

Laboratory 11.

Application Techniques of Logic Controllers

The aim of the laboratory

The measurement laboratory widens the student's knowledge of process control technologies, control devices and design methodologies through practical exercises. During the laboratory the students learn how to implement a control system using Programmable Logic Controllers (PLC). The laboratory gives an easy introduction to process controls, PLC programming, system identification and control system design. The measurement configuration uses a wide scope of hardware and software technologies: Siemens compact PLC modules for control, PROFIBUS network for communication, WinCC operating system for man-machine interface, Simatic manager for PLC programming, Matlab for control design and a simple physical process. The main component of the configuration is a thermal process; a heatsink element that can be heated by a power transistor and cooled by an adjustable ventilator. During the laboratory the students get an overview of a simple industrial process control system.

Measurement configuration

The subject of the laboratory is an application of Siemens PLCs for a process control system.

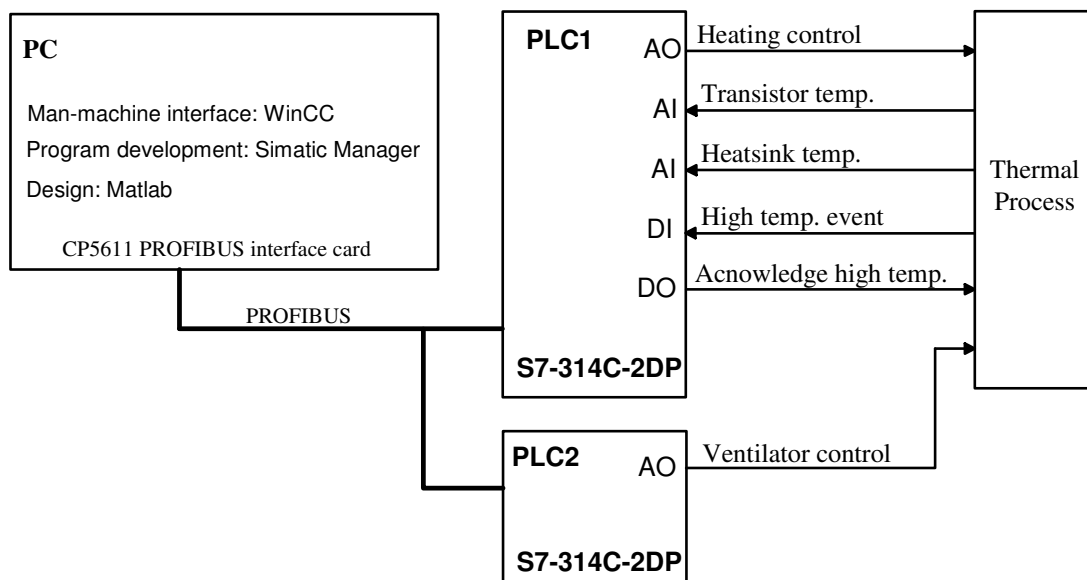


Figure 11-1. System configuration (AI: Analog Input, AO: Analog Output, DI: Digital Input, DO: Digital Output)

Measurement instruments:

PC + PROFIBUS interface	WinCC, Simatic Manager, MATLAB
Power supply	24V dc, 5A
S7-314C-2DP PLC	CPU: 314C-2DP
PLC #1 és PLC #2	4 analog inputs: $\pm 10V$ dc, 11 bits + sign 2 analog outputs: $\pm 10V$ dc, 11 bits + sign 24 digital inputs: 24V dc 16 digital outputs: 24V dc

Thermal Process

Thermal Process

The main object of the thermal process is the heatsink. The temperature (ϑ) of the heatsink can be increased by a power transistor. The temperature of the heatsink and the transistor is measured by two temperature sensors. The heatsink is cooled by a controllable ventilator. The thermal process is connected to the two PLCs by an interface card.

The interface card has an RS latch for the overheating protection and various buffering and level shifting circuits for the analog and digital signals. The two temperature signals are monitored by comparators. If the temperature exceeds the limit (approximately 80°C) the RS latch is set indicated by a red LED. In this state the heating circuit is turned off and the “*High temperature*” output signal goes to low state. The RS latch can be reseted by the “*Acknowledge high temperature*” signal. The reset signal brings back the circuit to normal operation. The interface card analog outputs and inputs voltage level is 0-10V. The input resistance is 100kOhm and the maximum load current of the outputs are 10mA. The acknowledge reset is an open collector active low signal.

The interface card is connected to the two PLCs by RJ-45 connectors (8 pin phone connectors). The PLCs are communicating with the PC over a PROFIBUS network.

Connections to PLC:

- 1 – Not connected
- 2 – Acknowledge high temperature event, resets to normal state (active falling edge)
- 3 – High temperature (open collector active low output)
- 4 - U_{V_cont} (ventilator control input)
- 5 - U_P (Transistor heating control input)
- 6 - $U_{\vartheta 2}$ (Transistor temperature output)
- 7 - $U_{\vartheta 1}$ (Heatsink temperature output)
- 8 - Common (GND)

The 24V power supply is connected by a two terminal green connector. The presence of the supply is indicated by a green LED.

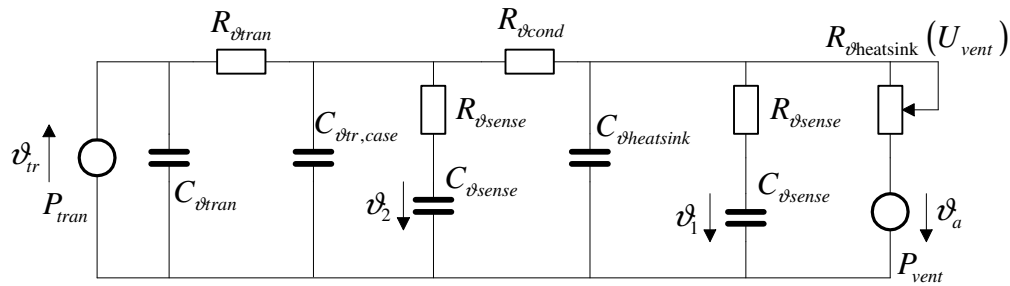


Figure 11-3. The simplified diagram of the thermal process

In the laboratory the thermal process modules are not identical. There are differences in the placement of the temperature sensors and the heat coupling between the heatsink and the transistor. Not every version measures the temperature of the transistor. The process is a second order system, because it has two temperature storage element, the heatsink and the transistor. The heatsink is cooled by the adjustable ventilator. The ventilator is used as a disturbance signal or to adjust the working point of the process.

Approximating the heating process with first order lag

Many simple physical systems can be approximated by a first order system. The thermal process can be approximated by a first order lag, described by a static gain (A) and a time constant parameter (T).

Its transfer function and step response are

$$P(s) = \frac{A}{1 + sT} = \frac{k}{s - p}$$

$$v(t) = A(1 - e^{-t/T}).$$

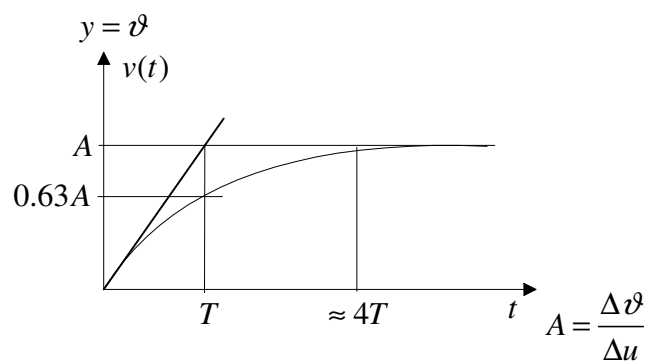


Figure 11-4. Step response of first order lag

The heating process has a linear static characteristic (Figure 11-4a) with a bias due to the ϑ_a ambient temperature. The slope of the static curve is the static gain of the first order lag; $A = \frac{\Delta \vartheta}{\Delta u}$. The step response can be generated by applying a step input in the (u_0, ϑ_0) working point. From the step response curve the A and T parameters can be identified.

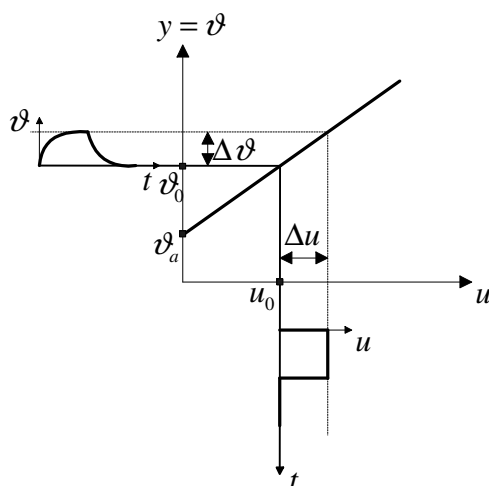


Figure 11-4a. Static characteristic of the heating process

Signal wiring

The analog and digital signals connect the PLCs to the heating process. The cross connections between the two PLCs allow the system to be configured for other tasks.

Process control PLC (PLC #1)

Signal	Variable	Physical meaning
AI0 Analog input	IW10 (word)	$U_{\vartheta 1}$ (heatsink temperature)
AI1 Analog input	IW12 (word)	$U_{\vartheta 2}$ (transistor temperature)
AO0 Analog kimenet	QW10 (word)	U_p (heating control)
DI0 Digital input	I0.0 (bit)	High temperature ($\vartheta > 80^\circ\text{C}$) signal
DO0 Digital output	Q0.0 (bit)	Acknowledge high temperature signal
DI8-DI15	I1.0–I1.7 bits IB1 byte	PLC #2 digital output connection
DO8-DO15	Q1.0–Q1.7 bits QB1 byte	PLC #2 digital input connection

Process tuning PLC (PLC #2)

Signal	Variable	Physical meaning
AO0 Analog output	QW10 (word)	U_{V_cont} (ventilator control voltage)
DI8-DI15	I1.0–I1.7 bits IB1 byte	PLC #1 digital output connection
DO8-DO15	Q1.0–Q1.7 bits QB1 byte	PLC #1 digital input connection

Siemens S7-300 PLC

A PLC is a special digital computer that was developed for industrial process control applications. A PLC can be expanded by analog and digital interfaces to connect and control the physical processes. The programming can be done by an external programming device, in this case with a regular PC.

The S7-314C-2DP PLC is from the Siemens S7-300 family of low-middle category compact controllers. The PLC has been designed for innovative systems solutions with the emphasis on production engineering, and is a universal automation platform providing optimum solutions for applications with central or distributed designs.

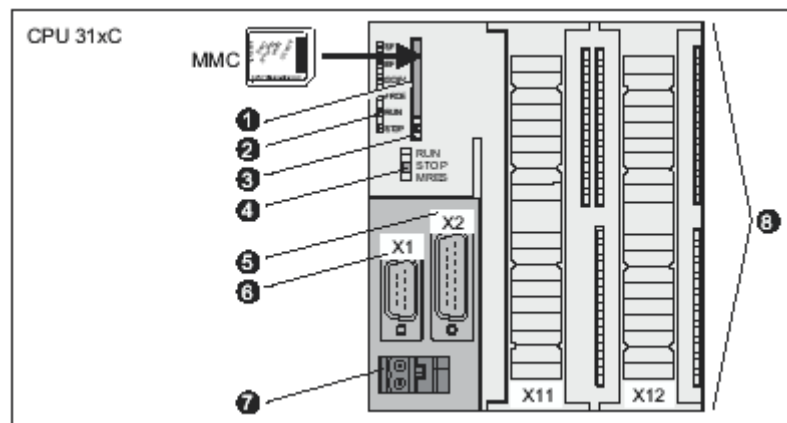


Figure 11-5. The PLC front panel

1. Slot for the SIMATIC Micro Memory Card
2. Status and error displays
 - SF –Hardware or software error (red)
 - BF – PROFIBUS error (red)
 - 5VDC – 5-V power for CPU and S7-300 bus is OK (green)
 - FRCE – Force job is active (yellow)
 - RUN – in start mode it flashes, in normal mode on (green)
 - STOP – CPU in STOP and HOLD or STARTUP (green)
3. MMC (Micro Memory Card) ejector
4. Mode selector
 - RUN – The CPU executes the user program.
 - STOP – The CPU does not execute a user program.
 - MRES –CPU memory reset.
5. X2 PROFIBUS connector
6. X1 MPI connector
7. 24V-os connection
8. Connections of the integrated I/O

A 314C-2DP compact PLC has 24 digital inputs, 16 digital outputs, 4 analog inputs and 2 analog outputs. The PC is connect to the PROFIBUS network by a CP5611 PROFIBUS interface card

PLC programming

The PLC configuration and programming is done with the Simatic Manager program. The man-machine interface of the process control system can be created by the WinCC program.

The programming language of the PLC is STEP7. This structured programming language consists of program modules (blocks). There are several types of blocks:

1. *OB - Organization Block*: Interface between operating system and user program.
2. *FC - Function*: Block with parameter transfer capability but no memory
3. *FB - Function Block*: Block with parameter transfer capability and memory
4. *DB - Data Block*: Block for storing user data
5. *SFB, SFC - System FB, System FC*: Characteristics similar to a FC and FB.
6. *SDB - System DB*: System configuration data

The operating system of the S7 CPU executes OB1 periodically. When OB1 has been executed, the operating system starts it again. Cyclic execution of OB1 is started after the startup has been completed. You can call other function blocks (FBs, SFBs) or functions (FCs, SFCs) in OB1. The operating system continuously monitors the run time of the block. If the run time exceeds the cycle time, the operating system calls the OB80 error handling block or switches to STOP mode. The operating system calls the OB35 block in every 100 ms. During the laboratory the user functions are called from this block.

Simatic Manager

The program development during the laboratory for the S7-314C-2DP PLC is carried out in the Simatic Manager software. The integrated user interface of Simatic Manager makes it possible to easily edit, compile and debug the operating programs and download them to the PLC.

STEP7 includes several low and high level programming languages:

- *STL - Statment List*
- *LAD - Ladder Logic*
- *FBD - Function Block Diagram*
- *GRAPH*
- *HiGraph – State graph*
- *SCL - Structured Control Language* – A high-level textual programming language which is based on PASCAL. It is also based on a standard for PLCs.

During the laboratory the *SCL* programming language is used.

WinCC

SIMATIC WinCC is a software program for the operator control and monitoring of automated processes. WinCC stands for Windows Control Center, meaning that it provides you with all the features and options of a standard Windows environment. WinCC is a visualization system that can be customized and comes with powerful features, an intuitive user interface, and easy-to-use configuration functions. SIMATIC WinCC provides a complete basic system for operator control and monitoring. It provides a number of editors

and interfaces that allow you to create highly efficient configurations for your specific application. All relevant configuration data is stored in a WinCC project.

Measurement Framework

The laboratory exercises are solved with the help of a measurement framework. This framework has two main components: the downloadable PLC blocks and a WinCC graphical user interface.

The framework project in Simatic Manager

The project can be viewed in Simatic Manager:

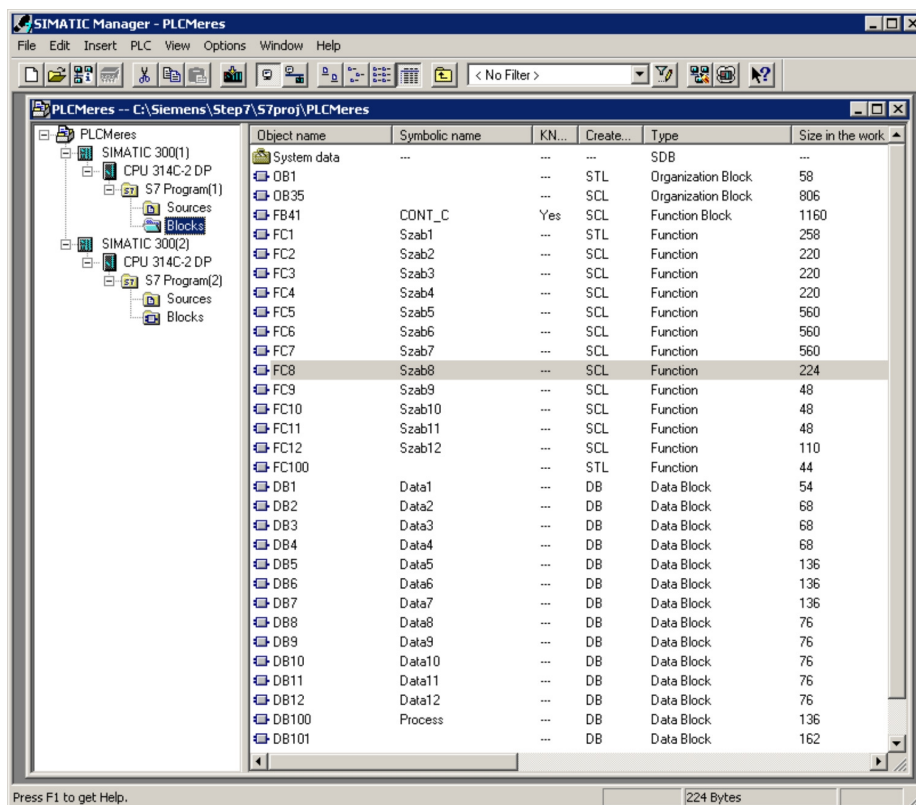


Figure 11-6. Framework project in Simatic Manager

If one of the function blocks are double clicked than the editor and the selected program block is loaded (for example FC4).

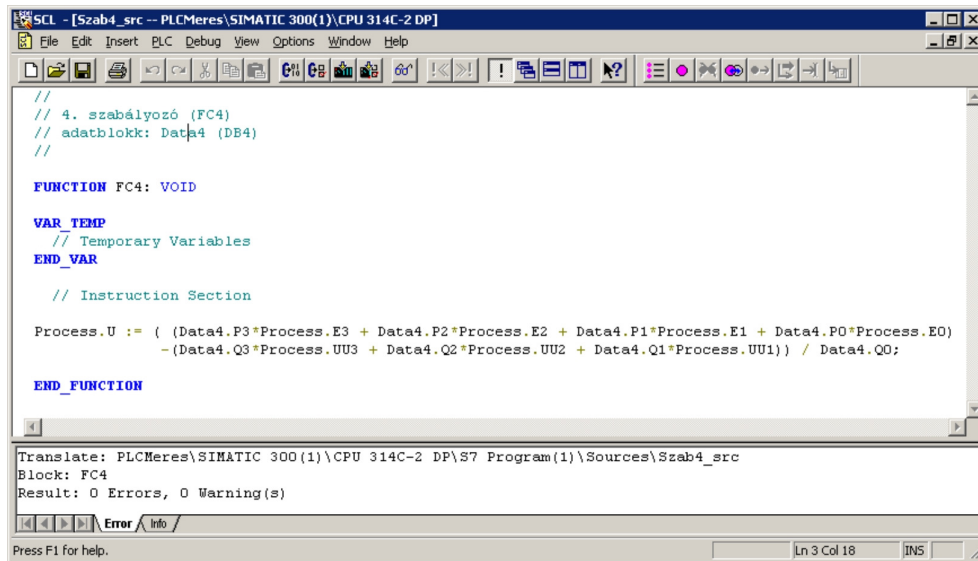




Figure 11-7. The SCL editor

The Simatic Manager and SCL editor user interface is very similar to any other windows based programs. The SCL program is compiled by the  button or File/Compile (Ctrl+B) menu command. The compilation errors are displayed in the lower portion of the window. The result of the successful compilation can be downloaded to the PLC by the  button or the PLC/Download (Ctrl+L) menu command.

WinCC userinterface

A user interface is available to control the thermal process and visualize the signals.



Figure 11-8. WinCC userinterface (for the translation of hungarian labels see next sections)

The upper portion of the window shows the block diagram of the system. The signals, the user interface controls, and the curves are color coded:

red - reference signal, ("Alapjel")

green (manual control input ("Kézi beav.") or control signal ("u-szab.),

blue and magenta – output signal

brown – error signal ("Hibajel")

The lower left corner shows the manual slider controls for the manual input, reference signal and ventilator setting. The ventilator ("Hűtés") can be controlled by the yellow slider.

The right side of the user interface controls the selection of the PLC programs. The red radio-buttons select the program and the pushbuttons display their parameter windows.

Temperatures exceeding the maximum temperature limit should be indicated by a high temperature event (red "Nyugta" button in the upper right corner). The overheating protection circuit turns the heating process off. The event is acknowledged and the system is started again by clicking the red "Nyugta" button.

The user interface makes it possible to record various signals. Start the data recorder by clicking the "Adatmentés start" button. In the "Adatmentés" window set the directory of the data file to ...*Matlab/work* and select the data file name (for example *datalog.m*). The system stores the measured data to the data file with $T_s=1\text{sec}$ sampling time. The data recorder can be stopped by clicking the "Adatmentés vége" button. The data file can be further processed in Matlab environment.

Programming the PLC

The actual variables of the process model are stored in the DB100 data block. They can be referenced by the *Process* name (for example *Process.Y* for the output):

<i>Name</i>	<i>Type</i>	<i>Function</i>
TIPUS	WORD	Type of controller
Y1_PER	WORD	Process output (I/O format) = IW10
Y2_PER	WORD	Process output (I/O format) = IW12
U_PER	WORD	Control signal (I/O format) = QW10
Y1	REAL	Process output T1
Y2	REAL	Process output T2
Y	REAL	Selected process output
Sp	REAL	Reference signal, "Alapjel"
U_MAN	REAL	Manual input, "Kézi beavatkozó jel"
U	REAL	Control signal
U1	REAL	k-1 Control signal
U2	REAL	k-2 Control signal
U3	REAL	k-3 Control signal
U4	REAL	k-4 Control signal
UU	REAL	Saturated Control signal
UU1	REAL	k-1 Saturated Control signal
UU2	REAL	k-2 Saturated Control signal
UU3	REAL	k-3 Saturated Control signal
UU4	REAL	k-4 Saturated Control signal
E0	REAL	k Error signal
E1	REAL	k-1 Error signal
E2	REAL	k-2 Error signal
E3	REAL	k-3 Error signal
E4	REAL	k-4 Error signal
MAX	REAL	Upper limit

MIN	REAL	Lower limit
YSZAB	BOOL	Output signal selection, 0:Y=Y1, 1:Y=Y2
HIBA	BOOL	High Temp. "Hiba"
NYUGTA	BOOL	Acknowledge High Temp. "Nyugta"
MINTAV	WORD	Sampling time* 100ms, "Mintavételi idő"

The cyclic execution of the PLC program is realized by the OB35 block. This block is called in every 100 msec by the operating system. The actual sampling time of the operation is *100 msec*, multiplied by the *Process.Mintav* variable. This adjustable sampling time is used to sample the signals and call the user programs. The *Process.Mintav* parameter can be adjusted from the user interface ("Ciklusidő").

From the user interface 12 programs can be selected and the actual program number is stored in the *Process.Tipus* variable. The parameters belonging to the program are stored in the data block with the same program number. For example the parameters of FC10 function block are stored in DB10 data block. The data blocks should be referenced by Data1, Data2, ... symbols.

For the measurement laboratory only programs 1, 2, 8, 9 and 10 are defined.

Program 1: Internal PID controller

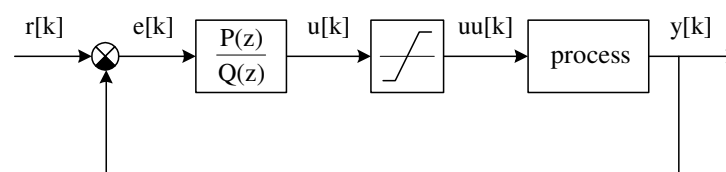
The internal PID controller (FB41) parameters are stored in the DB1 (Data1) data block:

Name	Type	Function
P	BOOL	P compensation on
I	BOOL	I compensation on
D	BOOL	D compensation on
K	REAL	Controller Kc
TI	TIME	Controller Ti (ms-ban)
TD	TIME	Controller TD (ms-ban)
T	TIME	Controller T (ms-ban)

$$C(s) = K \left(1 + \frac{1}{sT_i} + \frac{sT}{sT_d + 1} \right)$$

Program 2: Discrete controller

The block diagram of the controller:



$$P(z) = p_0 + p_1 z^{-1} + p_2 z^{-2} + p_3 z^{-3}$$

$$Q(z) = q_0 + q_1 z^{-1} + q_2 z^{-2} + q_3 z^{-3}$$

Figure 11-9. Discrete controller

The controller parameters are stored in the DB2 (Data2) data block:

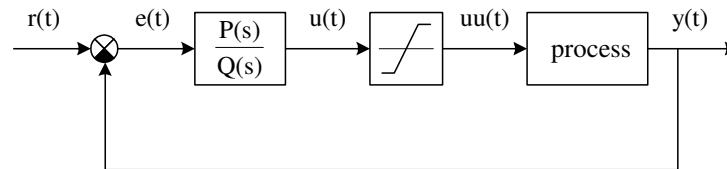
Name	Type	Function
Q0	REAL	q ₀ parameter
Q1	REAL	q ₁ parameter
Q2	REAL	q ₂ parameter
Q3	REAL	q ₃ parameter
P0	REAL	p ₀ parameter
P1	REAL	p ₁ parameter
P2	REAL	p ₂ parameter
P3	REAL	p ₃ parameter

The control algorithm can be found in the FC2 (Szab2) function block:

```
Process.U := ( (Data2.P3*Process.E3 + Data2.P2*Process.E2 + Data2.P1*Process.E1
+ Data2.P0*Process.E0) - (Data2.Q3*Process.UU3 + Data2.Q2*Process.UU2 +
Data2.Q1*Process.UU1)) / Data2.Q0;
```

Program 8: Continuous controller

The block diagram of the controller:



$$P(s) = p_0 + p_1 s^{-1} + p_2 s^{-2} + p_3 s^{-3}$$

$$Q(s) = q_0 + q_1 s^{-1} + q_2 s^{-2} + q_3 s^{-3}$$

Figure 11-10. Continuous controller

The controller parameters are stored in the DB8 (Data8) data block:

Name	Type	Function
Q0	REAL	q ₀ parameter
Q1	REAL	q ₁ parameter
Q2	REAL	q ₂ parameter
Q3	REAL	q ₃ parameter
P0	REAL	p ₀ parameter
P1	REAL	p ₁ parameter
P2	REAL	p ₂ parameter
P3	REAL	p ₃ parameter
X1	REAL	x ₁ parameter
X2	REAL	x ₂ parameter
X3	REAL	x ₃ parameter

The control algorithm can be found in the FC8 (Szab8) function block:

```
FUNCTION FC8: VOID

VAR_TEMP
    tmv : REAL;
END_VAR

tmv := WORD_TO_INT(Process.Mintav) * 0.1;

// integrátorok aktualizása
```

```

Data8.X3 := Data8.X3 + tmv * (Process.E0*Data8.P3 -
    Process.UU*Data8.Q3)/Data8.Q0;
Data8.X2 := Data8.X2 + tmv * ((Process.E0*Data8.P2 -
    Process.UU*Data8.Q2)/Data8.Q0 + Data8.X3);
Data8.X1 := Data8.X1 + tmv * ((Process.E0*Data8.P1 -
    Process.UU*Data8.Q1)/Data8.Q0 + Data8.X2);

// beavatkozó jel számítása
Process.U := Data8.X1 + Process.E0*Data8.P0/Data8.Q0;

```

Program 9: Continuous PI controller with anti-windup (FOXBORO controller)

In this measurement configuration the control signal is limited to the 0,+10V range due to the hardware realization. If the controller output reaches the maximum or the minimum value the signal saturates and the control loop falls out of the linear operating region. When the system returns to the linear region the operation significantly degrades due to the mismatches of the state variables. There are several solutions for this problems (they called anti-windup solutions). A simple and practical solution for PI controllers is the FOXBORO controller.

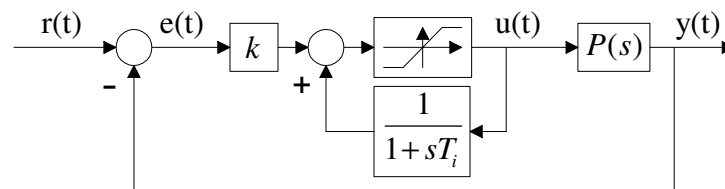


Figure 11-11. A FOXBORO controller

The FOXBORO controller in the linear region is identical with a traditional series PI controller.

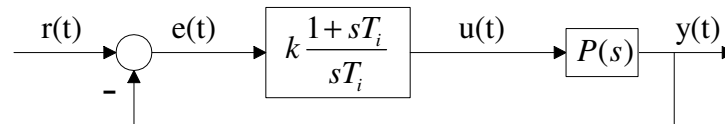


Figure 11-12. Series PI controller

The identity can be seen easily,

$$C(s) = k \frac{1}{1 - \frac{1}{1 + sT_i}} = k \frac{1 + sT_i}{sT_i}.$$

The FOXBORO controller solves the saturation problem by limiting the input of the controller and process the same way. The k and T_i parameters can be set from the user interface parameter window. The Szab9 operating program is

```

FUNCTION FC9: VOID

VAR_TEMP
    tmv : REAL;
END_VAR

tmv := WORD_TO_INT(Process.Mintav) * 0.1;
IF Process.Init THEN
    Data9.X0 := Process.UU * Data9.Ti;

```

```

END_IF;

Data9.X0 := Data9.X0 + tmv * (Process.UU - Data9.X0/Data9.Ti);
Process.U := Data9.K*Process.E0 + Data9.X0/Data9.Ti;

END_FUNCTION

```

Program 10: User program

Program 10 (FC10) is used for program development during the laboratory. This user defined program makes it possible to test different types of algorithms. A data block with 10 real parameters belongs to the FC10 function block. The parameters can be changed from the user interface and from the program they are referenced as Data10.Par1, Data10.Par2, ... , Data10.Par10.

Name	Type	Function
Par1	REAL	1. parameter
Par2	REAL	2. parameter
Par3	REAL	3. parameter
Par4	REAL	4. parameter
Par5	REAL	5. parameter
Par6	REAL	6. parameter
Par7	REAL	7. parameter
Par8	REAL	8. parameter
Par9	REAL	9. parameter
Par10	REAL	10. parameter

Sample SCL program

Here a simple example is provided to demonstrate the programming capabilities of the SCL language. Let's write a program for the PLC that compares the temperature reading of the heatsink to an adjustable parameter and based on the result sets a digital output.

Open the Szab10_src (the source code of FC10) program in Simatic Manager. The problem can be solved by the following program:

```

FUNCTION FC10: VOID

VAR_TEMP
    Limit : REAL;
END_VAR
    Limit := Data10.Par1;
    IF Process.Y > Limit THEN Q1.0 := 1; ELSE Q1.0 := 0; END_IF;
    Process.U := Process.U_MAN;
    Data10.Par2 := Process.Y;
END_FUNCTION

```

A *Limit* local variable should be declared between the *VAR_TEMP* and *END_VAR* commands. The parameters typed from the user interface are stored in the *Data10.Par1*, *Data10.Par2*, ... variables. These parameters not only store the entered values but they also display their present values. The value of the temperature reading is stored in the *Process.Y* variable. This value is also displayed in *Data10.Par2* variable. The Q1 digital port can be set as a byte with the *QB1:=0* assignment or by bits (*Q1.0 := 1;*). The state of the digital port is visualized in the user interface and on the PLC front panel. The bits can be also changed directly from the user interface. The *Process.U := Process.U_MAN* command assigns the value of the green slider control to the heating level (control signal) variable. This is necessary to verify the program operation. Compile the program and if there are no errors download it to the PLC. The program is activated by selecting the program number with the red radio buttons. The parameter editor window opens by the "10. Mintaprogram" push button. Set the value of 1.Paraméter to the desired temperature limit value.

Matlab processing

The previously stored data file has the following format:

$T_s = 1;$

$data = [..., ..., ..., ..., ...;, ..., ..., ..., ...];$

$t = data(:,1); y = data(:,2); u = data(:,3); uu = data(:,4); a = data(:,5);$

The data can be loaded into the Matlab workspace by evaluating the data file. Start Matlab and type the file name with no extension (for example *datalog*). The *mfile* file separates the data matrix into vectors with the following variable names: *t*(time), *y*(output signal, temperature), *u*(control signal), *uu*(saturated control signal), *a*(reference signal). The data can be visualized and processed by Matlab commands. Recommended Matlab commands to use: *plot*, *step*, *dsgain*, *tf*, *lsim*, *stairs*.

The *A* and *T* parameters of first order approximation can be read from the step response curve. The step response of the estimated transfer function should be compared to the measured step response:

$P=tf(A,[T,1])$

$ysim=lsim(P,u-u0,t)+y0;$

$plot(t,y,t,ysim), grid$

The resulted curves can be exported to a document by the *edit*, *copy figure* menu command.

Measurement exercises

1. Identification of system components

- 1.1. With the help of the laboratory instructor verify the measurement configuration: identify the hardware components, the connections of the thermal process and the software components of the man-machine interface. List the following system components:

The hardware components of the system:

The components of the man-machine interface:

2. Basic operation of the thermal process and the measurement framework

- 2.1. At zero heating and 100% ventilator settings measure the output temperature of the heating process (ambient temperature, ϑ_a).

<results >

- 2.2. By raising the heating voltage verify that the overheating protection circuit operates. Temperatures exceeding the maximum temperature limit should be indicated by a high temperature event (red "Nyugta" button in the upper right corner). Verify that the overheating protection circuit turns the heating process off. Measure the turn-off temperature. Acknowledge the overheating signal (click the red "Nyugta" button) and verify that the heating process is operating normally.

< results >

- 2.3. Measure the steady-state temperature value at a given heating and ventilator settings provided by the lab instructor. These heating and ventilator settings are used as the working point values for further measurements.

Working point heating voltage (u_0):

Working point ventilator value:

Measured steady-state temperature ($y_0 = \vartheta_0$):

- 2.4. Measure the maximum heating voltage for which the overheating protection does not shut off. Use the constant ventilator setting from exercise 2.3.

< results >

3. Identification of the thermal process

- 3.1. Identify the thermal process for the working point (heating voltage, ventilator setting) given in exercises 2.3. Set the heating cooling values and wait until the system settles (the output temperature is constant). Start the data recorder by clicking the “*Adatmentés start*” button. In the “*Adatmentés*” window set the directory of the data file to *...Matlab/work* and select a name for the data file. Raise the heating voltage value by 2V (step response input). When the system settles again stop the recording by clicking the “*Adatmentés vége*” button. Visualize the results in Matlab environment (use the *plot* command and the *edit, copy figure* menu command to import the curves). Verify that the MATLAB and WinCC environments provide similar results.

< results, Matlab figures >

- 3.2. Approximate the heating process by a first order system. From the measured data and curves identify the static gain and the time constant of process. Compare the identified process to the measured data. (Use the *step* and *lsim* commands).

< results, Matlab figures, conclusions >

4. Closed loop control of the temperature.

- 4.1. Based on the identified process parameters design a PI controller. Adjust the ventilator to the same level used for the identification. Select the continuous PI controller with saturation (9. controller). From the user interface apply a constant reference signal (red, “Alapjel”) and verify that the output temperature follows it. If the system is stabilized apply small step input signals (for example 5°C step input) and record and evaluate the results. Adjust the controller parameters that the response is fast and the output overshoot does not exceeds 10%.

< results, curves, conclusions >

- 4.2. Examine the parameter sensitivity of the control loop. Set constant reference signal. When the system settles apply a disturbance signal by reducing the ventilator setting to half of its previous value. Record and evaluate the results. How the output and control signal is changed? How much static error is resulted by applying the disturbance?

< results, curves, conclusions >

5. Programming the PLC with Simatic Manager

- 5.1. Write a PLC program for FC10 function block which executes the task chosen by the laboratory instructor. The program should be compiled, downloaded and its operation verified. **Since the FC10 function block is evaluated automatically in every sampling period, a separate program cycle is not needed.**

- a. Switch the digital outputs depending on the temperature.
- b. Indicate by a selected digital output if the temperature stays in a given interval.
- c. Indicate by a digital output if the temperature exceeds a given limit for longer than a given time.
- d. Indicate by a flashing digital output if the temperature exceeds a given limit.
- e. Implement a combinatorial logic circuit given by the instructor.
- f. Realize a simple two state controller (bing-bang control) for the temperature control.

<PLC program >
