:	<u>+</u>
	\Logical\MATLAB_includes\R2008a\	
	μ	
	OK Cancel App	aly

Fig. 35: Task property settings

B&R Automation Studio Target for Simulink®

# INFO

If the generated task is moved to a different development system, please make sure to adapt all absolute include directories.

# 4.1.10 Debugging: B&R Automation Studio Watch

The result of the Automatic Code Generation can be easily verified by opening the B&R Automation Studio Watch window after the download to the target.

Name	Туре	Scope	Force	Value	
🗆 🖗 a	LREAL[02]	local			
– 🏈 a[0]	LREAL			1.0	
– � a[1]	LREAL			2.0	- 4
└ 🏈 a[2]	LREAL			3.0	
🗆 🖗 Ь	LREAL[02]	local			
⊢ 🖉 Ь[0]	LREAL			2.0	-
— 🖗 Ь[1]	LREAL			2.0	- 6
∟∲ ь[2]	LREAL			2.0	
🗆 🖗 с	LREAL[02]	local			
– � c[0]	LREAL			30.0	
∲ c[1]	LREAL			40.0	
L 🖉 c[2]	LREAL			50.0	
🖗 k	LREAL	local		10.0	

Fig. 36: Automation Studio Watch window

# 4.2 Function block generation

# Example:

The following example shows the automatic generation of Automation Studio function blocks with *Automation Studio Target for Simulink*. The automatically generated Automation Studio library containing the function block is integrated into an existing project and can be used in any programming language.

Error handling can be directly implemented in Simulink (e.g. based on Embedded MATLAB functions).

The example contains an algorithm that integrates the sum of the two inputs a and b and then divides the result by d. In order to avoid division by zero an Embedded MATLAB function is included. In addition an error number is generated as soon as a division by zero would occur.



Fig. 37: Basic example for function block generation

To enable function block generation instead of task generation the 'Create function block' setting has to be activated.

Solver       -Solver         -Solver       -Data Import/Export         -Optimization       Browse         -Diagnostics       Automation Studio task or function block name [myFub         -Data Vinburg       Compatibility         -One at Vinburg       Create zip-file         Zip-file path       Ciprojects         Ziprojects       Real-file         Zip-file path       Ciprojects         Simulation       Automation Studio configu	Select:		
Optimization       Browse         □ - Sample Time       Automation Studio task or function block name [myFub         □ - Sample Time       □ Create zip-file         □ - Sample Time       □ Create zip-file         □ - Sample Time       □ Create zip-file         □ - Compatibility       □ Create zip-file         □ - Compatibility       □ Create zip-file         □ - Saving       □ Add task or library to hardware configuration         ■ Model Referencing       □ Add task or library to hardware configuration         ■ Model Referencing       □ Automation Studio PLC name         □ - Simulation Target       ■ Automation Studio PLC name         □ - Symbols       □ Create function block         □ - Comments       □ Simulation Code         □ - Symbols       □ Create function block         □ - Symbols       □ Create function block         □ - Stand PIL Verification       □ Create function block         □ - Tamplates       □ Code Style         □ - Tamplates       □ Code Style         □ - Tamplates       □ Code Style         □ - Tamplates       □ Code Strings         □ - B&R Advanced Settings       □ - Statings	Solver	Automation Studio project path C:\project_1	
→Optimization       Automation Studio task or function block name myFub         →Sample Time       Automation Studio task or function block name myFub         →Data Validity       □ Create zip-file         →Dopatbility       □ Create zip-file         →Connectivity       Zip-file path C:lprojects         →Connectivity       □ Add task or library to hardware configuration         →Hardware Implementation       →Add task or library to hardware configuration         →Hardware Implementation       Automation Studio PLC name Simulation         →Simulation Target       Automation Studio PLC name PLC1         →Symbols       Automation block         →Real-Time Workshop       I Create function block         →Report       →Comments         →Symbols       →Locatom Code         →Debug       →Interface         →Simulation       File Placement         →Debug       →Interface         →Sta Type Replacement       →B&R Basic Settings         →B&R Basic Settings       →B&R Basic Settings	Data Import/Export		1
-Sample Time       Addition Social Case of function block frame (myrdb)         -Data Validity       Create zip-file         -Connectivity       Zip-file path         -Connectivity       Zip-file path         -Connectivity       Zip-file path         -Connectivity       Zip-file path         -Conpatibility       Browse         -Saving       Image: Connectivity         -Compatibility       Add task or library to hardware configuration         -Model Referencing       Automation Studio Configuration name         -Symbols       Automation Studio PLC name         -Custom Code       Image: Connectivity         -Report       Comments         -Symbols       Custom Code         -Debug       Interface         -Interface       Studie PL Verification         -Code Style       -Templates         -Code Style       -Templates         -Code Style       -Bar Type Replacement         -Data Type Replacement       -Data Type Replacement         -Bar Advanced Settings       Bit Studie Settings	Optimization	Browse	
Sample Time       -Data Validity         -Data Validity       Create zip-file         -Connectivity       Zp-file path C:\projects         -Compatibility       Browse         -Model Referencing       Automation Studio configuration         -Saving       ✓ Add task or library to hardware configuration         -Model Referencing       Automation Studio configuration name         Simulation Target       Automation Studio configuration name         -Symbols       Automation Studio PLC name         -Costom Code       V         -Symbols       Create function block         -Real-Time Workshop       ✓ Create function block         -Real-Time Workshop       ✓ Create function block         -Symbols       -Custom Code         -Debug       -Interface         -Stal and PIL Verification       -Code Placement         -Dota Type Replacement       -Data Type Replacement         -Memory Sections       BBR Basic Settings	-Diagnostics	Automation Studio task or function block name myEub	
-Type Conversion       -Connectivity         -Compatibility       Browse         -Model Referencing       Saving         -Hardware Implementation       Implementation         -Model Referencing       Automation Studio configuration name         Simulation Target       Automation Studio configuration name         -Symbols       Automation Studio PLC name         -Coursents       -Symbols         -Coursents       -Symbols         -Custom Code       Implement         -Symbols       Implement         -Comments       -Symbols         -Custom Code       Implement         -Symbols       Implement         -Code Style       Implement         -Stan Type Replacement       -Code Styles         -Code Style       -Code Strings         -B&R Baki Settings       -B&R Advanced Settings			
-Connectivity       Zp-file path       Cityprojects         -Compatibility       Browse         -Model Referencing       Image: Cityprojects         -Simulation Target       Automation Studio configuration name         -Symbols       Automation Studio PLC name         -Custom Code       Image: Cityprojects         -Real-Time Workshop       Image: Cityprojects         -Report       -Comments         -Symbols       -Custom Code         -Debug       -Interface         -Timplates       -Code Style         -Code Style       -Code Style         -Templates       -Code Style         -BBR Basic Settings       -BBR Advanced Settings		Create zip-file	
-Compatibility       Browse         -Model Referencing       Browse         -Swing       ✓ Add task or library to hardware configuration         -Model Referencing       Automation Studio configuration name Simulation         -Simulation Target       Automation Studio PLC name PLC1         -Comments       ✓ Create function block         -Symbols       ✓ Create function block         -Courter face       -SIL and PIL Verification         -Code Placement       -Deta Type Replacement         -Memory Sections       BBR Basic Settings         -BBR Basic Settings       -BBR Basic Settings			
Model Referencing       Browse        Saving      Madel Referencing        Hardware Implementation       Automation studio configuration name        Symbols		Zip-file path C:\projects	
Image: Referencing       Image: Referencing         Swing       Add task or library to hardware configuration         Model Referencing       Automation Studio configuration name         Symbols       Automation Studio PLC name         Custom Code       Automation Studio PLC name         Real-Time Workshop       Image: Referencing         Report       Comments         Symbols       Image: Referencing         Custom Code       Debug         Interface       Interface         Stude APIL Wrification       Code Style         Templates       Code Relearent         Data Type Replacement       Data Type Replacement         BBR Basic Settings       BBR Basic Settings		Province	ł.
Hardware Implementation     Model Referencing       Simulation Target     Automation Studio configuration name       Symbols     Automation Studio PLC name       Custom Code     Real-Time Workshop       Real-Time Workshop     Image: Create function block       Comments     Symbols       -Symbols     Create function block       Comments     Symbols       -Symbols     Custom Code       Debug     Thterface       -StL and PLL wrification     Code Style       -Code Style     Templates       -Code Placement     Odd Placement       -Data Type Replacement     Memory Sections       -B&R Basic Settings     B&R Basic Settings		browse	I.
Hardware Implementation       Automation Studio configuration name       Simulation         →Model Referencing       Automation Studio configuration name       Simulation         ⊖Symbols       Automation Studio PLC name       PLC1         ⊂Custom Code       Comments       Symbols         −Symbols       ✓       Create function block         −Comments       −       Symbols         −Custom Code       −       Debug         −Interface       −       −         −St. and PIL Verification       −       Code Style         − Templates       −       −         − Ode Accement       −       −         − BBR Basic Settings       −         −BBR Basic Settings       −		Add task or library to hardware configuration	
B-Simulation Target Symbols -Custom Code Regort -Comments -Symbols -Custom Code -Debug -Interface -SIL and PIL Verification -Code Style -Templates -Code Placement -Data Type Replacement -B&R Basic Settings -B&R Advanced Settings -B&R Advanced Settings -B&R Advanced Settings -B&R Advanced Settings -B&R Advanced Settings -B&R Advanced Settings -Symbols -Code Style -B&R Advanced Settings -B&R Setings -B&R Settings -B&R Settings -B&R Se			
Symbols     Custom Code     Real-Time Workshop     Report     Comments     Symbols     Custom Code     Debug     Interface     Sit, and PIL wrification     Code Style     Templates     Templates     Code Style     Templates     Code Style     Templates     Code Style     Templates     Style     S		Automation Studio configuration name Simulation	
Custom Code     Real-Time Workshop     Report     Comments     Symbols     Custom Code     Debug     Debug     Interface     SIL and PIL Verification     Code Style     Templates     Code Placement     Data Type Replacement     Memory Sections     B&R Basic Settings     BBR Advanced Settings			
Real-Time Workshop     Report     Comments     Symbols     Custom Code     Debug     Interface     Style     Code Style     Code Style     Templates     Code Placement     Data Type Replacement     Memory Sections     BBR Basic Settings     BBR Advanced Settings		Automation Studio PLC name   PLC1	
Reprit     Comments     Symbols     Custom Code     Debug     Interface     Still and PIL verification     Code Style     Templates     Code Placement     Dedu Type Replacement     Memory Sections     B&R Basic Settings     B&R Advanced Settings		Create function block	
-Comments       -Symbols       -Custom Code       -Debug       -Interface       -SIL and PIL Verification       -Code Style       -Templates       -Code Placement       -Data Type Replacement       -B&R Basic Settings       -B&R Advanced Settings			
-Symbols -Custom Code -Debug -Interface -SIL and PIL Verification -Code Style -Templates -Code Placement -Data Type Replacement -Memory Sections -B&R Basic Settings -B&R Advanced Settings			
-Custom Code       -Debug       -Interface       -SIL and PIL Verification       -Code Style       -Templates       -Code Placement       -Data Type Replacement       -B&R Basic Settings       -B&R Advanced Settings			
-Debug       -Interface       -SIL and PIL Verification       -Code Style       -Templates       -Code Placement       -Data Type Replacement       -Memory Sections       -B&R Basic Settings       -B&R Advanced Settings			
-Interface -SIL and PIL Verification -Code Style -Templates -Code Placement -Data Type Replacement -Memory Sections -B&R Basic Settings -B&R Advanced Settings			
-SIL and PIL Verification -Code Style -Templates -Code Placement -Data Type Replacement -Memory Sections -B&R Basic Settings -B&R Advanced Settings			
Code Style Templates Code Placement Data Type Replacement Memory Sections B&R Basic Settings B&R Advanced Settings			
Templates Code Placement Data Type Replacement Memory Sections 			
Code Placement Data Type Replacement Memory Sections B&R Basic Settings B&R Advanced Settings			
Data Type Replacement Memory Sections B&R Basic Settings B&R Advanced Settings			
Memory Sections    B&R Basic Settings			
—B&R Basic Settings     —B&R Advanced Settings			
B&R Advanced Settings			
LB&R Information			
OK Cancel Help A		OK Cancel Help Appl	

Fig. 38: Enable 'Create function block' setting

 $\mathcal{O}$ 

B&R Automation Studio Target for Simulink®

The automatic code generation process then automatically generates an Automation Studio library containing the corresponding function block.



Fig. 39: Automatically generated Automation Studio library

The automatically generated function block can be used in any Automation Studio task.

0001	(**************************************
0002	* COPYRIGHT Bernecker + Rainer
0003	***************************************
0004	* Program: test_fub
0005	* File: test_fub.st
0006	* Author: B4R
0007	***************************************
0008	* Implementation of program test_fub
0009	***************************************
0010	
0011	PROGRAM _INIT
0012	(* call method is 'initialize' (init part) *)
0013	<pre>myFub_1.ssMethodType := SS_INITIALIZE;</pre>
0014	(* call FUNCTION block *)
0015	myFub_1();
0016	END_PROGRAM
0017	
0018	
0019	PROGRAM _CYCLIC
0020	(* call method is 'output' (cyclic part) *)
0021	<pre>myFub_1.ssMethodType := SS_OUTPUT;</pre>
0022	(* call FUNCTION block *)
0023	myFub_1();
	END_PROGRAM
0025	
	PROGRAM _EXIT
0027	(* call method is 'terminate' (exit part) *)
0028	<pre>myFub_1.ssMethodType := SS_TERMINATE;</pre>
0029	(* call FUNCTION block *)
0030	myFub_1();
0031	RND_PROGRAM

Fig. 40: Automation Studio task (Structured Text) calling the automatically generated function block

The 'ssMethodType' option is automatically added to the function block structure. It determines the current function call step. SS\_INITIALIZE ... initialize function SS\_OUTPUT ... cyclic update and output function SS\_TERMINATE ... exit function

# 4.3 Structure interface blocks

# Example:

The following example shows the use of the structure interface blocks. To use structure variables for automatic code generation a corresponding Automation Studio type file (\*.TYP) has to be created. The type file can then be imported into Simulink and be used for the B&R Structure Input and B&R Structure Output blocks.

In the first step a type file has to be exported from Automation Studio.

#### IMPORTANT

Type files should not be modified or deleted anymore after the B&R Structure blocks have been inserted in order to avoid inconsistencies.

<b>Fi</b> lcial			
	bal.typ [Data Type Declaration	] ×	
Name		Туре	& Reference
	* COPYRIGHT Bernecker + Ra * File: Global.typ * Author: wallnerp * Created: May 21, 2010 * Global data types of project proj	*****	
Ξ 🕇	myInputType		
	< in_REAL	REAL	
	🧼 in_INT	INT[02]	
	🔗 in_BOOL	BOOL	
Ξ 👖	myOutputType		
	🤣 out_REAL	REAL	
	🤣 out_INT	INT[02]	
	🧼 out_BOOL	BOOL	

Fig. 41: Automation Studio type file

# IMPORTANT

Large type files can significantly slow down the performance of the B&R Structure interface blocks. Try to divide large type files into smaller pieces.

B&R Automation Studio Target for Simulink®



Fig. 42: Basic sample model with structure interfaces

In Simulink the type file is imported on the 'B&R Advanced Settings' tab.

🍇 Configuration Parameters: sa	mple/Code Generation ERT based (Active)
Select: Solver	Automation Studio package name
Data Import/Export Optimization	Add TYP files
	Add source files
Model Referencing	Add include directories
⊡-Real-Time Workshop ⊡-Report	Additional compiler switches
Comments Symbols Custom Code Debug	Version Encyption password
	Enable continous time support     Enable expert mode     Enable logging
Data Type Replacement Memory Sections B&R Basic Settings 	Enable automatic transfer     Enable External Mode     External Mode buffer memory size 1000000
B&R Information	External Mode IP address 127.0.0.1
0	OK Cancel Help Apply

Fig. 43: Add Automation Studio type files

# Working with B&R Automation Studio Target for Simulink

🛃 Add TYP files		
C:\projects\project_1\Logical\Globaltyp Add		
Delete		
		ľ
OK	ncel	

Fig. 44: Import Automation Studio type files

As soon as at least one valid type file has been added to the model, structure elements can be selected in the B&R Structure interface mask.

	📣 B&R Structure Block	
	Variable Name	
	Input	
	Variable Scope	
	LOCAL	*
	Type File	
	C:\projects\project_1\Logical\Global.typ	*
	Structure Name	
	myInputType	-
	Invition the	
	Structure Element	
	in_REAL	-
	Variable Data Type	
	REAL	
	Array Dimension (0Scalar)	
	p	_
	Ok Cancel	
$\boldsymbol{\boldsymbol{\wedge}}$		

Fig. 45: B&R Structure interface mask

The generated source code contains the interface to the according structures selected in the block mask.

Name	Туре	Scope	Force	Value
🖻 🍭 Input	myInputType	local		
⊢∲ in_REAL	REAL			1.0
中� in_INT	INT[02]			
	INT			0
	INT			0
└└� in_INT[2]	INT			0
└ � in_BOOL	BOOL			FALSE
🗄 🖗 Output	myOutputType	local		
⊢ � out_REAL	REAL			2.0
Ė- � out_INT	INT[02]			
│	INT			0
│	INT			0
└ � out_INT[2]	INT			0
L 🔗 out_BOOL	BOOL			FALSE

Fig. 46: Structure interface in Automation Studio

The type file itself is not automatically copied to the Automation Studio project by default. However, the 'Add source files' tab can be used to copy the file to the project without user interaction.

🍇 Configuration Parameters: sar	nple/Code Generation ERT based (Active)	×
Select:	Automation Studio package name	1
Data Import/Export Optimization	Add TYP files	
⊕-Diagnostics	Add source files	
Hardware Implementation Model Referencing	Add include directories	
	Additional compiler switches	
Symbols Custom Code Debug	Encyption password	
Interface SIL and PIL Verification	Enable continous time support     Enable expert mode	
Code Style Templates	T Enable logging	
Code Placement Data Type Replacement	Enable automatic transfer	
Memory Sections B&R Basic Settings	Enable External Mode	
B&R Advanced Settings	External Mode buffer memory size 1000000	
	External Mode IP address 127.0.0.1	
<b>U</b>	OK Cancel Help Apply	

Fig. 47: Automatically copy type files to the Automation Studio project

### 4.4 Automatic transfer

With the 'automatic transfer' option being enabled the generated program is included into the Automation Studio project, the entire project is compiled and then transferred to the target system automatically.

### INFO

In order to be able to use the automatic transfer feature the automatically generated program has to be assigned to hardware configuration on the target system (see chapter 3).

Configuration Parameters: sa	mple/Code Generation ERT based (Active)	
lect: Solver	Automation Studio package name	
"Data Import/Export "Optimization	Add TYP files	
Diagnostics Hardware Implementation	Add source files	
Model Referencing Simulation Target	Add include directories	
Real-Time Workshop Report Comments	Additional compiler switches Version	
Symbols Custom Code	Encyption password	
Debug Interface	Enable continous time support	
SIL and PIL Verification Code Style	Enable expert mode	
Templates Code Placement	Enable automatic transfer	
Data Type Replacement Memory Sections	Enable External Mode	
	External Mode buffer memory size 1000000	
B&R Information	External Mode IP address 127.0.0.1	
)	OK Cancel Help A	Apply

Fig. 48: Automatic transfer setting

	### Starting Real-Time Workshop build procedure for model: sample
	### Successful completion of Real-Time Workshop build procedure for model: sample
	Building project project_1, configuration Config1
	Analyzing project
	Generating header files
	Compiling C:/projects/project_1/Logical/sample/main.c
	Compiling C:/projects/project_1/Logical/sample/sample.c
	Linking C:/projects/project_1/Temp/Objects/Config1/PLC1/sample/a.out
	Building program "sample" as "sample"
	Building configuration object "iomap"
	No relevant changes.
	Generating transfer list
	Generating PVI Transfer Tool list C:\projects\project_1\Binaries\Config1\PLC1\Transfer.pil
	Build: O error(s), O warning(s)
	* Connecting (Device: "/IF=tcpip /LOPO=11159 /SA=1", Connection: "/DAIP=127.0.0.1 /CKDA=0 /REPO=11160 /ANSL=1" )
	* Connecting to AROD/V3.06 0K.
	* Connecting to WRODO/V3.06 CK. * Transferring sample (UserROM, Vers: V1.00, 01.02.2011, 648 Byte, Path: C:\projects\project 1\Binaries\Config1\PLC1\)
	* Transferring sample (bserkon, vers: v1.00, 01.02.2011, 646 byte, Path: C:\projects\project_1(binaries(config(Path)) * Transferring sample ok
	>> Hanslering sample ok
÷	<i>"</i>

#### Fig. 49: Compilation and download to the target system in MATLAB

# 4.5 External Mode

The External Mode feature allows the developer to connect to the target and debug automatically generated programs directly from Simulink. Therefore values on the target system can be directly shown in Simulink (e.g. using a 'Scope' or a 'Dsplay' block) and parameters on the target system can be modified from Simulink (e.g. 'Gain' or 'Constant' blocks).

#### INFO

When checking the External Mode option additional source code will be generated and run on the target system. Therefore using External Mode is not recommended for generating production code.

For further information see the corresponding chapters in the MathWorks Real-Time Workshop documentation.

To be able to use the External Mode feature the according option has to be checked on the settings page. The buffer memory size and IP address can also be set in the 'B&R Advanced Settings' section.

🍇 Configuration Parameters: sa	mple/Code Generation ERT based (Active)
Select:	Automation Studio package name
Data Import/Export Optimization	Add TYP files
Diagnostics     Hardware Implementation	Add source files
Model Referencing	Add include directories
⊕-Simulation Target ⊡-Real-Time Workshop	Additional compiler switches
Report Comments	Version
Symbols Custom Code	Encyption password
Debug Interface	Enable continous time support
····SIL and PIL Verification ····Code Style	Enable expert mode
···· Templates ···· Code Placement	Enable logging
Data Type Replacement	Enable automatic transfer
····Memory Sections ····B&R Basic Settings ····B&R Advanced Settings	Enable External Mode External Mode buffer memory size 1000000
<sup>i</sup> B&R Information	External Mode IP address 127.0.0.1
	<u> </u>
0	OK Cancel Help Apply

Fig. 50: External Mode settings

After the download of the generated program the values and parameters can directly be accessed from Simulink as soon as the connection has been established (see External Mode section in the MathWorks Real-Time Workshop documentation).

🙀 sample				- D ×	
File Edit View Simulation Format					
🗋 🛛 🖉 🖬 🎒 🕹 😽 🕞 🛛	->{  <u></u>	🔳 👿 İnf	External 💌	🎦 🛗	
B&R CONFIG	● 1 Gain	↓	cope B&R OUT _LOCAL LREAL b		
			1 Display		
External	100%	T=31.200	FixedStepDiscrete	11.	

Fig. 51: External Mode connection in Simulink

Block parameters on the target system can only be changed during execution if the 'Inline parmeters' option is disabled.

🍇 Configuration Parameters: sa	mple/Code Generation ERT based (Active)		×
Select:	Simulation and code generation		_ <u> </u>
Solver Data Import/Export	₩ Block reduction	Conditional input branch execution	
Optimization	🔽 Implement logic signals as Boolean data (vs. double) 🔽	Signal storage reuse	
Diagnostics     Hardware Implementation	Inline parameters	Configure	
Model Referencing ⊕Simulation Target	Application lifespan (days) inf		
⊡-Real-Time Workshop Report	Use integer division to handle net slopes that are reciproc	als of integers	
Comments Symbols	Code generation		
Custom Code	Parameter structure: NonHierarchical		
Debug Interface	C Optimize using the specified minimum and maximum values	5	
SIL and PIL Verification	_Signals		
Code Style Templates	Fable local block outputs	Reuse block outputs	
Code Placement Data Type Replacement	Eliminate superfluous local variables (expression folding	) 🔽 Inline invariant signals	
Memory Sections	Minimize data copies between local and global variables	🗖 Simplify array indexing	
	☐ Use memcpy for vector assignment		
B&R Information	Pack Boolean data into bitfields		
0		OK Cancel	Help Apply

Fig. 52: Inline parameters option

# INFO

For use of the External Mode feature the Automation Studio libraries 'AsArLog' and 'AsTCP' have to be part of the project.

# 5. EXAMPLES

The following examples provide a small overview of the extensive possibilities for utilizing *B&R* Automation Studio Target for Simulink in the field of automation technology. In the first example the fast and easy implementation of a simple **discrete-time PID controller** is demonstrated. In the second part a Hardware-in-the-Loop application representing a **simulation model of a temperature system** is realized using *B&R* Automation Studio Target for Simulink. In the last example a second Hardware-in-the-Loop system modelling a hydraulic valve and cylinder is shown.

# 5.1 PID controller

#### Example:

Using Simulink it is easy to implement a simple PID controller. After the control deviation has been calculated from the set and actual values, the equations listed below are used to calculate the manipulated variable directly on the controller's output. All that is needed to install the controller on the target system is to add the B&R blocks described before and start the Automatic Code Generation.



Fig. 53: PID controller

The control concept for the PID controller is:

 $Y_p = K_p \cdot (W - X)$ ... Proportional element $Y_I = \frac{K_p}{T_n} \cdot \int (W - X) \cdot dt$ ... Integral element $Y_D = K_p \cdot T_v \cdot \frac{d}{dt}(W - X)$ ... Differential element $Y = Y_p + Y_I + Y_D$ ... Entire manipulated variable

Since the controller code is executed on the target system in equidistant time cycles, it is recommended to ensure that all integrator and differentiator blocks are also discrete-time.

B&R Automation Studio Target for Simulink®

## 5.2 Temperature model

In order to properly test the controller created in section 5.1 without having to have a real system at hand, a simplified model of a temperature system can be created and transferred to the target system using *B&R Automation Studio Target for Simulink* in only a few steps.

# Example:

The system is based on the following mathematical model:

$$G(s) = \frac{\widetilde{\vartheta}(s)}{y(s)} = \frac{k_s}{(1+s \cdot T_1) \cdot (1+s \cdot T_2)} \cdot e^{-s \cdot T_{tu}}$$

The simulation model is enhanced with a white noise block representing the measurement process as well as the quantization to tenths of a degree by the sensor. All system parameters - like the ambient temperature for instance - are accessible as local parameters.



Fig. 54: Temperature System

Since the simulation model is a continuous-time model, support for continuous-time systems has to be enabled.

# Examples

🍇 Configuration Parameters: sa	ample/Code Generation ERT based (Active)
Select: Solver	Automation Studio package name
Data Import/Export Optimization	Add TYP files
⊕-Diagnostics	Add source files
Hardware Implementation	Had source mes
Model Referencing	Add include directories
i⊡-Simulation Target ⊡-Real-Time Workshop	
Report	Additional compiler switches
Comments	Version
Symbols	
Custom Code	Encyption password
Debug Interface	✓ Enable continous time support
	Enable expert mode
Code Style	
···· Templates	Enable logging
Code Placement	Enable automatic transfer
Data Type Replacement Memory Sections	E Foable External Mode
	External Mode buffer memory size 1000000
	External Mode IP address 127.0.0.1
0	OK Cancel Help Apply

Fig. 55: Settings for continuous-time Simulink models

In order to be able to run the continuous-time system on the target system with fixed equidistant steps, a fixed step solver (e.g. ode1 - Euler) must be selected.

🍇 Configuration Para	meters: Sim_Temperature/Code Generation (Active)			×
Select:	Simulation time			
Solver	Start time: 0.0	Stop time: inf		
Data Import/Export Optimization	,			
⊕-Diagnostics	Solver options			
-Hardware Implement	ation Type: Fixed-step	Solver: ode1 (Euler)		-
Model Referencing	Fixed-step size (fundamental sample time): 10			
-Report	The step size (randamental sample and).			
Comments	Tasking and sample time options			
Symbols	Periodic sample time constraint:	Unconstrained		-
Custom Code Debug	Tasking mode for periodic sample times:	SingleTasking	Constrain model's discrete sample time	es to specified values
Interface	Automatically handle rate transition for data transfe			
Code Style	<ul> <li>Higher priority value indicates higher task priority</li> </ul>			
Templates Data Placement	I Higher priority value indicates higher task priority			
-Data Type Repla	cement			
Memory Section				
Dok Information				
			OK Cancel Help	Apply
Fig. 56: Fixed	step solver			
-15. 50. Tixed				
<b>—</b>				
b-				

B&R Automation Studio Target for Simulink®

Or, alternatively, the system can be converted to an adequate discrete-time system. This can be done either manually using transformations that are described in detail in the corresponding literature, or by using the 'Model Discretizer' provided with the Simulink toolboxes *Control System Toolbox* and *Simulink<sup>®</sup> Control Design*. It can be found under **Tools**  $\rightarrow$  **Control Design**  $\rightarrow$  **Model Discretizer** (see section 6.1).



Fig. 57: Model Discretizer

#### 5.3 Hydraulics applications

In Fig. 58 a Simulink model of a linear hydraulic actuator is depicted. It consists of the hydraulic servo valve and the hydraulic cylinder. The servo valve is modelled via a nonlinear curve, describing the dependence of its opening with respect to the voltage, and its non-linear hydraulic resistance, i.e. the relationship between pressure drop across the valve and volume flow. The hydraulic cylinder is described via four differential equations, two for the pressure build-up in the two cylinder chambers and two for the mechanical movement. For the implementation of the respective equations Embedded MATLAB Function blocks are used. The model includes friction and leakage of cylinder and valve. In addition the cylinder has two end positions with modelled damping. The differential equations are discretized, thus for simulation and code generation a discrete solver can be used. The inputs to the model are the valve voltage and an external load force acting on the cylinder. In addition the model has many parameters, e.g. for the geometric dimensions of the cylinder, the leakage and friction coefficients and the nominal values of the valve, which are accessible via parameter blocks. The model outputs are the states of the system, i.e. the two chamber pressures, the cylinder position and speed. In addition the sensors are modelled in so far, as the physical values of the signals are scaled to the corresponding sensor outputs.

The Simulink model represents a typical valve controlled hydraulic drive application, and is used for hardware-in-the-loop tests of hydraulic controllers and hydraulic control trainings.



Fig. 58: Application of a hydraulic drive including servo valve and cylinder

B&R Automation Studio Target for Simulink®

TM140 TM140

# 5.4 Inverted Pendulum on a Cart Model

In this section the model of an inverted pendulum is considered and linear as well as non-linear closed-loop controllers are designed with the help of MATLAB. The design is verified using simulation in Simulink and based on these simulations the realtime code is generated for hardware-in-the-loop testing and rapid prototyping.



Fig. 59: Application sketch of the inverted pendulum

Fig. 59 shows a sketch of an inverted-pendulum-on-a-cart model. It consists of a cart of mass M which is driven by a motor generating an input force F acting on the cart. A pendulum of length l can rotate freely about the point P of the cart. At the tip of the pendulum a mass m is mounted. The position of the cart is denoted by x, the displacement angle of the pendulum by  $\theta$  (with respect to the vertical position).

Typical benchmark control problems for this model are

- the closed loop control of the vertical (unstable) position of the pendulum,
- the swing up of the pendulum,
- the positioning of the cart with the attached pendulum.

The equations of motion have the form

$$M \cdot \frac{d^2}{dt^2} x + m \cdot \frac{d^2}{dt^2} (x + l \cdot \sin \Theta) = F,$$
  
$$m \cdot \frac{d^2}{dt^2} (x + l \cdot \sin \Theta) \cdot l \cdot \cos \Theta - m \cdot \frac{d^2}{dt^2} (l \cdot \cos \Theta) \cdot l \cdot \sin \Theta = m \cdot g \cdot l \cdot \sin \Theta.$$

with

 $\dot{x} = v, \ \dot{\Theta} = \omega$ 

the non-linear system can be described as a system of explict differential equations

$$\dot{x} = v,$$
  

$$\dot{v} = \frac{F \cdot \cos \Theta - (M + m) \cdot g \cdot \sin \Theta + m \cdot l \cdot \cos \Theta \cdot \sin \Theta \cdot \omega}{m \cdot l \cdot \cos^2 \Theta - (M + m) \cdot l}$$
  

$$\dot{\Theta} = \omega,$$
  

$$\dot{\omega} = \frac{F + m \cdot l \cdot \sin \Theta \cdot \omega^2 - m \cdot g \cdot \cos \Theta \cdot \sin \Theta}{M + m \cdot \sin^2 \Theta}.$$

Please refer to the Simulink file 'pend\_mod\_nlin.mdl' for a simulation of the non-linear model without controller.

5.4.1 Linear controller design for the linearized pendulum system

In order to design a controller for the unstable (upper) equilibrium position of the pendulum the linearized equations of motion are of interest. The linearization at the equilibrium point

 $\Theta = 0, \ \omega = 0, \ x = 0, \ v = 0$ 

results in a model of the form

 $\Delta \mathbf{x} = \mathbf{A} \Delta \mathbf{x} + \mathbf{b} \Delta u$  $\Delta \mathbf{y} = \mathbf{C} \Delta \mathbf{x} + \mathbf{d} \Delta u$ 

with

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{mg}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{g}{l} & 0 \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ -\frac{1}{Ml} \end{bmatrix}, \ \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \ \mathbf{d} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Because this is the linearization about the unstable equilibrium position (pendulum up), the Eigenvalues of the linear model are unstable.

As the controller is intended to run in discrete time, the model is discretized using the MATLAB command 'c2d' to derive the system matrices of a discrete time system of the form

 $\Delta \mathbf{x}_{k+1} = \mathbf{A}_{disc} \Delta \mathbf{x}_k + \mathbf{b}_{disc} \Delta u_k$  $\Delta \mathbf{y}_k = \mathbf{C} \Delta \mathbf{x}_k + \mathbf{d} \Delta u_k .$ 

For this discrete time linear model a linear state regulator of the form

 $\Delta u = \mathbf{k}^T \Delta \mathbf{x} + l \Delta r$ 

is designed.

Using the MATLAB command 'acker' the *Formula of Ackermann* is applied to directly assign the desired Eigenvalues to the closed loop system.

With the *Linear Quadratic Regulator* design (MATLAB command 'lqr') an objective function is minimized to derive the optimal controller in the sense of the provided weights.

The MATLAB script 'init\_pend\_ctrllin.m' does the mentioned derivations and initializes the respective simulation parameters. The Simulink model 'pend\_ctrl\_lin.mdl' shows a simulation of controller and linearized model. Fig. 60 shows the simulation model and Fig. 61 a step response of the controlled pendulum on a cart. The initial condition of the pendulum is set to  $\Theta = 0.01 rad$  and at t = 4 s a reference step of 0.1m is applied.



Fig. 60: Simulation model of pendulum and controller



Fig. 61: Step response of pendulum and controller

Remark: For some applications the controller is directly calculated for the continuous time linear model although it runs in discrete time. However, this can lead to undesired behaviour of the controlled system, especially when the sample time is increased.

# 5.4.2 The inverted-pendulum-on-a-cart model library

The library 'pend\_lib.mdl' (see Fig. 62) represents a collection of different modeling examples for the linear as well as for the non-linear inverted pendulum model to visualize how the same modeling goal can be reached using Simulink with different modeling approaches.

In addition, the model 'inverted\_pendulum\_IOs does not provide a physical model, but the inputs and outputs of the controller.

The configurable subsystem 'Inverted Pendulum (Simulation / IOs)' incorporates all different modeling blocks and provides the user with the possibility to select on demand which model this subsystem represents. In the following sections it will be shown, how with the help of this configurable subsystem the same simulation model which is used for simulating the controller with model in Simulink can easily be used for codegeneration.

Similar facts hold for the section 'Inputs' and 'Controller'. Again a configurable subsystem incorporates two different forms of inputs and controllers, respectively, which can be selected within the simulation model on demand.



Library:pend_lib ≥ Edit View Format Help )				
Inverted Pendulum - Linearized Mod		inverted_pendulum_lin_Es	Inverted Pendulum - IOs:	Inverted Pendulum - Configurable Block
Inverted Pendulum - (Full) Nonlines	rr Models:	inverted_pendulum_catt	inverted_pendulum_IOs	Inverted Pendulum (Simulation / IQs)
Reference Input: ref > Reference step	ref > B&R reference input	Template ref > Reference Input		
Inverted Pendulum - Controller Bloc Closed Loop Controller State Feedback Controller (0%s)	Closed Loop Controller State Feedback Controller (embm)	Closed Loop Controller State Feedback Controller (UCs)	>ref >state Configurable Controller Subsystem	
dy				100% Unlocked

Fig. 62: Inverted Pendulum library showing different modeling features and configurable subsystems

Fig. 63 shows the Simulink model 'pend\_ctrl\_lin.mdl' using the configurable blocks 'Inverted Pendulum (Simulation / IOs)' and 'Reference Input'.



Fig. 63: Simulink model with configureable subsystems 'Reference Input' and 'Inverted Pendulum'

# 5.4.3 Code generation of the state feedback controller

To prepare the model for code generation of the tested controller, in the 'B&R Config' block the corresponding 'Code Generation' choice has to be made.



Fig. 64: 'B&R Config' dialog

As the Simulink model 'pend\_ctrl\_lin.mdl' uses the configurable blocks 'Inverted Pendulum (Simulation/IOs)' and 'Reference Input', with the control 'Block Choice' in the context menu these two blocks can be configured to represent the desired choice. With the choice 'BR reference input' for the block 'Reference Input' and 'inverted pendulum IOs' for the block 'Inverted Pendulum' the entire simulation is configured in such a way, that just the code of the controller is left and the interface to the other system parts (like reference input and plan) is given with the respective 'B&R Input' and 'B&R Output' blocks.

# 5.4.4 Hardware-in-the-Loop Simulation

For hardware-in-the-loop simulations of the controller against a simulation model running also on a target system (see Section 1.2.2) it is necessary to generate real-time code from the simulation model. The respective setup is prepared in the model 'pend\_mod\_nlin.mdl' in which the inputs and outputs of the (exact, non-linear) simulation model are connected to 'B&R Inputs' and 'B&R Outputs' respectively.

By setting in the 'B&R Config' block the option to 'Code generation' and adding the desired values in the 'Real Time Workshop  $\rightarrow$  Options' dialog this model can be generated and represents a task in an Automation Studio project.

The sample Automation Studio project 'PenPrj' contains the task 'pend\_mod' which represents the simulation model of the pendulum and the task 'pend\_ctr' which contains the controller in the two packages 'Controller' and 'Model' (see Fig. 65). The global variables are used for interfacing these two tasks when running on the target system.

🎦 🖆 🖆 🚰 🎒 👗 🖹 隆 🏩 🗠 👓 🗙 gical View	<b>▼</b> ₽ ×	1	pvm [Watch]	Contraction of the local distance of the loc	and the second s		tel tskoctel tsk st [Steu	tured Text]	
) 🔟 🖂 🖪 🕢 🗟 🌢 象 🔶		House Contraction Contraction		and the second states	.omgaraci		auTenturent Tenta	concertexel 1	
bject Name	Desc	Object Name			Transfer	Size (bytes)	Source	Source File	Descrit
) 🗁 PendPrj	Defau		J	1.0101011	Transford	0.00 (0,000)	0.04100	000000710	10000in
🗄 🚽 🖬 Global typ	Globa		Cyclic #1 - [10 ms]	-					
🖬 🛛 🐼 Global.var	Globa		# pend_ctr	1.00.0	UserROM	2560	Controller.pend ctr	\Cpu.sw	
F,	Car fc		# pend mod	1.00.0	UserROM		Model.pend mod	\Cpu.sw	
er er	Refer Invert	0	Cyclic #2 - [200 ms]	a secolation of					1
	Invert		Cyclic #3 - [500 ms]	1					
🖅 🗁 Libraries	Globa	- 0	Cyclic #4 - [1000 ms]						1 1
🗄 😇 Model	Pendi	- 0	Cyclic #5 - [2000 ms]						
🗄 🖻 pend_mod		- 0	Cyclic #6 - [3000 ms]						
🚊 😳 Controller	Pendi	- 0	Cyclic #7 - [4000 ms]						
庄 🚽 pend_ctr		- 0	Cyclic #8 - [5000 ms]						1 11
i± 🔁 ctrl_tsk ∵ 🦳 MATLAB includes	Task Defau	<b>b</b> − <b>0</b>	Data Objects						
	Derau		Nc Data Objects						
		🚨	Visualisation						
			Binary Objects						1 11
		<u>⊕</u> (2)	Library Objects						1
	•	÷ 😭	Configuration Objects						
Logical View 🏽 🍓 Configuration View 🛛 🏹 Physical '	View	4		- M					Þ
tput									

Fig. 65: Automation Studio project with controller and model tasks

Via the Automation Studio diagnostic tools like 'Watch' and 'Trace' the correct operation of the controller which controls the upper (unstable) equilibrium position of the pendulum can be verified. Fig. 66 shows a step response of the setup to a reference step of 0.1m.



Fig. 66: Step response in Automation Studio Trace

# 5.4.5 Code generation of Function blocks

Another feature of *B&R Automation Studio Target for Simulink* is the generation of function blocks for Automation Studio (see chapter 4.2). In the sequel it is shown, how a function block can be designed and how the generation of function block error codes can be included in the simulation model.

Fig. 67 shows the model ,pend\_ctrl\_fun.mdl' which contains the Embedded MATLAB function 'State feedback controller with error handling'. Besides the interface variables 'ref', 'u' and 'x' this block has several additional inputs used for the parameterization of the corresponding limits. If one of these limits is exceeded the block responds either with a warning (control is continued) or an error (controller output u is set to zero), respectively. This behaviour can, of course, be simulated in Simulink.

By switching the 'B&R Config' parameters to 'Code Generation', selecting the 'B&R reference input' and the 'inverted\_pendulum\_IOs' option in the two configurable subsystems and adapting the desired options in the 'Real Time Workshop  $\rightarrow$  Options' dialog a function block can be generated having the interface depicted in Fig. 68.



Fig. 67: Simulink model 'pend\_ctrl\_fun.mdl'

B&R Automation Studio Target for Simulink®



Fig. 68: Automation Studio project with the generated function block

By using an instance of the function block 'pctr\_fun' in the task 'ctrl\_tsk', connecting inputs and outputs of the instance to the plant signals and to the reference signal, again plant and controller can be realized in a 'hardware-in-the-loop' setup.

```
PROGRAM _INIT
ctrlFUB.ssMethodType := SS_INITIALIZE;
ctrlFUB();
END_PROGRAM
PROGRAM _CYCLIC
ctrlFUB.ref := ref;
ctrlFUB.x[0] := x[0];
ctrlFUB.x[1] := x[1];
ctrlFUB.x[2] := x[2];
ctrlFUB.x[3] := x[3];
ctrlFUB.ssMethodType := SS_OUTPUT;
```

Fig. 69: Structured Text task using the automatically generated function block

ect Nam	ne	Version	Transfer	Size (bytes)	Source	Source File	Description
😰 🖸	PU						
ġ <b>(</b>	🖸 Cyclic #1 - [10 ms]						
	🖅 pend_ctr	1.00.0	UserROM	0	Controller.pend_ctr	\Cpu.sw	Pendulum Controller
	🖅 ctrl_tsk	1.00.0	UserROM	1120	Controller.ctrl_tsk	\Cpu.sw	Task running the controller FUB
	🖅 pend_mod	1.00.0	UserROM	11696	Model.pend_mod	\Cpu.sw	Pendulum model
(	🙄 Cyclic #2 - [200 ms]						
(	🙄 Cyclic #3 - [500 ms]						
(	Cyclic #4 - [1000 ms]						

Fig. 70: Software configuration including the controller task 'ctrl\_tsk' and the modeling task 'pend\_mod' for 'hardware-in-the-loop' testing

# 5.4.6 Swing Up control of the pendulum

The model 'pend\_ctrl\_swingup.mdl' shows one possibility for swing-up control of the pendulum. The MATLAB Stateflow Toolbox is used to switch between a swing-up control algorithm and the stabilizing linear controller for the upper (unstable) equilibrium position of the pendulum.

5.4.7 Using the MATLAB OPC Toolbox for Online-MATLAB/Simulink debugging

Please contact B&R for examples on this topic.

# 6. APPENDIX

#### 6.1 Real-Time Workshop Embedded Coder

Real-Time Workshop Embedded Coder is not mandatory for the use of *B&R Automation Studio Target for Simulink*. However, it is recommended to generate more efficient and optimized source code for your target system.

For detailed information regarding **Real-Time Workshop Embedded Coder** please visit <u>http://www.mathworks.com/products/rtwembedded</u> or contact MathWorks directly.

#### 6.2 Simulink block support

Nearly all standard Simulink blocks are supported by the Automatic Code Generation. You can get an overview in MATLAB using the command *showblockdatatypetable*. However, it is not recommended to use continuous-time blocks for industrial applications. These blocks should be replaced by corresponding discrete-time blocks. One option is to use the Model Discretizer mentioned in section 5.2.

More detailed information regarding the company **MathWorks**, Inc. can be found at <u>http://www.mathworks.com/products</u>.

B&R cannot guarantee problem-free implementation of blocks other than the standard Simulink blocks. In this case it is recommended to contact the MathWorks **support** to inquire whether a particular block is supported by the products **Real-Time Workshop** and **Real-Time Workshop Embedded Coder**.

Technical support: http://www.mathworks.com/contact\_TS.html



# 6.3 Suggestions

# 6.3.1 Recommended blocks

- Simulink standard blocks
- Time-Discrete blocks
- Embedded MATLAB functions
- Inlined s-functions

# 6.3.2 Supported but not recommended for production code

- Time continuous blocks
- Large look-up tables
- Non-inlined s-functions

# 6.3.3 Not supported blocks

- Level-2 m-file s-functions
- MATLAB functions

# 6.3.4 Further suggestions

The B&R Parameter block only works for tuneable block parameters. For non-tuneable blocks the parameter value will be directly used for the generated source code. If you want to make sure that the block parameter is accessible during runtime, it is strongly recommended to use B&R Input blocks instead of B&R Parameter blocks.



## 6.4 Additional links

The following links will take you to MathWorks' corporate website. B&R can therefore not make any guarantees regarding the site's content. Any questions should be directed to the MathWorks support.

#### Contact information: MathWorks

For questions regarding MathWorks products, you can find the appropriate contact information here: http://www.mathworks.de/company/aboutus/contact\_us/

To contact the technical support department for MathWorks (for customers with a valid maintenance contract), it is recommended to use a 'MathWorks

Account' - free registration at

<u>http://www.mathworks.com/accesslogin/createProfile.do</u> - since this is the only way to submit an online query regarding the current processing status: <u>http://www.mathworks.com/accesslogin/createProfile.do</u>

#### Using Simulink with B&R Automation Studio

MathWorks web portal summarizing latest news and application reports on 'Automation Studio Target for Simulink' and other interface products between Simulink and Automation Studio <u>http://www.mathworks.com/br</u>

# Industrial automation and industrial machines

Industry-specific portal with access to the most important sources of information regarding implementation of MATLAB & Simulink, including user reports, book program and event calendar. http://www.mathworks.com/industries/iam/

#### Tech notes / How-to guides

Tips & tricks for MATLAB & Simulink. http://www.mathworks.com/support/tech-notes/list\_all.html

# MATLAB Central

Public exchange platform for MATLAB & Simulink users, including exchange for files and links as well as access to the public MATLAB Newsgroup.

http://www.mathworks.com/matlabcentral/

# Product documentation

Online access to the complete HTML & PDF documentation for the current MATLAB release.

Note: Access to individual product documentations, such as 'Real-Time Workshop Embedded Coder', requires a 'MathWorks Account' - free registration at <u>http://www.mathworks.com/accesslogin/createProfile.do</u>. <u>http://www.mathworks.com/accesslogin/createProfile.do</u>

# MATLAB tutorial

Online tutorial for introduction to MATLAB. <u>http://www.mathworks.com/academia/student\_center/tutorials/launchpa</u> <u>d.html</u>

# Simulink tutorial

Online tutorial for introduction to Simulink. <u>http://www.mathworks.com/academia/student\_center/tutorials/index.ht</u> <u>ml?link=body#</u>

# MathWorks Newsletter

Access to MathWorks' newsletter 'News&Notes' as well as other publications. http://www.mathworks.com/company/newsletters/?s\_cid=HP\_NL

# **Recorded Webinars**

Presentation of the latest MathWorks solutions in approx. 1 hour long online presentations (in English and German). http://www.mathworks.com/company/events/archived\_webinars.html?s\_ci d=HP\_E\_RW

# Overview of training modules

TM210 - The Basics of Automation Studio TM211 - Automation Studio Online Communication TM213 - Automation Runtime TM220 - The Service Technician on the Job TM223 - Automation Studio Diagnostics TM230 - Structured Software Generation TM240 - Ladder Diagram (LAD) TM241 - Function Block Diagram (FBD) TM246 - Structured Text (ST) TM250 - Memory Management and Data Storage TM261 - Closed Loop Control with LOOPCONR TM400 - The Basics of Motion Control TM410 - The Basics of ASiM TM440 - ASiM Basic Functions TM441 - ASiM Multi-Axis Functions TM445 - ACOPOS ACP10 Software TM446 - ACOPOS Smart Process Technology TM450 - ACOPOS Control Concept and Adjustment TM460 - Starting up Motors TM480 - Hydraulic Drive Control

TM500 - The Basics of Integrated Safety Technology TM510 - ASiST SafeDESIGNER TM540 - ASiST SafeMC

- TM600 The Basics of Visualization
- TM610 The Basics of ASiV
- TM630 Visualization Programming Guide
- TM640 ASiV Alarm System, Trend and Diagnostic TM670 - ASiV Advanced
- TM700 Automation Net PVI
- TM710 PVI Communication
- TM711 PVI DLL Programming
- TM712 PVIServices
- TM730 PVI OPC

TM800 - APROL System Concept TM810 - APROL Setup, Configuration and Recovery TM811 - APROL Runtime System TM812 - APROL Operator Management TM813 - APROL Operator Management TM830 - APROL Project Engineering TM840 - APROL Project Engineering TM840 - APROL Parameter Management and Recipes TM850 - APROL Controller Configuration and INA TM860 - APROL Library Engineering TM865 - APROL Library Guide Book TM870 - APROL Python Programming TM890 - The Basics of LINUX

# **CORPORATE HEADQUARTERS**

Bernecker + Rainer Industrie-Elektronik Ges.m.b.H. B $_{\rm H}$ R Strasse 1 5142 Eggelsberg Austria Tel.: +43 (0) 77 48/65 86 - 0 Fax: +43 (0) 77 48/65 86 - 26 info@br-automation.com www.br-automation.com TM140TRE.30-ENG 0211 ©2011 by BBR. All rights reserved. All registered trademarks are the property of their respective owners. We reserve the right to make technical changes.

155 offices in 60 countries - www.br-automation.com/contact



Argentina • Australia • Austria • Belarus • Belgium • Botswana • Brazil • Bulgaria • Canada • Chile • China • Colombia • Croatia • Cyprus Czech Republic • Denmark • Egypt • Emirates • Finland • France • Germany • Greece • Hungary • India • Indonesia • Ireland Israel • Italy • Japan • Korea • Luxemburg • Kyrgyzstan • Malaysia • Mexico • Mozambique • Namibia• The Netherlands New Zealand • Norway • Pakistan • Peru • Poland • Portugal • Romania • Russia • Serbia • Singapore • Slovakia • Slovenia • South Africa Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • USA • Venezuela • Vietnam • Zimbabwe