



VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Mika Niemi

Simulink to PLC Compatibility

A Heating System

Tekniikka
2023

TIIVISTELMÄ

Tekijä	Mika Niemi
Opinnäytetyön nimi	Simulink to PLC Compatibility
Vuosi	2023
Kieli	englanti
Sivumäärä	36 + 2 liitettä
Ohjaaja	Smail Menani

Opinnäytetyön tarkoituksena oli tutkia MATLAB Simulinkin PLC Coder -ohjelman ja Siemensin TIA Portal -ohjelman käyttöä ohjelmoitavien logiikkaohjaimien (PLC) konfiguroinnissa. Simulink- ja TIA Portal -ohjelmia käytetään VAMK:n tietotekniikan ja sähkö- ja automaatiotekniikan kursseilla.

Tutkimuksen tavoitteena oli PLC-laitteen liittäminen lämpösensoriin, lämmityslähteeseen ja ulkoiseen virtalähteeseen. Simulink -ohjelma on simulaatioympäristö, jossa voi tehdä erilaisia simuloitavia systeemiä. Opinnäytetyössä Simulink -ohjelmaa käytetään lämpösystemin simulointiin ja siinä käytettävien sisään- ja ulostulo porttien logiikan muodostamiseen. Simulink -systeemi siirretään TIA Portal ohjelmaan PLC Coder -lisäosan avulla. PLC Coder käyttää IEC 61131- standardia, kun se muokkaa tiedostot logiikkaohjaimille.

Tutkimuksen tarkoituksena on tukea Simulinkin ja TIA Portal -ohjelmien samanaikaista oppimista ja käyttöä. Tutkimuksessa käytetään TIA Portal V16 versiota ja MATLAB R2021a versiota. PLC -laitteita käytetään teollisuuden monilla aloilla, ja Simulink -ohjelma on käytössä monilla yrityksillä erilaisten simulaatioiden luomisessa ennen kuin testejä tehdään oikeilla laitteilla. Tutkimusaineistona oli Siemensin omat manuaalit PLC laitteille ja MathWorksin ohjeet Simulinkin käyttöön.

Opinnäytetyön tuloksena on systeemi, josta voi mitata lämpötilan TIA Portal -ohjelmalla, joka on tehty Simulinkin avulla. Tätä opinnäytetyötä voidaan tulevaisuudessa käyttää ohjelmien opetuksessa Vaasan ammattikorkeakoulun sähkötekniikan ja tietotekniikan kursseilla sulautettujen järjestelmien opinnoissa.

ABSTRACT

Author	Mika Niemi
Title	Simulink to PLC Compatibility
Year	2023
Language	English
Pages	36 + 2 Appendices
Name of Supervisor	Smail Menani

The purpose of this study was to investigate the use of MATLAB Simulink's PLC Coder software and Siemens' TIA Portal program in configuring programmable logic controllers (PLCs). Simulink and TIA Portal programs are used in VAMK's information technology and electrical engineering and automation technology courses.

The aim of this research was to connect a PLC device to a heat sensor, a heating source, and an external power source. Simulink is a simulation environment where simulated systems can be created. In this thesis Simulink was used to model a heating system and its input and output ports. The Simulink model of the heating system was then transferred using Simulink PLC Coder, which generates files for TIA Portal. The PLC Coder follows the standard IEC 61131 when generating files.

The purpose of the thesis was to create a system to support the simultaneous learning and use of Simulink and TIA Portal programs. The programs used are TIA Portal V16 version and MATLAB R2021a version with Simulink and PLC Coder add-on for Simulink. PLC computers are used in many industrial areas and Simulink is used by many companies to create different simulations before any tests are performed with real hardware. The research consisted of Siemens' own manuals for PLC devices and MathWorks' instructions for using Simulink.

The thesis resulted in a system where the temperature can be measured with the TIA Portal program, which was made using Simulink. Following this, the results of the thesis can be used in teaching Simulink and TIA Portal in electrical engineering and information technology for embedded system courses at Vaasa University of Applied Sciences.

Keywords Programmable logic controller, Simulink, simulation, TIA Portal, Siemens, and IEC 61131

CONTENTS

1	INTRODUCTION	9
1.1	Objectives.....	9
2	PROGRAMMABLE LOGIC CONTROLLERS	11
2.1	TIA Portal.....	13
2.2	Controlling the PLC	14
3	MATLAB	17
3.1	Simulink.....	17
3.2	PLC Coder	17
3.3	Generation of Files.....	19
3.3.1	TIA Portal Configuration.....	23
3.3.2	PLC Tags.....	23
4	HEATING SYSTEM	25
4.1	Equipment Used in the System.....	25
4.2	Configuration of Analog Input for PLC	27
4.3	Heater Model from Simulink to TIA Portal	30
4.3.1	Measurement Ranges	30
4.3.2	Switch Module Used	31
4.4	Combining all systems in TIA Portal.....	31
4.5	Hardware	33
5	CONCLUSION AND FURTHER DEVELOPMENT	35
	REFERENCES	36
	APPENDICES	

LIST OF FIGURES AND TABLES

Figure 1. Siemens S7-1200 PLC	p.12
Figure 2. Device overview of S7-1200	p.13
Figure 3. TIA Portal block menu	p.14
Figure 4. Programming language menu	p.15
Figure 5. Structured Control Language file	p.16
Figure 6. Target IDE list for generated code	p.18
Figure 7. Fixed- and variable step solver comparison	p.20
Figure 8. Simulink solver settings	p.20
Figure 9. PLC Code Generation settings	p.21
Figure 10. Block parameters settings	p.21
Figure 11. Atomic subsystem and PLC Code ready for generation	p.22
Figure 12. Generated file data types	p.23
Figure 13. Database block for ON/OFF switch	p.23
Figure 14. Default tag table for heating system	p.24
Figure 15. Adjustable power supply for PTC heater	p.25
Figure 16. PeakTech TF-20 Thermocouple	p.25
Figure 17. PTC heater used in the system	p.26
Figure 18. Carlo Gavazzi ZRLS1-2NA relay and connector block	p.26
Figure 19. K-Type Thermocouple to VDC 0-10V converter	p.27
Figure 20. Normalization and scale blocks	p.28

Figure 21. Analog Scale block	p.29
Figure 22. Analog scale with optimized values	p.29
Figure 23. Heat indicator system with values in Simulink	p.31
Figure 24. Simulink switch module	p.31
Figure 25. TIA Portal project tree	p.32
Figure 26. TIA Portal view of switch turned on and off	p.32
Figure 27. TIA Portal view of heat indicator values	p.33
Figure 28. PLC and components connected to it	p.34
Figure 29. PTC Heater and thermocouple in a steel container	p.34
Table 1. Datatype table for Siemens PLCs	p.19

ABBREVIATIONS

PLC = Programmable Logic Controller

IEC = International Electrotechnical Commission

IDE = Integrated Development Environment

TIA Portal = Totally Integrated Automation Portal

I/O = Input/Output

FBD = Function Block Diagram

SCL = Structured Control Language

LAD = Ladder Logic

CPU = Central Processing Unit

Int8 = 8-bit signed integer

SINT = Signed Integer (8-bit)

PTC = Positive Temperature Coefficient

DC = Direct Current

V = Volt

TMH - Temperatur Messelemente Hettstedt GmbH

LIST OF APPENDICES

APPENDIX 1. Instructions on how to use

APPENDIX 2: Code generated for the system using PLC Coder

1 INTRODUCTION

1.1 Objectives

The main objective of the thesis was to create a working model with Simulink that can be exported to Totally Integrated Automation (TIA) Portal using Programmable Logic Controller (PLC) Coder. The secondary objective was to have a second model and a temperature environment to test actual values. The goal of these objectives was to gather knowledge on PLC systems and the connectivity between the two software.

1.2 Background and Motivations

PLCs are used in a variety of different workplaces that require automation, so to gain more knowledge on the subject and create new methods are the reasons why this topic was chosen. There is a lack of instructions and compatibility information of Simulink to PLC, the situation which hopefully is improved with the results of this thesis.

PLCs are used in many applications, from a washing machine at home to an industrial factory controlling multiple outputs. A PLC is a computer which is built to withstand industrial environments such as differing temperatures, moisture, and dust.

The first versions of PLCs were developed by Richard E. Dick Morley in 1968 while working at Bedford and Associate. Morley is considered as one of the fathers of PLCs, but never took full credit saying that "it was really invented by 50 people, each of whom invented half of it!". (Dunn 2009)

PLCs were then integrated first to car manufacturers' factories, such as General Motors and General Electric. PLCs replaced the relay-based cabinets that needed to be controlled individually. The introduction of PLCs made the old way of using relay-based control systems expensive and time consuming as they lack the accessibility, which a PLC has, such as the ability to program many different modules and controls in different circumstances.

The topic of this thesis work is to test MATLAB Simulink's program Simulink PLC Coder. Simulink PLC Coder is an add-on to Simulink which allows generating hardware-independent IEC 61131 compatible structured control language or ladder diagrams from Simulink models. The Simulink modules are generated as different documents which can then be deployed on different PLC software, such as TIA Portal that Siemens systems use.

2 PROGRAMMABLE LOGIC CONTROLLERS

PLCs are small computers designed to withstand high temperatures, vibration, and electrical noise. "A PLC can be thought of as a 'ruggedized' digital computer" is how a PLC is described by Vidya Muthukrishnan (2021).

At first their main purpose was to control industrial automation process but have since moved to control various things ranging from traffic control lights to roller coaster systems in amusement parks. How a PLC works is described well by Ben Wayand (2020): "The PLC receives data from associated sensors or information devices, processes the information, and triggers outputs dependent on pre-customized parameters". Signal values are the current and electricity fluctuations that the PLC operates on after the logic is set by the user thus being automated.

PLCs operate on a scan cycle continuously. This means it reads its own inputs, does a program scan, and changes its output according to its programming. The PLC continues this cycle any time a CPU or an I/O module check triggers the scan. (Wayand 2020)

The benefits to using a PLC over a relay-based control system are:

- Flexibility to change and fix according to problems or changes made to overall systems
- More reliable as PLCs are made to withstand difficult industrial and outside environments and PLC components tend to last longer
- Faster response time
- Cost effective for a large or complex system and smaller in size
- Troubleshooting and programming ease with integrated development environments and used standards

The cost effectiveness of a PLC compared to relay-based control systems comes from not having to wire every single module and switch to a single relay or a relay board, but to the PLC and it then controls all of them. PLCs are controlled with each manufacturer's own integrated development environment (IDE), for example

TIA Portal. The IDE works as a station in which the user can change the parameters that the PLC follows. (Egan 2021, Brown 2000)

Every PLC has three main modules:

- Central processing unit (CPU): The CPU is the center of the PLC. It performs different tasks that it accesses from its memory
- Power supply: Transfers the 24 VDC current for the CPU, its memory and I/O electric circuits
- Input and output ports: Bridges to connect different input and output devices such as sensors or switches to the PLC

PLCs are controlled with software (IDE) that it comes with and communication to a PLC goes through an ethernet cable to an RJ45 ethernet port.

The PLC used in this thesis is a Siemens SIMATIC S7-1200 PLC. It has a 1214C DC/DC/DC CPU, 100 kilobyte integrated work memory and 4 Megabyte of load memory.



Figure 1. Siemens SIMATIC S7-1200 PLC

Device overview							
Module	Slot	I address	Q address	Type	Article no.	Firmware	
	103						
	102						
	101						
▼ PLC_1	1			CPU 1214C DC/DC/DC	6ES7 214-1AG40-0XB0	V4.2	
DI 14/DQ 10_1	1 1	0...1	0...1	DI 14/DQ 10			
AI 2_1	1 2	64...67		AI 2			
	1 3						
HSC_1	1 16	1000...10...		HSC			
HSC_2	1 17	1004...10...		HSC			
HSC_3	1 18	1008...10...		HSC			
HSC_4	1 19	1012...10...		HSC			
HSC_5	1 20	1016...10...		HSC			
HSC_6	1 21	1020...10...		HSC			
Pulse_1	1 32		1000...10...	Pulse generator (PTO/P...			
Pulse_2	1 33		1002...10...	Pulse generator (PTO/P...			
Pulse_3	1 34		1004...10...	Pulse generator (PTO/P...			
Pulse_4	1 35		1006...10...	Pulse generator (PTO/P...			
► PROFINET interface_1	1 X1			PROFINET interface			

Figure 2. Device overview of the S7-1200 PLC which shows the PLCs general information ranging from number of ports to what firmware its run on

As seen in Figure 2, the PLC has 2 analog inputs (AI), 14 digital inputs (DI) and 10 digital outputs (DQ). These are controlled with addresses which are later labeled %IW64 for analog input, %IO.0 and %Q0.0 for digital inputs and outputs. The PLC is run on firmware update V4.2 and is controlled using V16 of TIA Portal.

PLCs have an international standard, IEC 61131, in which the standard languages, safety and requirements of PLCs are universal between different manufacturers. The IEC 61131 standard was first published in 1993 and the current version 3 has its stability date set for 2025. For an engineer, this means that switching to another PLC only requires getting to know how to control different ports and the ins and outs of their IDE, not the language that every PLC is able to operate with. (IEC webstore)

2.1 TIA Portal

TIA Portal is software combining multiple previous Siemens' platforms, -such as SIMATIC STEP and SIMATIC WinCC into one hub (Siemens page on TIA Portal, Software in TIA Portal). TIA Portal is used to control, monitor, and program the PLC. TIA Portal has the hundreds of different PLCs available with their different versions that Siemens has released. (Siemens, n.d., online)

2.2 Controlling the PLC

The controlling of a PLC happens in their manufacturers IDE. Organization blocks are the main block of a system in an IDE. They can have different functions in them or be a blank slate that a function block can then be attached to. Organization blocks can have a variety of different settings from controlling a servo motor to different kinds of interrupts as seen in Figure 3, but in this thesis, it is used as program cycle that loops a function block.

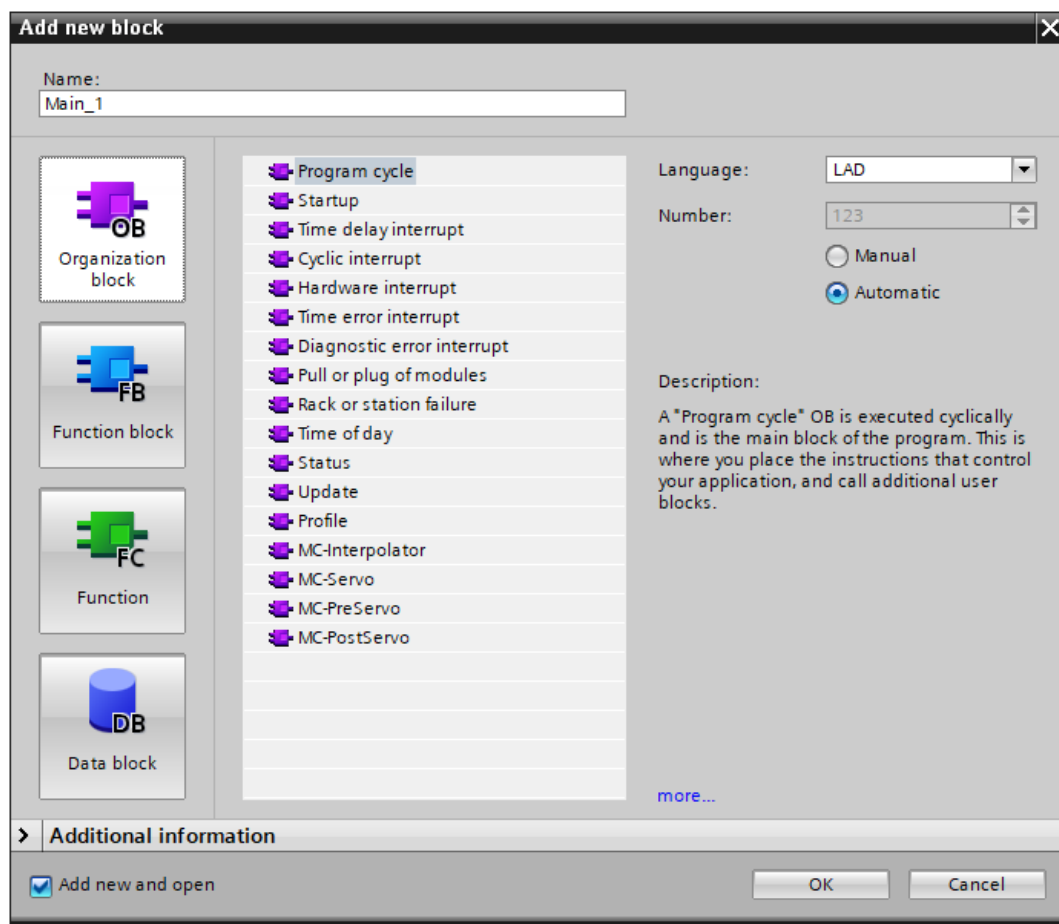


Figure 1. Block adding menu which has different pre-made organization blocks and a selection for language that the program is then made of

The programming language options for TIA Portal V16 and 1214CPU PLC has are LAD, FBD and SCL, as seen in Figure 4.

- LAD: Ladder Diagrams, is a graphical form of creating instructions and functions for the PLC. It is also referred as Ladder Logic
- FBD: Function Block Diagram, is also a graphical form of controlling a PLC. It is a beginner friendly to users with programming background since it has very clear blocks with their inputs and outputs
- SCL: Structured Control Language, is a text-based way of controlling a PLC. SCL is based on the Pascal programming language

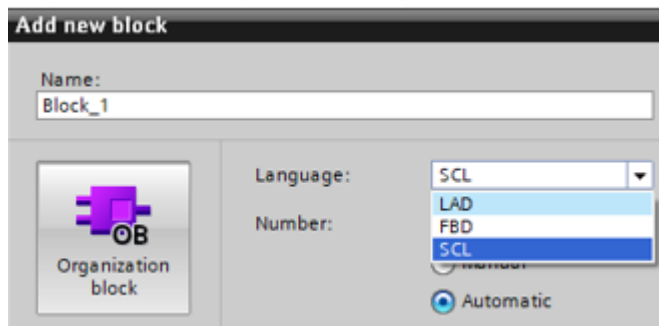


Figure 4. Programming language choices with TIA Portal

This thesis includes two languages, FBD and SCL languages. FBD is used to create normalization blocks for the temperature, more of it in Chapter 4.2 and SCL is used by PLC Coder when it generates its files from the Simulink model (Figure 5).

IF...	CASE... OF...	FOR... TO DO...	WHILE... DO...	(*...*)	REGION
-------	------------------	--------------------	-------------------	---------	--------

```

1 CASE #ssMethodType OF
2     0:
3         (* SystemInitialize for Atomic SubSystem: '<Root>/Atomic Subsystem' *)
4         (* SystemInitialize for IfAction SubSystem: '<S1>/If Action Subsystem' *)
5         (* SystemInitialize for Output: '<S2>/Out1' incorporates:
6         *   Inport: '<S2>/In1'
7         *   Outport: '<Root>/Out1' *)
8         #Out1 := FALSE;
9         (* End of SystemInitialize for SubSystem: '<S1>/If Action Subsystem' *)
10
11        (* SystemInitialize for IfAction SubSystem: '<S1>/If Action Subsystem1' *)
12        (* SystemInitialize for Output: '<S3>/Out1' incorporates:
13        *   Inport: '<S3>/Input 1'
14        *   Outport: '<Root>/Out2' *)
15        #Out2 := FALSE;
16        (* End of SystemInitialize for SubSystem: '<S1>/If Action Subsystem1' *)
17        (* End of SystemInitialize for SubSystem: '<Root>/Atomic Subsystem' *)
18    1:
19        (* Outputs for Atomic SubSystem: '<Root>/Atomic Subsystem' *)
20        (* If: '<S1>/If' incorporates:
21        *   Constant: '<S1>/Constant'
22        *   Constant: '<S1>/Constant1'
23        *   Inport: '<S2>/In1'
24        *   Inport: '<S3>/Input 1'
25        *   Outport: '<Root>/Out1'
26        *   Outport: '<Root>/Out2' *)
27        IF #sensorvalue < 2 THEN
28            (* Outputs for IfAction SubSystem: '<S1>/If Action Subsystem' incorporates:
29            *   ActionPort: '<S2>/Action Port' *)
30            #Out1 := TRUE;
31            (* End of Outputs for SubSystem: '<S1>/If Action Subsystem' *)
32        ELSE
33            (* Outputs for IfAction SubSystem: '<S1>/If Action Subsystem1' incorporates:
34            *   ActionPort: '<S3>/Action Port' *)
35            #Out2 := FALSE;
36            (* End of Outputs for SubSystem: '<S1>/If Action Subsystem1' *)
37        END_IF;
38        (* End of If: '<S1>/If' *)
39        (* End of Outputs for SubSystem: '<Root>/Atomic Subsystem' *)
40 END_CASE;

```

Figure 5. Structured Control Language file generated from Simulink

3 MATLAB

MathWorks' MATLAB is a programming and numeric computing environment that is used to create models, analyze data, and develop algorithms. MATLAB has a vast library of toolboxes ranging from control system models to machine learning.

Version 2021a of MATLAB was used throughout this thesis.

3.1 Simulink

MATLAB's most popular toolbox is a graphical programming tool Simulink. "Simulink is a block diagram environment for multidomain simulation and Model-Based Design" (Mathworks Simulink TIA Portal). It is used to make simulations and used with MATLAB to calculate and simulate complex systems. The main industries that use Simulink are fields that require meticulous testing and using a simulation tool for testing cuts a lot of costs from traditional testing, for example, in the automotive industry, creating a simulation of the car engine and how it performs in certain situations. MATLAB and Simulink work together seamlessly since they share the same libraries and Simulink is opened via MATLAB to make it compatible.

3.2 PLC Coder

PLC Coder is software for MATLAB specifically for Simulink. It allows generation of a Simulink model to IEC-61131 standard language for many integrated development environments as seen in Figure 6. PLC Coder was first introduced in 2010.

Using PLC Coder to create systems for a PLC can take a while to get used to. For a user that has never used MATLAB or Simulink, using PLC Coder to create a system for a PLC will take more time than doing it in PLC's own IDE. But for a user who knows Simulink and has an idea of what the system needs to have, PLC Coder can reduce the time used in simulating the system and in generating the code that the system will run on.

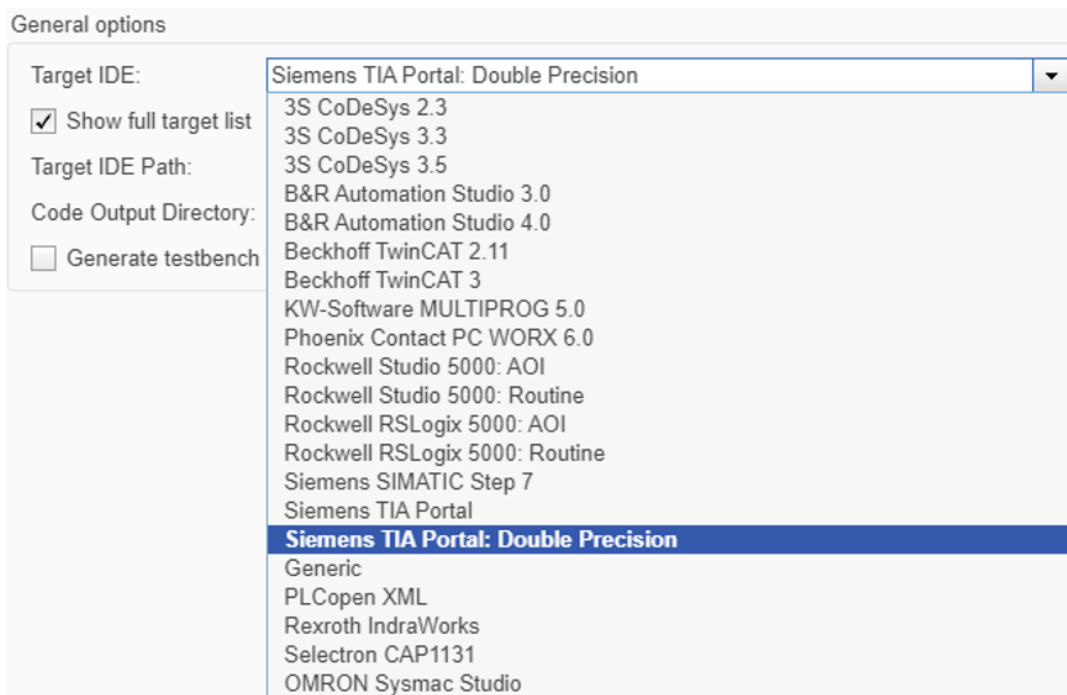


Figure 6. Target IDE list that PLC Coder can generate code for

Siemens' systems have two options when generating a file for the PLC. TIA Portal and TIA Portal: Double Precision, the difference between the two are the number of bytes they can store. Double precision has double the amount that single precision has. Double precision (64 bits of memory used) has 11 bits for the exponent and 52 bits for the fraction when dealing with decimals. Single precision (32 bits of memory used) has 8 exponent bits and 23 fraction bits.

The reason there are two options mainly comes down to that the old generation S7-300 and S7-400 PLCs used single precision and the new generation S7-1200 and S7-1500 uses double precision.

Table 1. Data types that each series of Siemens PLCs support (Siemens IDE Requirements)

Target PLCs and Supported Data Types

To choose your target PLC based on supported data types, see the options in this table.

Data Type	S7-300/400	S7-1200	S7-1500
BOOL	Yes	Yes	Yes
BYTE	Yes	Yes	Yes
WORD	Yes	Yes	Yes
DWORD	Yes	Yes	Yes
LWORD	No	No	Yes
SINT	No	Yes	Yes
INT	Yes	Yes	Yes
DINT	Yes	Yes	Yes
USINT	No	Yes	Yes
UINT	No	Yes	Yes
UDINT	No	Yes	Yes
LINT	No	No	Yes
ULINT	No	No	Yes
REAL	Yes	Yes	Yes
LREAL	No	Yes	Yes

3.3 Generation of Files

After creating a model in Simulink, PLC Code generation needs a few settings to be changed first, in order to create the files the PLC needs.

As PLCs run continuously, they use a scan cycle which means that they check the values of its inputs and outputs a certain amount per second. This means that a PLC uses a fixed time when scanning. To get an accurate definition in the simulation the solver needs to be changed to a fixed-step solver (Figure 8). This is done in the solver section in configuration parameters (Figure 9).

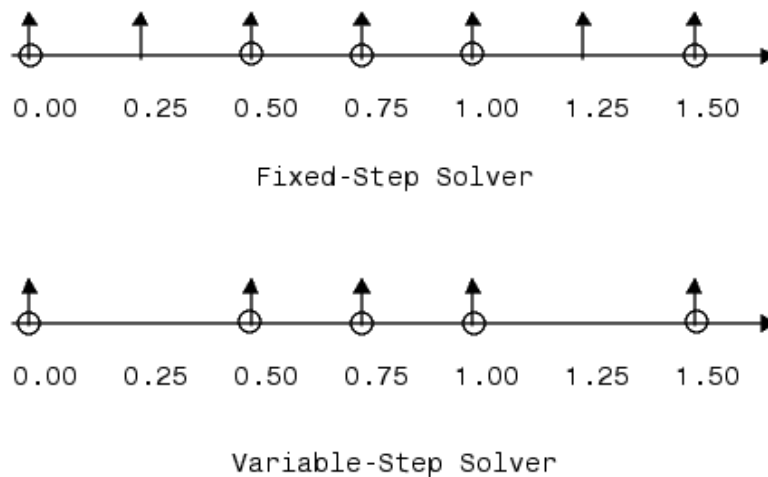


Figure 7. Example of steps when fixed- and variable-step solver acquire data (Siemens page on comparing solvers)

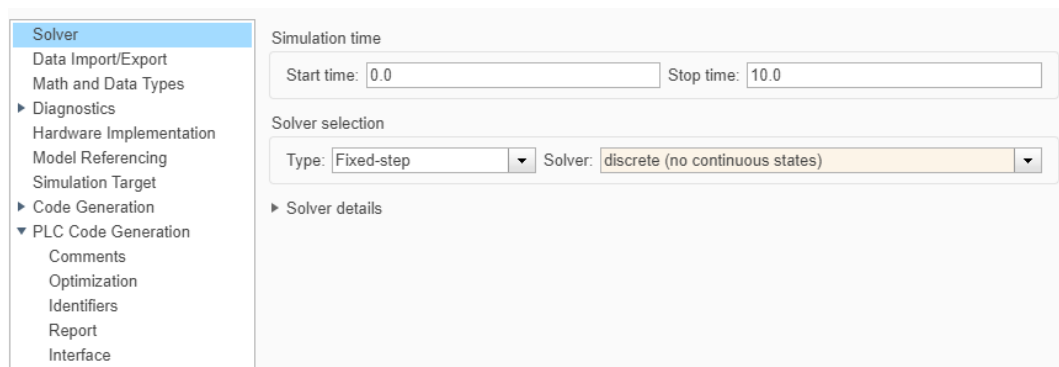


Figure 8. Configuration parameters, solver settings in Simulink

The second setting before generating files is regarding PLC code generation. The target IDE needs to be changed to the PLCs own IDE platform. This is done in general options under PLC Code Generation (Figure 10).

After changing the required settings, the created system needs to be treated as an atomic subsystem. An atomic subsystem is considered a level above the constants it takes into the subsystem. For example, in figure 12 the sensor value and the two outputs are level 0, and the subsystem is level 1. This means that the atomic subsystem executes its parameters separately in its own order. In comparison, a non-atomic subsystem which has one level of execution, and all its executions done in a single level.

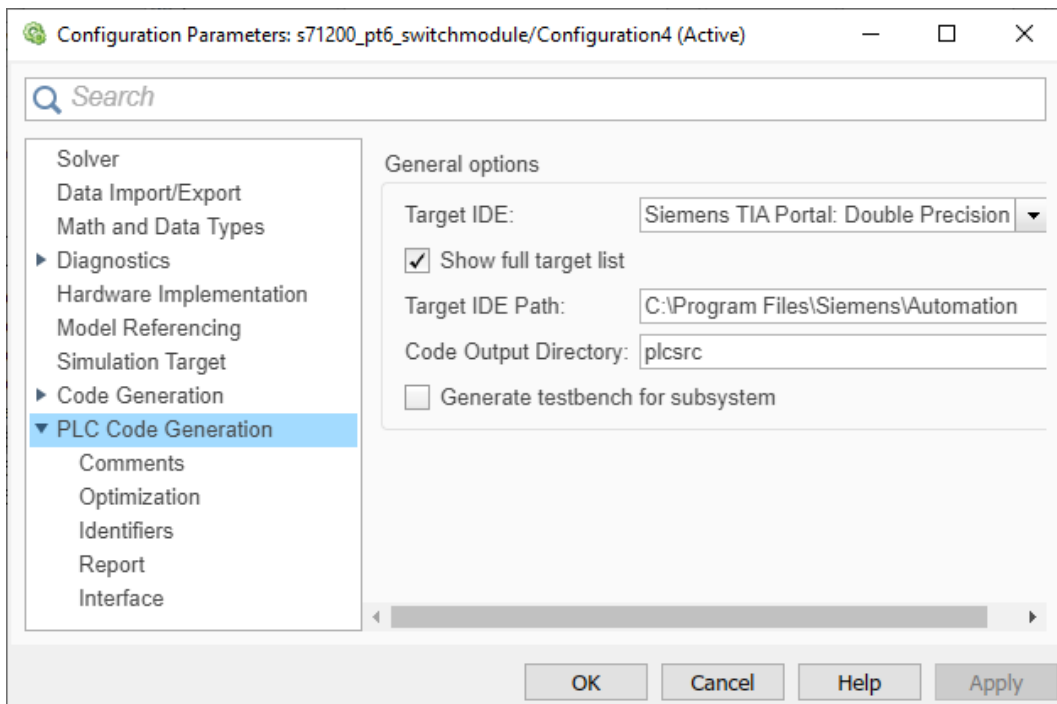


Figure 9. PLC Code Generation settings

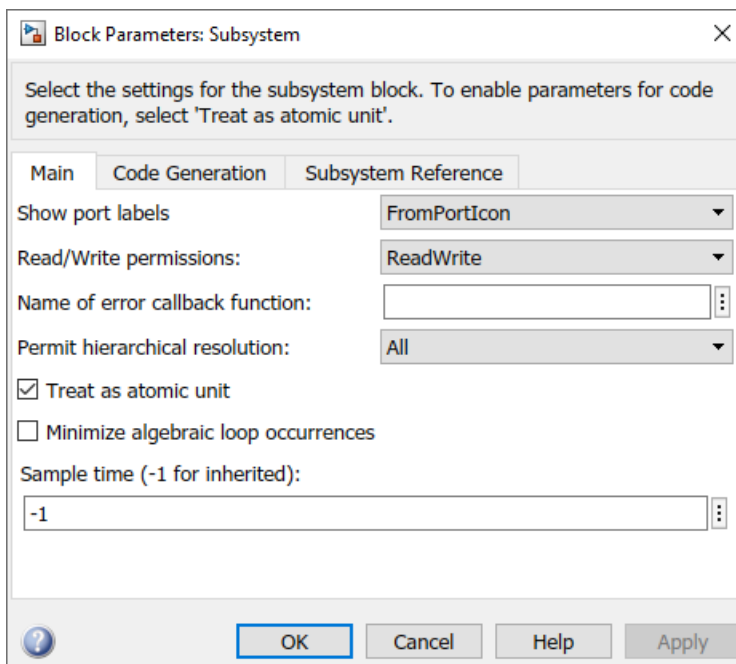


Figure 10. Block parameters settings with treat as atomic turned on

The block is generated into an atomic subsystem by entering its block parameters (Figure 10), this can be done by right clicking the said system. When a system is

made atomic, and all the necessary settings changed clicking the Atomic Subsystem in the model window will make the “Generate PLC Code” option available to use at the top of Simulink’s interface as seen in Figure 11.

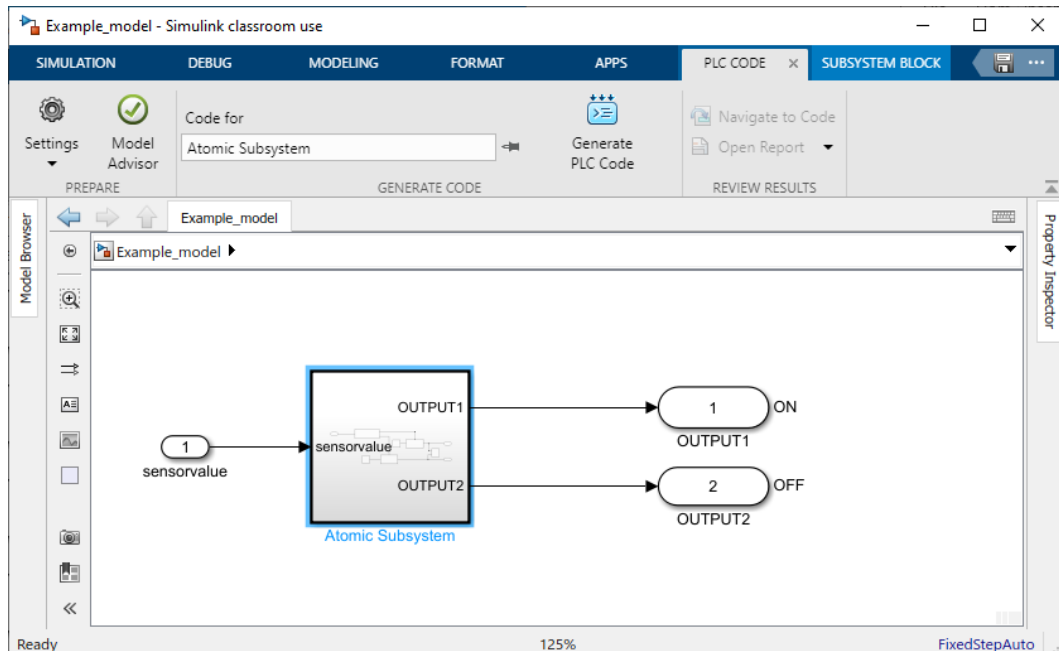


Figure 11. Atomic subsystem and PLC Code ready for generation

After generating the PLC Code, a new window will open with the code generation report. In the report Simulink will report what it has eliminated or has deemed virtual blocks, such as the inputs and outputs that are not a part of the atomic subsystem in Figure 11. The inputs and outputs will have their data types changed according to the selected PLC data types. For example, sensor value has “int8” as its data type in Simulink which is changed to “SINT” in the generated SCL file, as this is how TIA Portal has it and can then be used in TIA Portal without resulting in an error when executing the program.

The ssMethodType is a special input argument that Simulink PLC Coder adds, to execute code for Simulink subsystem blocks, it is not used in TIA Portal.

```

22 FUNCTION_BLOCK Atomic
23 VAR_INPUT
24     ssMethodType: SINT;
25     sensorvalue: SINT;
26 END_VAR
27 VAR_OUTPUT
28     Out1: BOOL;
29     Out2: BOOL;
30 END_VAR

```

Figure 12. Generated file data types

3.3.1 TIA Portal Configuration

After generating the files to be transferred to TIA Portal, they can be accessed from the external source files section, by adding external source files. After adding the generated SCL file into external source files, it can be converted into a function block by selecting “Generate blocks from source”. TIA Portal will then create a block from the SCL file and add the block into the main block of the program, TIA Portal also creates a database block that has the converted datatypes that Simulink previously made according to the selected system (Figure 13).

Switch_DB							
	Name	Retain	Accessible f...	Writa...	Visible in ..	Setpoint	Comment
1	▼ Input	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
2	■ Input1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Switch Input
3	▼ Output	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
4	■ Output1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Switch Output
5	InOut	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
6	Static	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Figure 13. Database block for ON/OFF switch

3.3.2 PLC Tags

Simulink cannot have input and output addresses, such as the PLCs output ports “%Q0.1”; this means that the tags that the PLC uses for its input and output ports need to be manually changed in TIA Portal after the export. They are changed in “PLC tags” section of the project tree under the default tag table.















Default tag table							
	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...
1	 Sensor value	Int 	%IW64 	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	 scaled output	Real	%MD4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	 Heater on	Bool	%Q0.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4	 Temp over 50	Bool	%Q0.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5	 On/Off switch	Bool	%I0.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6	 Normalized value	Real	%MD0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
7	 Temp over 70	Bool	%Q0.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
8	 Temp over 90	Bool	%Q0.4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
9	 Temp over 110	Bool	%Q0.5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
10	 Temp over 150	Bool	%Q0.6	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
11	 Temp under 30	Bool	%Q0.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
12	 Temp over 200	Bool	%Q0.7	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
13	<Add new>			<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figure 14. Default tag table for heating system

4 HEATING SYSTEM

The heating system was made to highlight the use of Simulink PLC Coder. It includes a K-Type thermocouple, a positive temperature coefficient heater (PTC), S7-1200 series PLC and a ZRLS1-2NA relay to convert current to the PTC.

4.1 Equipment Used in the System

Adjustable DC power supply (PS23023DL) (Figure 15), S7-1200 series PLC with 1214C CPU DC/DC/DC version (Figure 1) was used in the created system. A small circuit board with 8 switches is attached to the PLC inputs %I0.0 - %I0.7 to control ON/OFF input for the PTC heater.



Figure 15. Adjustable power supply for PTC heater



Figure 16. PeakTech TF-20 Thermocouple

The K-Type thermocouple with a temperature range from -50 °C to 900 °C (Figure 16) was also used, as well as a PTC heater with a 40 °C to 290 °C operating temperature range (Figure 17), which is connected to the PLC via ZRLS1-2NA relay

(Figure 18). The relay is used to get the correct voltage and current to heat up the PTC.



Figure 17. PTC heater used in the system

The PTC heater is connected to the external power supply's positive output and to the relay on port 14 in Figure 18.

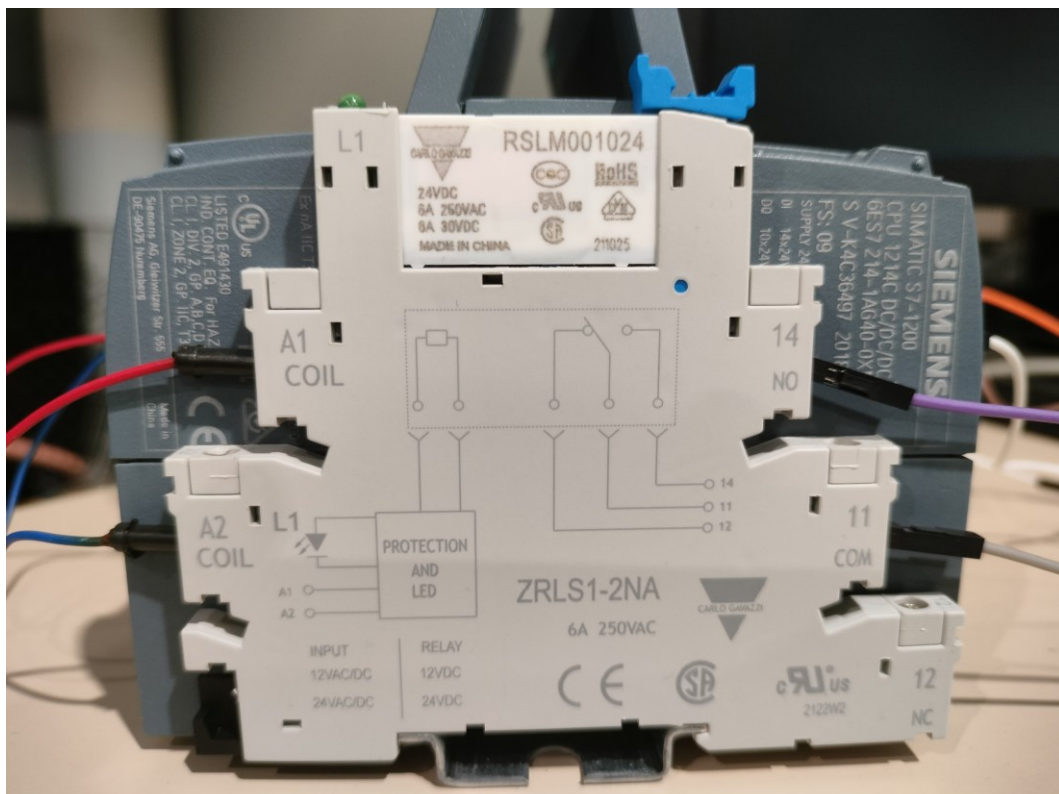


Figure 18. Carlo Gavazzi ZRLS1-2NA relay and connector block

The relay is connected in between the PLC, PTC heater and the external power supply, meaning the red and blue wire from A1 and A2 ports are connected to PLC

outputs %Q0.1 and blue connected to 3M port which is the PLCs own power from +24 V main outlet. The purple wire from 14 NO (Normally open) port is connected to the PTC heater, while the grey wire from 11 COM (Common) port is connected to the negative output of the external power supply.

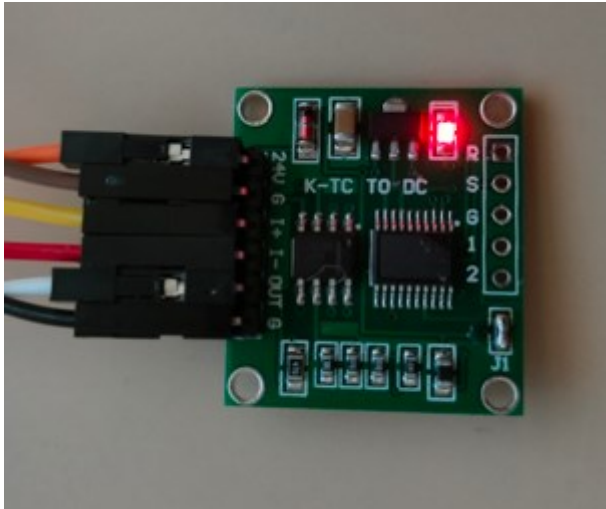


Figure 19. K-Type Thermocouple to VDC 0-10V converter connected to PLC, Thermocouple, and external power supply

Thermocouple to VDC converter is used to convert the value from the main sensor of the system, Peak-Tech TF-20 Thermocouple. The thermocouple range for -50°C to 900°C corresponds to a value of $-1,889\text{ mV}$ – $37,326\text{ mV}$ according to the manufacturer (TMH) of the thermocouple element in the sensor.

The converter is connected to the main power supply and the ground port of the PLC from the orange and brown wires (Figure 19). It is also connected to the K-Type thermocouple from the yellow and red wires. And finally connected to the PLCs ground and input port %IW64 to send the converted value from the thermocouple.

4.2 Configuration of Analog Input for PLC

Before adding the Simulink generated files there are a few things to do in TIA Portal. The sensor needs to have its sent data converted to a scale that the PLC can then read. The analog input full-scale data range for this PLC is 0 to 10 V this data range is converted to 0 – 27648 by the PLC. To have a more commonly used data

range to work with, the sensor data is converted to Celsius by a scale and a normalization block in TIA Portal (Figure 20).

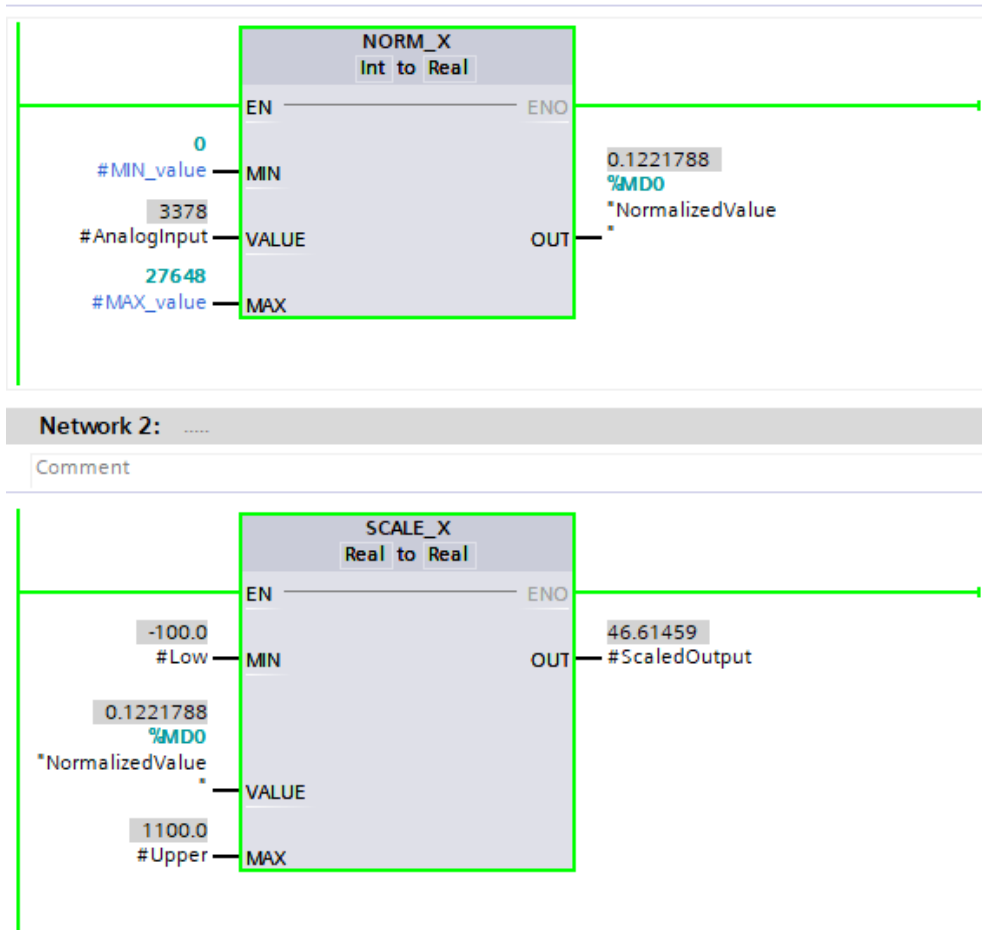


Figure 20. Normalization and scale blocks

The scale and normalization blocks are then converted into a main function block called **AnalogScale** in Figure 21. The analog scale block then takes the sensor value from **#AnalogInput** port 1 which is connected to **%IW64** analog port. TIA Portal and the PLC then converts it to output Celsius.

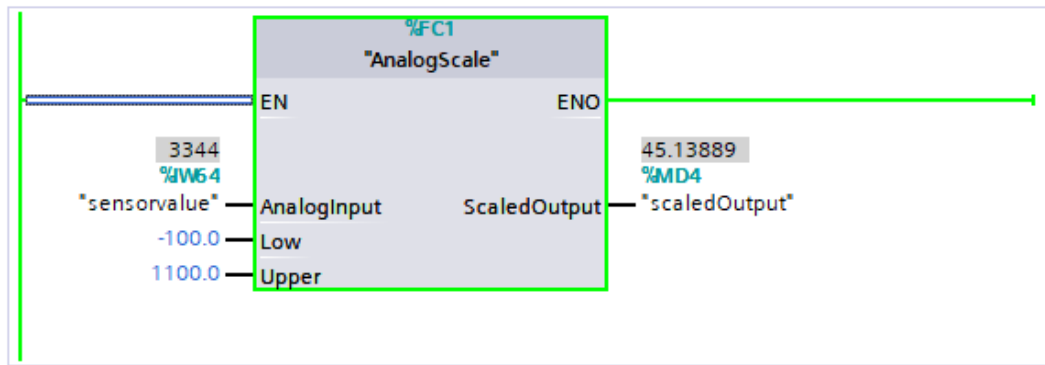


Figure 21. TIA Portal Analog scale block

The resolution for the K-type converter from its manual is 0.008333 V / °C. This means that its measurement range is the following:

$$\frac{10 \text{ V}}{0.0083333 \text{ } ^\circ\text{C}} = 1200,0048 \text{ } ^\circ\text{C}$$

The result accurately depicts the -100 °C – 1100 °C range given by the manufacturer. Although this range is set by the manufacturer, it gave an overall +20 °C to actual temperature in the room, the comparison calculated with two different multimeters. Dropping the “AnalogScale” low and upper values to -100 °C and 900 °C gave more precise readings as seen in Figure 22. As the sensor was not measuring the temperature of the system at the time so the temperature was close to room temperature.

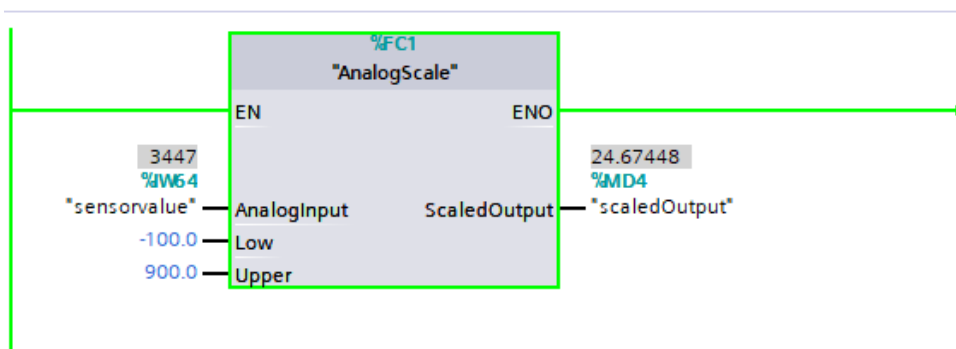


Figure 22. Analog scale with optimized values

The overshoot of the temperatures could be caused by the difference with the minimum and maximum values for the sensor and the converter. The sensor range

being $-50\text{ }^{\circ}\text{C}$ – ($-900\text{ }^{\circ}\text{C}$) for the K-Type sensor and $-100\text{ }^{\circ}\text{C}$ – ($-1100\text{ }^{\circ}\text{C}$) for the converter.

4.3 Heater Model from Simulink to TIA Portal

The initial system of the heater was created with Simulink and was then made more human readable by TIA Portal with its scaling blocks converting the value from the sensor to Celsius. After transferring the Simulink model to TIA Portal, it will run in the PLCs memory after it is compiled and downloaded there with TIA Portal.

4.3.1 Measurement Ranges

The heating model from Simulink consists of seven blocks. These blocks have a set temperature value that the sensor gives and in turn the PLC turns on a LED on the PLC output when a certain threshold is met. The first LED is on when the system is idle in a constant below 30 degrees Celsius. After the temperature raises $20\text{ }^{\circ}\text{C}$, the second LED is turned on to indicate that the heat is rising until $110\text{ }^{\circ}\text{C}$. The last two LEDs are for $150\text{ }^{\circ}\text{C}$ and $200\text{ }^{\circ}\text{C}$, as seen in Figure 24.

In this case the value in the compare block represents a value ranging between 0 and 27648 which is the range of the analog input of the PLC, 0 being $-100\text{ }^{\circ}\text{C}$, 7700 indicating $200\text{ }^{\circ}\text{C}$ and 27648 represents the maximum $900\text{ }^{\circ}\text{C}$.

The heat monitor is then converted into a SCL generated file, which is used in TIA Portal as mentioned in chapter 3.2.1.

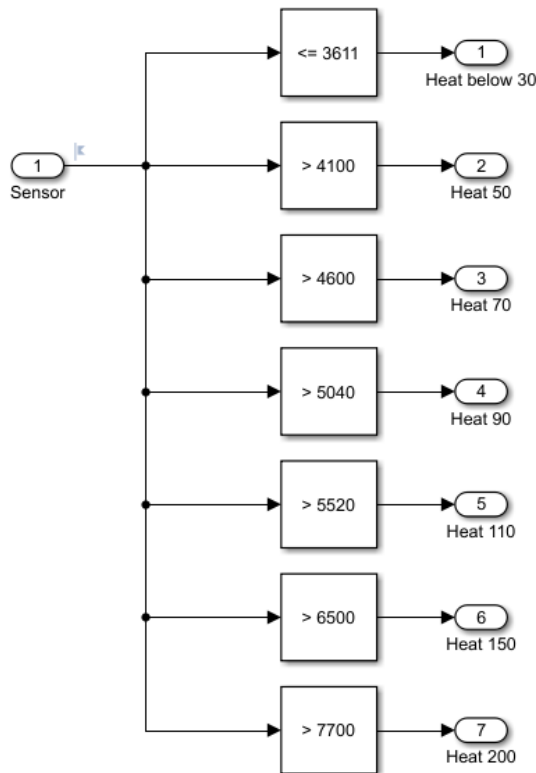


Figure 23. Heat indicator system with values in Simulink

4.3.2 Switch Module Used

A switch module to turn the heater on and off is also generated with Simulink. It consists of an input and an output; these are later labelled %I0.0 and %Q0.1 in TIA Portal.

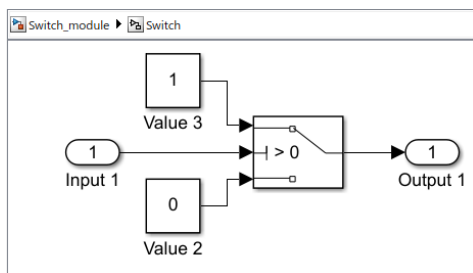


Figure 24. Simulink switch module that is used to turn the heater on or off

4.4 Combining all systems in TIA Portal

After all applications were added into TIA Portal, the project tree consisted of Analog scale, a switch, heater, and their database blocks as seen in the following Figure 25.

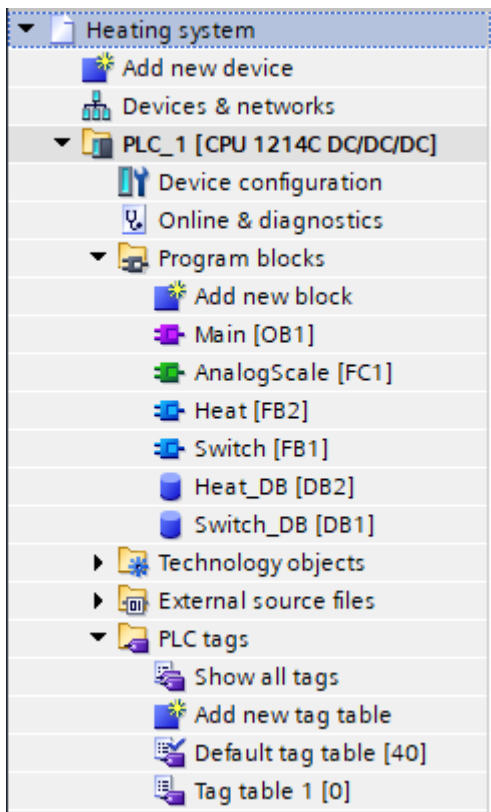


Figure 25. TIA Portal project tree after exporting the generated switch and heater modules and creating the analog scale function block

Manually added PLC tags are also included in the project tree under default tag table.

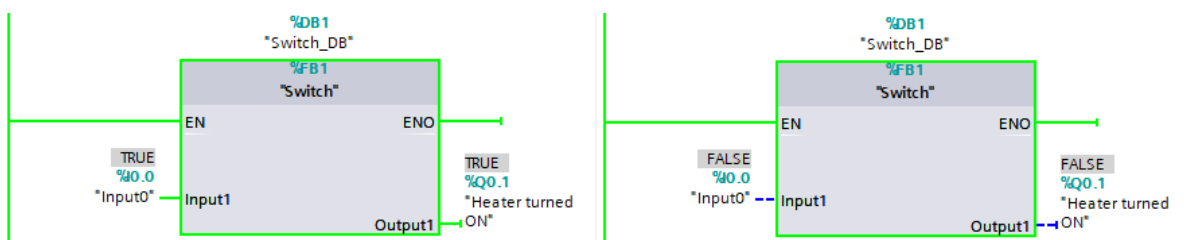


Figure 26. TIA Portal view of switch turned on and off

Figure 26 shows the TIA Portal view of the PLC running the switch model from the generated Simulink module.

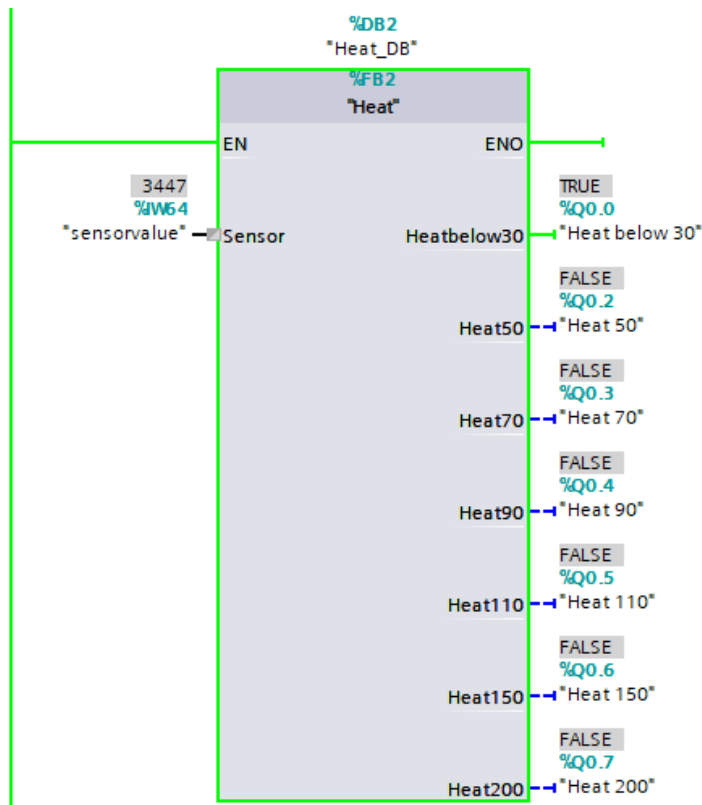


Figure 27. TIA Portal view of heat indicator values

Figure 27 shows the values from the Simulink model running on the PLC sending the thermocouple values also to the IDE.

4.5 Hardware

This chapter has the final figures of the system hardware that was used in its connected state.

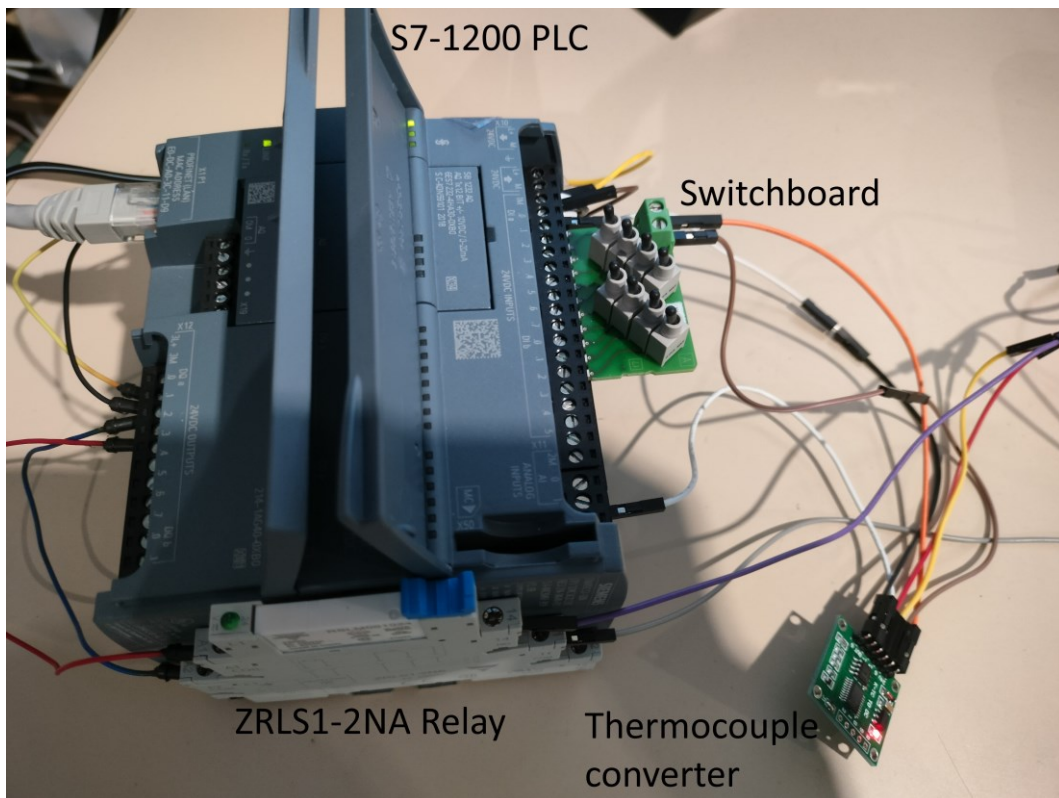


Figure 28. PLC and components connected to it

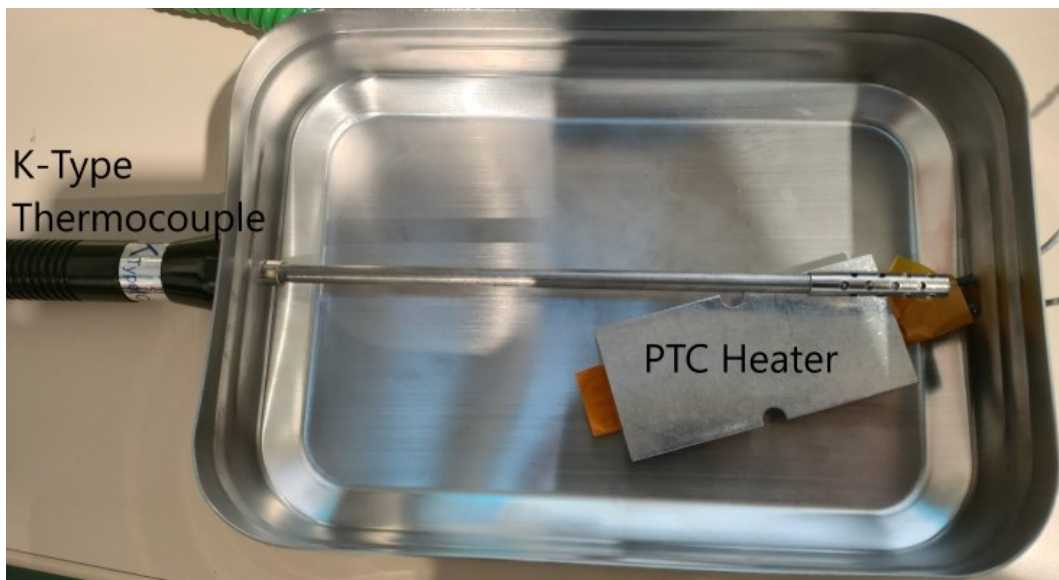


Figure 29. PTC Heater and thermocouple in a steel container

As seen in Figure 29, a steel container was used to keep the heater in a safe environment and to get accurate readings with a closed container, allowing to get the maximum value the heater could put out.

5 CONCLUSION AND FURTHER DEVELOPMENT

The objective of this thesis was to test, use and create instructions to use Simulink to PLC Coder with TIA Portal. The theory used for the thesis is from manuals, instructions, and guides for both software. The most time-consuming part of the thesis was getting the right parts and hardware combined to make a heating system.

For usability Simulink to PLC Coder is a good tool that can reduce the time of creating a PLC system, although this requires more Simulink knowledge because making a whole system in Simulink is not an easy task. Most programmers might not be too familiar with the ladder logic or function blocks so having to use a structured control language might be easier for students with a programming background. On the other hand, automation students are more used to automation software IDEs, such as TIA Portal, meaning function blocks and the ladder logic are more familiar.

The objective for this thesis was reached by creating a heating system in which the code produced by Simulink was tested. This was done by exporting it to TIA Portal with Simulink PLC Coder software. The objective was completed and instructions and software requirements for further use are found in Appendix 1.

If teaching both Simulink and PLC Coder simultaneously is successful, a more complex system could be modeled with Simulink and exported to Siemens or other automation hardware. Additionally, using TIA Portals PLC Simulator, a version of this could be done without any hardware needed.

REFERENCES

Dunn, A. 2009 The father of invention: Dick Morley looks back on the 40th anniversary of the PLC. Manufacturing Automation. Accessed 2.2.2023.

<https://www.automationmag.com/855-the-father-of-invention-dick-morley-looks-back-on-the-40th-anniversary-of-the-plc/>

IEC 61131 Standard. Accessed 29.3.2022. <https://webstore.iec.ch/publication/4552> ; <https://webstore.iec.ch/webstore/webstore.nsf/xpFAQ.xsp?Open&id=ADMN-7NALM5>

MathWorks Help Center. Simulink TIA Portal. Accessed 30.1.2023.

https://se.mathworks.com/help/plccoder/index.html?s_tid=CRUX_lftnav

Muthukrishnan, V. 2021. Programmable Logic Controllers (PLCs): Basics, Types & Applications. Accessed 2.2.2023. <https://www.electrical4u.com/programmable-logic-controllers/>

MathWorks Help Center. Compare Solvers. Accessed 2.2.2023. <https://se.mathworks.com/help/simulink/ug/compare-solvers.html>

MathWorks Help Center. Siemens IDE Requirements. Accessed 30.3.2022. <https://se.mathworks.com/help/plccoder/ug/siemens-simatic-step-7-considerations.html>

Siemens. TIA Totally Integrated Automation Portal – Always ready for tomorrow. Accessed 2.2.2023. <https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html>

Wayand, B. 2020. What is a PLC? Industry News, PLC Applications. Accessed 2.2.2023. <https://www.mroelectric.com/blog/what-is-a-plc/>

Egan, M. 2021. Relays vs PLCs. Electrical & Automation solutions. Accessed 1.2.2023. <https://easwaikato.co.nz/relays-vs-plcs/>

Brown, J. 2000. PLCs get smaller, do more. MachineDesign. Accessed 2.2.2023. <https://www.machinedesign.com/markets/article/21827755/plcs-get-smaller-do-more>

APPENDICES

APPENDIX 1

Software requirements for using PLC Coder are TIA Portal V16, MATLAB R2021a with Simulink and PLC Coder. Older versions of both software are likely to be sufficient as well. Simulink PLC Coder add-on is installed from “APPS” tab in Simulink.

Hardware requirements are a Siemens PLC and preferably an input to control, for example a switch connected to the PLC.

Instructions to use PLC Coder with TIA Portal.

Simulink:

- First create or download a Simulink model example to use in a system
- Turn on “Treat as atomic unit” from Block Parameters settings for the model used
- Switch the Target IDE in PLC Code Generation settings to Siemens TIA Portal: Double Precision
- Select fixed-step for the type of solver in solver settings
- The tab for PLC CODE will have the model’s name under “Code for” and “Generate PLC Code” available
- The generated file can be accessed from the code generation report or by going to the file directly in file explorer. File location can be found in MATLAB after clicking the file name in the report

TIA Portal:

- Create a project and choose the Siemens PLC from controllers and add it to the project
- In the project tree the generated file can be used from “External source files” and adding it to your project

- After the .scl file is in external files it can be used to generate blocks which appear in “Program blocks” after creation
- The different I/O ports need to be manually changed in TIA Portal under “PLC tags” in the project tree, for example the main switch to control the system is the first switch on the switchboard meaning %I0.0 in TIA Portal terms
- To get the program to run on the PLC a connection must be first made, this is done from “Go online” button in TIA Portal
- The program can then be compiled and downloaded to the PLC and run on it

APPENDIX 2

(*

*

* File: FurnaceModel_3.scl

*

* IEC 61131-3 Structured Text (ST) code generated for subsystem "FurnaceModel_3/Heat monitor"

*

* Model name : FurnaceModel_3

* Model version : 1.11

* Model creator : Mika_Niemi

* Model last modified by : Mika_Niemi

* Model last modified on : Tue Nov 15 16:29:50 2022

* Model sample time : 0.2s

* Subsystem name : FurnaceModel_3/Heat monitor

* Subsystem sample time : 0.2s

* Simulink PLC Coder version : 3.4 (R2021a) 14-Nov-2020

* ST code generated on : Tue Nov 15 16:29:59 2022

*

* Target IDE selection : Siemens TIA Portal: Double Precision

* Test Bench included : No

*

*)

FUNCTION_BLOCK Heat

VAR_INPUT

Sensor: LREAL;

END_VAR

VAR_OUTPUT

Temperature30: BOOL;

Temperature50: BOOL;

Temperature70: BOOL;

```
Temperature90: BOOL;

Temperature110: BOOL;

Temperature150: BOOL;

Temperature200: BOOL;

END_VAR

(* Outputs for Atomic SubSystem: '<Root>/Heat monitor' *)

(* Output: '<Root>/Temperature < 30' incorporates:

* Constant: '<S7>/Constant'

* RelationalOperator: '<S7>/Compare' *)

Temperature30 := Sensor <= 3661.0;

(* Output: '<Root>/Temperature > 50' incorporates:

* Constant: '<S2>/Constant'

* RelationalOperator: '<S2>/Compare' *)

Temperature50 := Sensor > 4214.0;

(* Output: '<Root>/Temperature > 70' incorporates:

* Constant: '<S3>/Constant'

* RelationalOperator: '<S3>/Compare' *)

Temperature70 := Sensor > 4767.0;

(* Output: '<Root>/Temperature > 90' incorporates:

* Constant: '<S4>/Constant'

* RelationalOperator: '<S4>/Compare' *)

Temperature90 := Sensor > 5320.0;

(* Output: '<Root>/Temperature > 110' incorporates:

* Constant: '<S5>/Constant'

* RelationalOperator: '<S5>/Compare' *)

Temperature110 := Sensor > 5873.0;

(* Output: '<Root>/Temperature > 150' incorporates:

* Constant: '<S6>/Constant'

* RelationalOperator: '<S6>/Compare' *)

Temperature150 := Sensor > 6979.0;

(* Output: '<Root>/Temperature > 200' incorporates:
```


* Constant: '<S8>/Constant'

* RelationalOperator: '<S8>/Compare' *)

Temperature200 := Sensor > 8362.0;

(* End of Outputs for SubSystem: '<Root>/Heat monitor' *)

END_FUNCTION_BLOCK