Automatic Code Generation with MATLAB/Simulink

TM140

Perfection in Automation

www.br-automation.com

Requirements

Training modules:	TM210 – The Basics of Automation Studio 3
Software:	Automation Studio 3 (Version 3.0.64 and higher)
	MATLAB® (Version 7.4 and higher)
	Simulink® (Version 6.6 and higher)
	Real-Time Workshop® (Version 6.6 and higher)
	Real-Time Workshop® Embedded Coder (Version 4.6 and higher)
Hardware:	None

Trademarks

Patents

Table of contents

1. INTRODUCTION

For years, the **MATLAB®** program package from **The MathWorks** company (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. Unlike numerous other computer algebra systems on the market, MATLAB® is designed to solve numerical problems. The biggest strength of the program lies in its handling of large matrices, as its name **MAT**rix **LAB**oratory suggests. MATLAB® can be expanded using various add-on packages, as **Simulink®** for instance. This program package allows graphic creation of simulation models used to adjust complex technical processes under realistic conditions.

Fig. 1: MATLAB® and Simulink® by The 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.

The biggest advantage of **automatic code generation** comes to those developers who already use MATLAB® und Simulink® for simulation and solutions design and to developers for whom it was once essential to tediously rework implemented structures in a language supported by Automation Studio. In the procedures listed below, this represents an innovation with unforeseen possibilities that help to productively reform the development of control-related systems.

The basic principle is simple: The module created in Simulink® is automatically translated using **Real-Time Workshop®** and **Real-Time Workshop® Embedded Coder** into the optimal language (ANSI-C) for the target system. Seamless and complete integration into an existing project in **Automation Studio** guarantees system conformity.

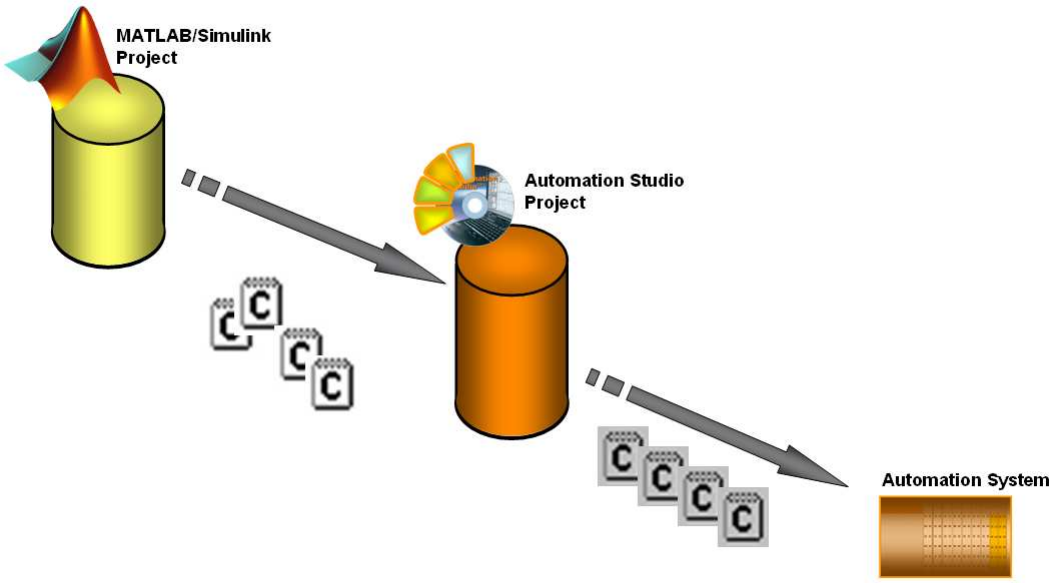


Fig. 2: Workflow

The elimination of extensive porting allows simple transfer of complex and involved simulation models to the controller (*Hardware-in-the-Loop*). Control-related solutions 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 new errors (*Rapid Prototyping*).
Thus, there are two different, basic instances in which the following Simulink® automatic code generation is used.

Rapid prototyping: Automatic code generation makes it possible to quickly and easily transform control or system-related innovations and implement them in a target system. Many potentially successful ideas are immediately rejected because of the large amount of time required for conversion into executable machine code and the risk of developing an error is not justified. The "Rapid Prototyping" concept brings an end to this. Using Simulink®, any system, no matter how complex, can be intuitively built, compiled and tested in a short amount of time. This practically eliminates implementation errors because the automatic code generation uses thousands of tested structures and does not make unnecessary mistakes.

Hardware-in-the-Loop: Every software modification bears the risk of damaging a system during commissioning. The fact that only a limited part of a system at a time can be conceived and modeled leads to control concept defects that can significantly damage a real system. Hardware-in-the-Loop is the key word that allows simple transfer of sophisticated system models developed in Simulink® to a target system. The prepared controller assumes the roll of the actual system for the duration of the first function test. This allows quick and safe testing of new control concepts without damaging costly machine parts. As a result, the controller and system simulation can run on the same target system.

Although there are numerous circumstances for using Simulink® automatic code generation, they have one thing in common: the possibility, **with the press of a button**, to generate ANSI-C code from Simulink®.

1.1 Objectives

After completing the installation described in section 2.1, simple access to professional application can be learned with the help of an example worked out in section 4. 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 blocks for Simulink is located in section 3.

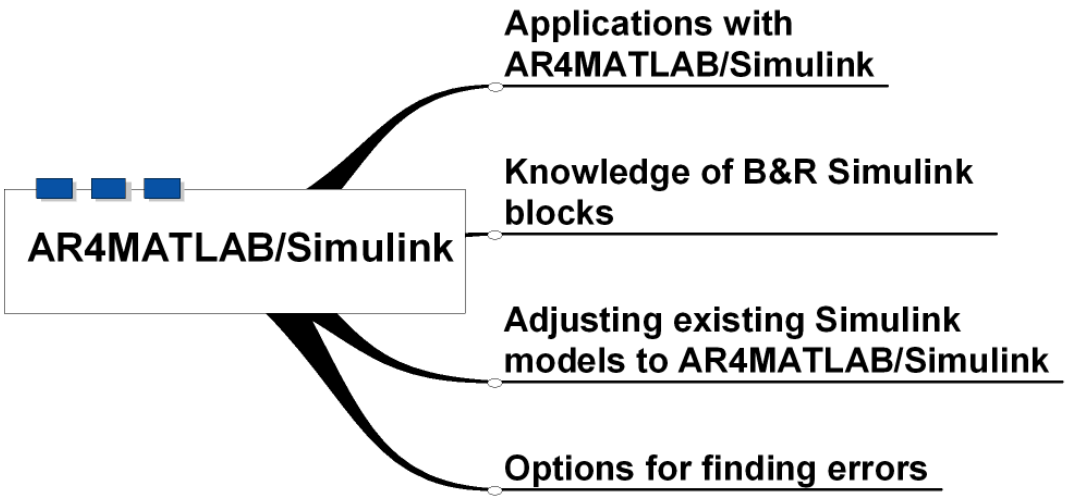


Fig. 3: Objectives

After successful completion of the training module, the user should be prepared to adjust and expand existing Simulink models so the automatic code generation can be executed using *AR4MATLAB/Simulink*. An additional part of this training module is the integration of automatically generated tasks in existing Automation Studio projects, like recognition of numerous options for error diagnosis. An introduction to the products of The MathWorks company is not included in the course of this module, but must instead be found in the documentation accompanying the respective products.

1.2 Definition

1.2.1 Rapid prototyping

As mentioned above, "Rapid Prototyping" offers unforeseen possibilities for quick and flexible implementation of sophisticated control and system-related solutions. Quickly formulated, innovative ideas that would have earlier been rejected because of time and resource restraints can now be quickly and easily developed using Simulink and transferred to a B&R controller using *AR4MATLAB/Simulink*. Tedious manual creation of source code, which always bears the risk of code error, is a thing of the past.

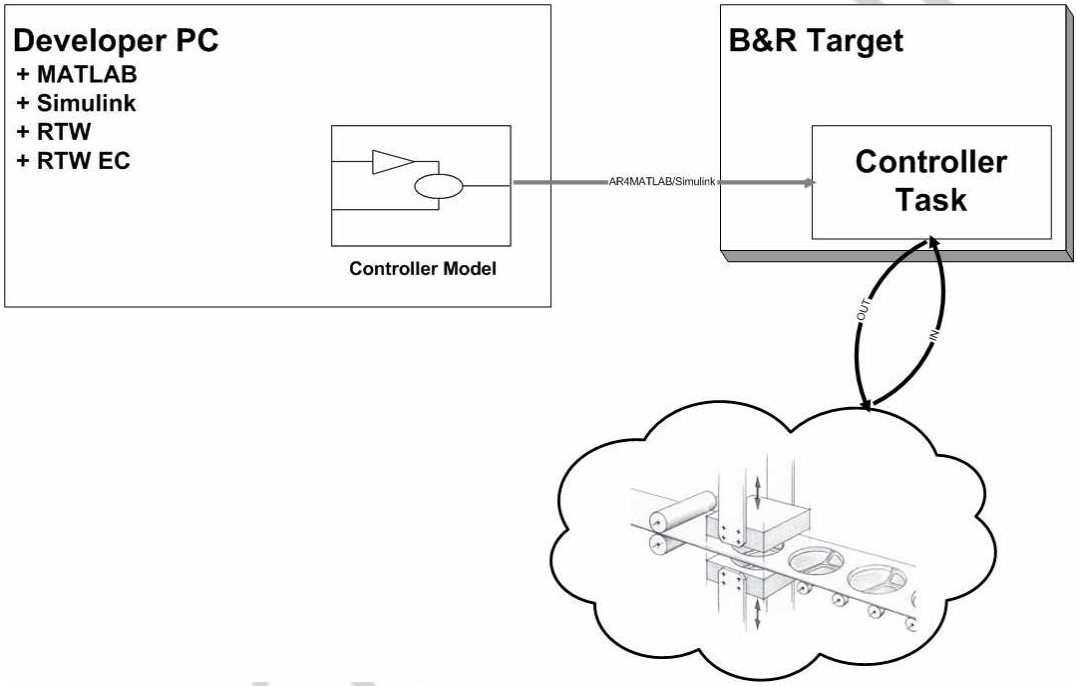


Fig. 4: Rapid prototyping

The procedure is quite simple – the task created in Simulink and transferred to the controller via *AR4MATLAB/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 actual system when applying newly developed algorithms, it is recommended that critical system parts are replaced in advance with an emulation system. For this purpose, a second target system is implemented in the form of a B&R controller using "Hardware-in-the-Loop". The emulation task runs on this system, which mimics the behavior of the actual system as accurately as possible. New developments are thus tested on the target system without putting the system operator at risk of experiencing damage to hardware components.

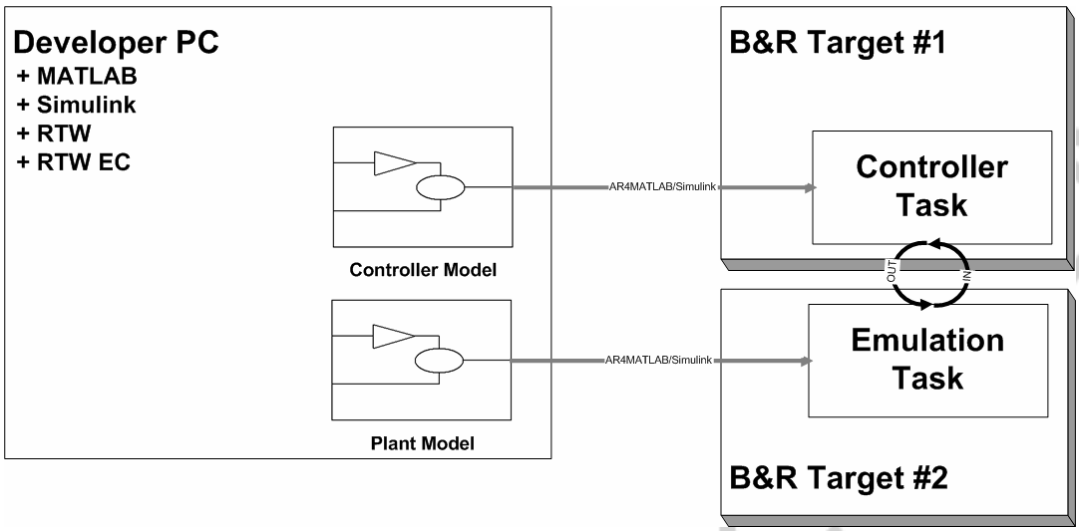


Fig. 5: Hardware-in-the-Loop on two separate target systems

Because there is generally enough free processing power available on the controller being used, both tasks can be run on the same B&R controller, thanks to the task structure of B&R systems. It must be noted, however, that the behavior of the actual inputs and outputs is only mimicked by the emulation model as long as is relevant for functionality.

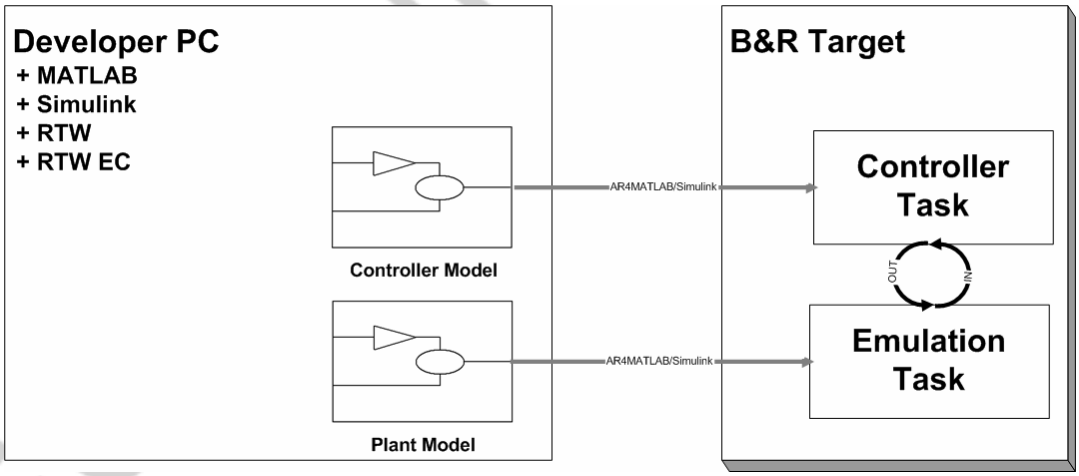


Fig. 6: Hardware-in-the-Loop

2. PREPARATIONS

2.1 Installation

The components required for using *AR4MATLAB/Simulink* can be installed easily and without any user input, using the standard installation of **Automation Studio** Version 3.0.71 and higher.

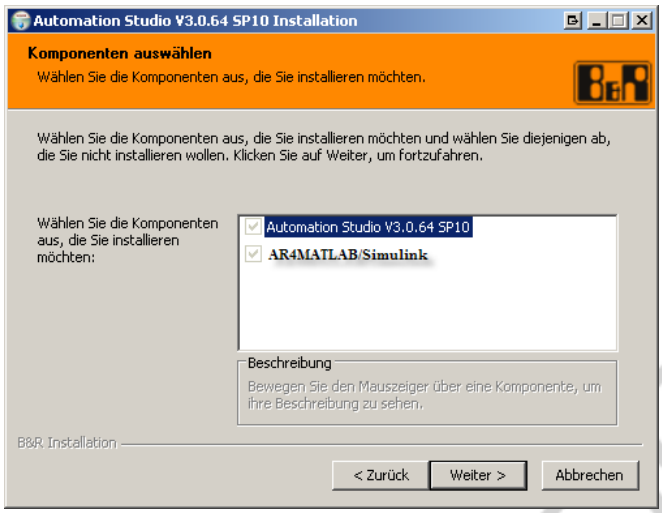


Fig. 7: Automation Studio setup

A prerequisite for using *AR4MATLAB/Simulink* is a one-time execution of **installation script** *"Install.m"* in directory *"X:\Programme\BrAutomation\AR4MATLAB"* ("X:" is the drive on which Automation Studio was installed – generally "C:"). The order in which Automation Studio and MATLAB are installed does not matter here.

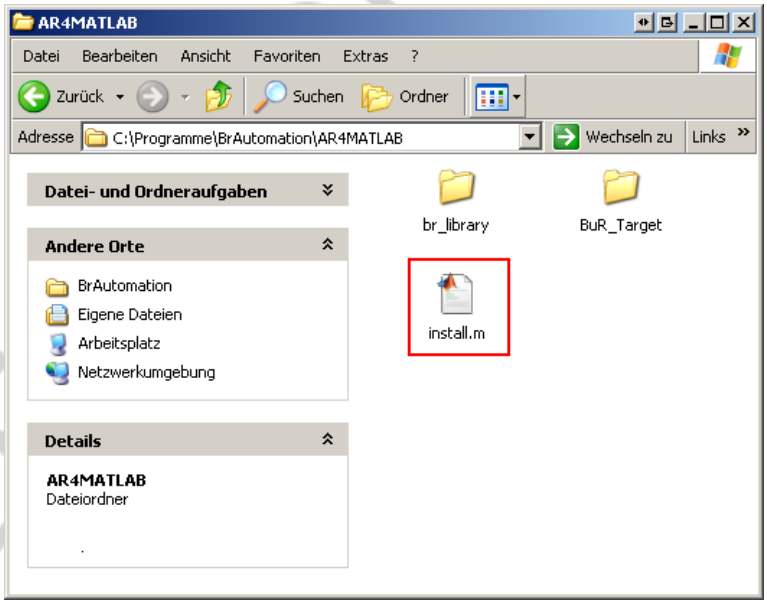


Fig. 8: Setup script for MATLAB

2.2 Advanced software requirements

For use of automatic code generation with *AR4MATLAB/Simulink*, the following software components are required:

- Automation Studio 3 (Version 3.0.64 and higher)
- The MathWorks Release 2007a
  - MATLAB® (Version 7.4 and higher)
  - Simulink® (Version 6.6 and higher)
  - Real-Time Workshop® (Version 6.6 and higher)
  - Real-Time Workshop® Embedded Coder (Version 4.6 and higher)

Also, the following components are needed depending on area of use and hardware configuration:

- Simulink® Fixed Point (Version 5.4 or higher)
- Fixed Point Toolbox (Version 2.0 or higher)
- Stateflow® (Version 6.6 or higher)
- Stateflow® (Version 6.6 or higher)

The following components are also recommended for *AR4MATLAB/Simulink* to function smoothly:

- Control System Toolbox (Version 8.0 or higher)
- Simulink® Control Design (Version 2.1 or higher)

2.3 Hardware support

Automatic code generation with *AR4MATLAB/Simulink* can be used on all B&R hardware components. For use on SG3 and SGC CPU systems, the following two software modules

- Simulink® Fixed Point and
- Fixed Point Toolbox

are recommended, because there is no support on these processor systems for floating point operations.

3. FUNCTION BLOCKS FOR AR4MATLAB/SIMULINK

In this section, the individual components of *AR4MATLAB/Simulink* are described and explained step by step.

- B&R Toolbox
- B&R ARConfig block
- B&R input block
- B&R output block
- B&R parameter block
- Monitor mode

3.1 B&R Toolbox

When the Simulink Library Browser is opened and *AR4MATLAB/Simulink* has been installed, there is an additional entry labeled "B&R Toolbox". This contains four different blocks that are described in the following sections.

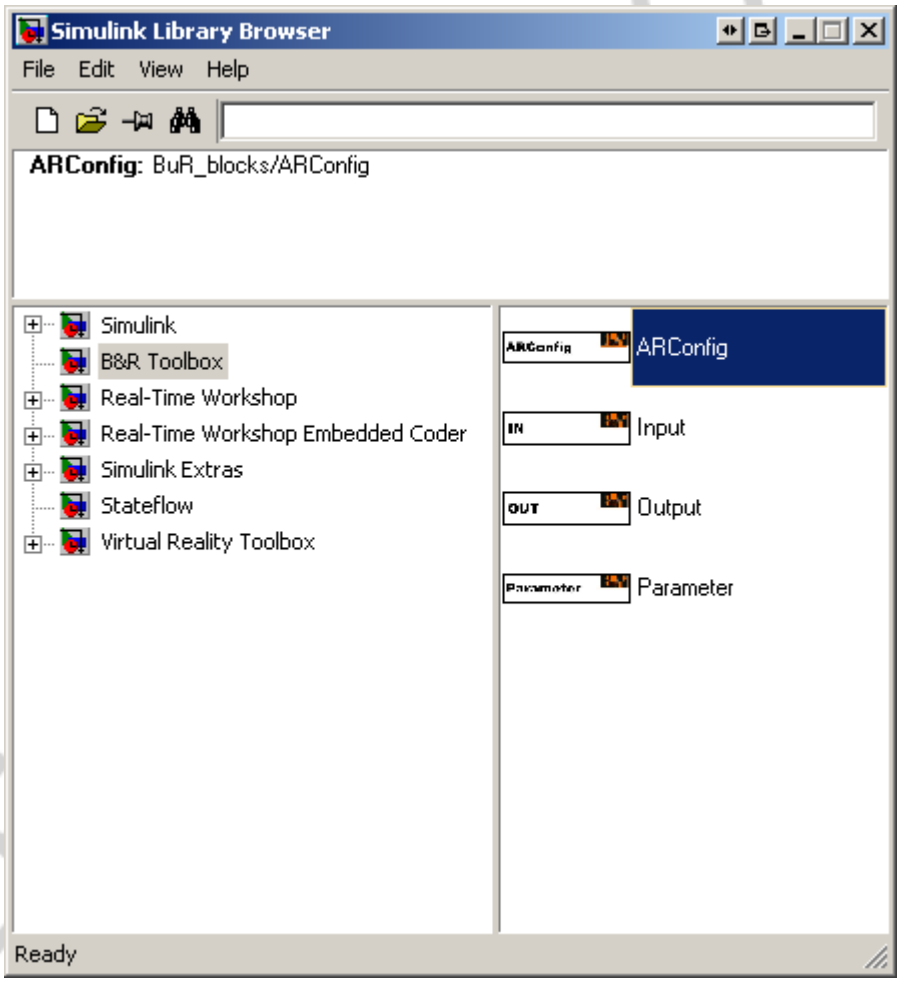


Fig. 9: B&R Toolbox in Simulink

3.2 B&R ARConfig block

During runtime, the "B&R ARConfig block" establishes a connection between the code executed on the the target system and the Simulink environment. The monitor mode described in paragraph 3.6 can be activated using this block's settings or deactivated for all the other blocks.

INFO
Activating the monitor mode only establishes the connection to the target system and adds the setting option "Monitor Mode" to the blocks described above. In order to access to individual variables during runtime, the option must be manually activated for all relevant blocks.

IMPORTANT
Exactly one instance of the B&R block ARConfig must always be contained in the Simulink model - even if no online connection has been established.

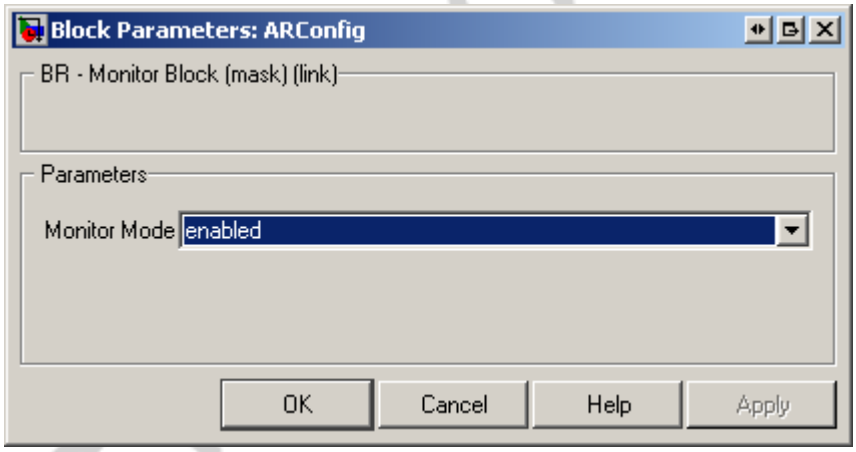


Fig. 10: B&R ARConfig block configuration

3.3 B&R input block

The "B&R input block" serves as the interface between the work environment on the controller and the Simulink model. For every "B&R input block", a variable is created on the target system that helps the model communicate with the other parts of the project.
In this instance, the following settings can be used:

Scope: This specifies scope (global or local) of the variables.

INFO
Local variables are created automatically. However, when the option "PREDEFINED (GLOBAL)" is selected, the user must ensure that the corresponding variables are defined in the Automation Studio project.

Type: The data type of the created variable can be selected from all 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 (Standard)	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 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.

Value: The start value of the variables is defined in this entry field.

Monitor mode: Monitor mode, described in section 3.6, can be activated or deactivated for each variable.

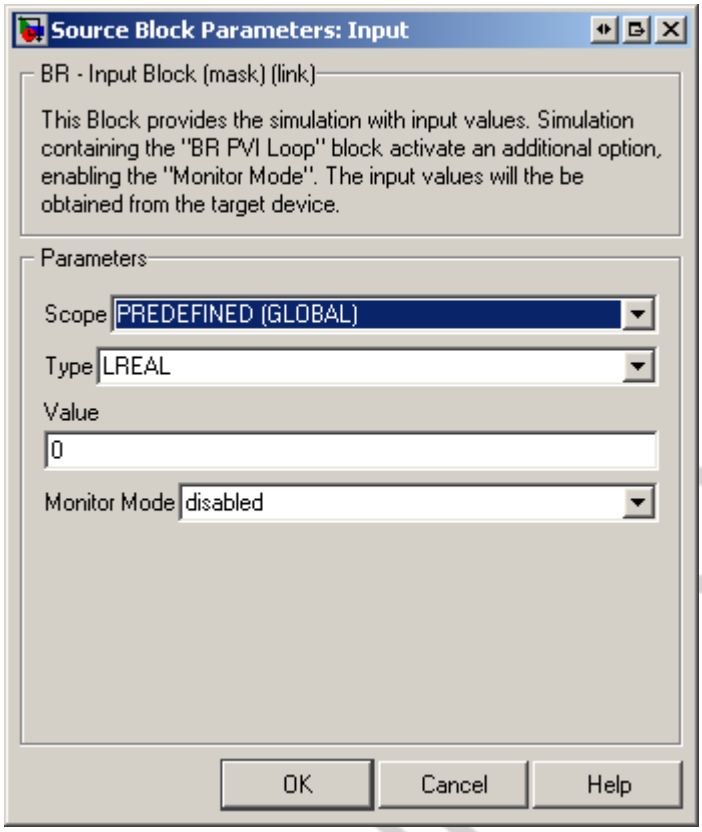


Fig. 11: B&R input block configuration

3.4 B&R output block

The "B&R output block" serves as the interface between the work environment on the controller and the Simulink model. For every "B&R output block", a variable is created on the target system that helps the model communicate with the other parts of the project.

In this instance, the following settings can be used:

Scope: The scope (global or local) of the variables is specified here.

> **INFO**
> Local variables are automatically created. However, when selecting the option "PREDEFINED (GLOBAL)", the user must ensure that the corresponding variables are defined in the Automation Studio project.

Type: The data type of the created variable can be selected from all 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 (Standard)	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 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.

Monitor mode: Monitor mode, described in section 3.6, can be activated or deactivated for each variable.

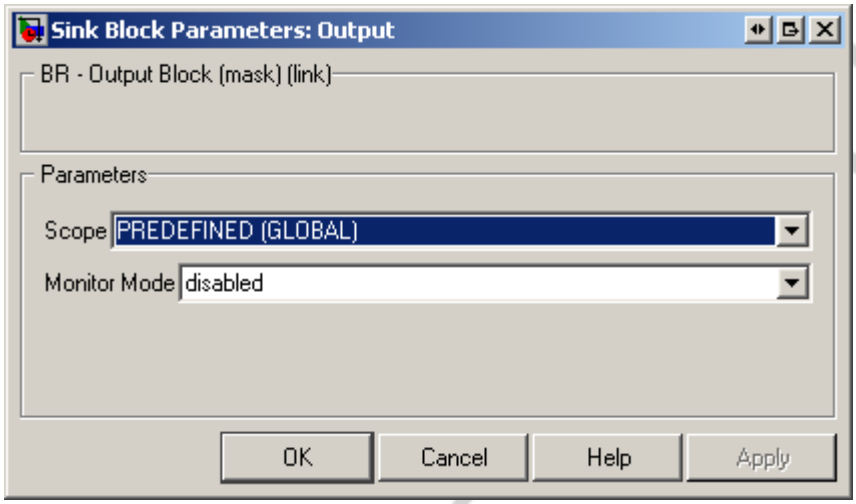


Fig. 12: B&R output block configuration

3.5 B&R parameter block

"B&R parameter blocks" are used to make the internal parameters of individual blocks accessible in Simulink and during operation on the target system. For every "B&R parameter block", a variable is created on the controller.

In this instance, the following settings can be used:

Scope: The scope (global or local) of the variables is specified here.

> **INFO**
> Local variables are automatically created. However, when selecting the option "PREDEFINED (GLOBAL)", the user must ensure that the corresponding variables are defined in the Automation Studio project.

Type: The data type of the created variable can be selected from all 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 (Standard)	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 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.

Value: The start value of the variables is defined in this entry field.

Monitor mode: Monitor mode, described in section 3.6, can be activated or deactivated for each variable.

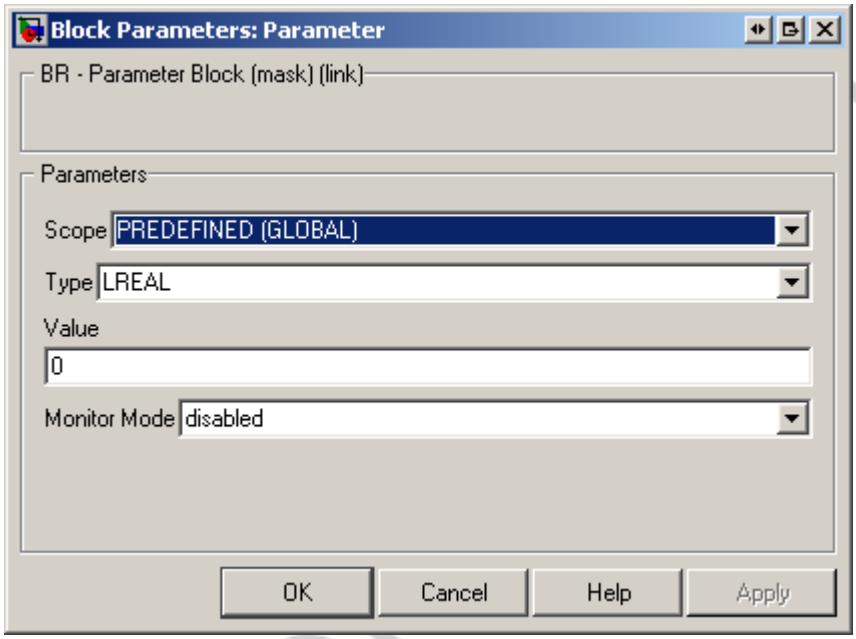


Fig. 13: B&R parameter block configuration

3.6 Monitor mode

Monitor mode is a simple and convenient option for **debugging**. The values from the application running in the target system can be accessed from Simulink via an **online connection**. It is therefore possible to read or modify controller values via the functions made available by MATLAB and Simulink.

Possible applications are value entry via an external MATLAB script or continuous, automatic evaluation of values read by the controller. The clear layout of the Simulink interface also makes it simple to compare simulation results with the actual system.

The **configuration** of monitor mode is very easy to operate. The mode for the entire model can be activated via the corresponding selection box in the ARConfig block. Each process variable can then be activated or deactivated separately.

The connection can be established via **TCP/IP** as well as via **serial interface**, although the serial connection is not recommended in fast systems.

4. WORKING WITH AR4MATLAB/SIMULINK

The example on the following pages will help clearly explain the use of the blocks introduced above.

Example:

The following introductory example illustrates, in simple steps, how an existing Simulink model is prepared for automatic code generation with AR4MATLAB/Simulink, and which preparations must also be made in Automation Studio.

- Origin
- Interfaces and parameters
- Project and target configurations
- Debugging

Using Simulink and preparing system and control-related models 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. 13 serves as the output in the implementation example, which adds the inputs a and b, multiplies them by a constant factor k and displays the result equal 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. It is recommended to those users who are not yet experienced enough with the use of MATLAB and Simulink to first read the appropriate sections in the appendix before proceeding.

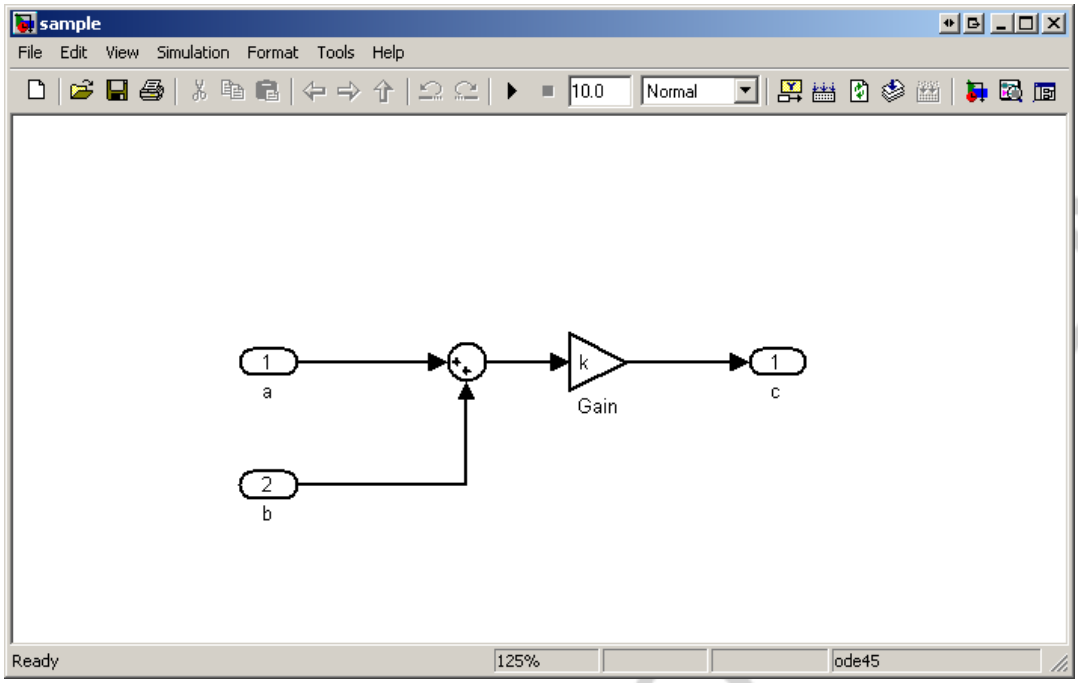


Fig. 14: Output model

4.1.2 Basic settings: B&R ARConfig block

Inserting the ARConfig block completes the first step towards automatic code generation. The ARConfig block not only serves as the online connection to the target system, but is also responsible for most of the basic settings and must, without exception, be present in every model that generates code.

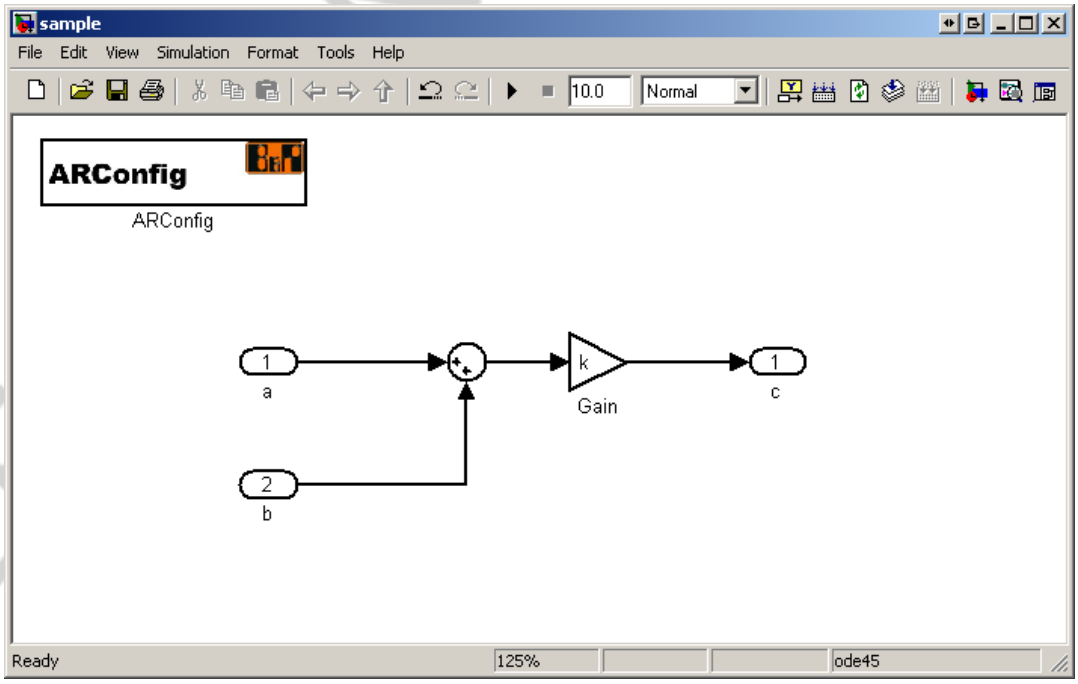


Fig. 15: Model with B&R ARConfig block

For now, monitor mode remains deactivated for the whole model.

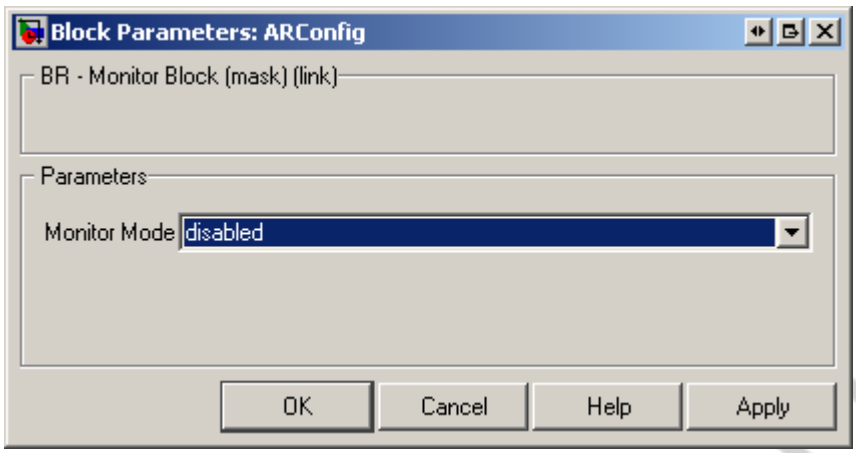


Fig. 16: B&R ARConfig block configuration

4.1.3 Interfaces: B&R input und output block

In order to make the process variables visible on the controller, 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 input and output block, which bears the name of the corresponding block.

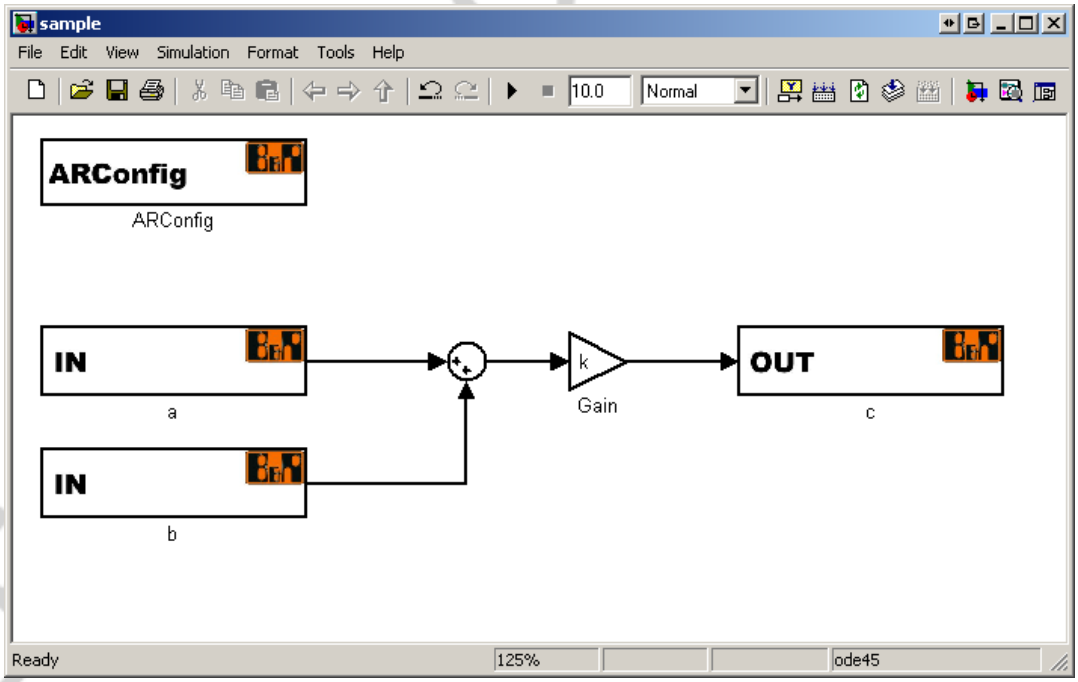


Fig. 17: Model with B&R input and output blocks

In addition to naming the variables, the scope, data type, start value, and data source must be adjusted. The value and source are only available for input variables.

The variables can be declared as either local or global using the scope option. In Automation Studio, local variables are created automatically in the course of the code generation. Global variables, however, must be declared manually by the user in the Automation Studio project.

WARNING
When manually declaring 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.

The data type of the variables can be modified using the Type entry. The following data types are allowed when using *AR4MATLAB/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 (Standard)	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

INFO
The correct data type for local output variables is internally defined in Simulink and cannot be set.

The start value of the active variables at initialization is defined in the value input field. Default value: 0, or FALSE

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

a: Scope: LOCAL
Type: LREAL
Value : 1

b: Scope: PREDEFINED (GLOBAL)
Type: INT
Value: 2

c: Scope: LOCAL

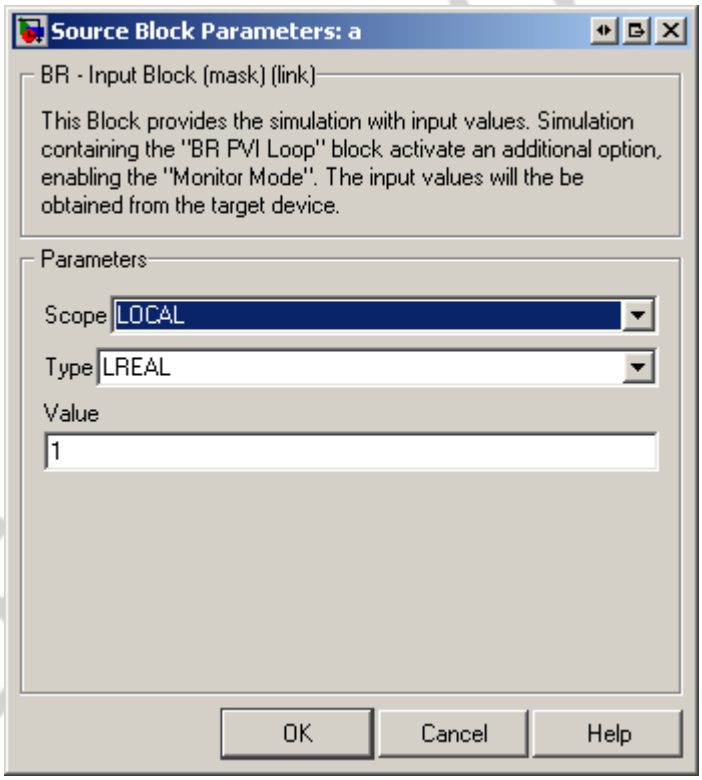


Fig. 18: Settings for variable a

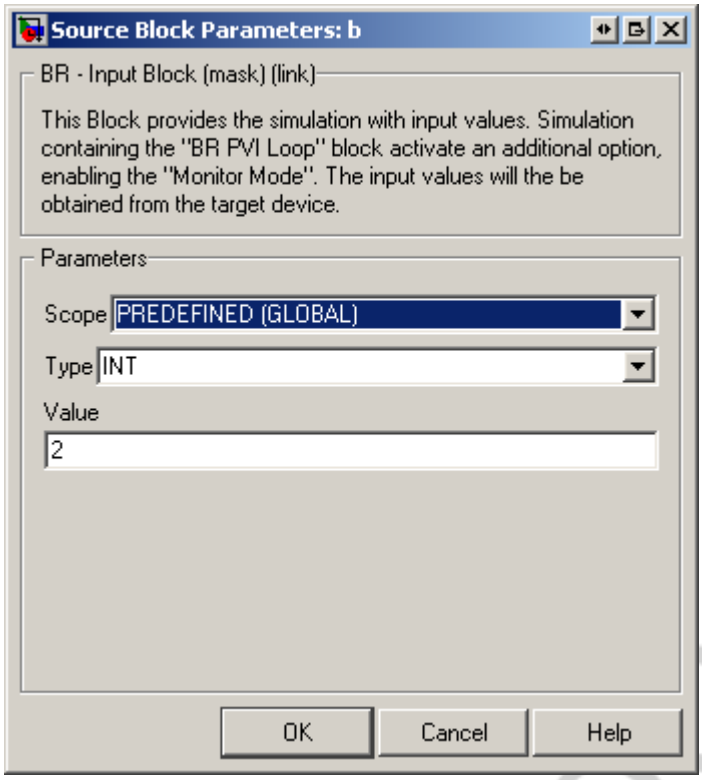


Fig. 19: Settings for variable b

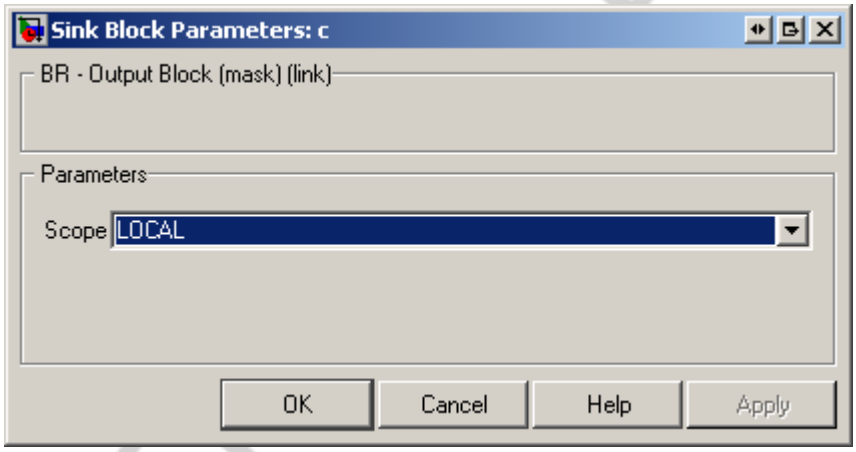


Fig. 20: Settings for variable c

4.1.4 Online configuration of factors: B&R parameter block

To make factor k accessible during operation, a B&R parameter block must be inserted. Again, the variable name is determined from the label located under the block – in this case, k.

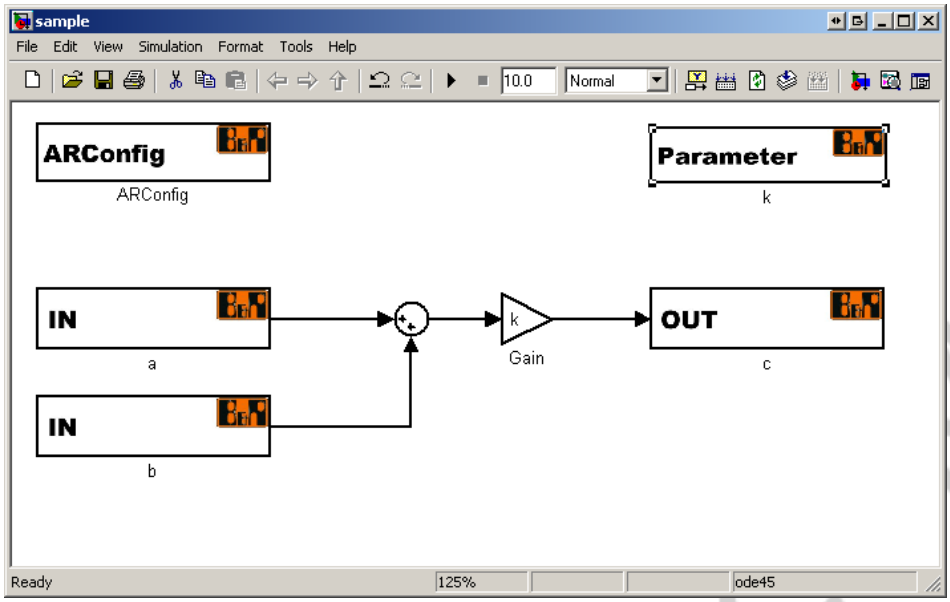


Fig. 21: Model with B&R parameter block

As described for the input and output blocks, the settings scope, type and value apply for parameter blocks.

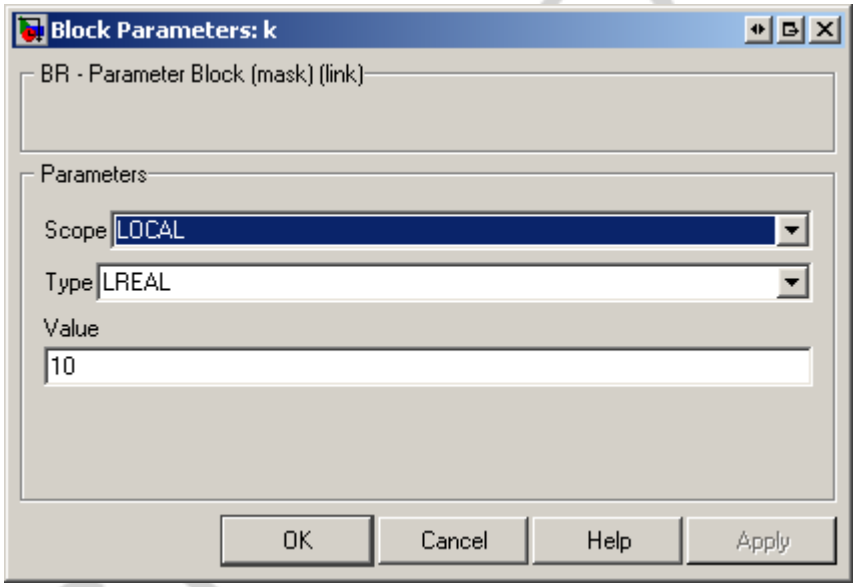


Fig. 22: Settings for variable k

4.1.5 Preparations: Automation Studio project

To allow automatic integration of code produced from the model in an existing Automation Studio project, a few minor preparatory measures must be met. First, an Automation Studio task is created, which bears the same name as the Simulink model – in our case "sample". ANSI C is selected as the programming language.

IMPORTANT
Before executing the automatic code generation, an empty task bearing the same name as the Simulink model must be created in Automation Studio.

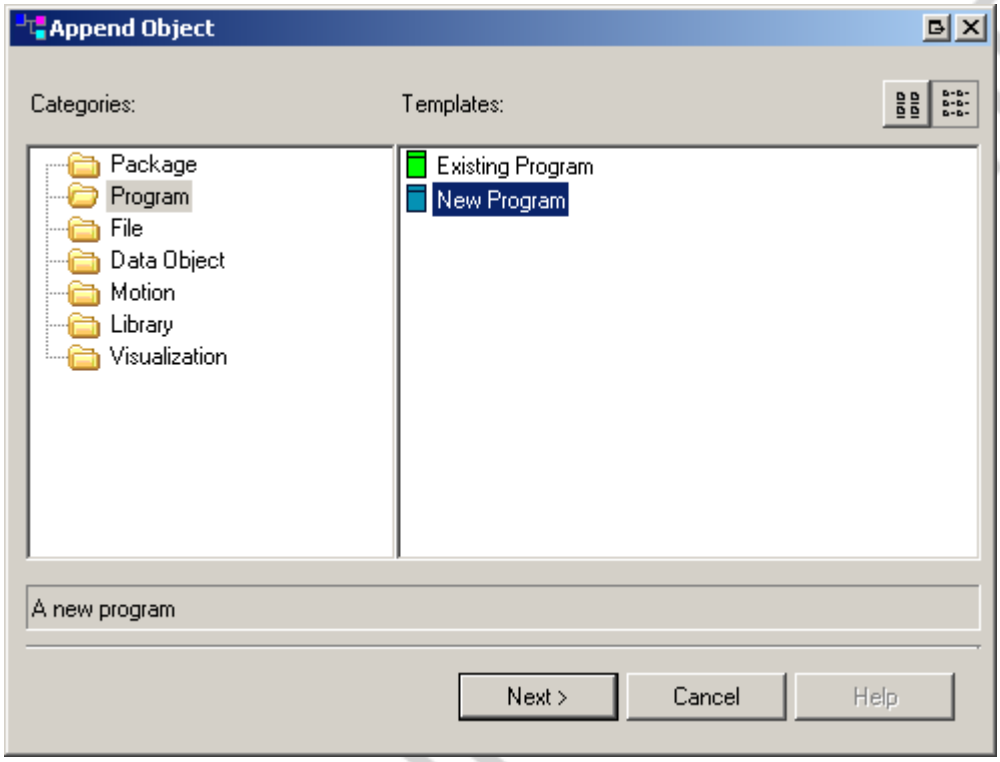


Fig. 23: Creating a new task in Automation Studio

When creating an Automation Studio task, it is imperative that its name matches that of the Simulink model.

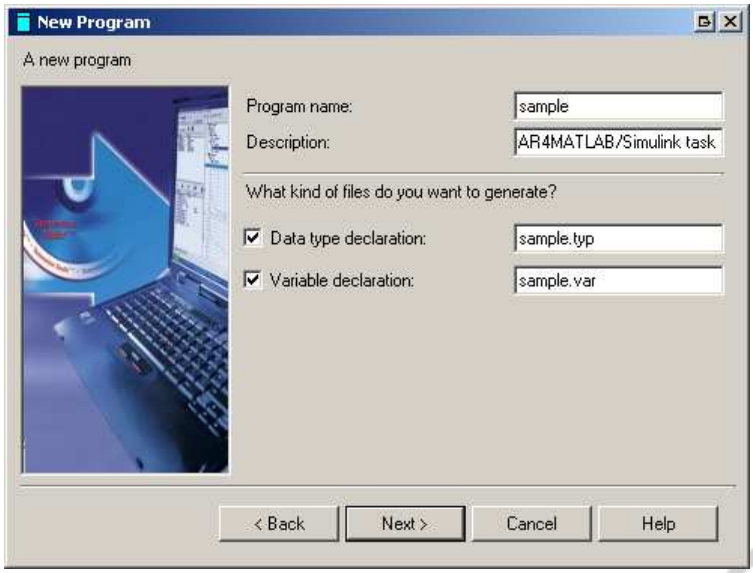


Fig. 24: Naming an Automation Studio task

It is also imperative that only one file is created for the initialization part, the cyclical part and the exit procedure.

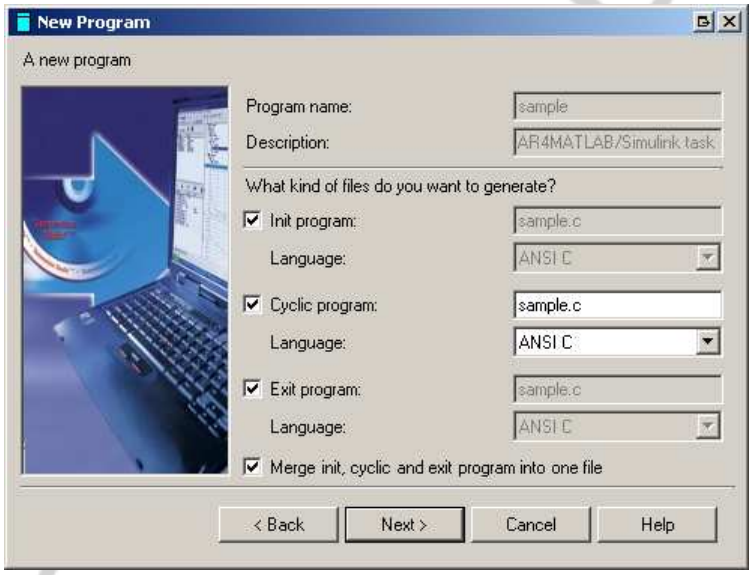


Fig. 25: Automation Studio task settings

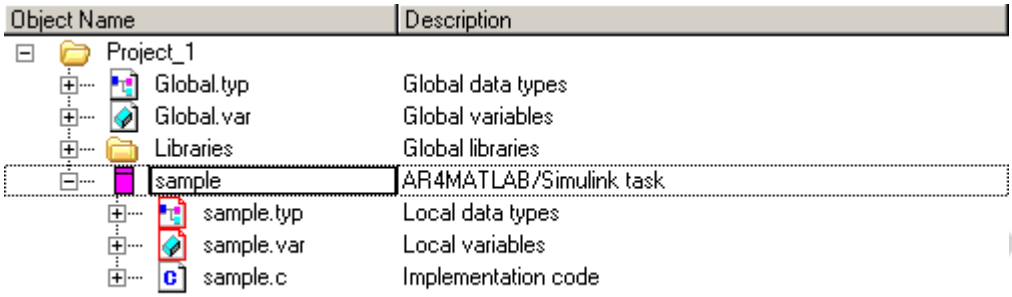


Fig. 26: Automation Studio task in the project directory

Additionally, all global variables used with the Simulink model must be created – in our case, only variable b. Local variables are created automatically during automatic code generation.

IMPORTANT
It is important to note that the type of global variables created in Automation Studio corresponds with the type selected in Simulink (see B&R input block).

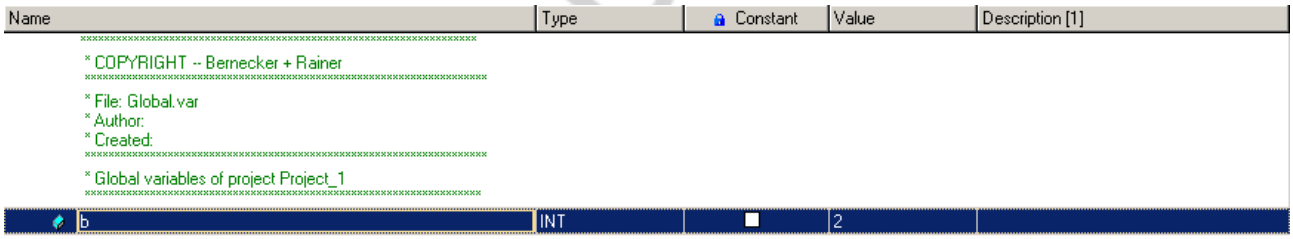


Fig. 27: Global variable b

4.1.6 Settings for the target system: B&R target preferences

Since *AR4MATLAB/Simulink* is used not only to generate code from Simulink models, but also to automatically integrate the source code in Automation Studio, settings must be made in MATLAB with respect to the current Automation Studio project. These can be found in the MATLAB Start menu under ***Start → Simulink → Automation Studio → Target Preferences***.

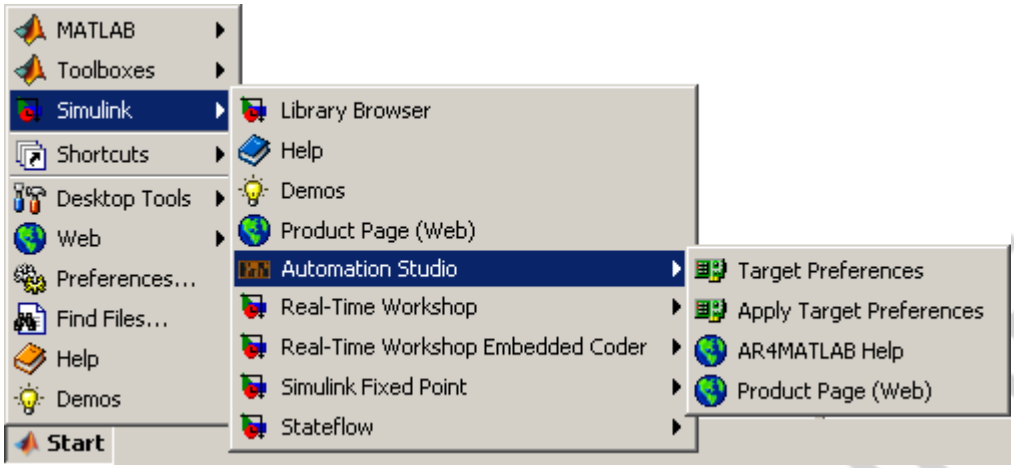


Fig. 28: B&R Target system settings

The following points are adjusted:

AS_Configuration_Name: Configuration name of the current Automation Studio project

AS_PLC_Name: PLC name of the current Automation Studio project

AS_Project_Directory: Root directory of the current Automation Studio project

Package_Name: Name of the package in the current Automation Studio project, in which the source code should be added

Simulink_Model_Directory: Directory in which the Simulink model is located

IMPORTANT
The entries listed above represent a frequent source of error in automatic code generation. All settings must match those of the Automation Studio project exactly.

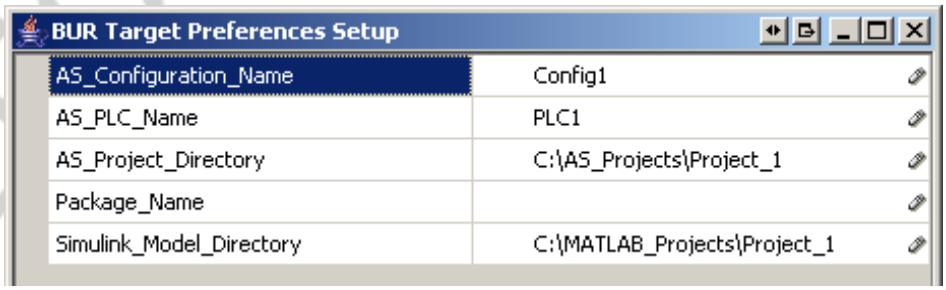
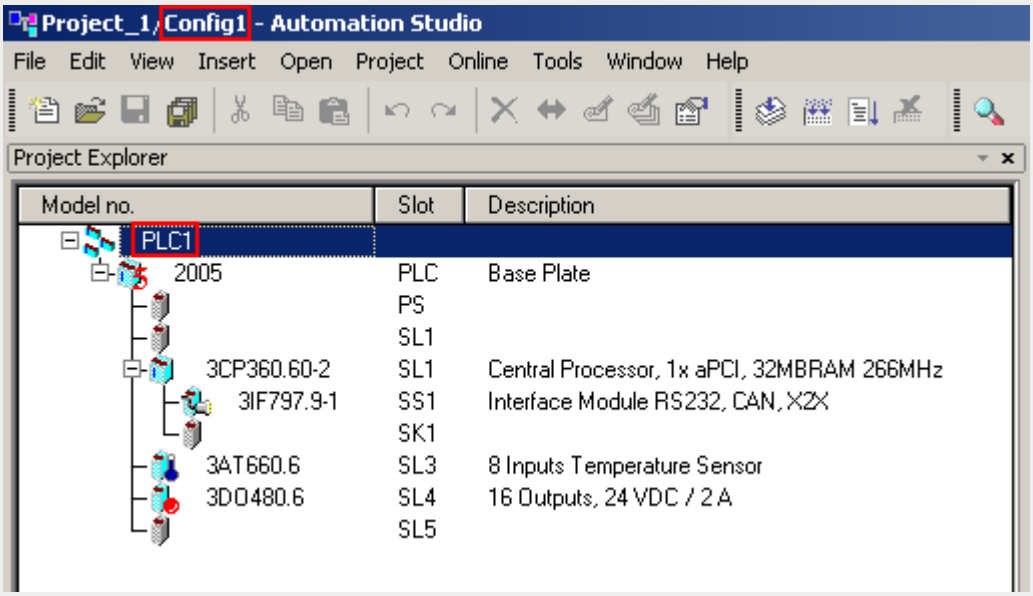


Fig. 29: Target settings for the example program

INFO
The configuration and PLC names of the current Automation Studio Project can be easily checked – as seen below – in Automation Studio.



The configuration created as described above must then be applied to the respective Simulink model. This is done using the menu item ***Start → Simulink → Automation Studio → Apply Target Preferences***. In the process, all the necessary basic settings - such as switching to a fixed step solver or increasing the duration of the simulation to infinite simulation (inf) - are adjusted in the model.

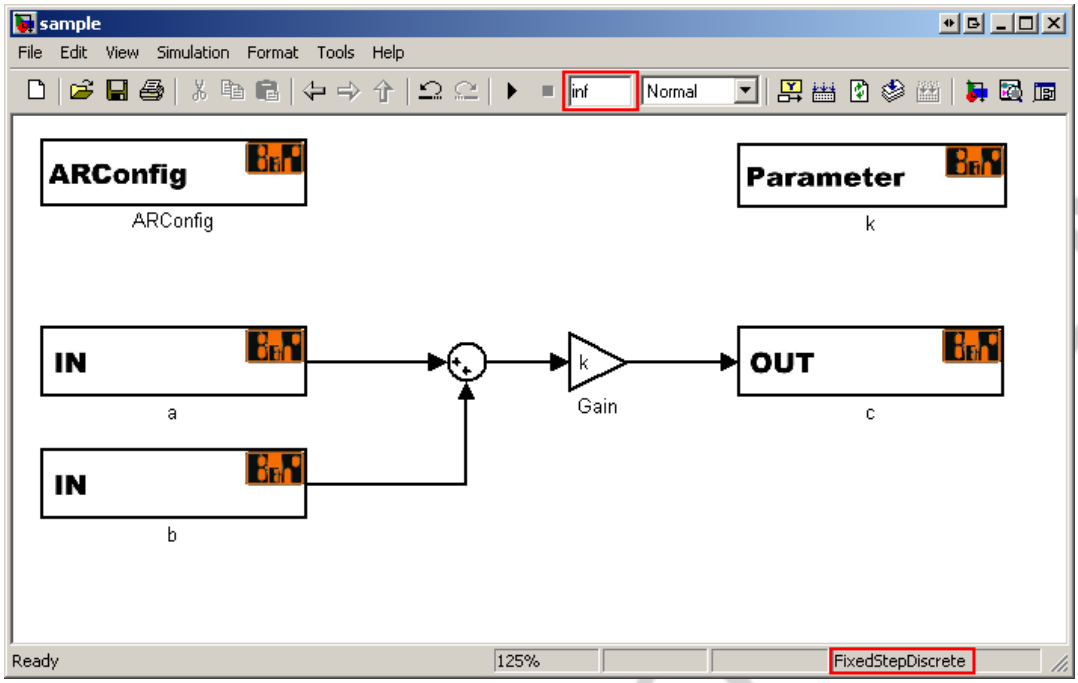


Fig. 30: Adjusted Simulink model

All preparations are now 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 begin with the automatic code generation in Simulink. This is done using the menu item ***Tools → Real-Time Workshop → Build Model... (Ctrl+B)*** or using the corresponding button on the toolbar.

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

```
### Cleaning up
### Generating Main File
### Postprocessing
### Logging variables for Automation Studio
### Relocating
### AS-Setup
### Successful completion of Real-Time Workshop build procedure for model: sample
>>
```

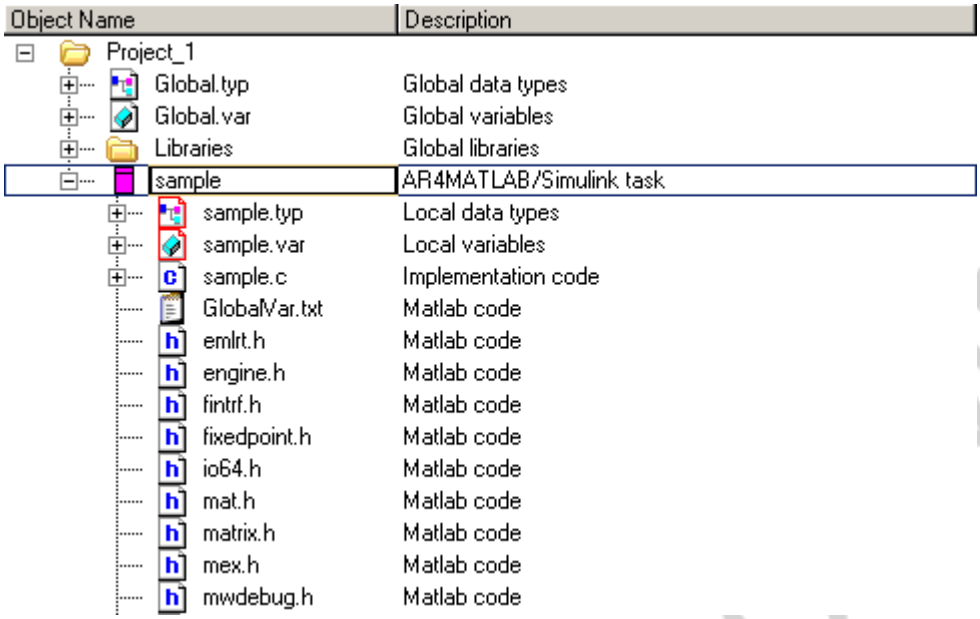


Fig. 31: Automatically integrated program code from the example program

4.1.8 Put it to the test: Debugging in Automation Studio

Taking a look at the watch window in the Automation Studio project, we can see that all the specified variables, including their data type and scope, can be found on the target system. The function test of our simple algebraic system also goes as expected.

Name	Type	Scope	Force	Value
a	LREAL	local		1.0
b	INT	global		2
c	LREAL	local		30.0
k	LREAL	local		10.0

Fig. 32: Watch window in Automation Studio

4.1.9 Online connection: Monitor mode

As already mentioned in section 3.6, it is possible to read out and edit process variables in Simulink on the controller. To do this, activate monitor mode in the B&R block ARConfig for the current model and for all relevant variables.

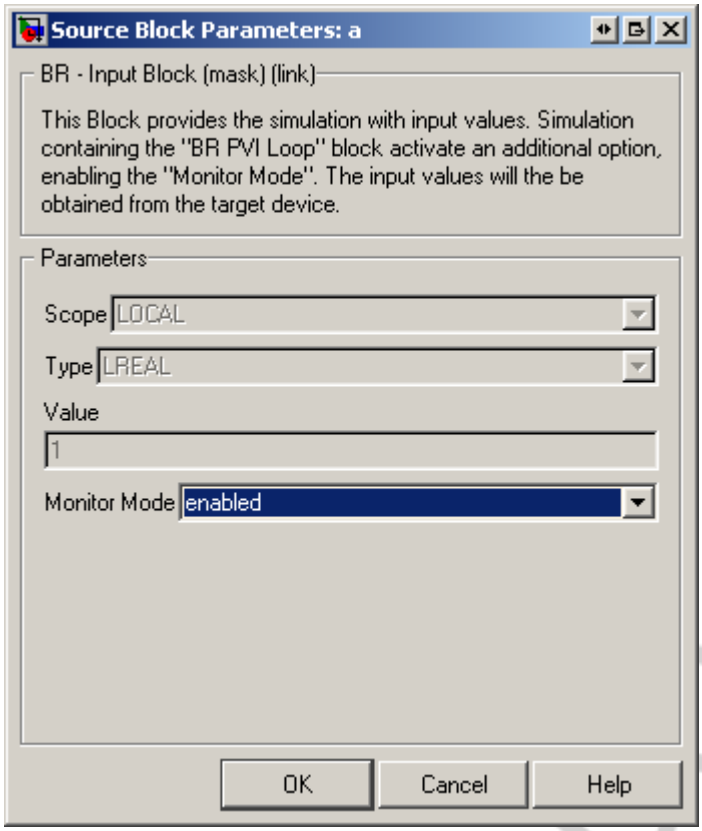


Fig. 33: Activate monitor mode for variable a

When the simulation is started in Simulink, the respective process variables are added in MATLAB Workspace, and can be read out and edited either directly or by a script.

INFO
Monitor mode can be activated without having to restart the automatic code generation.

IMPORTANT
To activate monitor mode, the simulation must be started.

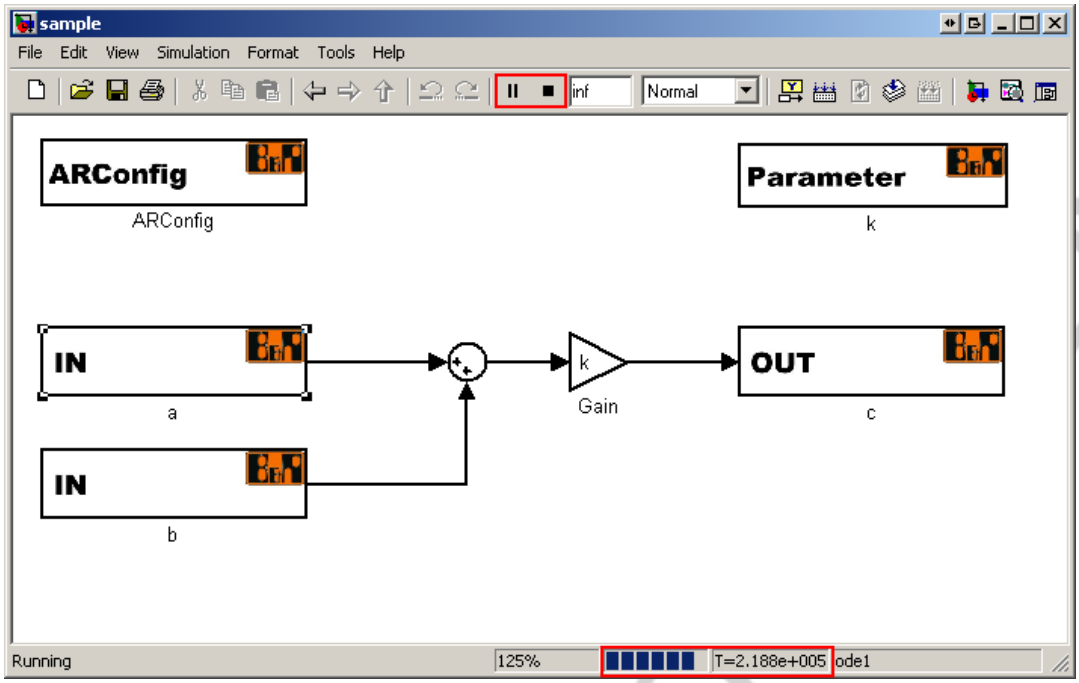


Fig. 34: Running simulation in monitor mode

In the MATLAB Workspace, the values of the variables can be read out on the target system.

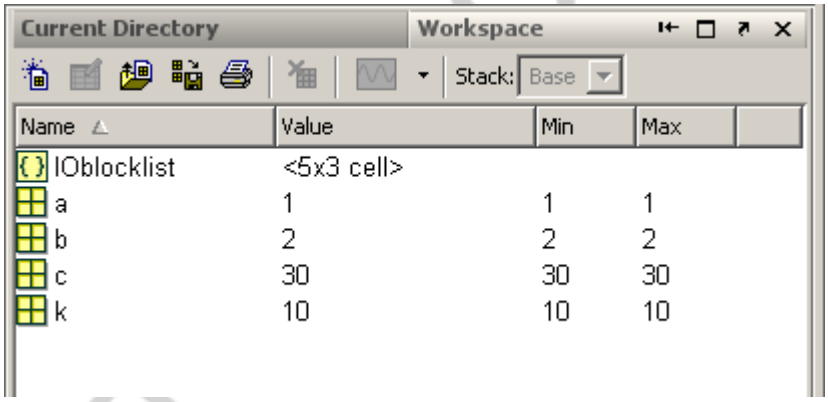


Fig. 35: MATLAB Workspace

5. EXAMPLES

The following examples will provide an overview of the extensive possibilities for utilizing *AR4MATLAB/Simulink* in the field of automation technology. Using a simple **discrete-time PID controller**, we will show how quickly and easily control technology solutions can be created using Simulink and transferred to the target system using *AR4MATLAB/Simulink*. Then, using a **simulation model of a temperature system**, we will demonstrate the implementation of a continuous-time simulation model for hardware-in-the-loop applications. In the third and final example, we will present the option of using **external, self-generated Simulink libraries** together with *AR4MATLAB/Simulink*.

5.1 PID controller

Example:

As you can imagine, it is easy to implement a simple PID controller using Simulink. After the control deviation has been established with 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 in section Function blocks for AR4MATLAB/Simulink and start the code generation.

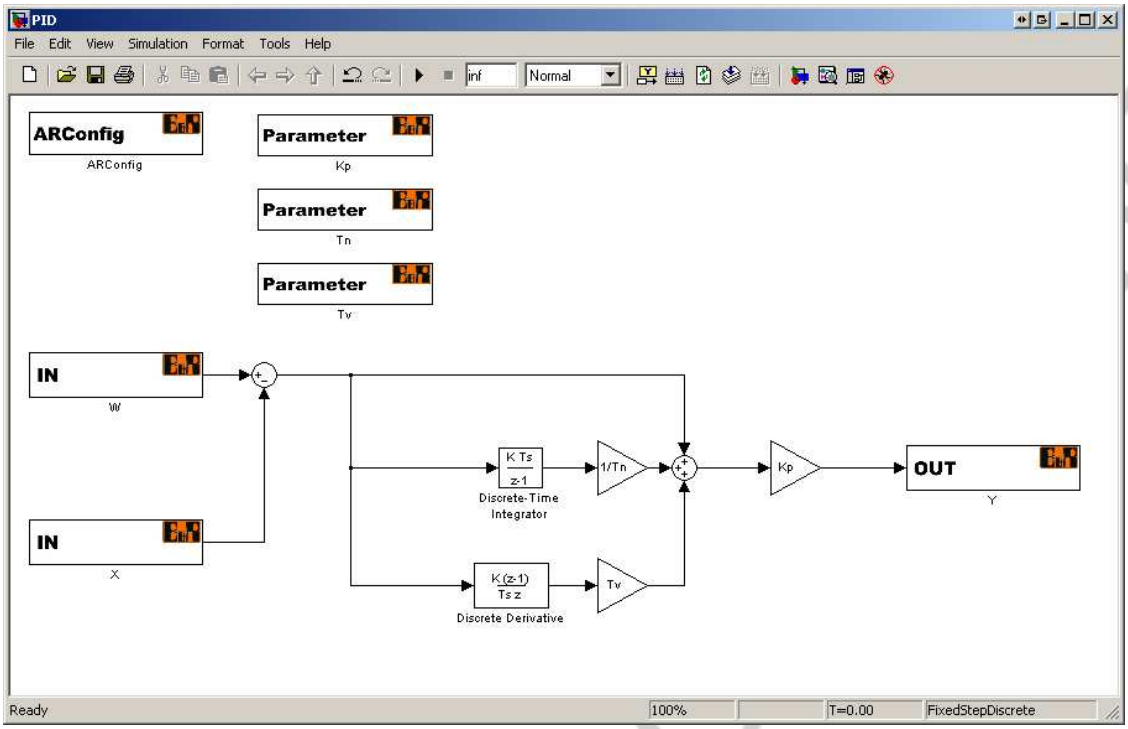


Fig. 36: Discrete-time PID controller

The control concept for the PID controller:

$Y_p = K_p \cdot (W - X)$ ... Proportional element

$Y_I = {K_p}/{T_n} \cdot \int (W - X) \cdot dt$ ... 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 fixed scan cycles, it is recommended to ensure that all integrator and differentiator blocks are also discrete-time.

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 must be created and transferred to the target system using *AR4MATLAB/Simulink*.

Example:

The system is based on the following mathematical model:

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

A simulation is also made of white noise during measurement and quantization to tenths of a degree by the measurement sensor. All system parameters are - just like the ambient temperature - accessible as local parameters.

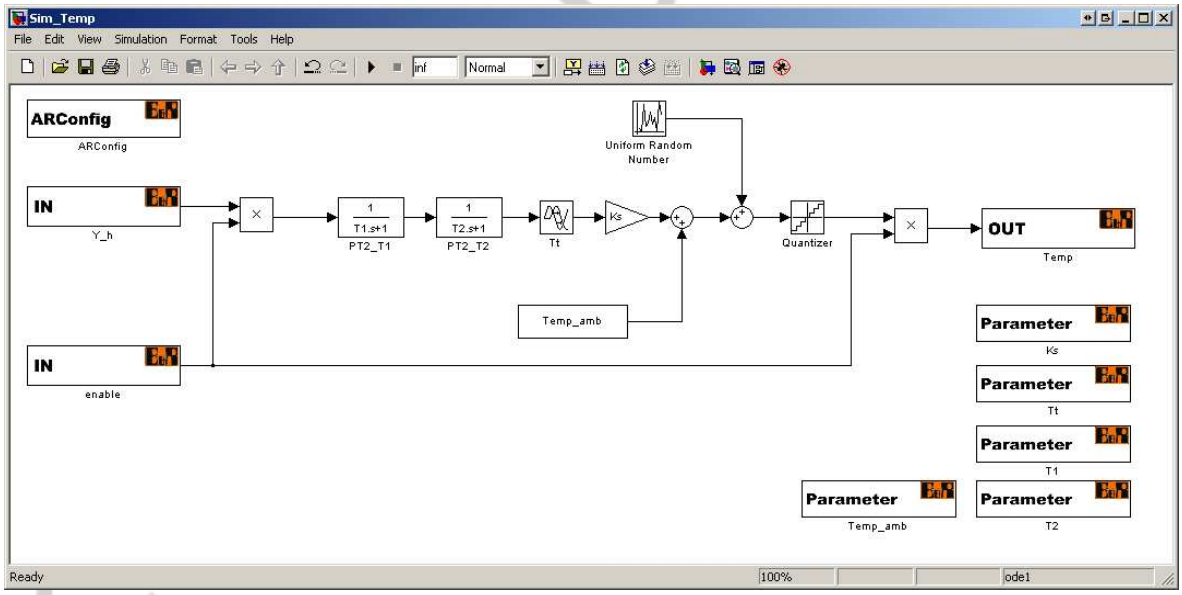


Fig. 37: Temperature system

Since the simulation model at hand is a continuous-time model, you must activate support for continuous-time systems.

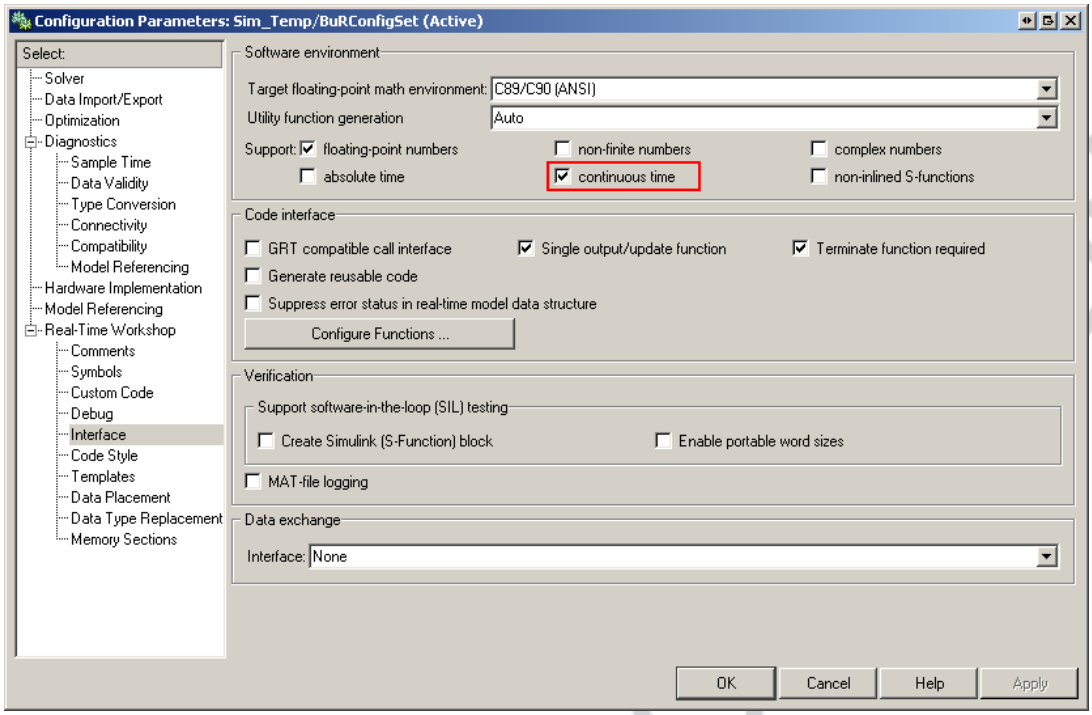


Fig. 38: Settings for continuous-time Simulink models

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

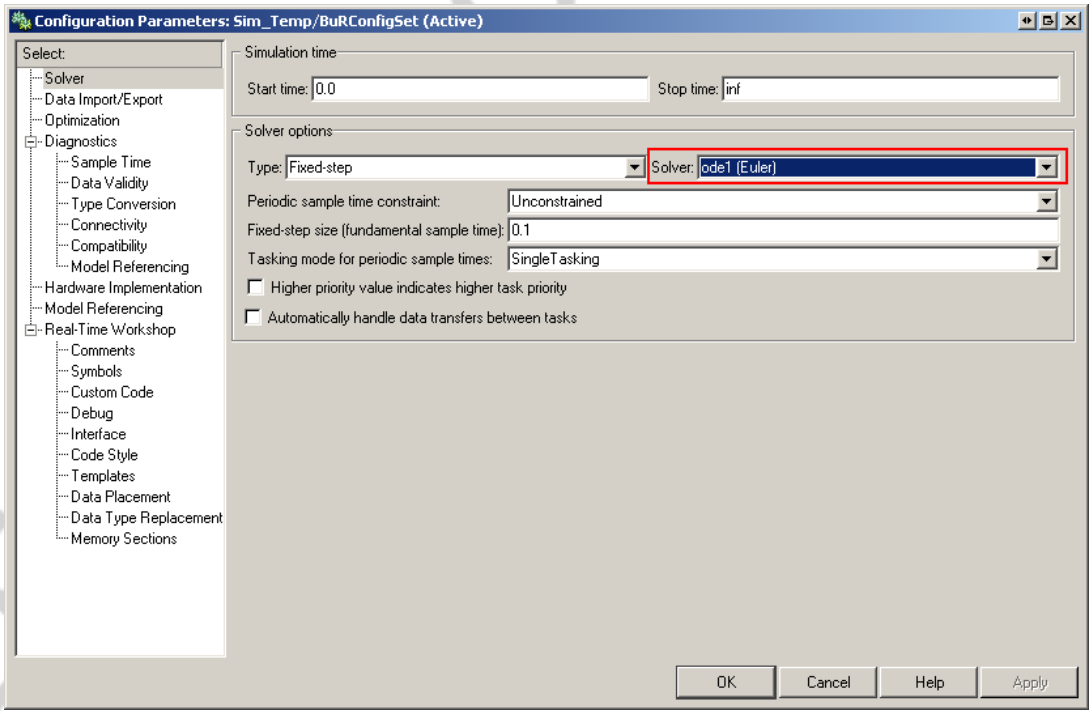


Fig. 39: Fixed step solver

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", a tool from the company The MathWorks. This can be found - as long as the two program packets *Control System Toolbox* and *Simulink® Control Design* are installed - in Simulink under **Tools → Control Design → Model Discretizer** (see section 6.1).

5.3 Hydraulics applications

Many external libraries can also be used with *AR4MATLAB/Simulink* (see section Appendix). This includes the hydraulics library "hydroLib" from the Institute of Machine Design and Hydraulic Drives of the Johannes Kepler University in Linz, Austria. This library is available at http://imh.jku.at/ftp/index.en.php, and is maintained and continuously expanded by members of the institute.

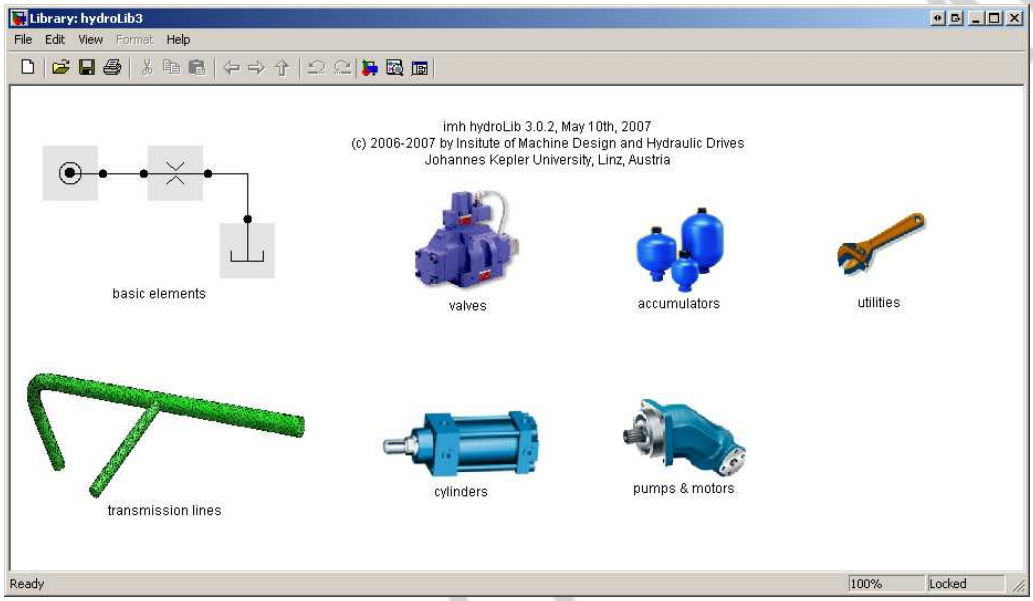


Fig. 40: Hydraulics library from the JKU Linz

The included example "example1.mdl" can be expanded using the B&R blocks described in section 3 so that you can generate complete, executable machine code for any B&R target system.

IMPORTANT
In order for external libraries to be supported, you must not use any Simulink function blocks that are not supported by Real-Time Workshop (see section Appendix). Also, "non-inlined S-functions" (see www.mathworks.com) cannot be used with AR4MATLAB/Simulink.

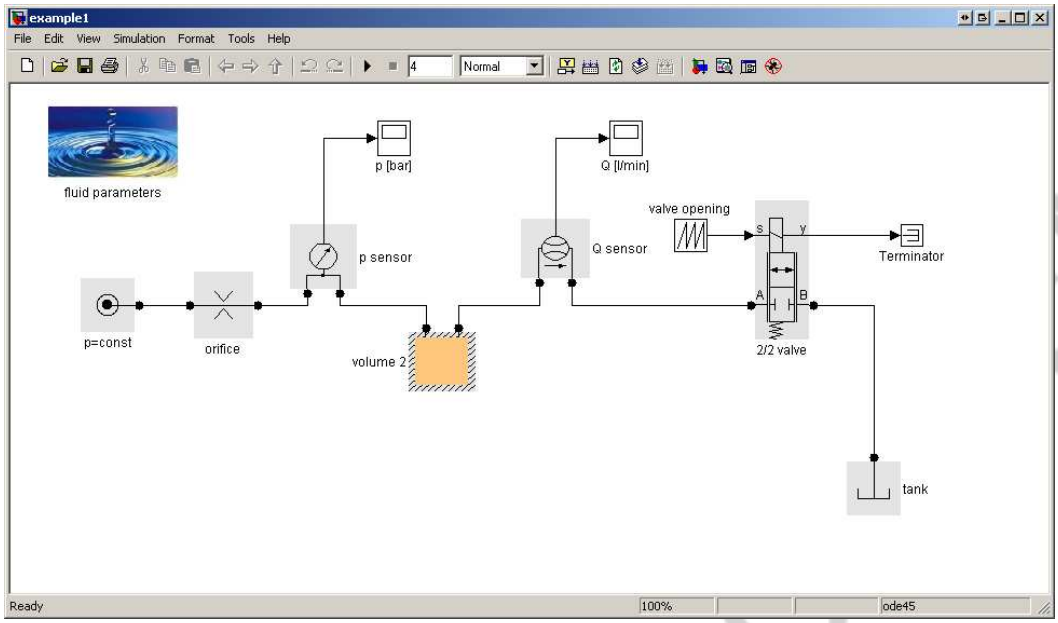


Fig. 41: Hydraulics model without B&R blocks

The addition of B&R blocks allows the example that accompanies the library to be transferred to the target system quickly and easily. The scan time must be set low enough (e.g. 1ms) to meet the demands of the highly dynamic system.

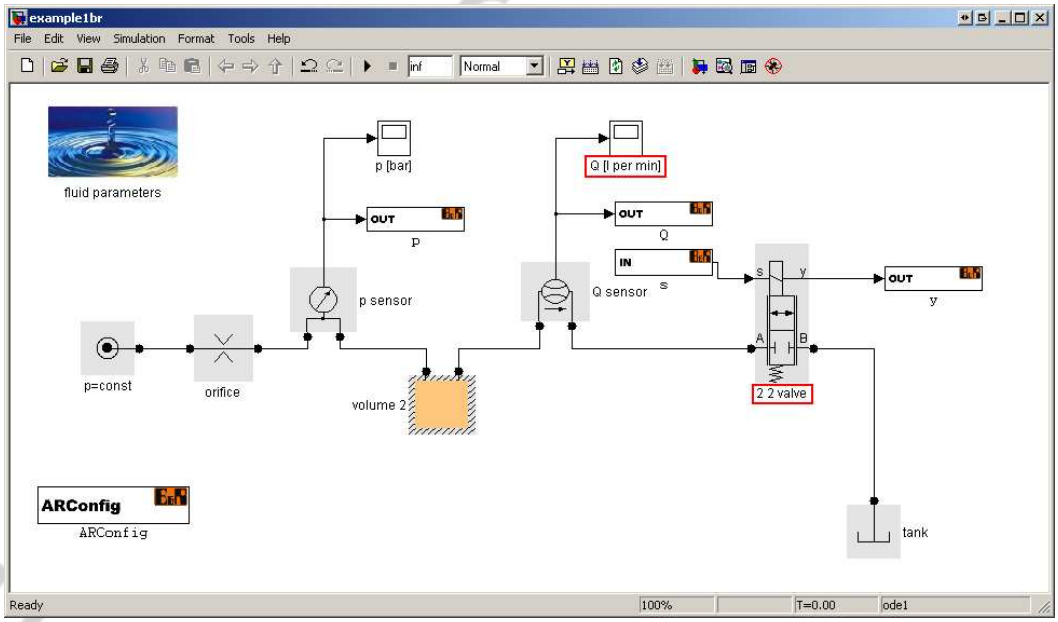


Fig. 42: Hydraulics model with B&R blocks

IMPORTANT
Special characters such as spaces at the end of the character string must be avoided when naming the Simulink blocks, as they cause problems during automatic code generation.

6. APPENDIX

6.1 Simulink block support

Nearly all standard Simulink blocks are supported by the automatic code generation. You can call up an overview in MATLAB using the command ***showblockdatatypetable***. Only a few blocks are explicitly incompatible with the Real-Time Workshop from **The MathWorks**. However, it is not recommended to use continuous-time blocks for industrial applications. These blocks should be replaced by a corresponding discrete-time scan system. One option here is to use the Model Discretizer mentioned in section Temperature model.

More detailed information regarding the company The MathWorks can be found at http://www.mathworks.com/products.

B&R cannot guarantee problem-free implementation of blocks other than the standard Simulink blocks. In this case we recommend contacting 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

6.2 Additional links

The following links will take you to The 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: The MathWorks

For questions regarding The 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 The MathWorks (for customers with a valid maintenance contract), it is recommended to use a "MathWorks Account" – free registration at
http://www.mathworks.com/accesslogin/createProfile.do – since this is the only way to submit an online query regarding the current processing status: http://www.mathworks.com/accesslogin/createProfile.do

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 – currently R2007b.
Note: Access to individual product documentations, such as "Real-Time Workshop Embedded Coder", requires a "MathWorks Account" – free registration at ***http://www.mathworks.com/accesslogin/createProfile.do***.
http://www.mathworks.com/accesslogin/createProfile.do

MATLAB tutorial

Online tutorial for introduction to MATLAB.
http://www.mathworks.com/academia/student_center/tutorials/launchpad.html

Simulink tutorial

Online tutorial for introduction to Simulink.
http://www.mathworks.com/academia/student_center/tutorials/index.html?link=body#

The MathWorks Newsletter

Access to The 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 The MathWorks solutions in approx. 1 hour long online presentations (in English and German).
http://www.mathworks.com/company/events/archived_webinars.html?s_cid=HP_E_RW

Notes

Overview of training modules

TM200 – B&R Company Presentation **
TM201 – B&R Product Spectrum **
TM210 – The Basics of Automation Studio
TM211 – Automation Studio Online Communication
TM212 – Automation Target **
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)
TM247 – Automation Basic (AB)
TM248 – ANSI C
TM250 – Memory Management and Data Storage
TM260 – Automation Studio Libraries I
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
TM450 – ACOPOS Control Concept and Adjustment
TM460 – Starting up Motors

TM500 – The Basics of Integrated Safety Technology
TM510 – ASiST SafeDESIGNER

TM600 – The Basics of Visualization
TM610 – The Basics of ASiV
TM630 – Visualization Programming Guide
TM640 – ASiV Alarm System
TM650 – ASiV Internationalization
TM660 – ASiV Remote
TM670 – ASiV Advanced

TM700 – Automation Net PVI
TM710 – PVI Communication
TM711 – PVI DLL Programming
TM712 – PVIServices
TM730 – PVI OPC

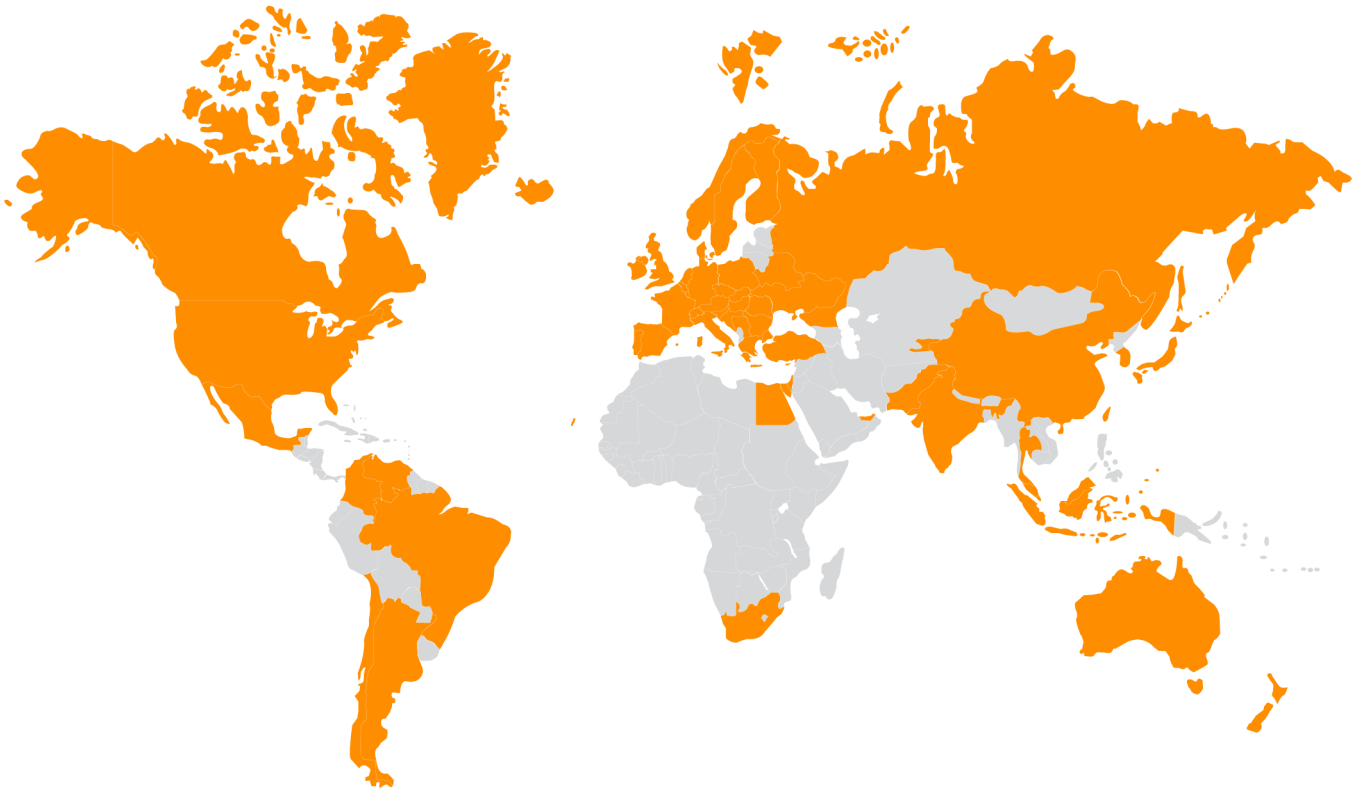
TM800 – APROL System Concept
TM810 – APROL Setup, Configuration and Recovery
TM811 – APROL Runtime System
TM812 – APROL Operator Management
TM813 – APROL XML Queries and Audit Trail
TM830 – APROL Project Engineering
TM840 – APROL Parameter Management and Recipes
TM850 – APROL Controller Configuration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

**) see Product Catalog

KONZERNZENTRALE

Bernecker + Rainer Industrie-Elektronik Ges.m.b.H.
B&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

Immer in Ihrer Nähe - 140 Büros in über 55 Ländern - **www.br-automation.com/contact**



Australia • Argentina • Austria • Belarus • Belgium • 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 • The Netherlands • New Zealand
Norway • Pakistan • Poland • Portugal • Romania • Russia • Serbia • Singapore • Slovakia • Slovenia • South Africa
Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • Ukraine • United Kingdom • USA • Venezuela • Vietnam