APPLICATION NOTE

AC500 WITH SIMULINK PLC CODER

HOW TO USE MATLAB/SIMULINK FOR ST CODE GENERATION

Contents

1 Introduction

1.1 Scope of the document

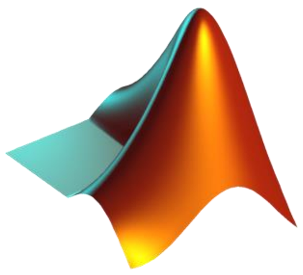
This document shows how MATLAB/Simulink with the Simulink PLC Coder can be used for automated ST code generation for AC500 V2 and AC500 V3 PLCs.

1.2 Compatibility

The application example explained in this document have been used with the below engineering system versions. They should also work with other versions, nevertheless some small adaptations may be necessary, for future versions.

- Automation Builder 2.2.5 or other
- Matlab R2020a or other
- MathWork Licenses for at least MATLAB, Simulink and PLC Coder

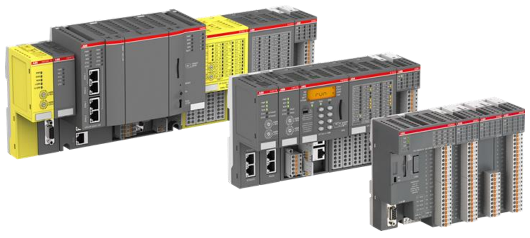
1.3 Overview



Model Design

Model Simulation

PLC Code generation



PLC Code implementation

Testbench verification

Compare between Simulation and real System

As shown above MATLAB/Simulink is used for a Model design, simulation and optimization. If the model is working as expected the PLC Code in Structured Text can be generated with the Simulink PLC Coder. The generated code can be imported and implemented in an AC500 PLC. To verify the code a testbench is included. Afterwards the PLC can be used to control the real application. Differences between the model and the real application are visible and can be used to optimize the system model and correct the system behavior.

2 Matlab, Simulink and PLC Coder

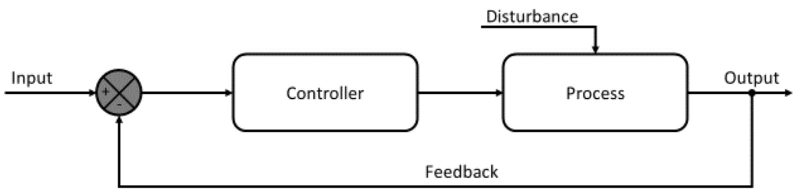
This chapter deals with the model design, simulation and PLC Code generation.

NOTICE

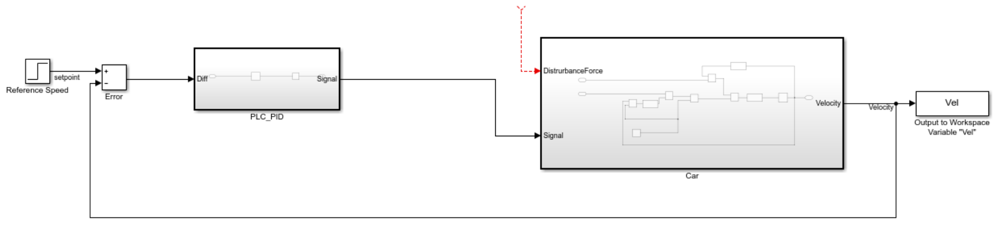
The document is not describing the MATLAB and Simulink Basics. Elementary knowledge is required to follow this description

2.1 Model creation and simulation

As base for this example a closed loop system is used.

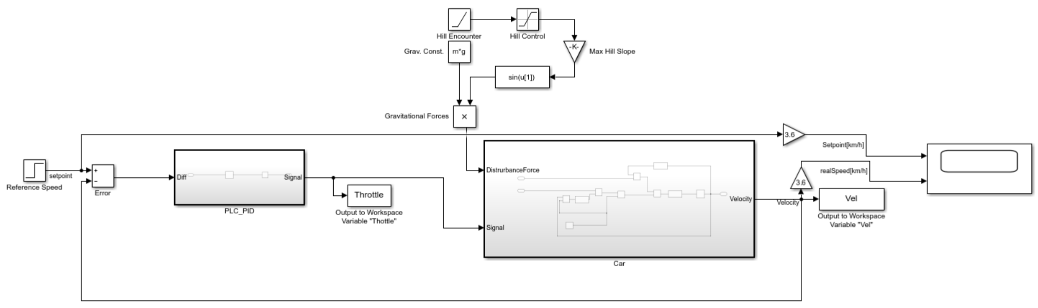


To discuss and simulate the general example above a car cruise control is used.



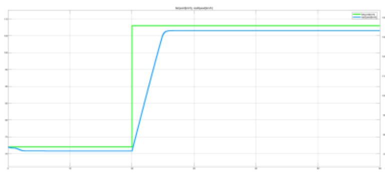
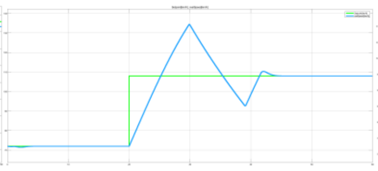
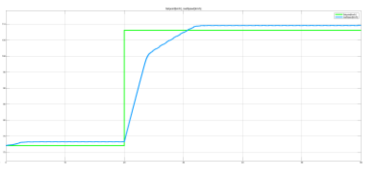
With the Ramp Block Reference Speed, the user specifies the speed the car should have. In the first step the current velocity (feedback) is compared to the desired velocity (input). The speed difference is given to the controller. In this example a PID controller is used, which shall be realized by a PLC.

The controller calculates his output values. The output is the percentage how much energy the car engine gets. This value is inputted as Signal to the Car block. The car speed is not only influenced by the gas but also by the load or the air resistance. These factors are part of the Car model itself and inside the subsystem. External Forces like wind or gradient are summarized as Disturbance Forces. Here nothing is connected. The process itself has the velocity as output which is given back to compare. To simulate and test the model some more logic is added.[1]

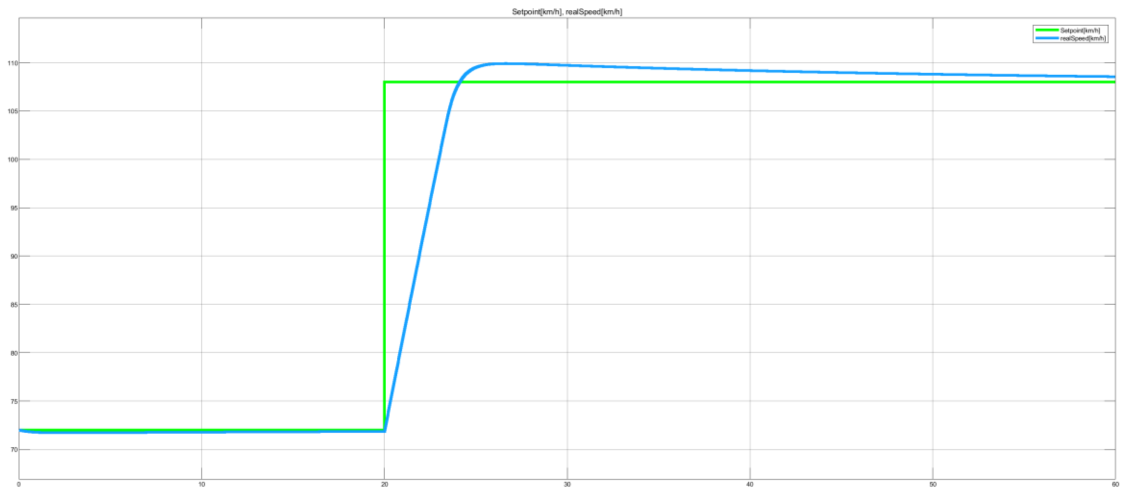


[1]Cf. Shreesh Mysore, Richard Murray http://www.cds.caltech.edu/~murray/amwiki/Cruise_control (18.05.2020)

As Disturbance Force a hill can be used. To compare the Setpoint and the real velocity a scope is added, where both signals are inputted. As the velocity is in m/s in the model it is multiplied by 3.6 to get the velocity in km/h.
For the model testing a small gradient starts after 5 seconds. After 20 seconds the setpoint is increased. Depending on the PID factors in the PLC Controller the behavior of the acceleration process can be changed. The following three little pictures are showing the response from not optimized P, PI and PD controllers.

In the simulation the PID Filter can be optimized and the robustness can be checked with the disturbance force. As visible in the picture below the setpoint changes from 72 km/h to 108 km/h. the cruise control accelerates to follow this new speed.



As the model in Simulink is now working as it should it can be transferred into PLC Code for the AC500. As the PLC has a defined cycle but Simulink can work with continues time signals only discrete blocks allowed in the subsystem for PLC Code generation. In the next chapter the steps for PLC Code generation are described.

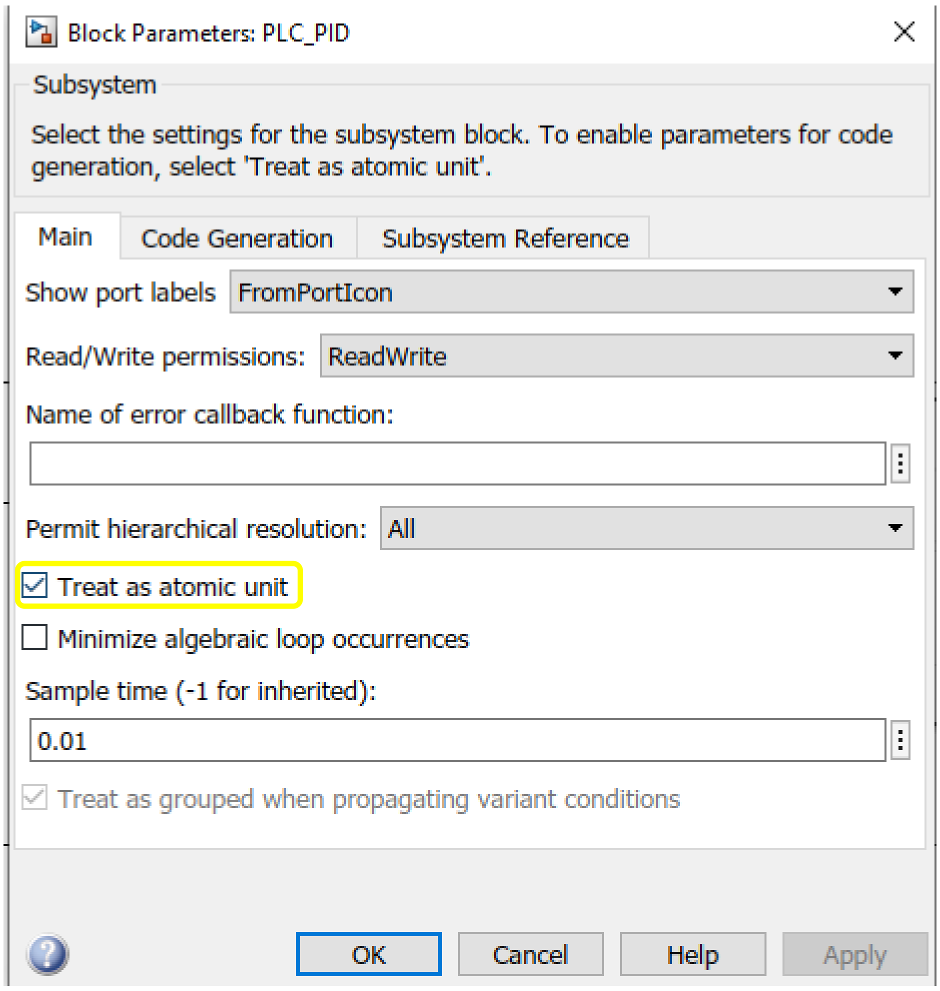
2.2 PLC Code Generation

This chapter describes how ST code for AC500 V2 and AC500 V3 can be created with the Simulink PLC Coder.

The Simulink Subsystem PLC_PID from the model described in the chapter above shall be transferred into PLC Code. Therefor no continuous block must be used. In case a block has to be discretized the Model Discretizer which can be found in the Apps can be used.

As the application is depending on the PLC cycle time the sample time of the discrete blocks can be set.

A subsystem which can be used for code generation has to be an atomic unit. Therefore, open the Subsystem Parameters and select the checkbox Treat as atomic unit as shown in the picture below.



Block Parameters: PLC_PID

Subsystem

Select the settings for the subsystem block. To enable parameters for code generation, select 'Treat as atomic unit'.

Main | Code Generation | Subsystem Reference

Show port labels: FromPortIcon

Read/Write permissions: ReadWrite

Name of error callback function:

Permit hierarchical resolution: All

[✓] Treat as atomic unit

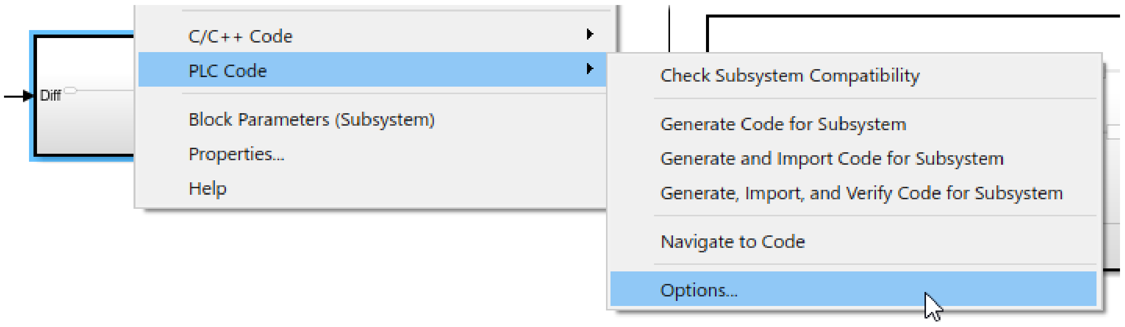
[ ] Minimize algebraic loop occurrences

Sample time (-1 for inherited): 0.01

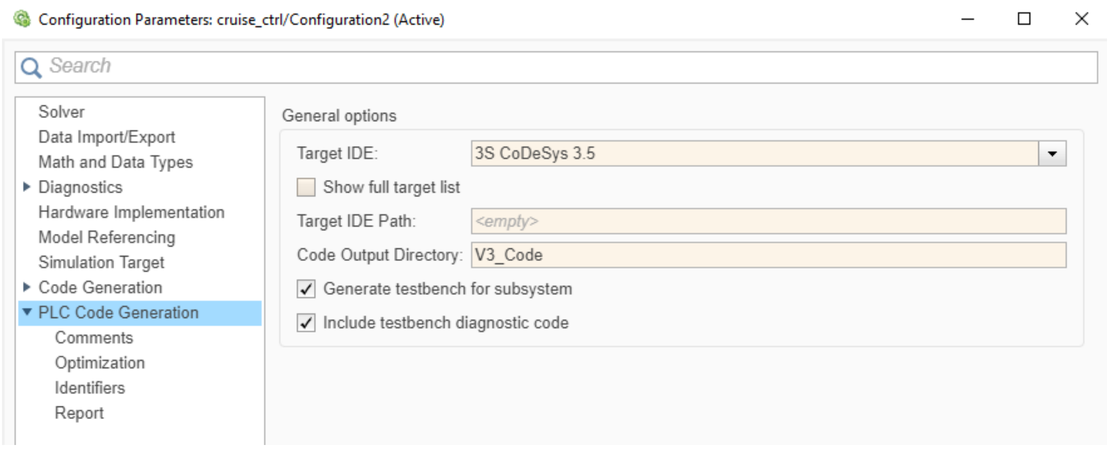
[✓] Treat as grouped when propagating variant conditions

After the steps above the subsystem is ready for code generation. By right clicking: PLC Code, Check Subsystem Compatibility the user can check if there are any issues in the subsystem which prevent the code generation.

The next step is to define the options for PLC Code generation.


Diff
C/C++ Code
PLC Code
Block Parameters (Subsystem)
Properties...
Help
Check Subsystem Compatibility
Generate Code for Subsystem
Generate and Import Code for Subsystem
Generate, Import, and Verify Code for Subsystem
Navigate to Code
Options...

Depending on the used PLC the Target IDE can be selected. For AC500 V2 PLCs 3S CoDeSys 2.3 has to be selected. For AC500 V3 PLCs 3S CoDeSys 3.5 or PLC Open XML has to be selected. In this application note the code generation for a V3 PLC is described. The steps for a V2 are nearly the same. The differences are short commented in this note.


Configuration Parameters: cruise_ctrl/Configuration2 (Active)

General options

Target IDE: 3S CoDeSys 3.5

☐ Show full target list

Target IDE Path: <empty>

Code Output Directory: V3_Code

☑ Generate testbench for subsystem

☑ Include testbench diagnostic code

The Code Output Directory is the path where the created PLC Code is saved.

Generate testbench for subsystem and Include testbench diagnostic code can be used to create a testbench. Simulink creates an array with input variables for the function block and an array with the output variables. The testbench loops through all input variables and compares the calculated PLC output with the output from the Simulink simulation. The user can check if the behavior of this function block is different in the PLC and the simulation.

After defining the options, the PLC Code can be generated. Therefor right click on the Subsystem again, select PLC Code and Generate Code for Subsystem. This might take several minutes.

```
### Emit PLC code to file.
### Creating PLC code generation report cruise_ctrl_codegen_rpt.html.
### PLC code generation successful for 'cruise_ctrl/PLC_PID'.
### Generated files:
V3_Code\cruise_ctrl.xml
```

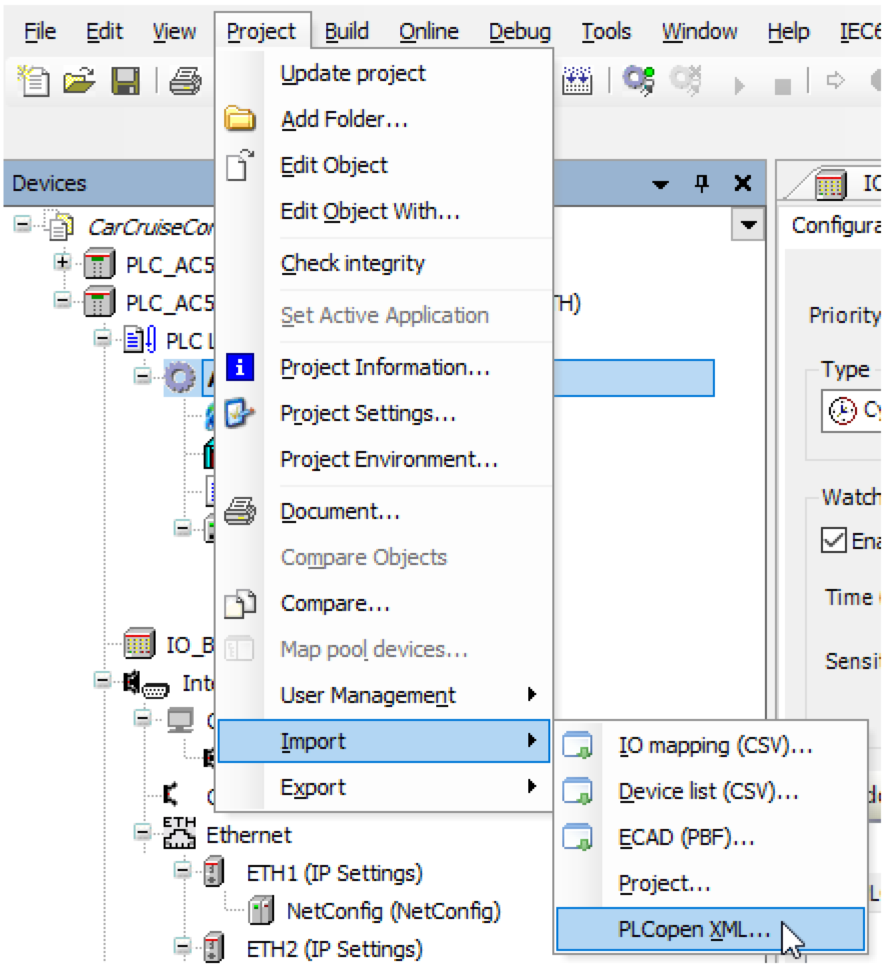
Component: PLC Coder | Category: PLC Coder

A report for the code generation is created in addition to the PLC Code. The progress can be followed in the Diagnostic Viewer.

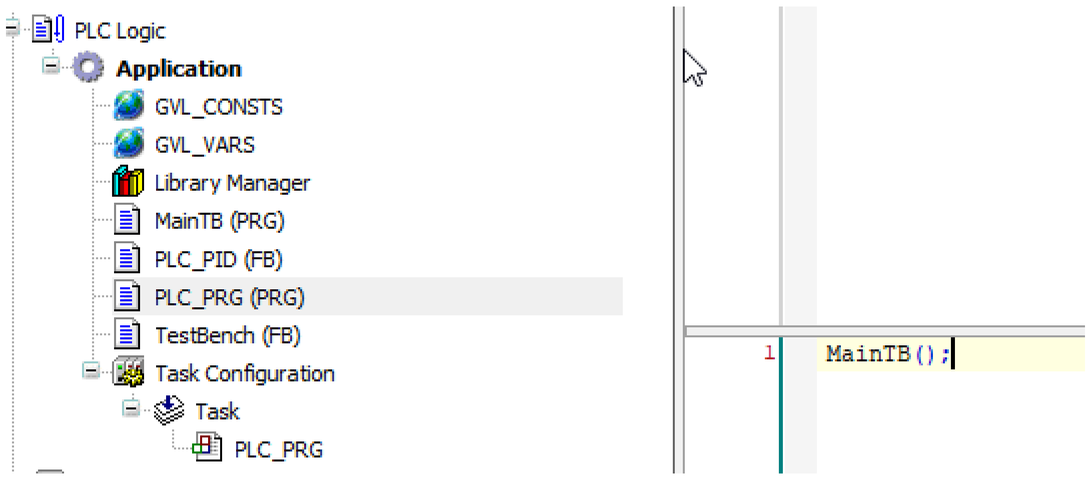
3 Automation Builder and CoDeSys

3.1 Code Import and Testbench

In Automation Builder the generated code can be imported. To import the code, select the Application and navigate Project – Import – PLC open XML to import the generated code. The Global Variable List GVL_CONSTS and GVL_VARS as well as the POUs MainTB, PLC_PID and TestBench are added to the project.

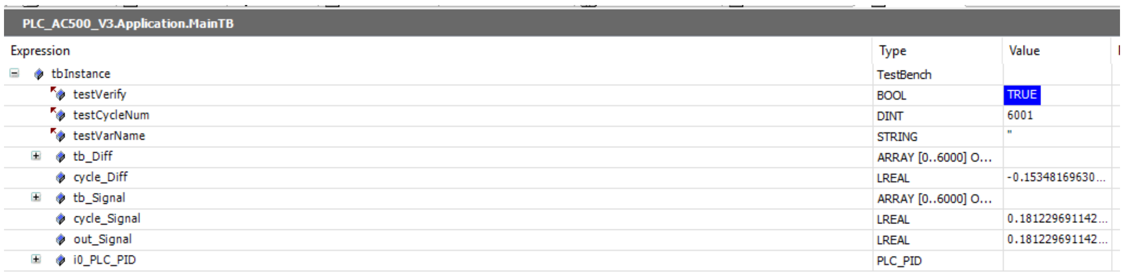


To run the Testbench the MainTB has to be called by a task or inside the PLC_PRG. Please make sure, that the task which is calling this function block has the corresponding cycle time to sample distance in Simulink. Here a sample distance of 0.01 s was configured so a 10 ms cyclic task is used.



```
MainTB();
```

When changing to run the testbench is executed. As visible from the screenshot below the instance from TestBench stepped through the 6000 array entries. As the variable testVerify is still true the test was successful.



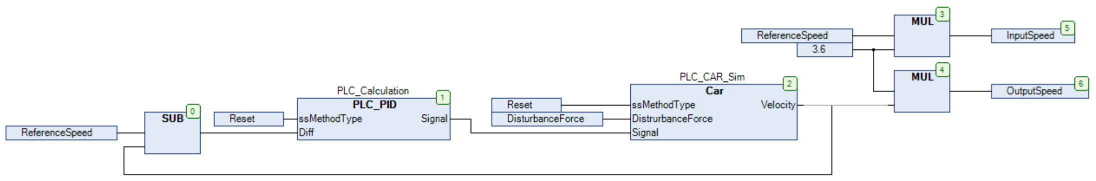
PLC_AC500_V3.Application.MainTB

Expression	Type	Value
tbInstance	TestBench	
testVerify	BOOL	TRUE
testCycleNum	DINT	6001
testVarName	STRING	''
tb_Diff	ARRAY [0..6000] O...	
cycle_Diff	LREAL	-0.15348169630...
tb_Signal	ARRAY [0..6000] O...	
cycle_Signal	LREAL	0.181229691142...
out_Signal	LREAL	0.181229691142...
i0_PLC_PID	PLC_PID	

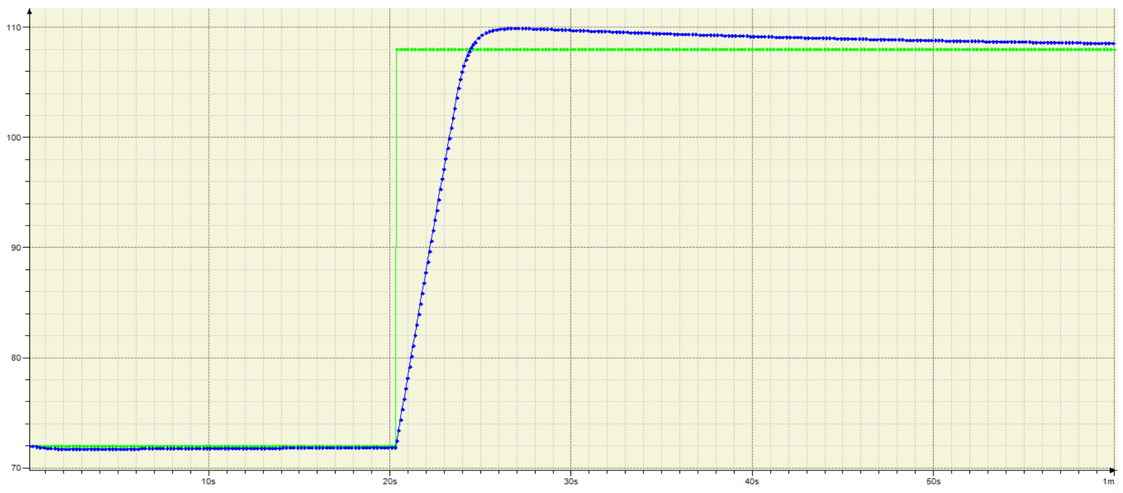
3.2 Compare Automation Builder & Simulink

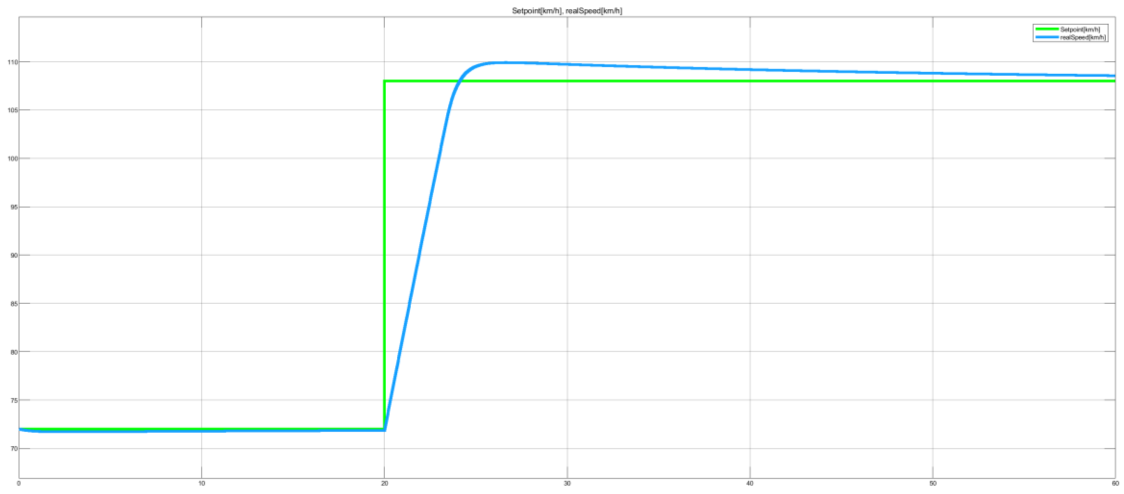
In the last chapter was shown that the output from the PLC_PID block is equal in Simulink and Automation Builder.

In Simulink the set Velocity is compared to the real velocity. To have the same layout in Automation Builder the Car block in Simulink is also converted into ST Code and the closed loop system is rebuild in CFC



Similar to the Scope in Simulink the Trace can be used to watch and compare the two velocity signals. The two pictures below show the compare between the trace in the AC500 and the Scope in Simulink. As visible the two graphs are identical.


110
100
90
80
70
10s
20s
30s
40s
50s
1m

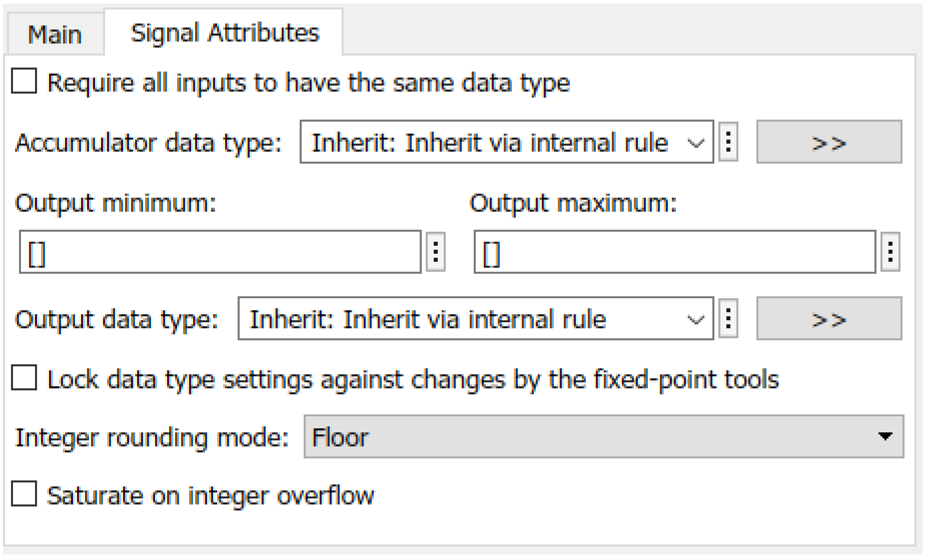

Setpoint[km/h], realSpeed[km/h]
Setpoint[km/h]
realSpeed[km/h]
110
105
100
95
90
85
80
75
70
0
10
20
30
40
50
60

4 Hints and possible issues

4.1 Data Types

By default, Simulink sources are using double values. This is corresponding with LREAL Values in the AC500. Calculations with floating point values take more CPU time than calculations with fixed point values especially when using a PM57X, PM58X or an ECO CPU. If possible, use Integer values instead to optimize the execution speed and reduce the CPU Load.

The data types are by default inherit. As the sources are double all Signal inputs and outputs will be double.



☐ Require all inputs to have the same data type

Accumulator data type: Inherit: Inherit via internal rule

Output minimum: []

Output maximum: []

Output data type: Inherit: Inherit via internal rule

☐ Lock data type settings against changes by the fixed-point tools

Integer rounding mode: Floor

☐ Saturate on integer overflow

The data type in Simulink has to be defined depending on the data types which shall be used in CoDeSys. Following table shows the corresponding data types in CoDeSys and Simulink.

Fixed point data types

CoDeSys	Matlab	Size (byte)
BYTE	uint8	1
WORD	uint16	2
DWORD	uint32	4
LWORD	uint64	8
(U)SINT	(u)int8	1
(U)INT	(u)int16	2
(U)DINT	(u)int32	4
(U)LINT	(u)int64	8

Boolean data types

CoDeSys	Matlab	Size (byte)
BOOL	boolean	1

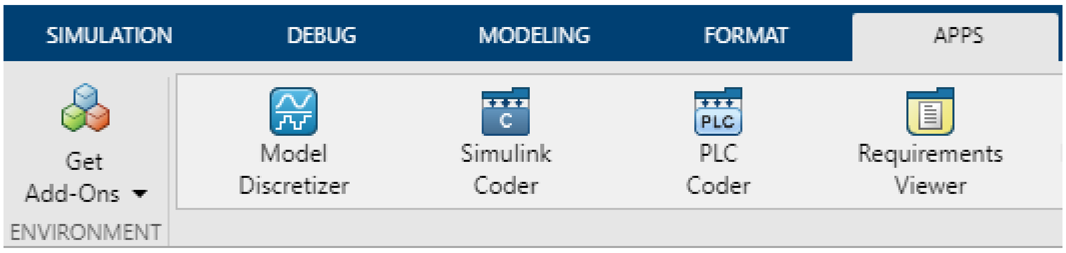
Floating point data types

CoDeSys	Matlab	Size (byte)
REAL	single	4
LREAL	double	8

4.2 Errors in PLC Code Generation

There are many different reasons why a code generation is not possible. At first please make sure that the Simulink model can be simulated by pressing run.

The most common issue is using not allowed blocks. For PLC Code generation continuous blocks are not allowed. These blocks need to be replaced by a discrete block. Therefor the Model Discretizer in the section apps can be used.



4.3 Test Bench reports a difference

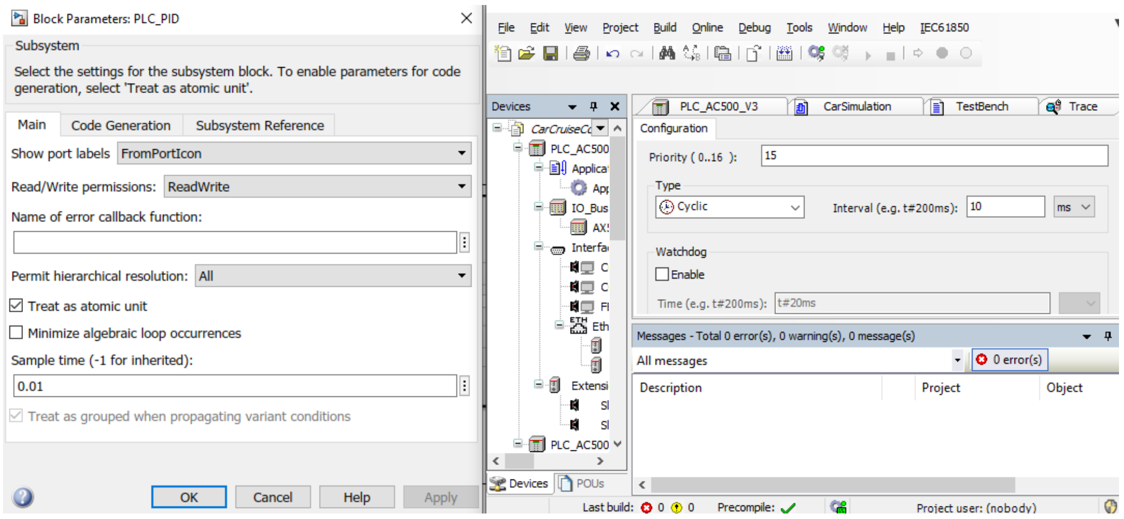
If the test bench reports an error, there is a difference between the PLC Code and the Simulink Model. The root case for this difference can for example be differences in calculations, behavior in error cases, …

Please check if already the calculations in the first cycle fails. This indicates a general difference. To check the difference the step function in Simulink can be used to find the initial values and the values after the first cycle. These values can be compared to the PLC Values to find the difference.

In special calculations cases the two systems might have differences. This can be data type conversations, integer overflow, dividing by 0, …

4.4 Timings between Automation Builder and Simulink are different

If the testbench runs successful but the PLC output is faster or slower as the simulation, it is an indication that the plc cycle time is different from the simulation sample time.


Block Parameters: PLC_PID

Subsystem

Select the settings for the subsystem block. To enable parameters for code generation, select 'Treat as atomic unit'.

Main | Code Generation | Subsystem Reference

Show port labels: FromPortIcon

Read/Write permissions: ReadWrite

Name of error callback function:

Permit hierarchical resolution: All

☑ Treat as atomic unit

☐ Minimize algebraic loop occurrences

Sample time (-1 for inherited): 0.01

☑ Treat as grouped when propagating variant conditions

OK | Cancel | Help | Apply

PLC_AC500_V3 | CarSimulation | TestBench | Trace

Configuration

Priority (0..16): 15

Type: Cyclic — Interval (e.g. t#200ms): 10 ms

Watchdog: ☐ Enable — Time (e.g. t#200ms): t#20ms

Messages - Total 0 error(s), 0 warning(s), 0 message(s)

All messages — 0 error(s)

Description	Project	Object

Last build: 0 0 Precompile: ✓ Project user: (nobody)

As visible in the screenshot above the cycle time in Simulink (0.01 s) and in Automation Builder (10 ms) are the same.

—

ABB Automation Products GmbH
Eppelheimer Straße 82
69123 Heidelberg, Germany
Phone: +49 62 21 701 1444
Fax: +49 62 21 701 1382
E-Mail: plc.support@de.abb.com
www.abb.com/plc