SIEMENS

SCE Training Curriculum

Siemens Automation Cooperates with Education (SCE) | 09/2015

PA Module P01-06

SIMATIC PCS 7 –
Control Loop and Other Control Functions

Cooperates with Education

Automation

SIEMENS

Matching SCE Trainer Packages for these curriculum

- **SIMATIC PCS 7 Software block of 3 packages**
  Order No. 6ES7650-0XX18-0YS5
- **SIMATIC PCS 7 Software block of 6 packages**
  Order No. 6ES7650-0XX18-2YS5
- **SIMATIC PCS 7 Software Upgrade block of 3 packages**
  Order No. 6ES7650-0XX18-0YE5 (V8.0 → V8.1) or 6ES7650-0XX08-0YE5 (V7.1 → V8.0)
- **SIMATIC PCS 7 Hardware Set including RTX Box**
  Order No. 6ES7654-0UE13-0XS0

Please note that these trainer packages may be replaced with subsequent packages.
An overview of the available SCE packages is provided at: siemens.com/sce/tp

Continuing education

For regional Siemens SCE continuing education, contact your regional SCE contact partner.
siemens.com/sce/contact

Additional information relating to SIMATIC PCS 7 and SIMIT

In particular, Getting Started, videos, tutorials, manuals and programming guide.
siemens.com/sce/pcs7

Additional information relating to SCE

siemens.com/sce

Note on Usage

The training curriculum for the integrated automation solution Totally Integrated Automation (TIA) was prepared for the program "Siemens Automation Cooperates with Education (SCE)“ specifically for training purposes at public educational and R&D facilities. Siemens AG is not liable for the contents.

This document may only be used for initial training on Siemens products/systems. This means it may be copied entirely or partially and handed to trainees for use within the scope of their training. Passing on or copying this document and communicating its contents is permitted within public training and continuing education facilities for training purposes.

Exceptions require written permission by Siemens AG. Contact person: Roland Scheuerer roland.scheuerer@siemens.com.

Violators are subject to damages. All rights including translation rights are reserved, particularly in the event a patent is granted or a utility model or design is registered.

Usage for industrial customer courses is explicitly not permitted. We do not agree to the commercial utilization of these documents.

We would like to thank the Technical University Dresden, particularly Prof. Dr. Leon Urbas and Annett Krause, MS, as well as the Michael Dziallas Engineering Corporation and those who provided support in preparing this SCE training document.

CONTROL LOOP AND OTHER CONTROL FUNCTIONS

TRAINING OBJECTIVE

In this chapter, the students will be introduced to the essential components and requirements of a block for the continuous control of process variables. This will enable them to create and configure a temperature control using the blocks PIDConL and PULSEGEN.

THEORY IN BRIEF

In the process industry, certain process variables will have to be kept at a certain value despite interferences (***disturbance characteristics***), and process variables have to be set in a stable mode to specified setpoints (***command behavior***). To this end, control loops are used as shown in Figure 1.

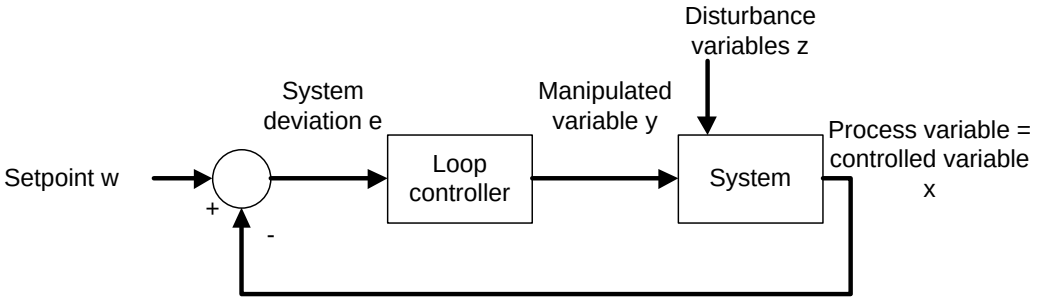


Figure 1: Control loop

For the plant in this training manual, the reactor temperature has to be set to a certain value for a specification-oriented response control. Disturbance variables are the ambient temperature and the materials used with different temperatures. For the temperature to be controlled, it first has to be determined through measurement. This measured value that corresponds to the ***actual value*** of the process variable is then compared with the desired value (***setpoint***). The difference between actual value and setpoint is called ***(system) deviation***.

If the system deviation is known, counter measures can be derived. In the case of temperature control, the heater is switched on when the measured actual value is less than the specified setpoint. For the process to react automatically, a closed loop controller is needed. A closed loop controller that calculates the manipulated variable based on the current deviation is called proportional controller (P-controller for short).

In practice, controllers that can be used for a wide variety of processes using only a few parameters have prevailed. These are referred to as ***PID controllers***.

The ***PCS 7 Advanced Process Library V80*** includes well-tried blocks that implement this functionality. Below, the block PIDConL is used.

THEORY

INTRODUCTION

The above mentioned P-controller represents the simplest controller. It processes according to the principle: the larger the current deviation, the larger the manipulated variable. This means its behavior is derived directly from the current system deviation– which makes it fast and relatively dynamic. However, certain disturbances are not completely adjusted, which means there is a permanent system deviation.

Not every process tolerates a permanent system deviation and additional steps have to be taken. One possibility consists of connecting an integral component; this turns the P-controller into a PI controller. The effect of the integral component consists of adding up a continuous system deviation. This means the manipulated variable increases even though the system deviation does not change.

If abrupt disturbances occur in a system, they can be counteracted quickly with an additionally differentiating component. The D-component calculates the manipulated variable from the time deviation of the system deviation. However, this behavior causes the stochastic interferences (noise) to be amplified. Here, an effective middle course has to be found.

A combination of P, I and D- component is called a PID controller. In the process industry, 95% of applications are implemented with these controllers, because the PID controller is set with only three parameters (gain, integral time and derivative time). These few parameters allow for a good adaptation to a variety of different dynamic processes.

However, setting the parameters presupposes knowledge of the system that is to be controlled. Knowledge of the system can be gained by experience, determined experimentally, or calculated by modeling the system. For a large variety of processes that are not dominated by dead time and that react in a similar manner to positive as well as negative changes of the manipulated variable intervention, different setting rules suitable for application in practice were found. Examples are the settings according to Chien, Hrones and Reswick [1], the Ziegler and Nichols method [2] as well as the T-Sum Rule [3].

The process control system ***PCS 7*** supports setting the parameters using a ***PID Tuner***.

For the closed loop control block PIDConL the parameter for gain is called GAIN, for the integral component TI (integral time) and for the differential component TD (derivative time). The time is specified in time units of seconds. The input variables of the controller are the controlled variable PV and the setpoint SP whose result is the system deviation ER. The manipulated value MV is the output variable for the controlled system; it is calculated according to the following formula:

$$MV = GAIN \cdot \left(1 + \frac{1}{TI \cdot s} + \frac{TD \cdot s}{1 + \frac{TD}{DiffGain} \cdot s} \right) \cdot ER .$$

INDUSTRIAL COMPATIBILITY OF CLOSED LOOP CONTROLLERS

For a closed loop controller to also work in everyday industrial applications, additional functions have to be implemented. These include above all:

– Bumpless transfer

– Anti-reset windup

– Support of different closed loop control structures

Bumpless transfer is to prevent an abrupt change of the manipulated variable when switching between manual and automatic mode, between internal and external setpoint entry or when parameters are changed. Bumpless transfer between manual and automatic mode is required, for example, when a process in process technology is running semi-automatically; this means when the startup is manual and normal operation is switched to automatic. In manual mode, the operator specifies the manipulated variable directly; in automatic mode, the PID algorithm calculates the manipulated variable.

The function Anti-Reset Windup (ARW) prevents that the integral component (reset) of the manipulated variable continues to increase (windup), because the system deviation cannot be adjusted based on the manipulated variable restriction.

The support of different closed loop control structures allows for optimizing the control without changing the controller. In the section ‘Expanded Closed Loop Control Structures’ some of these closed loop control structures are explained in greater detail. With PIDConL from the ***PCS 7 Advanced Process Library V8.1***, the following closed loop control structures can be implemented:

– Fixed setpoint control

– Cascade control

– Ratio control

– Feed forward control

– Split range control

– Smith predicator control

– Override

EXPANDED CLOSED LOOP CONTROL STRUCTURES

In some applications, single control loops are not sufficient so that expanded closed loop control structures have to be used to reach the desired goal.

If the command behavior and the disturbance characteristic cannot be optimized satisfactorily for a process variable at the same time, a feed forward/auxiliary variable control or a cascade control can be used.

If the disturbance variable and its point of application are known, a compensation of the disturbance variable can be feed-forwarded to the controller input or controller output. With the ***feed forward control*** the disturbance variable can be compensated for completely, so that the closed loop control can be set to the optimized command behavior.

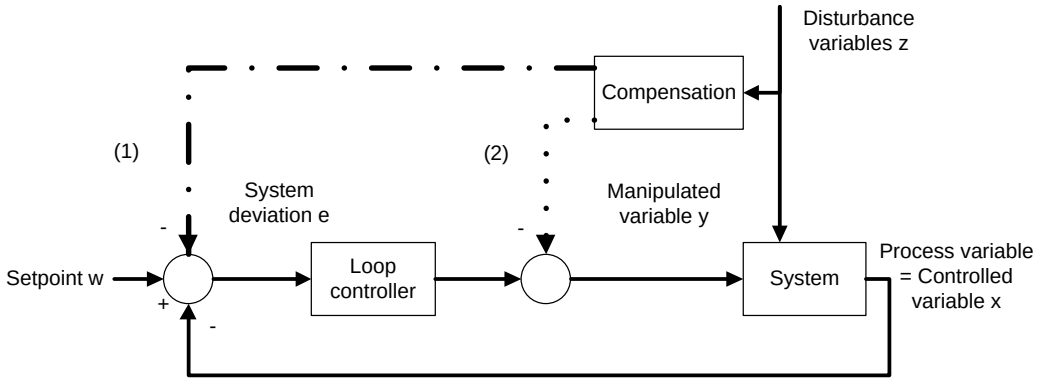


Figure 2: Feed forward control at the input (1) or the output (2) of the loop controller

If the disturbance variable cannot be measured but in its place another variable in the system, this auxiliary variable is switched with a controller to the controller input. ***Auxiliary variable feed forward*** reduces the influence of the disturbance variable, but does not compensate for it completely.

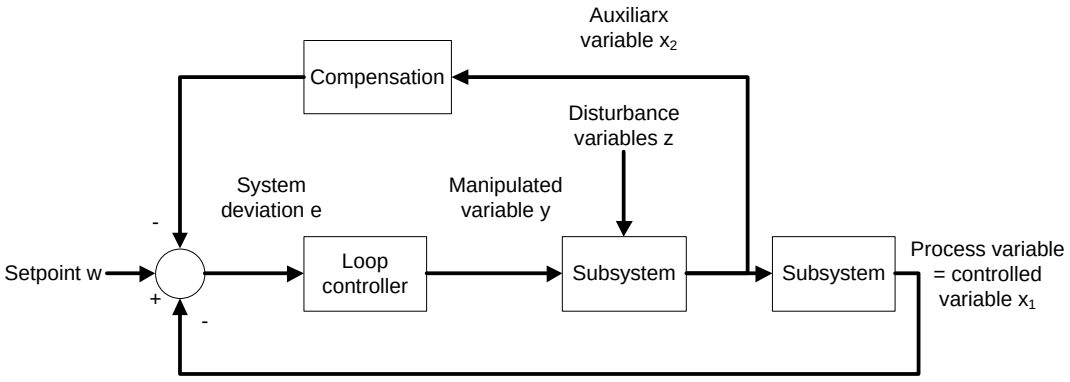


Figure 3: Compensation by auxiliary variables

If it is injected at the controller input, compensation and controller are not independent of each other. This means if the controller parameters are adjusted, the compensation has to be adjusted as well.

If the feed forward control and the auxiliary feed forward control are not sufficient, if the point of application of the disturbance variables cannot be determined with sufficient accuracy, or if the subsystems cannot be modeled sufficiently, a two or multi-loop ***cascade control*** is used.

When designing a cascade control, it is assumed that the lower level control loops (Loop controller 2 in Figure 4 – a so-called slave controller) react faster than the higher level control loops (Loop controller 1 in Figure 4 – a so-called master controller). The loop controller is thus always optimized from the inside to the outside.

Cascade controls reduce the influence of the disturbance variable and speed up controlling the reference variable. For the cascade control to be used, correspondingly measurable variables have to exist.

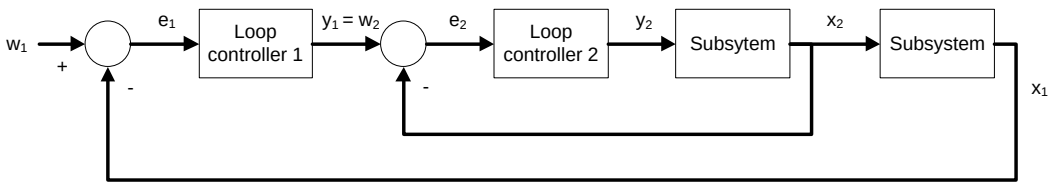


Figure 4: Cascade control with two loops

Ratio controls are used if the process variable is determined in dependence on another variable. For example, the ratio control of two liquid flows that are to be mixed, which means controlling the composition of the mixture, or the ratio control of combustion gas and fresh air at a gas burner for optimum combustion. The setpoint of the process variable w_2 is calculated from the ratio V_w and the process variable x_1.

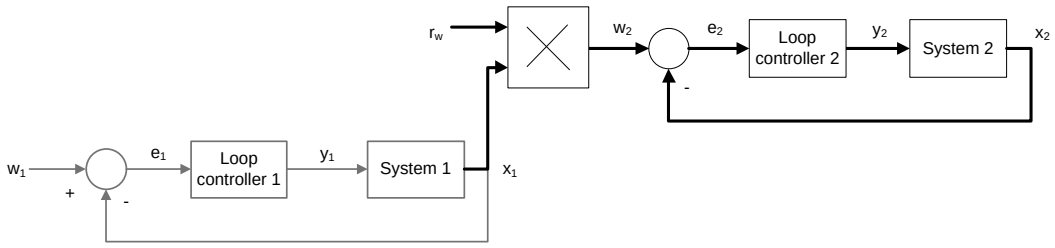


Figure 5: Ratio control

CONNECTION TO PROCESSES

The continuous output signal of the loop controller is not always read out directly to the process. This is not advisable particularly in the case of great forces or large flows so that a binary connection is implemented. To this end, the analog signal is converted into a binary signal by means of ***pulse width modulation***. In the ***CFC Library***, the elementary building block PULSEGEN [4] is provided.

By modulating the pulse width, the function PULSEGEN transforms the input variable INV (= LMN manipulated value of the PID controller) into a pulse train with a constant period. It corresponds to the cycle time that is used for updating the input variable and has to be parameterized in PER_TM.

The duration of a pulse for each period is proportionate to the input variable. However, the cycle parameterized with PER_TM is not identical with the processing cycle of the function block PULSEGEN. As shown in Figure 6, the cycle PER_TM ② consists of several processing cycles ① of the function block PULSEGEN. The number of PULSEGEN calls for each PER_TM cycle represents a measure for the accuracy of pulse width modulation.

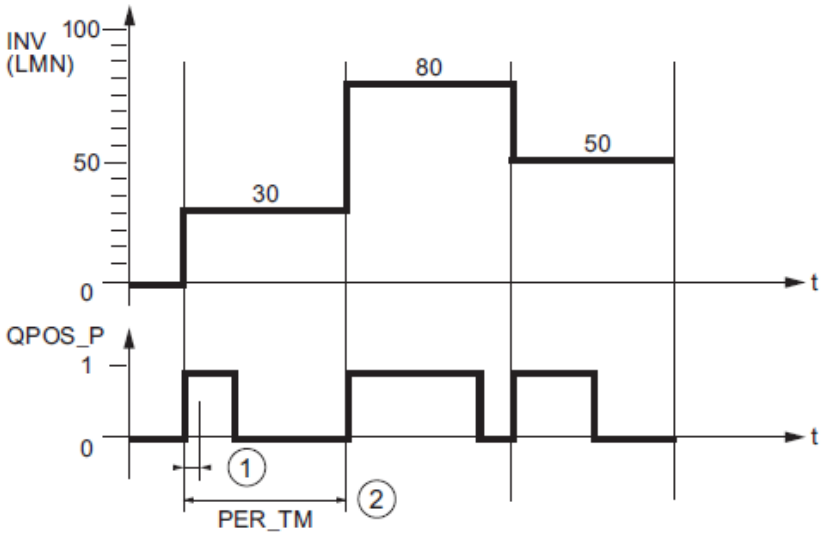


Figure 6: Time curve of input INV to output QPOS_P with PULSEGEN [4]

An input variable of 30% at 10 PULSEGEN calls for each PER_TM means the following:

– 1 at output QPOS for the first three calls of PULSEGEN (30% of 10 calls)

– 0 at output QPOS for seven additional calls of PULSEGEN (70% of 10 calls)

The pulse duration is recalculated at the beginning of each period. Through a sampling ratio of 1:10 (CTRL_PID calls to PULSEGEN calls), the manipulated value accuracy is limited in this example to 10%. Specified input values INV can be mapped only in a grid of 10% to a pulse length at output QPOS. Correspondingly, the accuracy increases with the number of PULSEGEN calls for each PIDConL call. If PULSEGEN is called 100 times and PIDConL only once, a resolution of 1% of the manipulated value range is reached.

Note: You have to program the down-scaling of the call frequency yourself.

LITERATURE

[1] Chien, Kun Li; Hrones, J. A.; Reswick, J. B. (1952): On the Automatic Control of Generalized Passive Systems. In: Transactions of the American Society of Mechanical Engineers, Vol. 74, Cambridge (Mass.), P. 175-185.

[2] Ziegler, J. G. and Nichols, N. B. (1942): Optimum settings for automatic controllers. In: Trans. ASME, 64, S. 759-768.

[3] Kuhn, U.: Eine praxisnahe Einstellregel für PID-Regler: Die T-Summen-Regel. Automatisierungstechnische Praxis, Nr. 5, 1995, S. 10-16. (Practice-Oriented Setting Rule for PID Controllers: the T-Sum Rule. Automation Engineering Practice, No. 5, Page 10 to 16)

[4] SIEMENS (2009): Process Control System PCS7: CFC – Elementary Blocks.

STEP BY STEP INSTRUCTIONS

TASK

Corresponding to the requirements in the chapter Process Description, we will be supplementing the CFCs from the chapter Functional Safety with the temperature control and the associated manual control of Reactor R001. The heater of the reactor is implemented using a PID controller with a series-connected pulse generator.

The following CFCs are created here:

– A1T2H008 (manual operation heater Reactor R001)

– A1T2T001 (heater Reactor R001)

When implementing the temperature control, the following interlock conditions are to be noted in the CFC.

– An actuator must only be operated if the main switch of the plant is switched on and the emergency stop switch is enabled.

– The temperatures in the two reactor must not exceed 60°C

– The heaters of both reactors must only be started up if they are covered with liquid (here: a minimum of 200ml in the reactor).

TRAINING OBJECTIVE

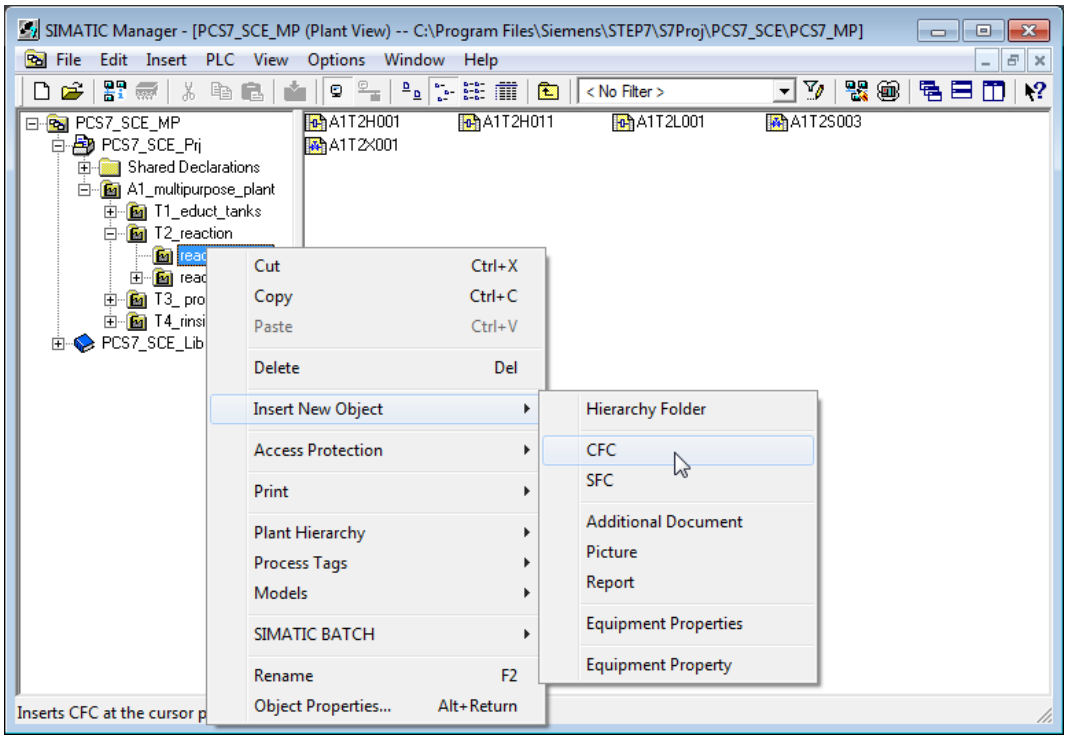
In this chapter, the student acquires:

– Knowledge for programming a continuous loop controller with pulse output and interlocks

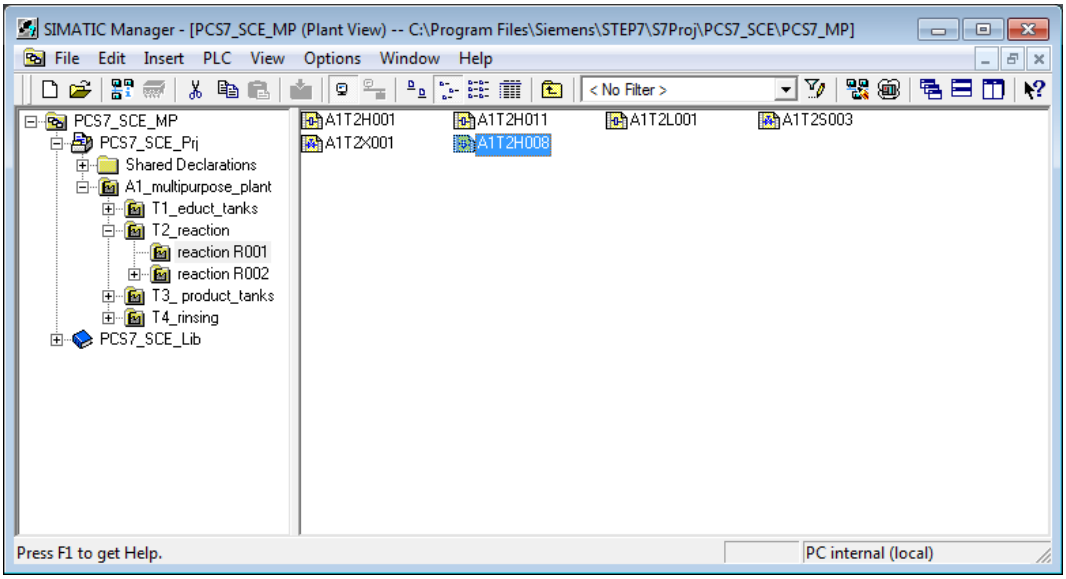
These instructions are based on the project PCS7_SCE_0106_Ueb_R1505_en.zip.

PROGRAMMING

1. First, insert a new CFC in the folder Reactor R001. In this CFC, we then implement the manual control for the heater.



2. The newly created chart is renamed A1T2H008.



3. The A1T2H008 connections differ from those of A1T2H011 only regarding the input and output signals (Pcs7DiIn and Pcs7DiOu) and in the last two reset conditions (block ‘Or08‘). The conditions refer on the one hand to the minimum level of 200.0ml and on the other hand the maximum temperature of 60.0°C.

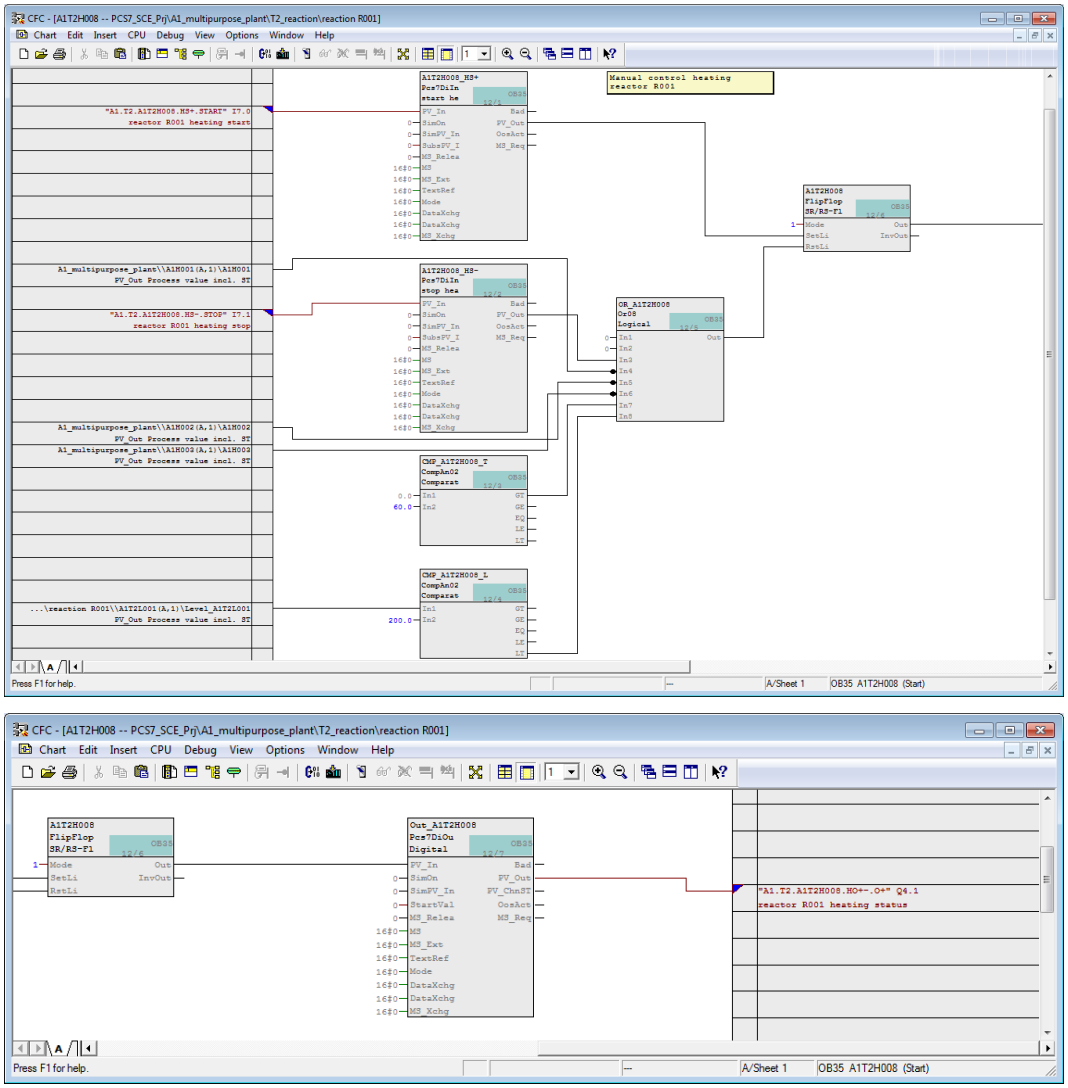


Table 1: New Blocks in Chart ‘A1T2H008’

Block	**Catalog/Folder**
Pcs7DiIn (2x)	Blocks/Channel
Or08	Blocks/LogicDi
CompAn02 (2x)	Blocks/LogicAn
FlipFlop	Blocks/LogicDi
Pcs7DiOu	Blocks/Channel

Table 2: Input Interconnections in Chart ‘A1T2H008’

Input	Connection to	Inverted
Pcs7DiIn.HS+ .PV_In	‘A1.T2.A1T2H008.HS+.START‘ / I7.0 / Reactor R001 Start heating	No
Pcs7DiIn.HS- .PV_In	‘A1.T2.A1T2H008.HS-.STOP’ / I7.1 / Reactor R001 Stop heating	No
Or08.In4	A1H001(A,1) / A1H001 PV_Out Process value incl. ST	Yes
Or08.In5	A1H002(A,1) / A1H002 PV_Out Process value incl. ST	Yes
Or08.In6	A1H003(A,1) / A1H003 PV_Out Process value incl. ST	Yes
CompAn02.T.In2	60.0	
CompAn02.L.In1	A1T2L001(A,1) / Level_A1T2L001 PV_Out Process value incl. ST	
CompAn02.L.In2	50.0	
FlipFlop.Mode	1	

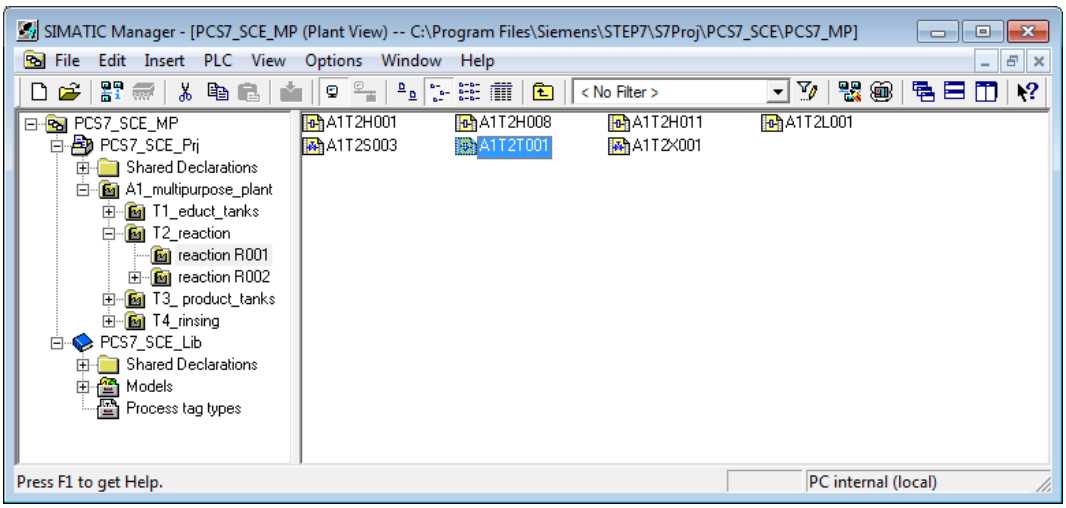
Table 3: Block Interconnections in Chart ‘A1T2H008’

Input	Output	Inverted
FlipFlop.SetLi	Pcs7DiIn.HS+.PV_Out	No
FlipFlop.RstLi	Or08.Out	No
Or08.In3	Pcs7DiIn.HS-.PV_Out	No
Or08.In7	CompAn02.GT	No
Or08.In8	CompAn02.LT	No
Pcs7DiOu.PV_In	FlipFlop.Out	No

Table 4: Output Interconnections in Chart ‘A1T2H008’

Output	Interconnection to	Inverted
Pcs7DiOu.PV_OUT	‘A1.T2.A1T2H008.HO+-.0+’ / Q4.1 / Reactor R001 Heating status value	No

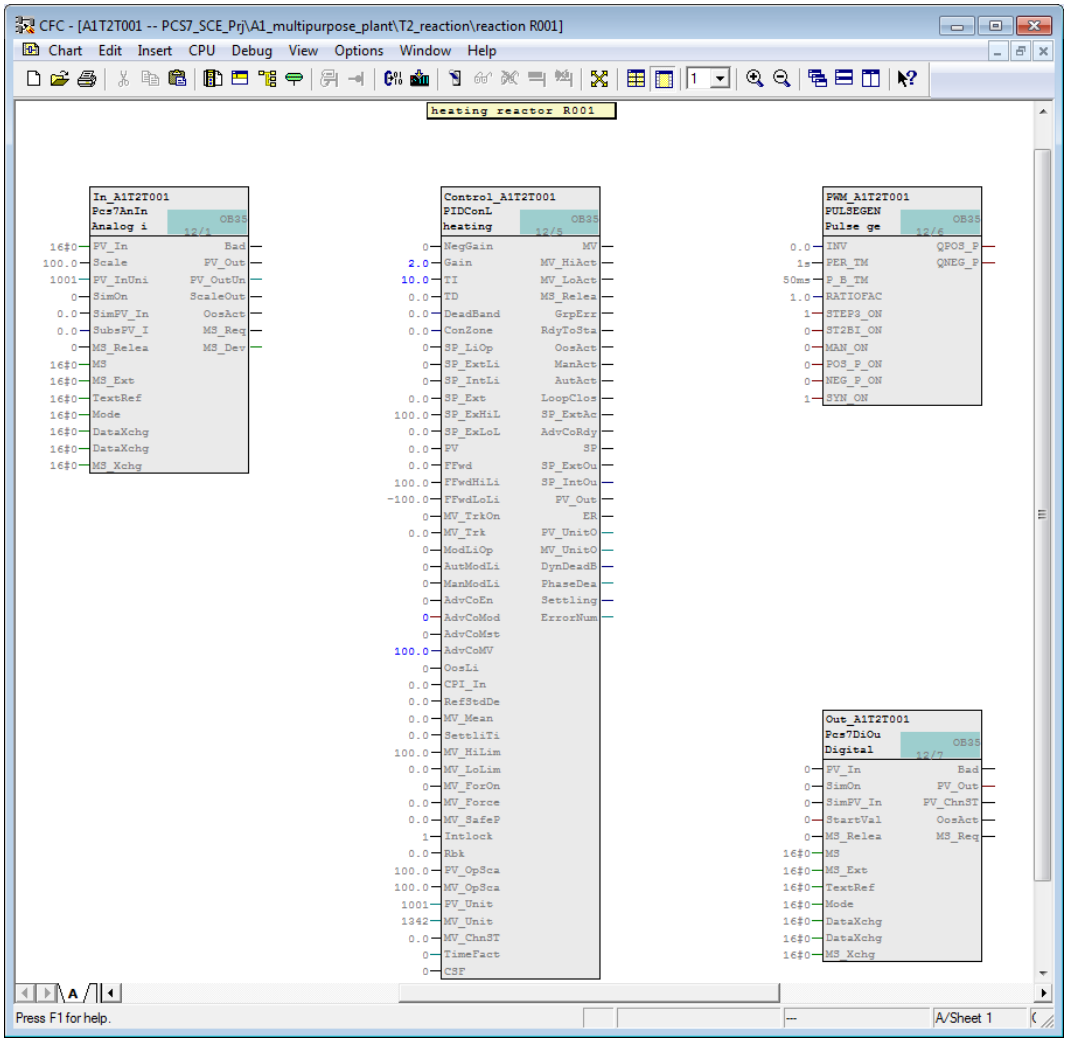
4. Now, a new CFC is set up with the name ‘A1T2T001‘. In this chart, implement the actual heater control for Reactor R001.



5. First add the following blocks and name them appropriately.

Table 5: New Blocks in Chart ‘A1T2T001’

Block	Catalog/Folder
Pcs7AnIn	Blocks/Channel
PIDConL	Libraries/PCS7 APL V81/Blocks+Templates\Blocks/Control
PULSEGEN	Libraries/CFC Library/ELEM400\Blocks/CONTROL
Pcs7DiOu	Blocks/Channel



6. Now, implement the basic interconnections as shown in the table below. Compare your result with the display.

Table 6: Input Interconnections in Chart ‘A1T2T001’

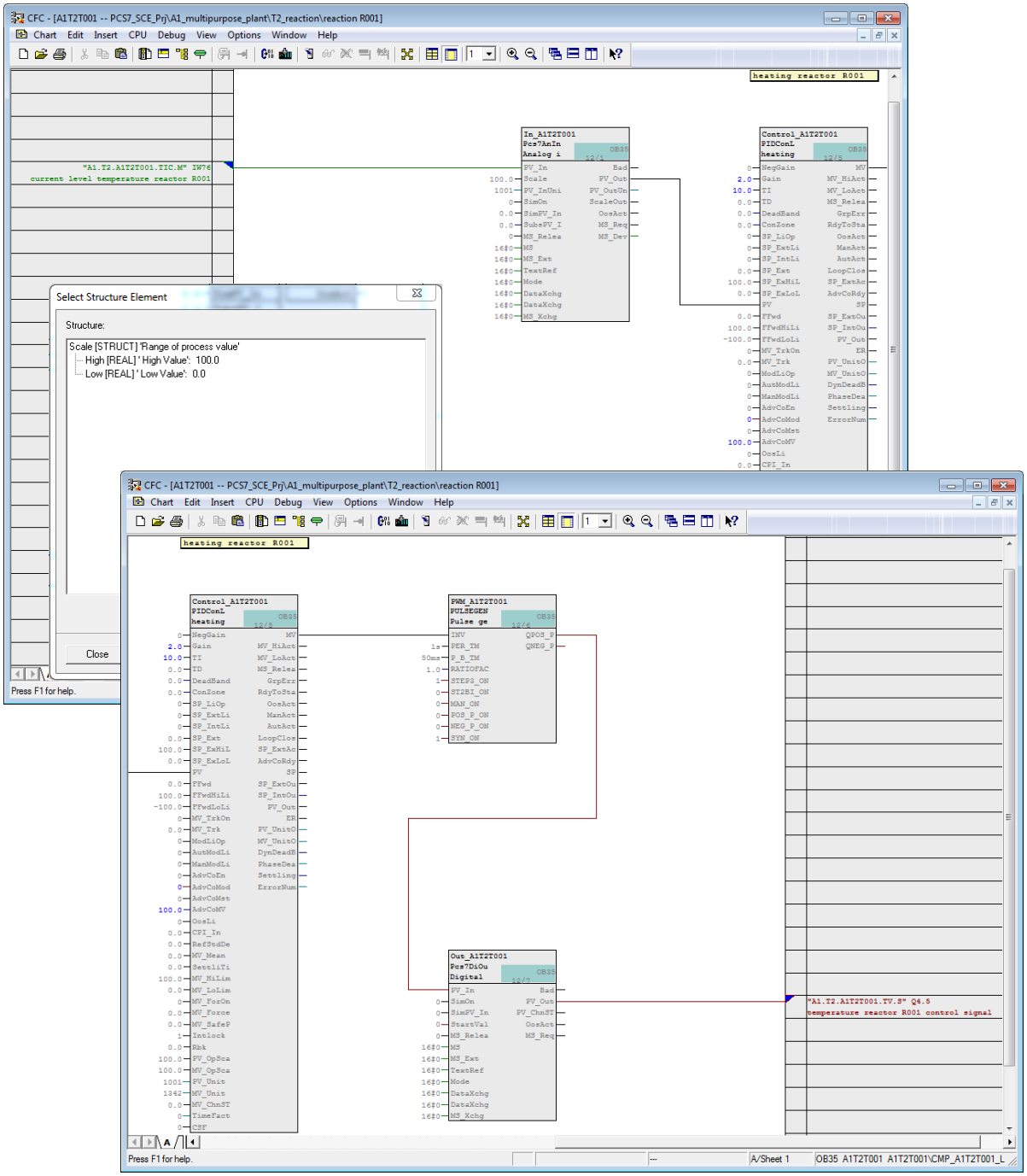
Input	Interconnection to	Inverted
Pcs7AnIn.PV_In	‘A1.T2.A1T2T001.TIC.M‘ / IW76 / Actual temperature value Reactor R001	
Pcs7AnIn.Scale	High value = 100.0, Low value = 0.0	

Table 7: Block interconnections in Chart ‘A1T2T001’

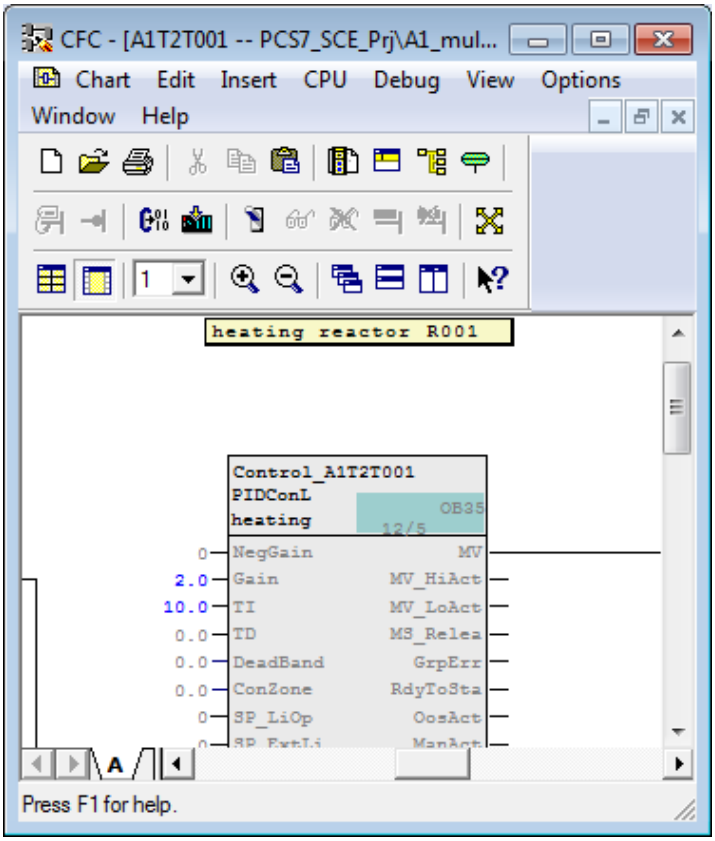
Input	Output	Inverted
PIDConL.PV	Pcs7AnIn.PV_Out	
PULSEGEN.INV	PIDConL.MV	
Pcs7DiOu.PV_In	PULSEGEN.QPOS_P	No

Table 8: Output Interconnections in Chart ‘A1T2T001’

Output	Interconnection to	Inverted
Pcs7DiOu.PV_OUT	‘A1.T2.A1T2T001.TV.S’ / Q4.5 / Temperature Reactor R001 actuating signal	No



7. Now, we parameterize the gains and the integral time of the PID controller by setting PIDConL.Gain = 2 and TI = 10.0.



8. Next, change to Sheet 2 and set up the interlocks shown below:

Table 9: New Block in Chart ‘A1T2T001/Sheet2‘

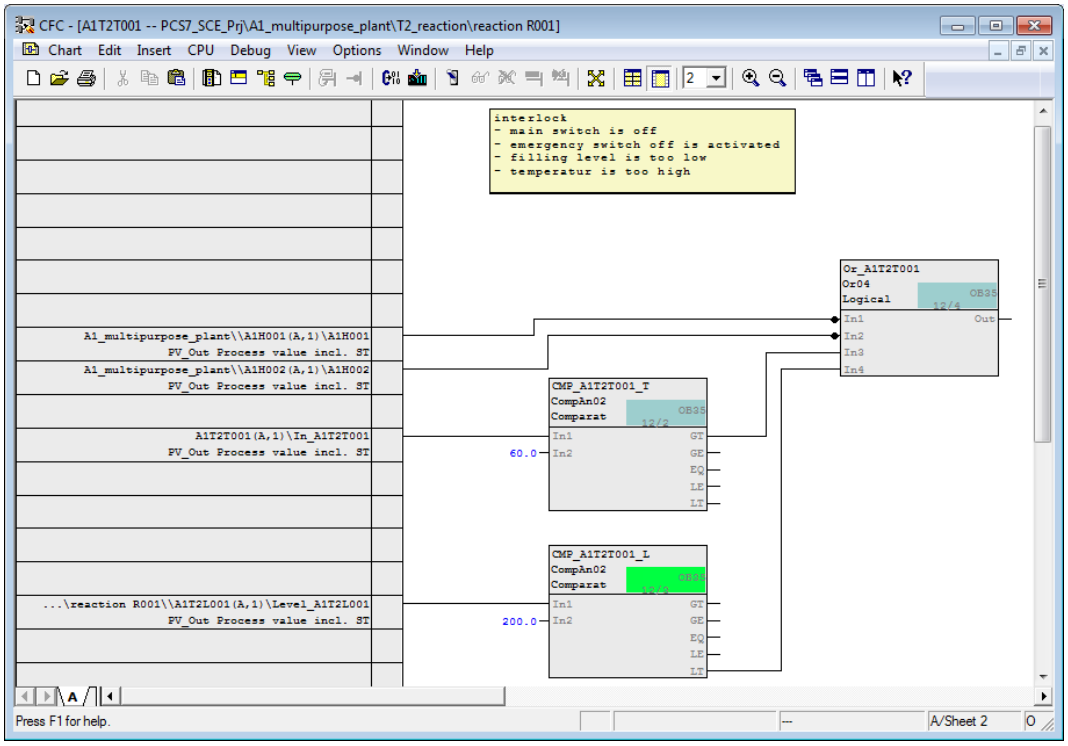
Block	Catalog/Folder
Or04	Blocks/LogicDi
CompAn02 (2x)	Blocks/LogicDi

Table 10: Input Interconnections in Chart ‘A1T2T001/Sheet2’

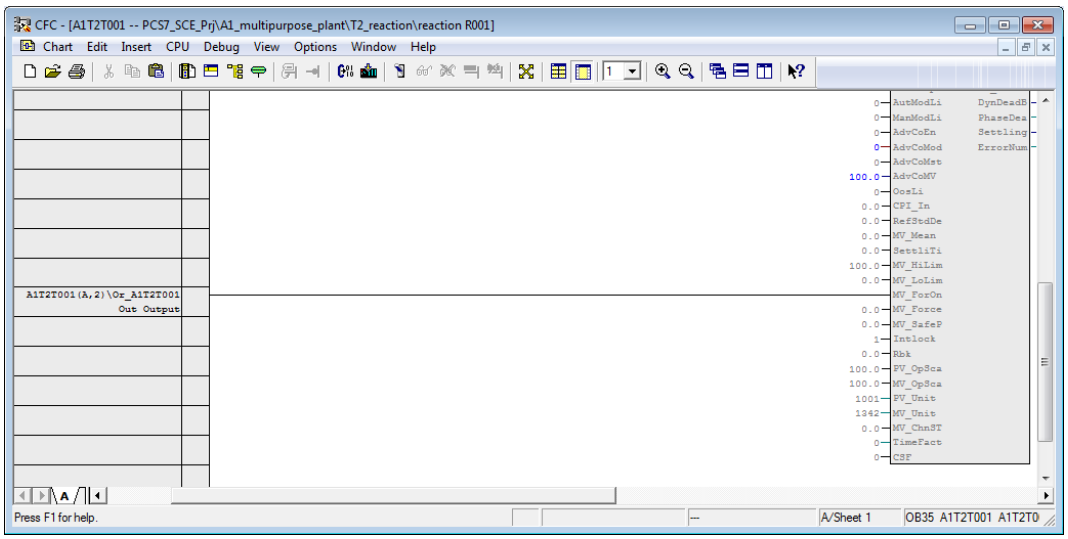
Input	Interconnection to	Inverted
Or04.In1	A1H001(A,1) / A1H001 PV_Out Process value incl. ST	Yes
Or04.In2	A1H002(A,1) / A1H002 PV_Out Process value incl. ST	Yes
CompAn02.T.In1	A1T2T001(A,1) / In_A1T2T001 PV_Out Process value incl. ST	
CompAn02.T.In2	60.0	
CompAn02.L.In1	A1T2L001(A,1) / Stand_A1T2L001 PV_Out Process value incl. ST	
CompAn02.L.In1	200.0	

Table 11: Block Interconnections in ‘A1T2T001’

Input	Output	Inverted
Or04.In3	CompAn02.T.GT	
Or04.In4	CompAn02.L.LT	



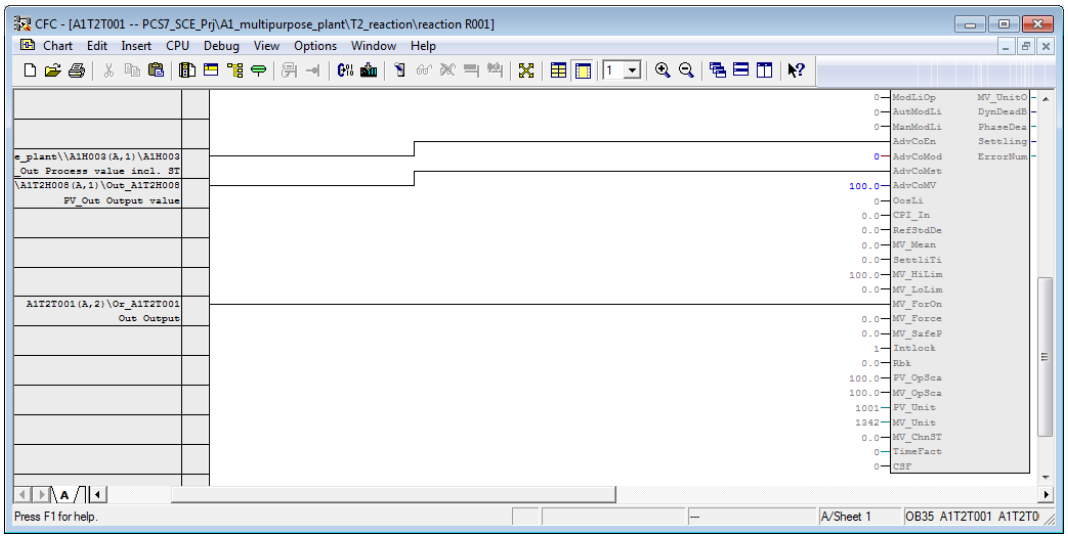
9. Now, connect output ‘Out‘ of block ‘Or04‘ to input ‘MV_ForOn‘ of block ‘PIDConL‘ and check that ‘MV_Force‘ = 0.0. This sets up the value ‘MV_Force‘ at output ‘MV‘ of the PID controller (manipulated value of controller) as soon as the interlocking conditions are met.



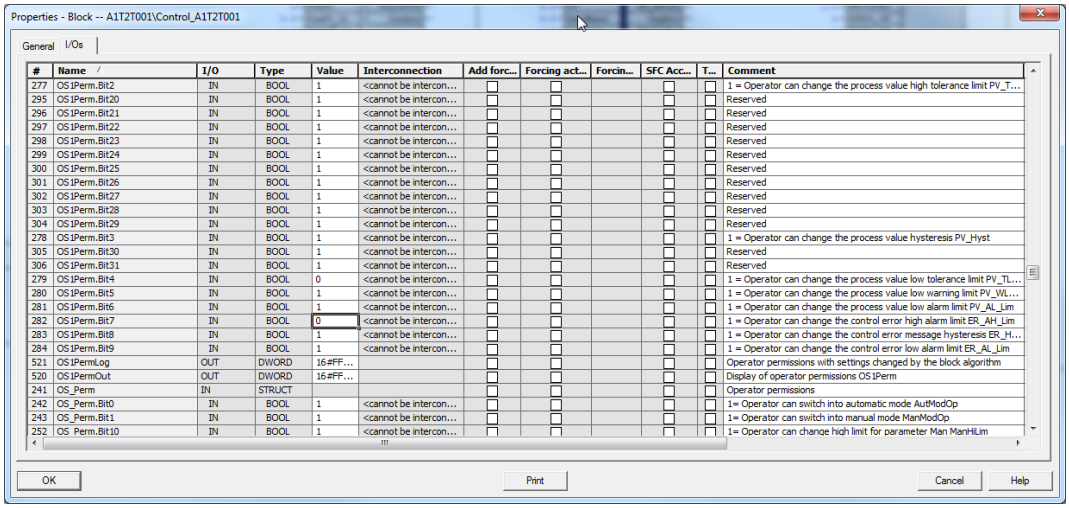
10. Next, we parameterize local operation. To this end, we are utilizing the programming mode. The programming mode is enabled by means of the input ‘AdvCoEn‘ and activated by means of ‘AdvCoMst‘. Parameter ‘AdvCoMod‘ is set to ‘0‘ so that in the local mode, the input ‘AdvCoMV‘ (invisible in the standard) is interpreted as manipulated value and not as setpoint. Then switch ‘AdvCoMV‘ visible and set it to 100. This starts the heater in the local mode without control.

Table 12: Input Interconnections in Chart ‘A1T2T001/Sheet1’

Input	Interconnection to	Inverted
PIDConL.AdvCoEn	A1H003(A,1) / A1H003 PV_Out Process value incl. ST	No
PIDConL.AdvCoMod	0	
PIDConL.AdvCoMst	A1T2H008(A,1) / Out_A1T2H008 PV_Out	No
PIDConL.AdvCoMV	100.0	



11. By parameterizing OS_Perm (bits 0 to 31), operator authorizations can be restricted. Set Bit 4 and Bit 7 to zero, so that the operator cannot switch on the programming mode and cannot change the manual requirement for the manipulated value (‘Man‘).



12. Finally, we interconnect chart ‘A1T2H008‘ and chart ‘A1T2T001‘.

Table 13: Block Interconnections between Chart ‘A1T2H008/Sheet1‘ and ‘A1T2T001/Sheet1’

Input	Output	Inverted
CompAn02.T.In1	A1T2T001(A,1) / In_A1T2T001 PV_Out Process value incl. ST	

EXERCISES

The exercises implement what we learned from Theory and the Step By Step Instructions. We will be using and expanding the existing multi-project from the step by step instructions (PCS7_SCE_0106_R1505_en.zip).

To prepare for the next chapter, we are implementing the last missing function of reactor R001 – the stirrer and the manual operation of the stirrer. The interlocking conditions are as follows:

– An actuator must be operated only if the main switch of the plant is switched on and the EMERGENCY STOP switch is enabled.

– The stirrers of the two reactors should be started up only when they come into contact with a liquid (here: 300ml minimum in the reactor).

In addition, further information about the PID controller, how it works and what parameters can be set is available. However, it is not necessary for the control functionality.

TASKS

1. Implement stirrer A1T2S001 in the chart folder ‘Reactor R001‘. Use the same process tag type for the stirrer as for the pumps. Connect the feedback and actuating signal. Assign appropriate names to the blocks. Then add the interlocks as explained above.
2. Implement the manual control A1T2H007 for the stirrer you just created. Here, implement the interlocking conditions as reset conditions.
3. Inform yourself about the inputs ‘ModLiOp‘, ‘AutModLi‘, ‘ManModLi‘ of the block ‘PIDConL‘. To this end, with the function key ‘F1‘ call Help for block ‘PIDConL‘. Select ‘PIDConL operating modes‘ and then the manual or automatic mode.
4. If you want to learn more about the inputs ‘SP_LiOp‘, ‘SP_ExtLi‘, ‘SP_IntLi‘, etc., in Help under tab ‘Search‘ enter the word Setpoint Input. Under the suggested title ‘Setpoint Input – Internal/External‘ information is provided.
5. What is the purpose of the parameters MV_HiLim and MV_LoLim? On your own, search Help for information regarding these inputs.