ISSN 0280–5316
ISRN LUTFD2/TFRT--5665--SE

Implementation of a PID Controller for Building Automation

Fredrik Holmberg

Department of Automatic Control
Lund Institute of Technology
March 2001

Department of Automatic Control **Lund Institute of Technology** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER THESIS *Date of issue* March 2001 *Document Number* ISRN LUTFD2/TFRT--5665--SE
Author(s) Fredrik Holmberg	*Supervisor* Henrik Olsson, TAC Tore Hägglund *Sponsoring organisation*

Title and subtitle

Implementation of a PID Controller for Building Automation
(Implementering av en PID-regulator för byggnadsautomation)

Abstract

This master thesis project has been performed in cooperation with TAC, a building automation company in Malmö and the Department of Automatic Control at LTH. TAC has experienced problems for some time with their PID controllers. This master thesis has therefore investigated the use of PID control at TAC, the problems with the existing controllers. The thesis also describes a new PID controller that has been implemented at TAC.

The issues treated in the thesis are not only technical but also deal with the processes within a company when changing a control algorithm. The non-technical issues are for instance how the force of habit affects the development and that backward compatibility must be ensured. The new algorithm that was decided on and implemented is of a parallel-positional form. This basic form of the PID algorithm is proven to be the best choice. From a control point of view the controller successfully controls the processes in TAC's area of business. The PID controllers at TAC are used to control HVAC (Heating Ventilation and Air Condition) systems. The conclusion that the new PID controller is successful is based on simulations as well as tests at a commercial system. The PID controller is described in detail in the thesis. Features of the controller like setpoint weighting is described, as well as a tracking signal that allows the controller to be forced to externally specified values. Other features that are described include an offset added during control without the integral part, bumpless transfer, a rate of change limiting mechanism and other necessary features to make the controller practically usable.

An investigation concerning possible concepts that may be useful in a new application programming tool has been performed as a part of the master thesis. The concepts determined to suit TAC are incorporated in IEC 61131. The concepts discussed in detail are amongst others SFC, the use of processes-tasks, instantiation and inheritance.

Key words

Classification system and/ or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280–5316		*ISBN*
Language English	*Number of pages* 47	*Recipient's notes*
Security classification		

Table of Contents

Acknowledgements

I wish to express my gratitude to several people for making this master thesis possible, above all Henrik Olsson at TAC for his support. I would also very much like to thank my advisor at LTH Prof. Tore Hägglund for his insights, as well as several employees at TAC, which I will not name so that I do not forget any single person.

1 Introduction

This thesis has been carried out in cooperation with the Department of Automatic Control at LTH and TAC in Malmö.

TAC is a multinational company within the field of building automation and has about 2000 employees around the world. The turnover for the company is about three billion SEK. The company's ambition is to develop, produce and market products and open system solutions and services which improve the indoor environment and provide the building owner with added value in the form of lower operating costs and more attractive properties.

TAC has experienced problems for some time with their PID controllers, which has prompted this master thesis. The aim of the master thesis has been to investigate the use of PID control at TAC and if necessary implement a new PID controller that regulates HVAC (Heating Ventilation and Air Condition) systems. The solution to the problem should be of high industrial standard and the implementation should be easy and functional to use for programming of larger systems. The solution should also be usable for many years ahead.

The aim of HVAC control is to maintain predefined levels of temperature, CO_2 and so on. These levels are specified in such a manner that they are within acceptable limits for a human being. The levels should also take into consideration the work environment and the kind of work that is carried out. Another consideration that is important for the industry is the energy consumption. This must be kept down not only to make the customers satisfied, but also to satisfy legal demands. The legal demands are for instance that heat is recovered from an airflow leaving a building.

Within the field of HVAC the processes are generally not complicated enough to require an advanced controller structure, it is generally enough to use a PID algorithm. This is also the case within TAC. The company uses two different PID controllers for their applications today. Considerations about the personnel working with the systems are also necessary, the application programmers and commissioning personnel are simply not allowed to spend too much time on the controller, for instance on tuning.

The master thesis work was initiated with a literature study to find information about the specific problems that may be experienced within the field of HVAC. The problems at TAC were also studied in detail during this phase. The work has then been focused on implementation and testing of a PID controller.

To be able to use a controller in an industrial environment a lot of testing is needed, and has been performed. The tests were performed both in a laboratory environment and in a commercial system.

An investigation concerning possible concepts that may be incorporated in a new application programming tool has been performed as a part of this master thesis. The concepts found to suit TAC are incorporated in IEC 61131.

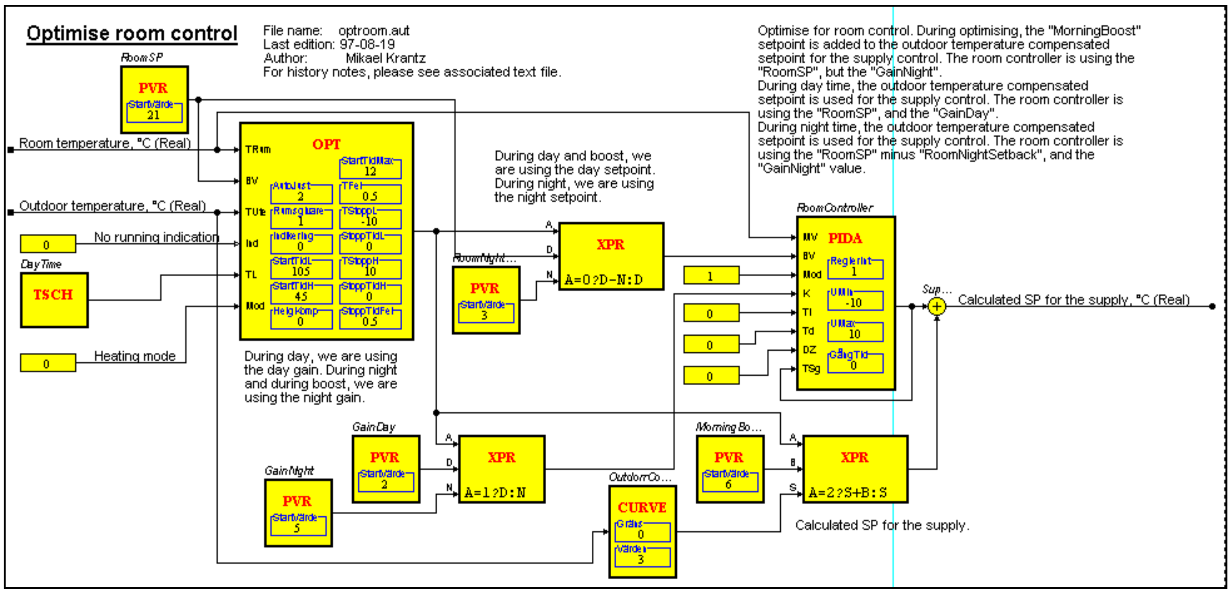
Chapter 2 of the thesis describes some of the background issues about the tools used at TAC and goes into detail about the PID algorithms and the problems with them today. A more extensive investigation about the features of the new algorithm is also given. Chapter 3 describes the new algorithm both in function and in the mathematics behind it. Implementation issues that have come up during the work are also described in Chapter 3. Chapter 4 contains simulation and testing results. In Chapter 5 the description of possible concepts for a new application programming tool is given. Conclusions are accounted for in Chapter 6.

2 Background

This chapter contains background material concerning both the tools used at TAC to program applications and the PID algorithms used at TAC, see Section 2.1. The examination of the PID algorithms also contains a problem description of the existing controllers, see Section 2.2. This problem description leads up to a specification of the new algorithm that will be accounted for in Chapter 3.

2.1 The Programming Tools

TAC uses a graphical programming tool called TAC Menta® to program control systems for HVAC. In this graphical tool, blocks are used for defining different operations, such as PID control and enthalpy calculations. In the programming tool, two different building blocks exist that uses a PID algorithm. These blocks are used for all the company's control applications. The tool also contains all the other different programming expressions that are needed for controlling such systems.


Optimise room control

File name: optroom.aut
Last edition: 97-08-19
Author: Mikael Krantz
For history notes, please see associated text file.

Optimise for room control. During optimising, the "MorningBoost" setpoint is added to the outdoor temperature compensated setpoint for the supply control. The room controller is using the "RoomSP", but the "GainNight".
During day time, the outdoor temperature compensated setpoint is used for the supply control. The room controller is using the "RoomSP", and the "GainDay".
During night time, the outdoor temperature compensated setpoint is used for the supply control. The room controller is using the "RoomSP" minus "RoomNightSetback", and the "GainNight" value.

During day and boost, we are using the day setpoint. During night, we are using the night setpoint.

During day, we are using the day gain. During night and during boost, we are using the night gain.

Figure 2-1 View from TAC Menta® of a macro block.

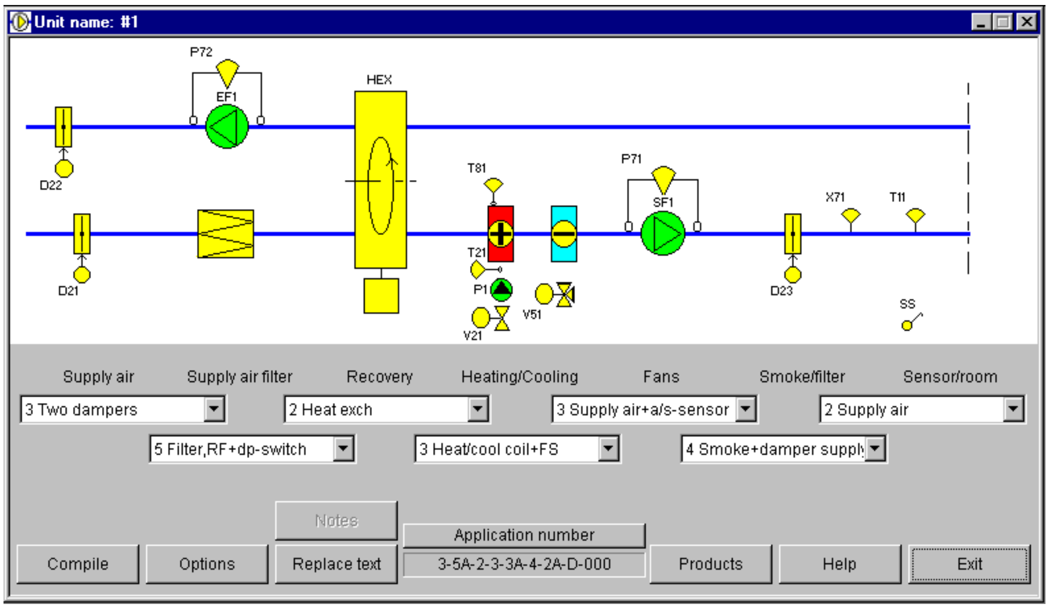
The two programming blocks in TAC Menta® that contain the PID algorithm are named PIDI and PIDA. Further details about the blocks and the differences between them are given in Section 2.2 below.

TAC´s development department has implemented several larger blocks, called macro blocks. These macro blocks solve different control application problems, see Figure 2-1 for an example of a macro block. Another macro block shows a very nice solution of how to perform sequential control of a heating coil, a heat exchanger and a cooling coil, see Figure 2-10.

With these blocks, a problem solving philosophy is spread to the users of the programming tool. These blocks are also intended as a timesaver during programming of applications, which is increasingly important in the automation business today.

TAC has also developed TAC Design+. This tool is programmed using symbols that show for instance a heat exchanger and a fan, see Figure 2-2. These blocks then correspond to specific applications from TAC Menta®. These applications have been pre-programmed in TAC Menta® and are added together to a complete Menta application. This application can be downloaded to the hardware and perform the needed tasks. The tool also produces labels for the hardware, several descriptions of for instance the functionality of the system and designs the valves.

The construction of all the applications produced by TAC Design+ is performed by using the predefined Menta applications. These applications are often not the most efficient ones for the specific task. Since the PID controllers are used in several of the blocks it is important to understand that the amount of time spent on a good control strategy is limited.



Supply air	Supply air filter	Recovery	Heating/Cooling	Fans	Smoke/filter	Sensor/room
3 Two dampers	5 Filter,RF+dp-switch	2 Heat exch	3 Heat/cool coil+FS	3 Supply air+a/s-sensor	4 Smoke+damper suppl	2 Supply air

Application number
3-5A-2-3-3A-4-2A-D-000

Figure 2-2 TAC-Design+ showing a typical programming view.

2.2 PID Algorithms Today at TAC

TAC uses the incremental form of the PID control algorithm. The PIDI block uses the basic incremental algorithm and the PIDA block uses an altered incremental version that is capable of giving a positional reference.

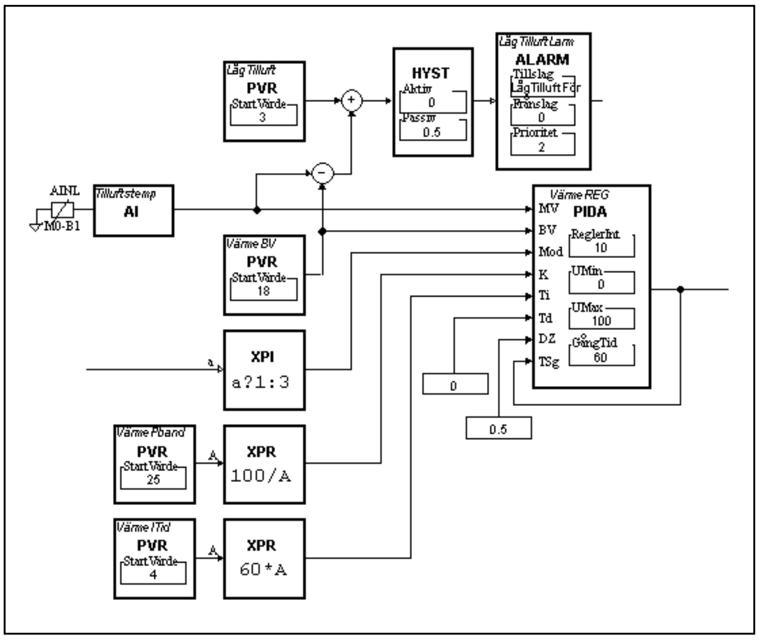


Figure 2-3 View from TAC Menta® of a PIDA block that is used as an ordinary positional controller.

The algorithm that is used in the PIDA block has been afflicted with some built-in problems that during the years have been experienced as troublesome. The problems have primarily been handled by using controllers that are careful, i.e. low gains and long integral times. The derivative part has not been used at all, partly because of an implementation that simply is not usable in practice. Since the problems were well known the employees could handle them. The PIDA block uses an old, and compared with controllers of today, not very efficient algorithm. This has also contributed to the problem with the controllers.

The effects of the problems are primarily discovered at building inspections and do not appear during normal operating conditions. This is a major reason why the control algorithms have not been changed. The control algorithms are generally not something that are changed easily because of the great impact it has on the behaviour of the company's products.

The Incremental Algorithm

To fully understand the problem, a short description of the incremental PID algorithm is given before details about the two algorithms are discussed.

Basic action of the incremental form, or velocity form, of the PID algorithm is to calculate the change in control signal that is needed, the increment. The name velocity form comes from the fact that the algorithm normally gives a velocity reference as output. This is efficient when the actuator has some sort of integrating action already built in. Like for example a valve that remains in position when the control signal is zero.

Another feature is that integrator windup is not a problem that has to be considered, since the valve cannot move beyond 100% open and cannot move beneath 0%. This means that integration stops automatically when the actuator reaches its end position, either max or min.

A bumpless transfer means that there are no changes in the output if the parameters are changed when the error is zero. All incremental algorithms meet this statement.

Eq 2-1 shows the variant of the incremental PID algorithm used at TAC, where MV(k) is the measured value and E(k) is the error between the setpoint and the measured value at time k.

$$du(k) = G\left(E(k) - E(k-1) + \frac{h}{T_i} E(k) - T_d \frac{MV(k) - 2MV(k-1) + MV(k-2)}{h} \right) \qquad \textbf{Eq. 2-1}$$

The PIDI and PIDA Blocks

The PIDI algorithm is based completely on the incremental algorithm and it is therefore only capable of calculating an increment. This makes it especially well suited for decrease/increase signals to a valve or a damper.

These decrease/increase signals are active for a specific amount of the stroketime, *dt*, for the valve or damper; see Eq. 2-2 below. Where *du* is the amount of increase, the increment, which has been calculated by Eq. 2-1.

$$dt = \frac{du[\%]}{100[\%]} \cdot StrokeTime \qquad \textbf{Eq. 2-2}$$

What makes the PIDA block different is the input for feedback of signals, see Figure 2-4. The reconnected feedback-signal is called the tracking signal, TSg. The PIDA algorithm is therefore capable of calculating a position reference. The position reference is possible because of the summation of increments, or velocities, that will be carried out automatically. The position reference can then be used for instance in cascaded control. The algorithm in the PIDA block is still basically the incremental algorithm. Figure 2-4 shows a block diagram of the PIDA algorithm. Figure 2-5 also shows the algorithm, with a focus on the information carried in the signals.

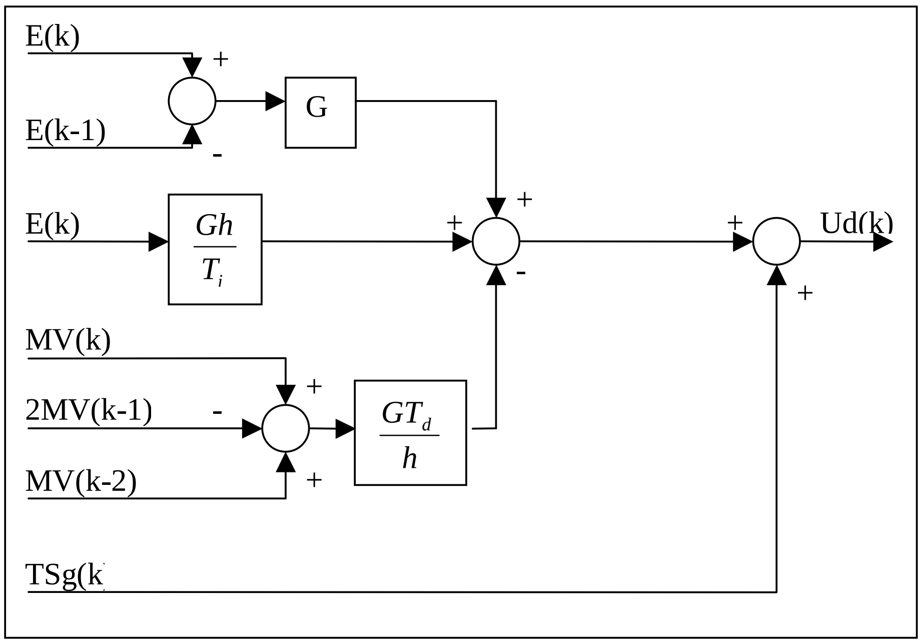


Figure 2-4 Block-diagram showing the incremental algorithm, together with the reconnected tracking signal TSg.

The output from the PIDA is limited both in rate of change and maximum and minimum i.e. the user sets limit for minimum and maximum values for both actions.

The reconnected signal is often manipulated in some way, before it is fed back. The manipulations aim to make sure that the controller does not reach some sort of forbidden state. For example a valve is often not allowed to be completely closed or completely open, to ensure some circulation of for example air.

The output from PIDA will be a position reference if the output is reconnected to the TSg input, instead of a velocity that is the normal signal from an incremental algorithm, as in the PIDI block. The TSg input could also be used for example to connect an offset value. In conclusion, the outputs from the two blocks are very different in their applications. The effects of these differences are given more attention in Section 2.4, when discussing merging of the two blocks.

The Problems with the Old Controllers

TAC has had problems with the controllers at saturation. These problems can be referred to the fact that the algorithm does not contain any internal state variables that have accurate and complete information about the amount of saturation or the history of the controller. The only information of this kind is incorporated in the reconnected signal, the signal that is called TSG in Figure 2-5. This variable is limited between a maximum and a minimum value and possibly altered in calculations between the output from and the input to the controller. During these limitations and calculations, information is lost.

This problem will be present as long as the TSG signal is a reconnection of the output, and consequently as long as the controller is producing positional references.

Figure 2-5 shows the structure of the PIDA algorithm as well as where the limitations and calculations are performed on the reconnected signal. It also shows what information the different signals are supposed to contain.

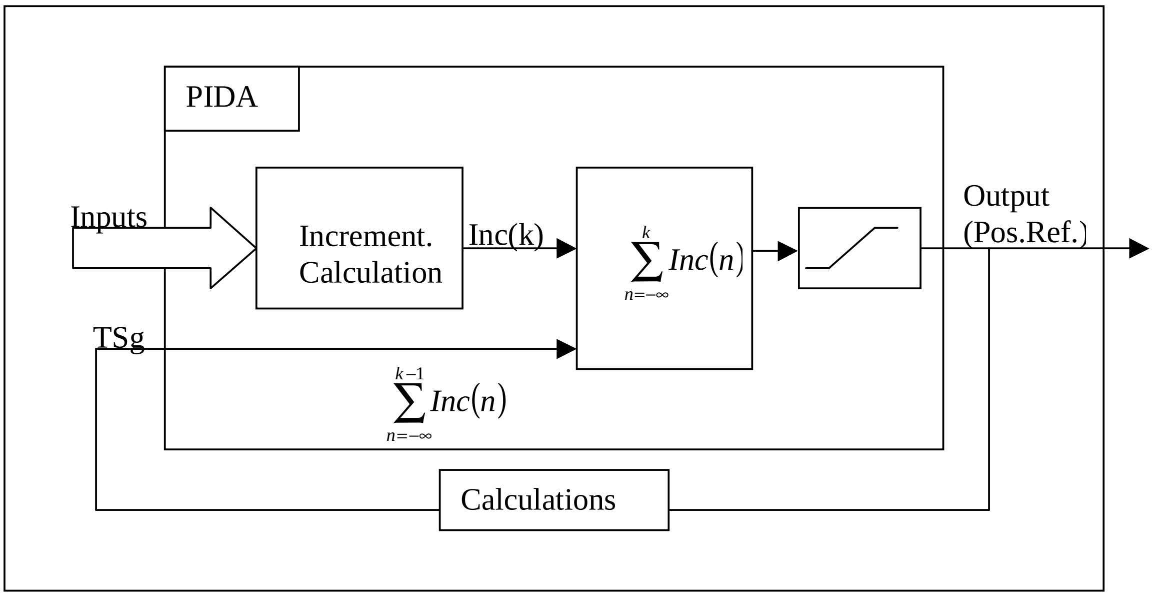


Figure 2-5 The overall structure of the PIDA algorithm. The figure also shows the intended information in the different signals.

What could and has happened is that a change in the setpoint, or the measured value, could dramatically change the control signal output. This change is independent of earlier values of the control signal.

Consider for example a setpoint of 20ºC and a measured value at 15ºC, this value is constant for a longer period of time and the controller saturates. An increase in the measured value however small will imply that the output from the controller is decreased. This behaviour is not acceptable because it is unlikely that since the controller has been saturated for a longer period of time that the energy need would decrease even if the measured value increases.

Figure 2-6 shows the effects mentioned above, the step in the measured value is in the plot 1ºC to make the effect visible. A plot of how a positional controller performs under these circumstances is also added as a reference.

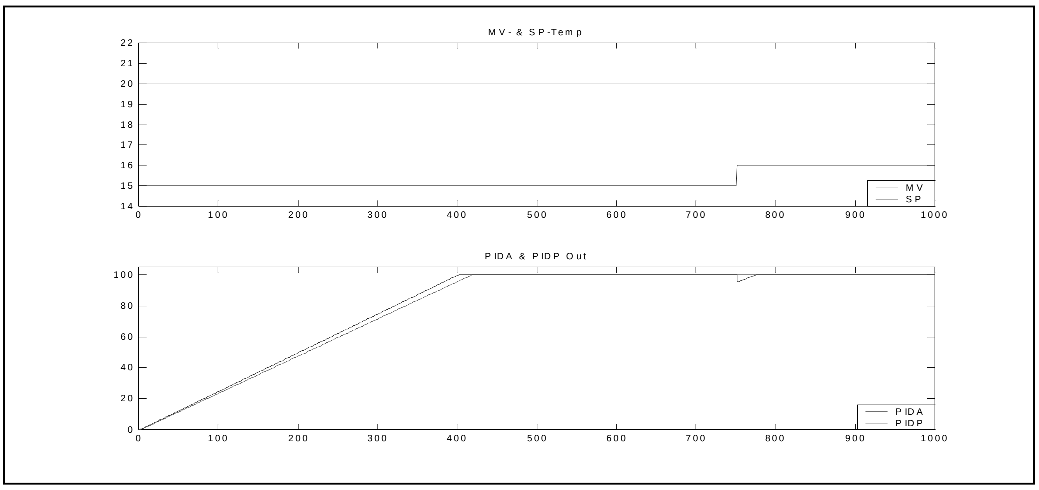


Figure 2-6 Plot showing the effects at saturation for an incremental and a positional controller.

TAC has had another problem with their controllers, the problem appears as a process output with a stationary oscillating fault, and an oscillating control signal see Figure 2-7. The problems appear when using a PI controller in fast and nonlinear processes.

The problem is known as “P-part dominance” at TAC.

The behavior can be referred to three factors:

- Badly tuned controllers.
- Large differences in process gains in nonlinear processes.
- A built in “error” in the PIDA algorithm.

The problems in Figure 2-7 are due to a nonlinear process and controller parameters that are not perfectly tuned. The figure nonetheless shows the behavior of the problems in all three cases.

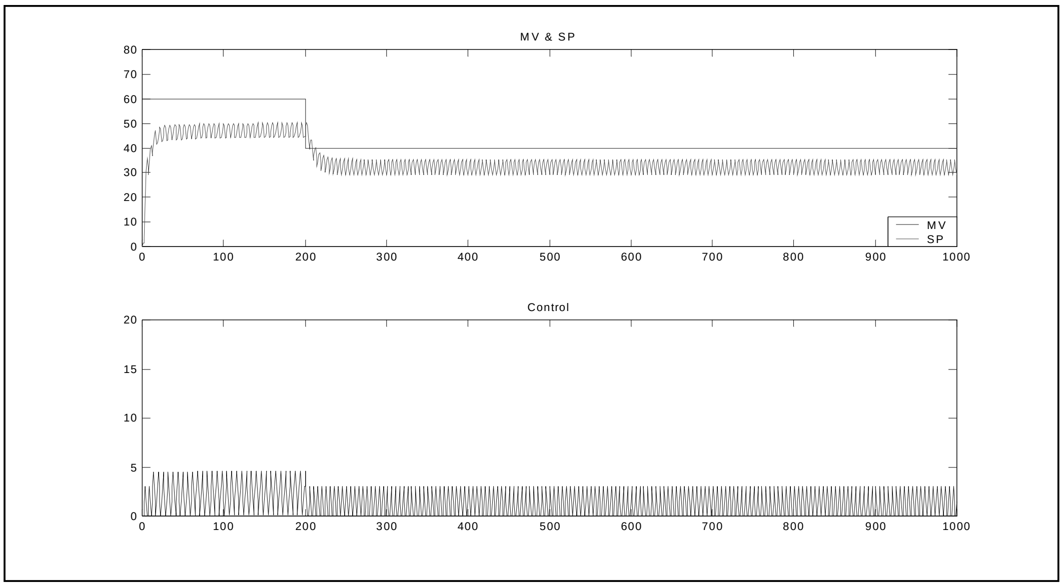


Figure 2-7 Plot showing the process output, the setpoint and the control signal for a PIDA block controlling a domestic hot water process.

That the controller is badly tuned is something that all controllers can be, this cannot be blamed on the algorithm. The problem with large differences in process gains is the same problem as with a badly tuned controller and therefore nothing that the algorithm can take care of.

However, this problem could easily be solved if TAC starts to use gain scheduling when controlling nonlinear processes. This problem is in reality limited to one of TAC's processes, the domestic hot water-process and the employees at TAC have very limited knowledge of gain scheduling. Because of this TAC does not use gain scheduling for this purpose.

The built in errors are something that a user cannot affect, but they can give just as powerful effects as a bad tuning.

The saturation problem and its negative effects on the control signal will affect also this type of problem. In short, vital information about the history of the control will be lost.

If the proportional part is broken out of the algorithm, the effects that are described below can be seen. See also Figure 2-4 and Figure 2-5:

$$U = \sum_{n=-\infty}^{k} Inc(n) \Rightarrow \sum_{n=-\infty}^{k} \Delta P(n) = G \sum_{n=-\infty}^{k} E(n) - E(n-1) = G \sum_{n=-\infty}^{k} (SP(n) - MV(n)) - (SP(n-1) - MV(n-1))$$

Eq. 2-3

Under the assumption that SP is constant, the proportional part is equal to:

$$\sum_{n=-\infty}^{k} \Delta P(n) = G \sum_{n=-\infty}^{k} MV(n-1) - MV(n) = G\left(\sum_{n=-\infty}^{k-1} \Delta P(n) + (MV(k-1) - MV(k)) \right) = -G \cdot MV(k)$$

Eq. 2-4

This will mean that the proportional part depends only of MV(k) as long as all the history is available, and no setpoint changes are made. Should on the other hand the history, $\sum_{n=-\infty}^{k-1} P(n)$ be lost because of limitations etc., the proportional part will mainly depend on the difference between two consecutive measured values. This is a derivative approximation. Because this is not intended and the controller is not tuned for this case, problems with stability might arise.

If the controller had internal variables that was not affected by limitations and calculations and that contained the total history of the controller this problem could have been solved. The algorithm would in this case have been close to the, in this case, better-suited positional algorithm. It would not be worth the effort to make the changes compared to implementing something altogether new.

Together and by themselves the causes discussed above have contributed to and caused the problem of "P-part dominance". These problems will disappear in a positional algorithm to some degree. The problems concerning bad tuning will remain, this problem does not depend of the algorithm and cannot be removed by changes in the PID algorithm. Bad tuning is something the user has to verify and control. The causes mentioned as built in will on the other hand disappear.

In the PIDA algorithm the setpoint is allowed to affect the controller output immediately. The example below describes some of the effects of this.

The controller is run as a PI-controller and the sampling period, h, is 1s.

Gain, G=5. Integral time T_i=100. A setpoint at 22ºC and a constant measured value at 14ºC. The proportional part will become zero since the error is considered constant at 8ºC for at least two samples. (The measured value can be considered constant if the process is slow.) Under the condition that the controller is at a state of 80%, the output will be according to Eq. 2-3.

$$U = (5(8-8)) + \left(\frac{5 \cdot 8}{100}\right) + 80 \approx 80\% \qquad \textbf{Eq. 2-5}$$

The assumption of controller state is randomly chosen just as the other values, as seen in Figure 2-8 it is however a relevant figure.

If a step change in the setpoint takes place for example to 19ºC, the following will happen with the control signal. The effect is shown in Figure 2-8 at t=200:

$$U = (5(5-8)) + \left(\frac{5 \cdot 5}{100}\right) + 80 \approx 65\% \qquad \textbf{Eq. 2-6}$$

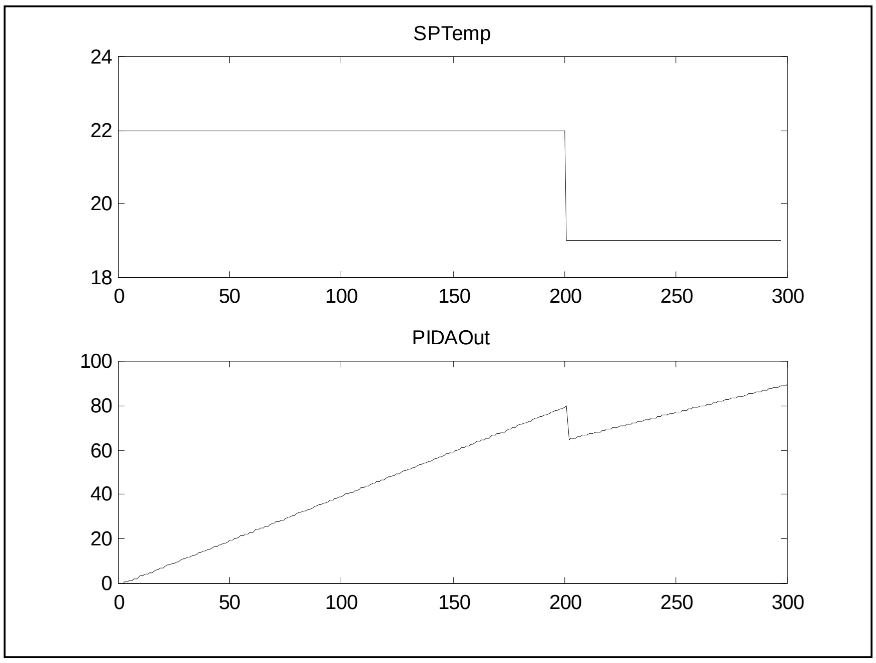


Figure 2-8 Showing the effects of a setpoint change from the example.

The conclusions that can be drawn from this example is that although the error between the measured value and the setpoint value has the same sign the control signal is decreased. The algorithm is designed to allow the setpoint to affect the control signal immediately, this effect is a feature in the algorithm. However, it will give undesirable effects. The design could mean that the controller is forced to unwanted and forbidden states.

The example above is only a small example to show the effects. When considering for example a controller with an output of 10% this setpoint step will force the controller from a heating mode to a cooling mode in one sample. This effect is not acceptable.

The derivative part is implemented in such a way that a high frequency disturbance or the high frequency part of a step in the measured value, is allowed to affect the control value without any filtering. Normally this is conducted via a built in low pass filter, since this is missing the derivative part is not practically usable.

The faults in the PIDA algorithm can be summarized as:

- Derivative action useless in practice.
- Poor performance for large setpoint steps and setpoint steps during high gains.
- The performance of the controller is difficult to analyse.

2.3 Use of PID Control at TAC

When discussing the topic of control at TAC it is important not to forget where the regulators actually are used. These basic units have a limited processor and memory space, because of money saving issues. The majority of TAC´s regulators control dampers in a room, a temperature valve or an airflow damper. The valves or dampers can be of different type and control different physical units such as heat, carbon dioxide concentration and airflow. The physical position of the valve can also differ greatly from a large fan room to a small office.

The performance of the single regulator is therefore in general not critical because it does not control a critical process by itself.

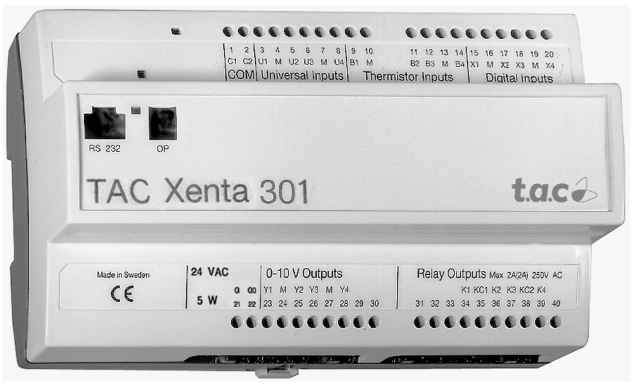


Figure 2-9 Picture of a TAC Xenta® 301, programmable regulator from TAC.

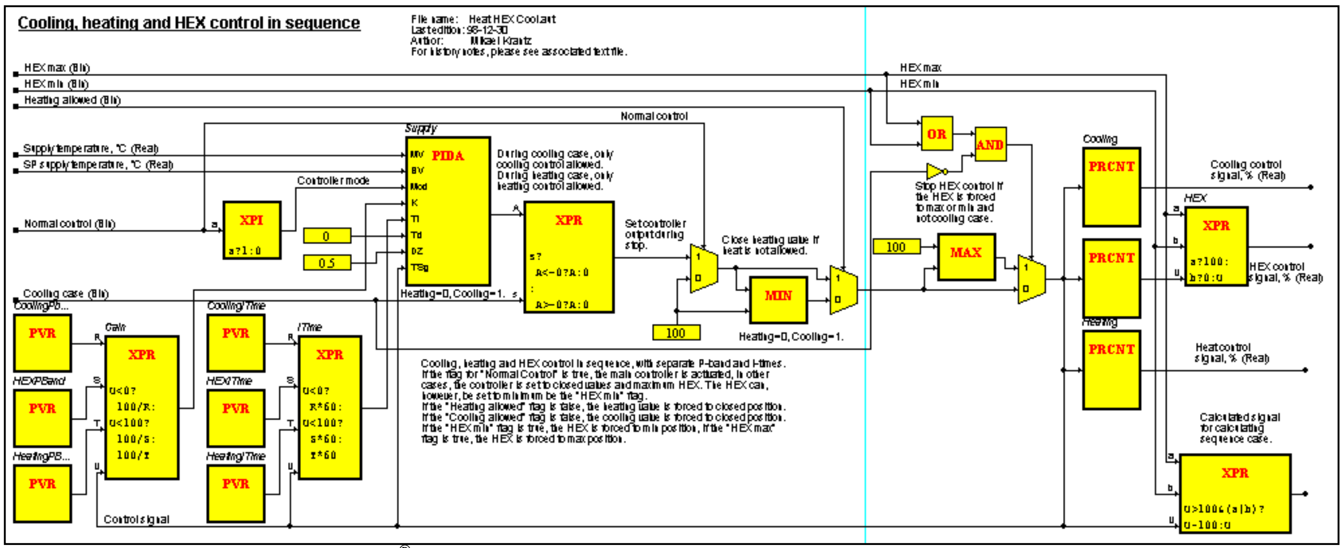
Two different situations for PID control were isolated in TAC´s systems.

Primary use for the PID block, either PIDA or PIDI, is as a controller that controls the position of a valve or damper. The difference between the two blocks is, as mentioned before, primarily the output.

The situations when a PIDI block is used for increase/decrease output signals to a valve are interesting. The increase signal means that a valve opening action is activated for a specified amount of time, and a decrease signal means the opposite. The amount of time is given by the PIDI signal. This situation is as concluded before very well suited for the incremental PID algorithm.

In the second situation a PIDA block generates a setpoint value for another PID block, of both types, so called cascaded control.

Common for the different situations are that the control signal often is limited in some way before it reaches the actuator, or in the cascade situation the second controller. This action means that many logical expressions are used to take care of different situations that can come up, see for instance Figure 2-7.


Cooling, heating and HEX control in sequence

File name: Heat HEX Coolant
Last edition: 98-12-30
Author: Mikael Krantz
For history notes, please see associated text file.

Cooling, heating and HEX control in sequence, with separate P-band and I-times.
If the flag for "Normal Control" is true, the main controller is actuated, in other cases, the controller is set to closed values and maximum HEX. The HEX can, however, be set to minimum be the "HEX min" flag.
If the "Heating allowed" flag is false, the heating valve is forced to closed position.
If the "Cooling allowed" flag is false, the cooling valve is forced to closed position.
If the "HEX min" flag is true, the HEX is forced to min position, If the "HEX max" flag is true, the HEX is forced to max position.

HEX max (Bin)
HEX min (Bin)
Heating allowed (Bin)
Supply temperature, °C (Real)
SP supply temperature, °C (Real)
Normal control (Bin)
Cooling case (Bin)
Normal control
Supply
PIDA
During cooling case, only cooling control allowed.
During heating case, only heating control allowed.
Controller mode
XPI
XPR
Set controller output during stop.
Close heating valve if heat is not allowed.
MIN
MAX
OR
AND
Stop HEX control if the HEX is forced to max or min and not cooling case.
100
Heating=0, Cooling=1.
PRCNT
Cooling
Heating
HEX
Cooling control signal, % (Real)
HEX control signal, % (Real)
Heat control signal, % (Real)
Calculated signal for calculating sequence case.
PVR
Gain
HEXPBand
HeatingPB...
CoolingPb...
CoolingITime
HEXITime
HeatingITime
ITime
Control signal

Figure 2-10 View from TAC Menta® of a macro block. Notice the manipulations of the reconnected signal.

Another situation that is common to both types of control, when using the PIDA block, is that the tracking signal is used in many places. For example:

- It is used to keep different controllers at the same state independently of which controller or logical expression that has computed the present control signal.
- To change controller parameters when the same controller is used for different processes, as in the case in the macro block in Figure 2-10. Where three different processes are controlled and different sets of parameters are necessary for each process.

2.4 Considerations when Changing the Algorithm

It is important to understand that the solving of a problem with the control algorithms are not performed over night, even if the solution should be relatively easy from a theoretical point of view.

The solution must be allowed to grow in the company, to make sure that the change is accepted and above all used in the programming in the future. Even if a fault could be proven to exist in a controller, the reasoning could very easily be that it has worked until now and why should it not work in the future.

Therefore, many discussions took place during the work of this thesis with people in different places in the corporation both geographically and administratively. During these discussions, an extensive mapping of how TAC uses PID control was made and several ideas were presented both to and from the staff at TAC.

The task given in the master thesis actually contains two different angles of the problem, the question of which and what kind of blocks that should be available and which algorithm the controller should use in the future. Primarily because of the limited memory available in the controllers, it has been a part of the master thesis to investigate whether the two blocks could be merged.

Since it is only in one of the blocks that TAC has experienced an actual fault, it would have been sufficient to take care of only that problem. The two existing blocks are also very similar in their actions and could possibly be merged into one.

The problem about the different blocks is in many ways a problem of philosophy. The problem concerns which operations that should be performed in the controller and how these actions are to be performed. To use another form of the PID algorithm will mean that the well-known responses, by TAC´s applications programmers, from the controller will be slightly altered. The tracking signal will for instance be affected. This means that a somewhat new thinking about the control will have to be used.

Which algorithm that should be used is more focused on engineering issues such as step responses in measured and setpoint values and implementation issues.

In conclusion, the arguments for and against a new algorithm could be summarized as:

For a change:

- The problems and actual faults in the algorithm. See for instance the examples in Section 2.2.
- Increased performance of the controller.

Against a change:

- The force of habit. The algorithm has proven successful for the company up to now.
- A slightly more complicated algorithm.

The part of the problem about which programming blocks that was going to be used, was the most complicated one to solve. When this problem was first brought to attention at TAC, a discussion started with employees in the company about which programming blocks that should be used. Since application programmers are individuals, it has been very difficult to find a solution that everybody are happy with, because it is primarily a question of what kind of philosophy the application programmers want to use.

The merging was natural since the two blocks were so similar in their implementation and internal action. This idea was discarded primarily because it was deemed to be too confusing for the users, in comparison to the relatively little gain in memory capacity that would be the primary gain of the operation. The merging would mean that a mode variable in the block would have to be used with a tremendous power, because of the effects of a faulty value. This mode variable should be used to separate the two cases of action, incremental and positional, that the new block would perform.

This whole reasoning about the variable may sound simple but the force of habit should not be overlooked. Since merging meant that some kind of signal had to be used, it would not be as obvious as desired.

Finally, the solution that was accepted was to make small improvements to the PIDI and PIDA block that is used in the controllers today and to further develop a new block that uses a more sophisticated algorithm. The new effective block should take care of the problems experienced today. The improvements in the PIDI and PIDA block were intended to make more efficient use of both memory and processor capacity. The new block should be able to take care of all the operations that the PIDA block does today and if possible a substitution would be made.

The issue of backward compatibility is important when a change like this is performed. This has however not been a specific consideration in the master thesis but TAC as a company has taken this matter very seriously.

3 The New Controller

The new controller should contain all the basic features of a modern PID controller. The improvements in the already existing blocks should not alter the performance of the actual PID control in any major way.

Other features of a PID controller of today such as gain scheduling should not be included in the algorithm. These actions are to be performed outside of the actual PID block.

3.1 The New Algorithm

The algorithm that was chosen is the positional parallel form. The basic idea behind this choice is that it is well known and easy to understand for everyone who has basic knowledge in mathematics and control theory. The algorithm was specified to contain bumpless transfer, setpoint weighting, anti windup and a possibility to use a reconnected tracking signal. The tracking signal should be used to perform limit actions for the integral part. The algorithm should also include a rate of change limiting mechanism. Internal states should be present in the algorithm, thereby solving several of the problems from the PIDA algorithm.

Below follows the mathematical formulas for the different parts of the algorithm and a block diagram of the algorithm. The notation used in these formulas and the block diagram differs somewhat from what is customary in the literature. The setpoint value is called SP(k), the measured value is called MV(k) and the error is called E(k). Other variables are more self-explanatory and follow the standard notation.

The Proportional Part

The proportional part is implemented with setpoint weighting. The setpoint weighting solves the problem that the earlier algorithm has experienced where a change in the setpoint dramatically affects the output. The original proportional part with setpoint weighting:

$$P(k) = G(b \cdot SP(k) - MV(k)) \qquad \textbf{Eq. 3-1}$$

Default value of the setpoint weighting is b=0 during control using the integral part, i.e. PID and PI control. This means that the proportional part is only influenced by the measured value, see Eq. 3-2.

$$P(k) = -G \cdot MV(k) \qquad \textbf{Eq. 3-2}$$

The proportional part is altered during P- and PD-control, during these types of control the controller uses b=1 and an offset value is added to the control signal. The offset given by Eq. 3-4 is used because it is assumed that the systems are balanced. A control signal between the maximum and minimum value is in this case preferable when the control error is zero.

The proportional part during control without the integral part:

$$P(k) = G(SP(k) - MV(k)) + Offset \qquad \textbf{Eq. 3-3}$$

The offset:

$$Offset = \frac{U_{Max} + U_{Min}}{2} \qquad \textbf{Eq. 3-4}$$

The Integral Part

The integral part uses the forward approximation. Anti windup strategy for the controller incorporates a tracking signal, TSg. The tracking signal is a feedback signal, the same signal as the one used in the PIDA block. The signal makes it possible to force the controller to states given by external logic or other controllers.

A general rule of thumb that is given in the literature for the tracking time of a PID controller is that it should be larger than T_d and smaller than T_i, and that a good choice is the square root of T_i times T_d. This rule gives rather short tracking times as the integral time grows, see Figure 3-1. It is also more rational to have an algorithm that uses a linear rule for the tracking time.

Since the Ziegler-Nichols method for tuning of PID controllers suggest that T_d=T_i/4.The choice of $T_t = \sqrt{T_i T_d} = T_i/2$ was natural, when using a PID controller. For the PI controller the choice was made to use a smaller time, T_t=T_i/4. A smaller tracking time gives faster responses to steps in measured or setpoint values.

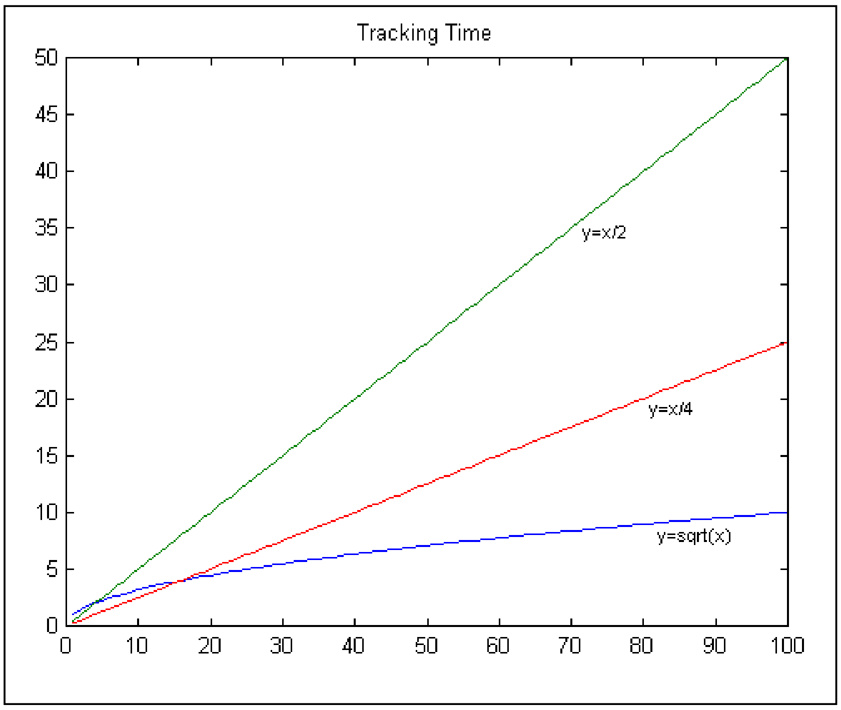


Figure 3-1 Plot showing the different rules discussed for the tracking time.

In conclusion, the tracking time is given by the integral time constant divided by 4 for a controller using only the I-part. For a controller using the D-part as well, it is equal to the more conservative T_i divided by 2. Naturally, the tracking time is set to infinity if the integral part is deactivated.

The tracking time constant is in its nature a delicate engineering problem. The motivation for different choices is that the derivative part alters the behaviour of the controller greatly. The derivative part could by reacting to disturbances cause saturation of the output. If a too small tracking time is used the integral part could be affected incorrectly and may be reset. It is therefore necessary to choose different rules for the tracking time and to use a longer tracking time when the derivative part is activated. The integral part, where Eq. 3-11 gives U_d, C_{Int} and C_{Track} are algorithm constants that will be precalculated:

$$\begin{aligned} I(k+1) &= I(k) + \frac{Gh}{T_i}\left(SP(k) - MV(k)\right) + \frac{h}{T_t}\left(T_{sg}(k) - U_d(k)\right) = \\ &= I(k) + C_{Int}\left(SP(k) - MV(k)\right) + C_{Track}\left(T_{sg}(k) - U_d(k)\right) \end{aligned} \qquad \textbf{Eq. 3-5}$$

The Derivative Part

For the derivative part a backward approximation is used, which is numerically stable. The filtering constant N is chosen by default to eight, the user cannot alter this value. The filter in the derivative part makes the derivative part practically usable in contrast to the earlier implementation. The derivative part in continuous time:

$$D(s) = -G\frac{T_d s}{1 + \frac{T_d}{N}s}MV(s) \qquad \textbf{Eq. 3-6}$$

In discrete time with the backward approximation:

$$D(k) = \frac{T_d}{T_d + Nh}D(k-1) - \frac{GT_d}{h}\left(1 - \frac{T_d}{T_d + Nh}\right)\left(MV(k) - MV(k-1)\right) \qquad \textbf{Eq. 3-7}$$

Which is rewritten as:

$$D(k) = -\frac{GT_d}{h}\left(1 - \frac{T_d}{T_d + Nh}\right)MV(k) + D_{State}(k-1) \qquad \textbf{Eq. 3-8}$$

Where D_{State} is given by:

$$\begin{aligned} D_{State}(k) &= \left(\frac{T_d}{T_d + Nh}\right)D(k) + \frac{GT_d}{h}\left(1 - \frac{T_d}{T_d + Nh}\right)MV(k) = \\ &= \frac{T_d}{T_d + Nh}\left(D_{State}(k-1) - \frac{GT_d}{h}\left(1 - \frac{T_d}{T_d + Nh}\right)MV(k)\right) + \frac{GT_d}{h}\left(1 - \frac{T_d}{T_d + Nh}\right)MV(k) = \\ &= \frac{T_d}{T_d + Nh}D_{State}(k-1) + \frac{GT_d}{h}\left(1 - \frac{T_d}{T_d + Nh}\right)\left(1 - \frac{T_d}{T_d + Nh}\right)MV(k) \end{aligned} \qquad \textbf{Eq. 3-9}$$

The entire rewriting of these equations aims on a more efficient use of processor capacity and the result is that D(k) can be calculated immediately when the MV(k) signal is available. This means that a new control signal value is available with a minimum of calculation delay.

The equations Eq. 3-7 and Eq. 3-8 are rewritten with algorithm constants as:

$$\begin{cases} D(k) = -C_{Der2}MV(k) + D_{State}(k-1) \\ D_{State}(k) = C_{Der1}D_{State}(k-1) + C_{Der3}MV(k) \end{cases} \qquad \textbf{Eq. 3-10}$$

Computational Aspects

The parameters used in the algorithm, the algorithm constants, are precalculated from G, T_i, T_d, and can be stored to achieve better performance and more efficient use of the processor. The constants are given in Eq. 3-5 and Eq. 3-10 and they are: C_{Int}, C_{Track}, C_{Der1}, C_{Der2} and C_{Der3}.

Block diagram

The figure below, Figure 3-2, shows the principal structure of the algorithm. The limitations are performed on $U_d(k)$ before it is made available as an output value.

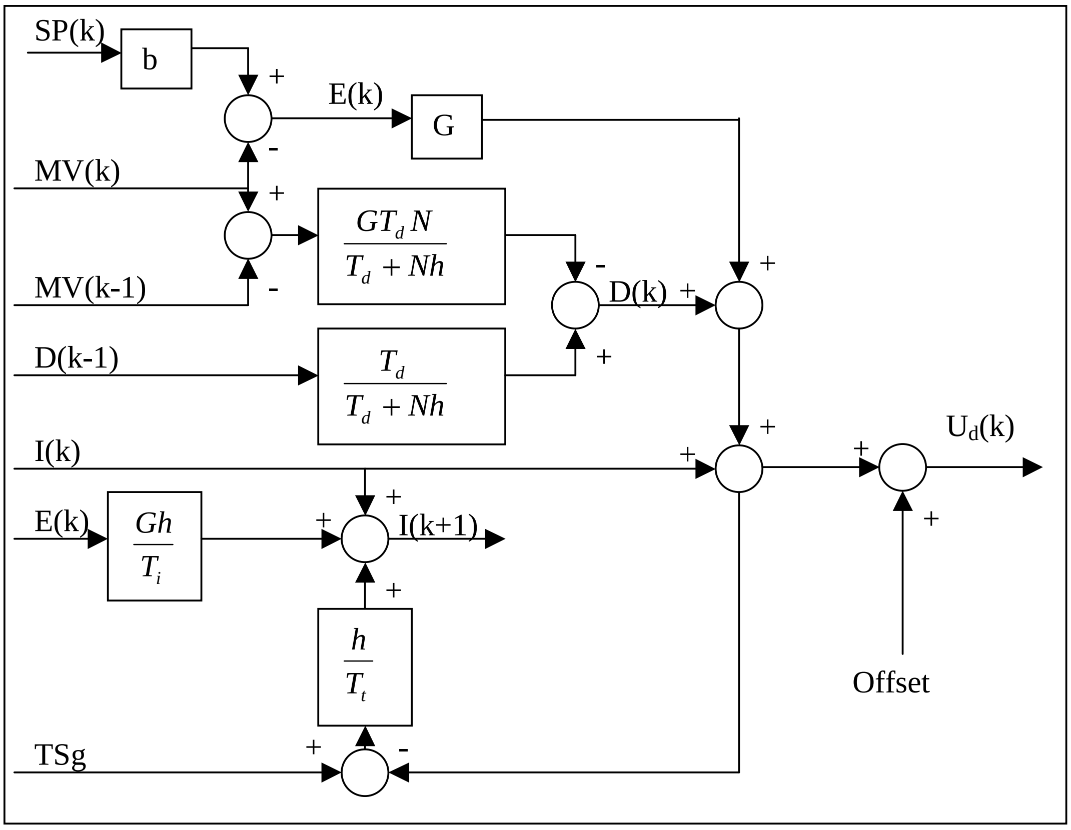


Figure 3-2 Block diagram of the new algorithm in discrete time.

Limitations and Bumpless Transfer

The unlimited control signal is given by:

$$U_d(k) = P(k) + I(k) + D(k) \qquad \textbf{Eq. 3-11}$$

Naturally, the control signal has to be limited. The maximum and minimum limitation is performed between two values that are given by the user, U_{Max} and U_{Min}. These values are the same ones that are used for the offset calculation in Eq. 3-4.

It is also possible to limit the rate of change of the control signal. This is done by a user-defined value of the stroketime for the valve or the object that is controlled. The limitation is then performed so that the rate of change never exceeds the limit that is calculated according to Eq. 3-12

$$C_{MaxChange} = \frac{(U_{Max} - U_{Min})h}{StrokeTime} \qquad \textbf{Eq. 3-12}$$

To achieve bumpless transfer, a check is performed that the parameters G, T_i and T_d have not been changed. If they have been changed a new set of algorithm constants are calculated.

By forcing the derivative part to zero and using the integral part as a memory for the present control signal value, bumpless transfer is achieved. The D_{State} is assigned as:

$$D_{State}(k) = \frac{GT_d}{h}\left(1 - \frac{T_d}{T_d + Nh}\right)MV(k) \qquad \textbf{Eq. 3-13}$$

This value will force D(k) to zero according to Eq. 3-8.

The integral part during bumpless transfer is given below, where U is the limited control signal output:

$$I(k) = U(k-1) - P(k) \qquad \textbf{Eq. 3-14}$$

These values of $D_{State}(k)$ and I(k) will ensure that the output is the same before and after the changes in G, T_i or T_d, when they are put into Eq. 3-11.

The transfer problem between automatic and manual control is not considered in this structure neither is it considered at TAC.

A deadzone is also incorporated that operates from the input, MV(k). The effect of the deadzone is that it freezes the control signal to the latest value. When the input value, MV(k), leaves the deadzone a bumpless transfer is needed and performed. Bumpless transfer is needed since the deadzone has introduced a non-linearity and the controller might have lost valuable information of its states.

Controller parameters should be tuned in the manner that is preferred by the different operators and no limitations have been imposed concerning this issue.

3.2 Implementation

The programming language that is used by TAC for embedded systems is C and the algorithm was implemented using this language. Implementation of the new block in TAC Menta® was made with great assistance of programmers from TAC´s development department, Engineering Tools, in Malmö. The implementation did not give rise to any problems above normal programming errors.

A weighting between memory usage and usage of processor capacity has been made and since the present controllers have a very limited amount of memory the emphasis has been placed on reducing this. However, TAC will in the future use a controller with more memory capacity and the algorithm has therefore been designed in a manner that easily can be altered to focus on reduction of processor usage, on the expense of memory usage.

4 Simulation and Testing

Simulation work has been focused on two issues:

- Differences between the new algorithm and the one used earlier in the PIDA block.
- Making sure that the new controller actually performs as well as expected.

This has meant that the controller has been run in a great number of cases, of which only a few key simulations are presented below.

Testing were performed to ensure that the controller did not encounter any problems when used in an actual application and an environment outside lab and computer:

4.1 Simulation Models

The modelling language that has been used is Simulink^{TM}. Three models have been built up from three different specific cases of control at TAC. Printouts from the simulation models are added in Appendix A-C

The models are:

- A zone controller that primarily is used to control a room. Zone Regulator-ZR.
- A controller that is used for control of a cooling coil, a heat exchanger and a heating coil, in sequence. Cooling, Heating, Heat Exchanger-CHHE.
- Control of a domestic hot water process. Domestic Hot Water-DHW.

To be able to use the models a set of process models had to be built. Printouts from the process models are added in Appendix D-G. The process models that have been used are:

- Model of a heating coil.
- Model of a cooling coil.
- Model of a heat exchanger.
- Model of a room.

The process models are first order transfer functions, with some expressions added to the models. The process models are kept quite simple because it is only of interest to make sure that the algorithm is working as it is supposed to do, not to study complicated effects in detail.

4.2 Simulation Results

The models have been linked up in such a manner that the result should mimic the reality, and have then been put through situations that can come up under normal operation. Situations such as load disturbances, measurements noise and setpoint step responses have been studied.

The following results are taken from the model CHHE, since this model demands the most from the controller. CHHE also includes all of the process models. The controllers have been run as PI-controllers in the experiments presented below. The use of the derivative part when comparing the two controllers gives completely irrelevant data because of the poor implementation in the PIDA algorithm.

The effects can be studied in Figure 4-1 and Figure 4-2, where the first contains the results from the new algorithm and the second contains the results from the old algorithm.

At time equal 4000 a setpoint change is introduced in the system, the setpoint is given an additional value of 5. The effects of this change are not dramatic and the controller handles the change satisfactorily.

A load disturbance is introduced in the system at time equal 6000, the controller adapts to this disturbance without any major problems.

At time equal 8000 a measurement disturbance is introduced in the system, it appears as a very small step of about 0.05% in the controller output.

The tests that have been performed on the controller depend highly on the process and its parameters. However, during simulations the controller has shown excellent handling of all the different process models and a wide variety of parameters.

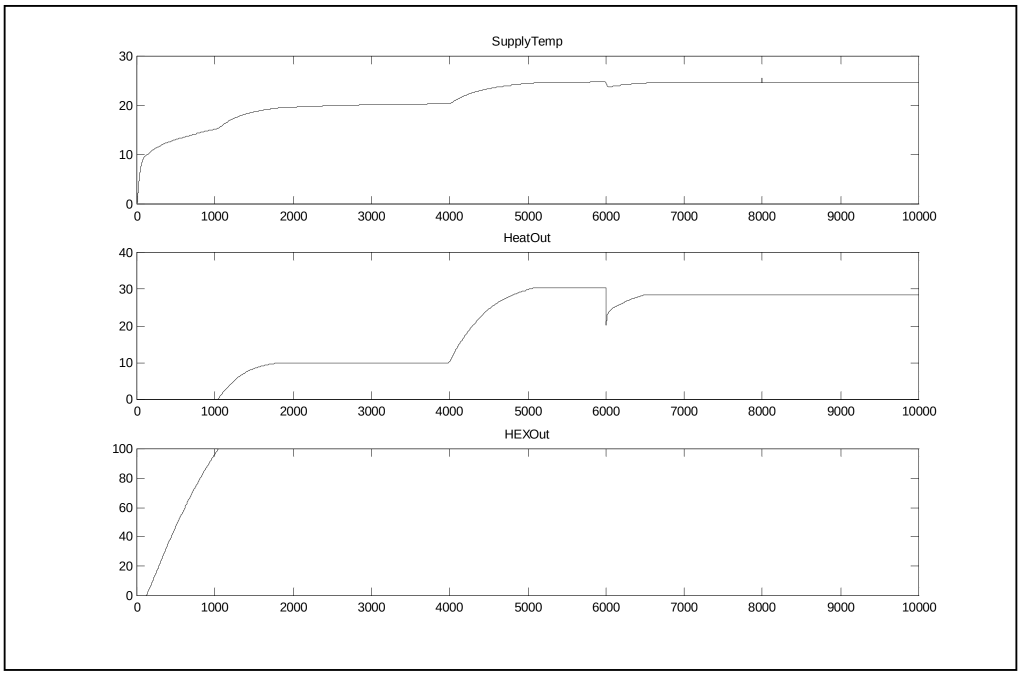


Figure 4-1 Plots showing load disturbance, setpoint handling and measurement noise for model CHHE, using the new algorithm.

Figure 4-2 shows the results for the same experiments as has been presented earlier. A simulated model of the earlier algorithm at TAC has been used this time. The results from the new and improved algorithm are in comparison with the old implementation dramatically improved. The setpoint change gives rise to a jump in the control signal that is not acceptable, the overshoot is too big and too fast. The load disturbance at time equal 6000 is handled just about the same as with new algorithm. The measurement noise affects the system terribly as can be seen by the noise introduced at time 8000.

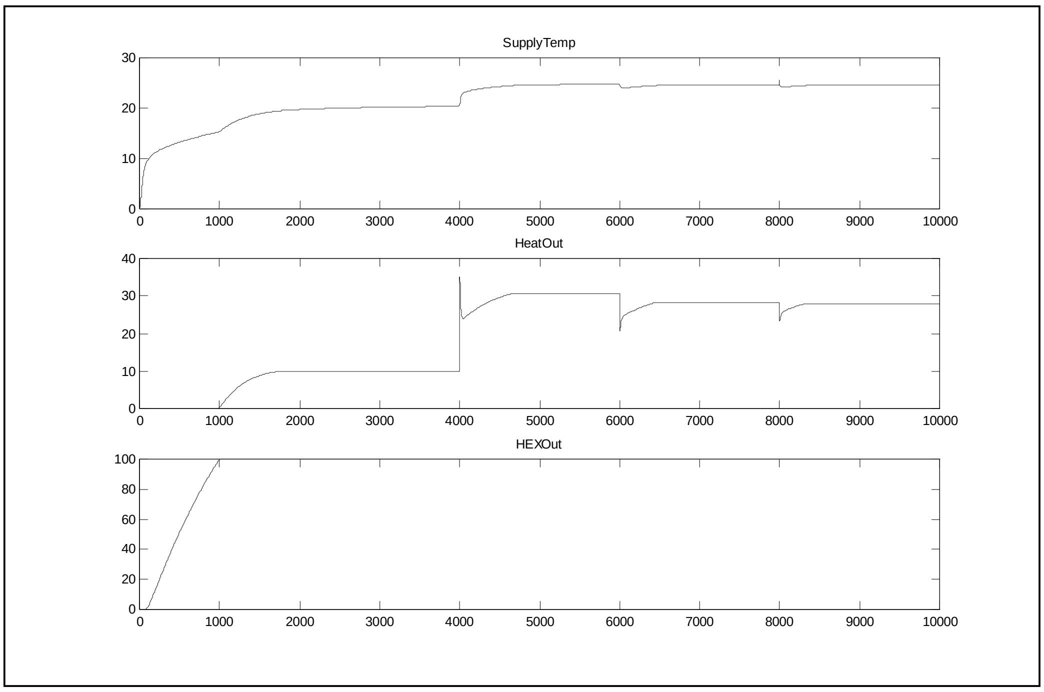


Figure 4-2 Plots showing load disturbance, setpoint handling and measurement noise for model CHHE, using the old algorithm.

4.3 Test Results

The controller has been run in the TAC building during a period of a little more than a week, it has there been controlling the temperature in the ventilation system. (The system is about the same as the model CHHE.) Since the outside temperature during this period has been approximately between -10°C and 10°C the controller has been controlling a heat exchanger and a heating coil. During the test period the controller has performed satisfactory.

To make any comparisons between the two algorithms based on this test is difficult. The system where the test has been carried out is not considered troublesome at TAC. The system is very well suited for a PI controller and during normal operating conditions, PIDA has had no problems with this system in the past.

However, one conclusion that can be made is that the new algorithm is correctly implemented since it has performed as expected.

To make accurate comparisons between the algorithms mean that they will have to be subjected to more advanced tests. There is however no need for more evidence of the behavior of the two controllers in this thesis.

An interesting test would have been to put the new controller in a system where TAC has experienced problems in the past. This test has not been performed during this master thesis, because of different circumstances.

4.4 Conclusions from Tests and Simulations

The greatest concern about the difference between the new and old algorithm has been when the controller saturates and how much time it spends in saturation. Another concern was how the new controller reacts to different logical and arithmetic operations to the reconnected tracking signal.

Simulations showed that the controller does behave differently in comparison with the previous algorithm, but the differences were very small, during normal operating conditions. These differences depend primarily on the fact that the new algorithm has a completely new implementation of how the tracking signal is incorporated. The problems experienced earlier with the PIDA block have been removed in the new algorithm.

Logical and arithmetic operations to the reconnected signal are part of the same question as the saturation, because the tracking time affects both issues. Therefore, the issues were treated simultaneously and a weighting has been made on how long the tracking time should be. The starting point was to make it as short as possible, to be able to get as fast answers from the controller as possible. The other consideration was if the controller could end up in forbidden states if the tracking time were too short, primarily because of the logical operations. Different strategies were simulated and the ones presented in Section 3.1 were accepted.

To conclude, the simulations and tests have shown that the new algorithm is more than adequate to use for the different cases of control that it can be used for at TAC.

5 Future Application Programming Tool

5.1 Introduction

As a part of this master thesis, an investigation of a future application programming tool has been carried out. Since the development of the existing tool, TAC Menta® started out some years ago, it is now time for TAC to take a look into what has happened during the years in this area and perhaps to develop their existing tools further.

The investigation has focused on the standard that is used in association with automation, IEC 61131, as well as thoughts and ideas within TAC about the future. The investigation includes possible improvements to Menta that could be an alternative to a completely new tool. A short comparative investigation between five commercial application programming tools is also included.

The present tool, Menta, was from the beginning developed in Spain by a smaller company that TAC acquired during the 90s. The tool has then been developed further in Malmö for several years and has been used as a practical programming tool “in the field” since 1996.

Discussions with several employees at TAC have been carried out to obtain the information and ideas from within the company. These employees are primarily people that have a background in industrial automation or have worked for a longer period with Menta. The ideas gathered during this part were very similar to the standard IEC 61131.

If a suitable tool exists already, which follows a standard, this should not be considered as a disadvantage. However, TAC has no need for all the features in a general PLC application programming tool used in the automation industry and specified in IEC 61131. The tool TAC is looking for should cover their specific needs and fit in with the other tools in TAC´s product family.

5.2 TAC Menta®

Introduction

TAC Menta® is a programming tool that uses a graphical programming language, so called Function Block language, which makes it possible for the engineers to develop specific applications. In the tool, several predefined blocks are used to calculate many different things as enthalpy, time-schedules, PID control and so on. These blocks as well as the tool in general are implemented in C++.

TAC Menta® is a 32-bit program fully adapted to Microsoft® Windows. The tool also contains tools for downloading of an application, tools for creating databases for network configuration, functions to alter values on-line, a simulation tool for debugging of applications and so on. Since this investigation focuses only on the programming of applications, using the graphical programming language, the other tools incorporated in Menta are not further analysed.

TAC Menta® has been experienced by employees at TAC as a stable and easy to use tool. The best feature of Menta is that it is easy to get started in Menta. The tool lacks many of the features found in a general PLC programming tool and it is therefore easier in Menta. That Menta does not contain some features is not something that immediately should be considered as a disadvantage, it is merely a specialisation for a specific area of automation.

Since Menta is a graphical programming tool, it has many similarities with the graphical programming language, FBD-Function Block Diagram, described in IEC 61131-3.

5.3 The Future Tool

Below is a short description of some concepts that could be interesting for TAC. The investigation of existing application programming tools is also found below.

Sequential Function Chart

SFC, Sequential Function Chart, is a graphical representation of the application program control flow and state transitions. SFC was developed for structuring of complex structures in applications. In a small application, it may not be necessary to use SFC. However, when the complexity of the applications increase, it is necessary to organize the applications in a manner that is understandable and readable.

The basic components of SFC are initial step, states and transitions. The initial step is the first step in the sequence where e.g. initialisations are performed. A transition is the condition or conditions that change state of the application. A transition between states is not allowed to occur unless the state prior to the transition is active. A state is where the application performs some kind of action, for instance calculates a new control signal or sets an alarm.

Three types of combinations exist in the control flow:

- Simple sequences.
- Branching - alternative sequences. One of several sequences is performed.
- Splitting - simultaneous sequences. Two or more sequences are performed simultaneously.

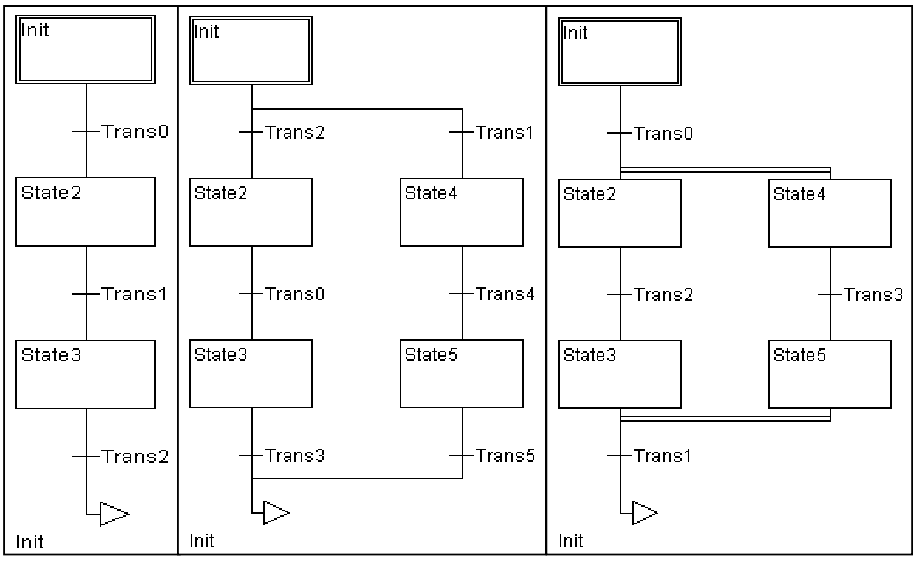


Figure 5-1 Figure showing the three types of combinations in SFC. From left to right: Simple, Branching and Splitting. The sequences have been implemented with CoDeSys® 2.1 by Smart Software Solutions.

A common situation in TAC's applications is that a variable is set or reset, for instance an alarm. Other examples could be during start-up or shutdown sequences when different valves and control signals have to be set or calculated in a specific order. These examples could be both event and time-driven, certain events or time specifications activate specific actions or sequences. The events are transitions and the setting of alarms or calculations of control signals are states.

The quote below, from R. W. Lewis, serves as a summation of why SFC is needed in an application programming tool at TAC.

SFC: "A graphical language for depicting sequential behaviour of a control system. It is used for defining control sequences that are time- and event-driven."

This is a very good description of how SFC could be used by TAC

SFC Examples

An interesting example is a start-up sequence, a similar example from TAC Menta® is not available since the start-up sequences are distributed in too many parts of the applications to fit in this report.

The example below describes a small start-up sequence of a heating system with a heating battery and a heat exchanger. First in this sequence, the temperature is increased in the heating battery by opening the valve in the heating battery. The amount of heat, i.e. the percentage the valve is opened, depends on for instance the outside temperature.

When the heat in the battery has reached a specified temperature a transition occurs and the supply fan is started. When the energy-boost has had a certain time to take effect, the next transition to the heat exchanger is made. The system will start the heat exchanger at maximum in the final state of this sequence.

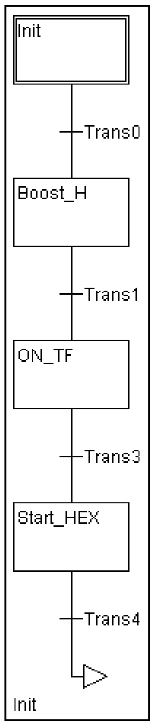


Figure 5-2 Figure showing an SFC of a start-up sequence for a heating system.

The advantage with SFC in this case, compared to TAC Menta®, is the compact manner it is described in. In comparison with TAC Menta® it is always obvious in which state the system is, thereby making debugging easy.

The short sequence above could be part of a sequence for controlling the entire system. In this case the sequence would return to its father sequence when it has performed the desired tasks. In the example presented below in Figure 5-3, the example from above is inserted in the Start_Up state.

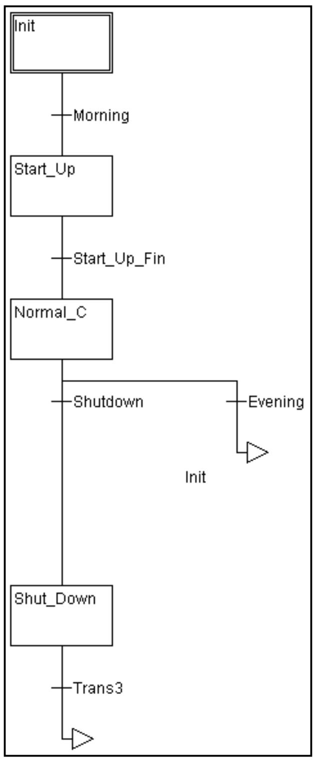
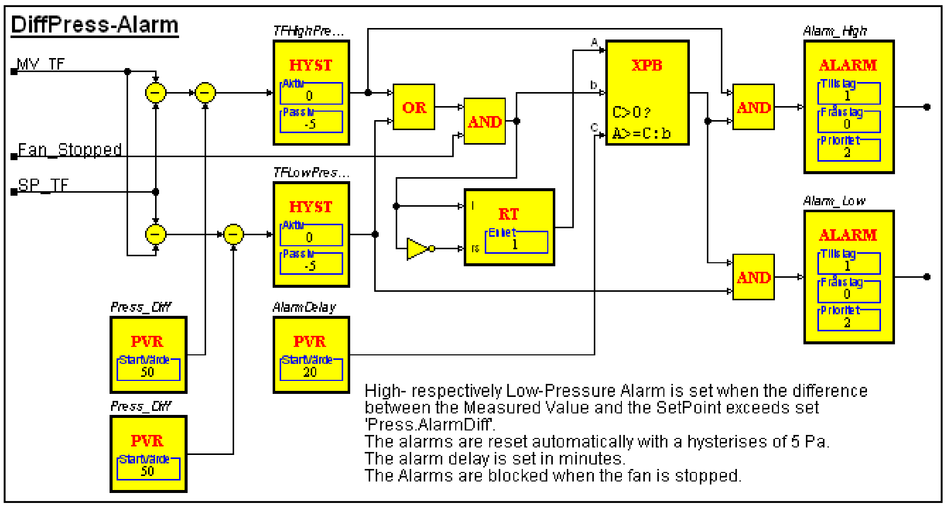


Figure 5-3 Figure showing an SFC for a state sequence of an entire heating system.

If the maximum heat recovered is not enough for the system, the heating battery will have to be activated again. This action is however done in the state of Normal_C, where the system will perform normal operations. Several additions to the example will have to be done for this sequence to be practically usable, like for instance alarm handling.

Figure 5-4 shows a small example from TAC Menta®, it is not too hard to read and understand. However, when smaller applications are connected the complexity grows and the applications become very difficult to read and understand.

The example is an alarm handler for pressures, if the pressure is not within acceptable limits the high- or low-alarm is triggered.



High- respectively Low-Pressure Alarm is set when the difference between the Measured Value and the SetPoint exceeds set 'Press.AlarmDiff'.
The alarms are reset automatically with a hysterises of 5 Pa.
The alarm delay is set in minutes.
The Alarms are blocked when the fan is stopped.

Figure 5-4 Figure showing the example implemented in Menta. The block RT counts the time it has been active, the Run Time. The other blocks are self-explanatory.

Figure 5-5 to Figure 5-9 show how this could be conducted with SFC. Since TAC has used graphical programming for some time now it would be nice for the employees to be able to continue using an environment that they are used to. All the programming of the transitions and the steps should therefore, in a possible future, be programmed using a graphical programming language. Figure 5-6 to Figure 5-9 below shows the transitions and steps programmed using FBD.

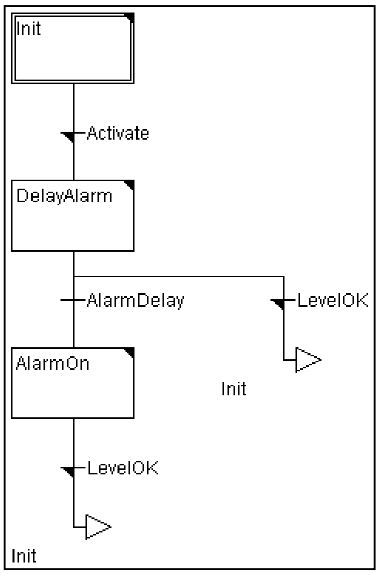


Figure 5-5 Figure showing the main SFC of the example Function block, implemented in CoDeSys® from Smart Software Solutions.

In the example the sequence will remain in its initial condition until the transition Activate is fulfilled, see Figure 5-6 below. The transition determines whether the measured value is within acceptable limits or not, a check is also performed that the fan is not stopped by the Fan_Stopped signal.

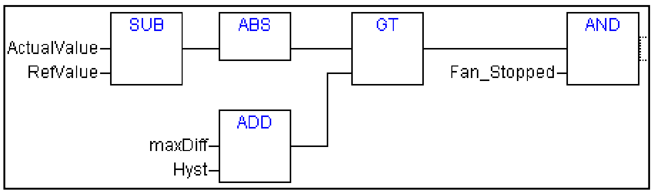


Figure 5-6 Figure showing the transition Activate.

In the state DelayAlarm, Figure 5-7, a timer delays the setting of the boolean variable AlarmDelay , this variable is used in the next transition as a condition for its activation. The effect will be that the transition is delayed for a specific amount of time the AlarmDelayTime.

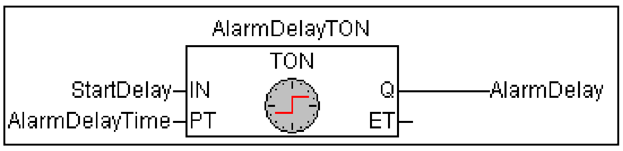


Figure 5-7 Figure showing the step action DelayAlarm.

According to IEC 61131-3 transitions are not allowed to contain function blocks that have internal state variables, like a counter that is present in the TON-function block. However, in several tools deviations are made from this.

If the system recovers during the delay time, see Figure 5-8, the sequence will automatically return to its first state and await a new Activate signal.

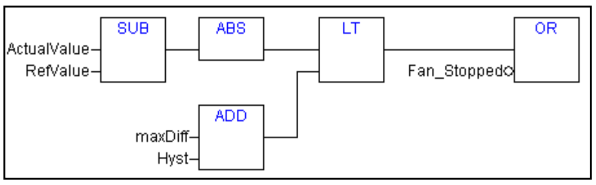


Figure 5-8 Figure showing the transition LevelOK

If the system does not recover the alarm is set by the state AlarmOn, Figure 5-9.

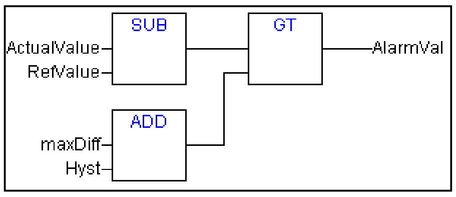


Figure 5-9 Figure showing the step action AlarmON.

SFC in TAC Menta®

An approach to SFC could be to incorporate it in the present tool Menta by developing existing features further. Possibly by taking an already existing block, the HFB (Hierarchal Function Block) and develop an enable-disable signal to this blocks. The enable-disable signal should prevent evaluation of the blocks within this block i.e. when the state is not active it is not allowed to set any values, neither internally within the SFC block, nor externally like for instance an alarm.

Within these new SFC blocks, the programming would be performed using the same set of function blocks as today. The transitions would return an output of boolean type. The states should perform actions like PID control and alarm handling.

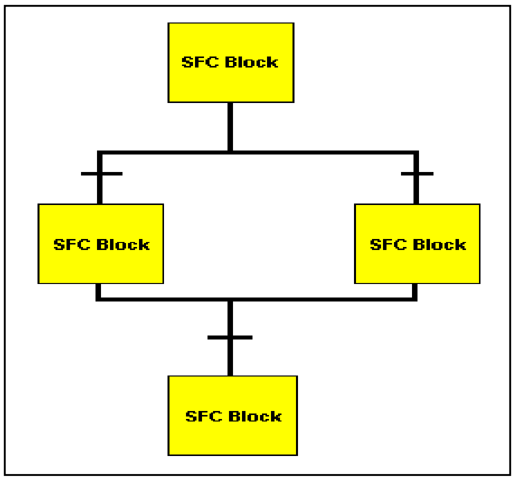


Figure 5-10 Figure showing a possible view of a new SFC-block in Menta.

The boolean variable that is returned from the transitions should be used to disable the preceding block and to enable the following block of the current transition. The enable-disable signals will have to be sent between the blocks at the moment of transition. The signals are necessary, to make certain which state that is active, i.e. it is significant that the block before the transition is disabled and the block after is enabled. Furthermore, the transition after the activated state will have to be activated i.e. allowed calculating its transition condition.

The signals that go between the blocks should be transmitted via the same line, preferably the line connecting the SFC-blocks, no matter of the type or the direction the signals are going in. This transmission of the signals is not vital for the function of the SFC blocks. The signals could for instance be sent via send and receive blocks inside the SFC blocks. It is however vital that the signals are sent via the same line to increase readability, and if any advantages are to be made in structure from this concept.

The enable-disable signal together with a structured signal transmission between the blocks could be an alternative to the use of the standardized SFC environment.

These changes will mean that a lot of development will have to be made in Menta because of the limitations today of amongst other things the lack of enable disable signals in the blocks.

Further features like jumps, branching and splitting should also be available when implementing this into Menta. These features are a question of which SFC block that should be activated.

Sophisticated and Specialized Function Blocks

A new tool could contain the possibility that highly specialized function blocks written in C, or another text based programming language, easily could be implemented and downloaded to a target system. This is a desired feature from employees at TAC’s R&D department. In most general tools for automation, it is possible to download C-code as a highly specialised function block.

When a new block is added to Menta today, for example a new PID controller, extensive programming efforts have to be made not only in Menta but also in the Xenta systems.

Structured Types

Menta only accepts simple types like REAL, INT and BOOL as signals between blocks unlike IEC 61131-3, which accepts structured types.

This deficit affects the structure of the program by making it hard to read and get a grip on because of the dispersion of signals in the applications.

Consider for instance a pump. The pump needs several signals, only a few presented below:

- An input value - for required speed.
- A status signal - determines whether the pump is operational or not.
- An output value - for actual speed.
- Run time- amount of active time.

If all of these signals were put in a pump data structure, instead of being transmitted one by one, the amount of transmission lines in the Menta applications would be decreased. This concept will also help structure the entire application, since all information concerning the pump is collected in one structure.

Processes

A process, or task, is an application or part of an application that normally is running on a different sample time and with a different priority than other parts of the application. Different priorities mean that the process is more or less important than other processes, the priority then decides the amount of access the process has to the central processing unit. It is not necessary that the processes run on different sample times and priorities.

In many cases it would be useful to give certain parts of an application smaller sample times, for instance to make sure that every change in the inputs are observed. This should be done without having to increase the amount of calculations and I/O operations performed in other parts of the application.

The concept of priorities could be very useful in many cases, especially in cooperation with different sample times. The concepts could be used for instance when controlling processes which have different dynamics.

Both of these concepts are described in IEC 61131-3 and are available in programming tools used in industrial automation.

Instantiation and Inheritance

The concepts of instantiation and inheritance are very interesting in a programming tool for automation. An instance is a copy of another programming block that is still linked with the original block. A change in the original block affects the copy, the instance, i.e. the instance inherits the changes and the new functionality. This is very useful especially when code is reused in several places in the same application. The instance will in this case only be a represented by one single block that is linked to the original structure.

These concepts should be central in a new programming tool and are available in tools for industrial automation. Today in TAC Menta® the use of macro blocks is the closest to this concept, more information about macro blocks is available in Section 2-1. A large drawback with the macro blocks is that the concept of instantiation and inheritance are lacking. When a macro block is used today, it is copied into the application and the concept of hierarchy is lost.

The concept of macro blocks and where the macro blocks are stored i.e. the macro block library is more developed in a general automation programming tool than in Menta. In a general tool, it is easier to use what is stored and above all this makes sure that it is used. The macro block library is applications or parts of applications that is saved and then loaded into the present application, the macro blocks are loaded using Microsoft's file system. This solution is crude and compared to what is found in a general automation programming tool not user friendly.

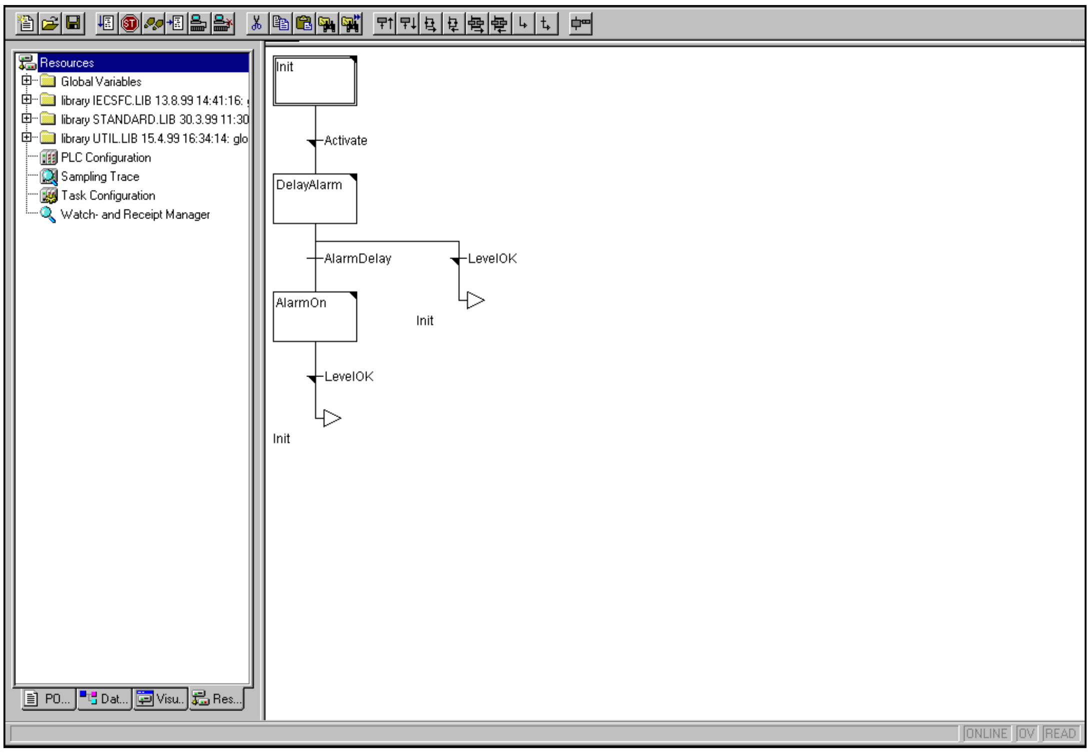


Figure 5-11. The figure shows a screenshot from a general programming tool, where several libraries are used. The libraries are located in folders in a separate window next to the programming area for easy access. Screenshot from CoDeSys® v 2.1by smart Software Solutions.

Other Possible Additions

In IEC 61131-3, hierarchy is a major concept and is built into the basis of the standard. Therefore all tools based on it, will have a highly functional hierarchy. The hierarchy supports and encourages reuse of code, and is closely linked to other concepts like instantiation.

The concept of hierarchy is present in TAC Menta®, by the HFB, Hierarchal Function blocks. However, these blocks are rarely used in practice since the implementation has not satisfied the users.

In Menta, there is very little support if two or more programmers need to work on the same application. There is no natural manner to divide the applications for these purposes, concerning both variables and functionality. Since a tool based on IEC 61131 has a more developed structure, it is possible to make a natural division of applications between several developers.

General IEC 61131 Programming Tools

To take a new approach to the problem may mean that a completely new programming tool could be used. A new tool will mean that a completely new strategy and philosophy of programming will have to be used at TAC. To make such a profound change could be unfortunate if the advantages were not obvious, since programmers, as well as people in general, tend to stick to what they are used to. It would also mean that many hours of education would have to be invested in every employee.

A completely new tool could be acquired from another company that has already developed it to near perfection, with some alterations to make it perfect for TAC. The list of suppliers could be made long, since it exists several suppliers of general IEC 61131 programming tools.

The tools for this purpose, found in the industry today, share several common features and are built up around the same ideas. To make a decision on a specific tool, several tests would have to be performed and several specifications concerning the tool would have to be conducted. This is outside the scope of this thesis, but a smaller investigation of five different tools has been performed.

The tools investigated are:

- ISaGRAF® Pro and ISaGRAF® 3.3 by CJ International.
- softCONTROL® 3.0 by Softing GmbH.
- CoDeSys® 2.1 by Smart Software Solutions GmbH.
- MULTIPROG® 2.1 by Klöpper und Wiege Software GmbH.
- OpenPCS® 4.0 by infoteam Software GmbH

The investigation was performed by implementing a small example found in TAC Menta®, the same example as found above in Figure 5-4, in the different tools. During this implementation, the programming environment and the different features in the tools were reviewed, as well as if some features were missing in any tool. During this investigation the tool concluded to be the most suitable for TAC was CoDeSys® by Smart Software Solutions.

This conclusion is subjective and a much more extensive investigation would as mentioned before have to be conducted by TAC, before any decision concerning a specific tool can be made. Since the basis of all of the tools is IEC 61131, the tools are built up around the same concepts. The tools also have a similar GUI and it is up to the specific user which tool that suits best.

5.4 Conclusions and Recommendations

Several, if not all, of the concepts above should be very interesting in a new tool for application programming at TAC.

All of the above-discussed concepts could of course be implemented in the existing tool TAC Menta®, problems might however arise if concepts from IEC 61131 are applied to the existing tool since it does not follow the standard in other aspects.

The conclusion of the investigation is that a new tool, or a fundamental revision of the existing tool, is needed. The concepts mentioned above, for instance SFC, are widely used in the automation industry of today. Even if TAC is not a company that is involved in any industrial automation systems, the advantages of concepts from this field of automation are clear and obvious even for the inexperienced user.

An estimate of the amount of work that has to be put in to alter Menta to make it fit the new concepts as well as to correct existing faults will have to be conducted at TAC. The amount of work this investigation finds that is needed to change Menta should be weighed against the cost of purchasing, or other ways of developing, a new tool. Several other costs will be associated with changing tool for instance the education cost for the new tool and adaptation of other tools in TAC´s product family.

The conclusion of this master thesis is that the best course of action is to purchase a new tool. Primarily since the problems and the new concepts in TAC Menta® will mean that a lot of work has to be performed if the old tool is to be used as a base. TAC has little to gain from producing a new tool that includes several features from IEC 61131, when considering that several companies specialises in producing these tools.

The recommendation based on the conclusions above is that TAC acquires an adapted tool from a supplier of a general IEC 61131 programming tool. Since the tools based on IEC 61131 are widespread and many suppliers of these tools exist, a tool should not be too expensive or difficult to acquire. An alternative tool is for instance CoDeSys® from smart Software Solutions. The adaptation of the tool should be e.g. to remove the programming languages that TAC has no need for and to make the tool work well together with TAC´s other tools.

6 Conclusions

The master thesis work that has been carried out at TAC and has been reported above focused not only on the implementation of a PID algorithm, but also on the process leading up to this implementation.

The process leading up to an implementation is an important topic for every engineer, but not easy to teach to students as it is something very specific for each company and each industry. It has been interesting for the author as a fresh engineer to experience some of these issues treated above.

The problems with the old PID controllers have been thoroughly investigated and described in the thesis. Since the old controllers were decided to remain in use at TAC the description of the problems could be useful for TAC’s application programmers to make sure they do not use the old controllers in a situation that could be troublesome.

The PID algorithm that was implemented has been proven to fulfil all the requirements TAC has put on it. This conclusion can be drawn from the simulations and tests performed in the thesis. In comparison with the previously used algorithm for PID control, the algorithm derived above has eliminated the problems experienced earlier at TAC. The new implementation has been met with approval from several employees at TAC who are working daily with control issues.

It could always be discussed if a more advanced structure should be used, for instance on the tracking time and the bumpless transfer. However, a more advanced structure would mean that more calculations would have to be made and more memory would have to be used, which in turn finally will mean that more money will have to be spent. It could also mean that more responsibility would be put on the application programmer, which should be avoided according to TAC’s philosophy.

During the work with the new concepts that could be used in a future application programming tool at TAC several practical issues have been discussed with employees at TAC, the outcome of these discussions have been that several questions have been raised at TAC. These questions will have to be solved within the company and this thesis has been an injection into an ongoing debate and an upcoming prestudy at the company.

7 Bibliography

Karl Johan Åström and **Tore Hägglund**: *PID Controllers: Theory, Design, and Tuning.* Instrument Society of America, Research Triangle Park, NC, 2nd edition, 1995.

Karl Johan Åström and **Björn Wittenmark**: *Computer Controlled System-Theory and Design.* Prentice-Hall, Englewood Cliffs, New Jersey, 2nd edition, 1990.

Gustaf Olsson and **Gianguido Piani**: *Computer Systems for Automation and Control.* Prentice Hall International, London, U.K., 2nd edition, 1998

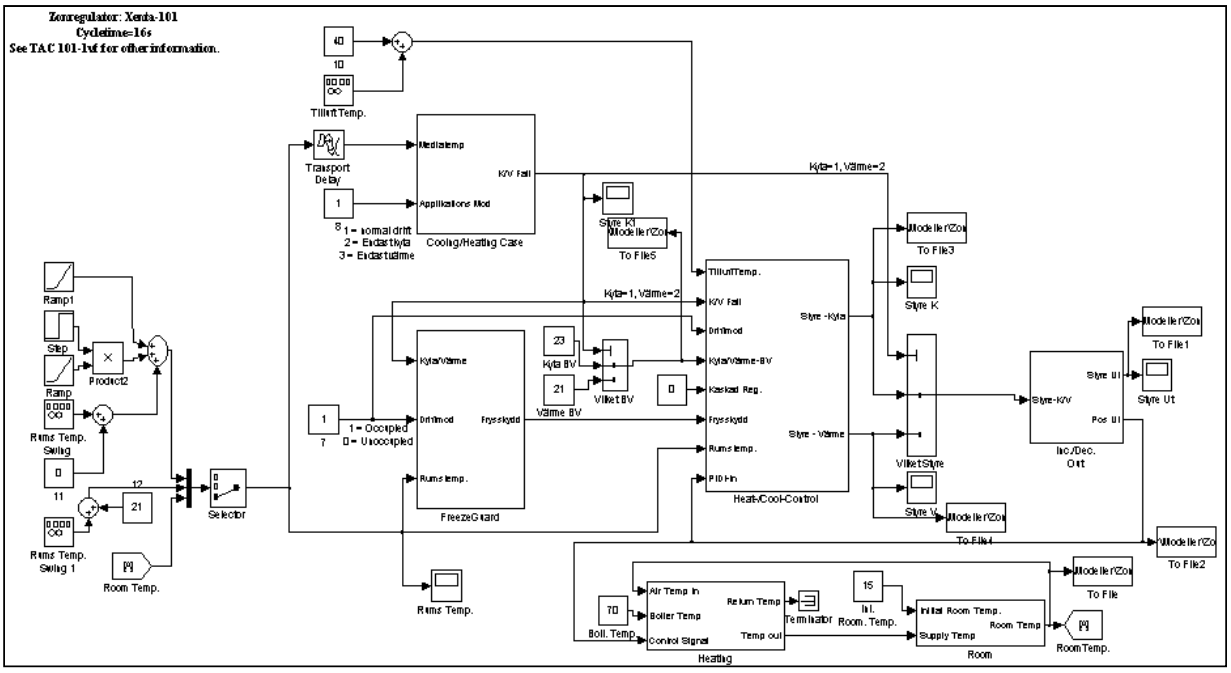
IEC: *IEC 61131 programmable controllers - part 3: Programming Languages*, Technical Report, International Electrotechnical Commission, 1993

R. W. Lewis: *Programming industrial control systems using IEC 1131-3*, The Institution of Electrical Engineers, London, 1995

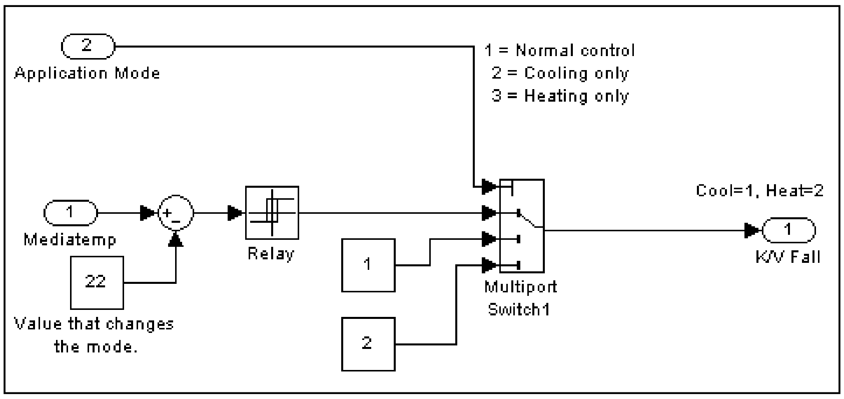
8 Appendix

8.1 Appendix A, The Model ZR

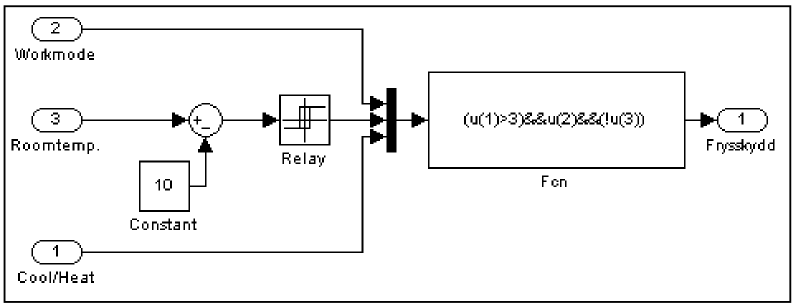
The main system of the model ZR.



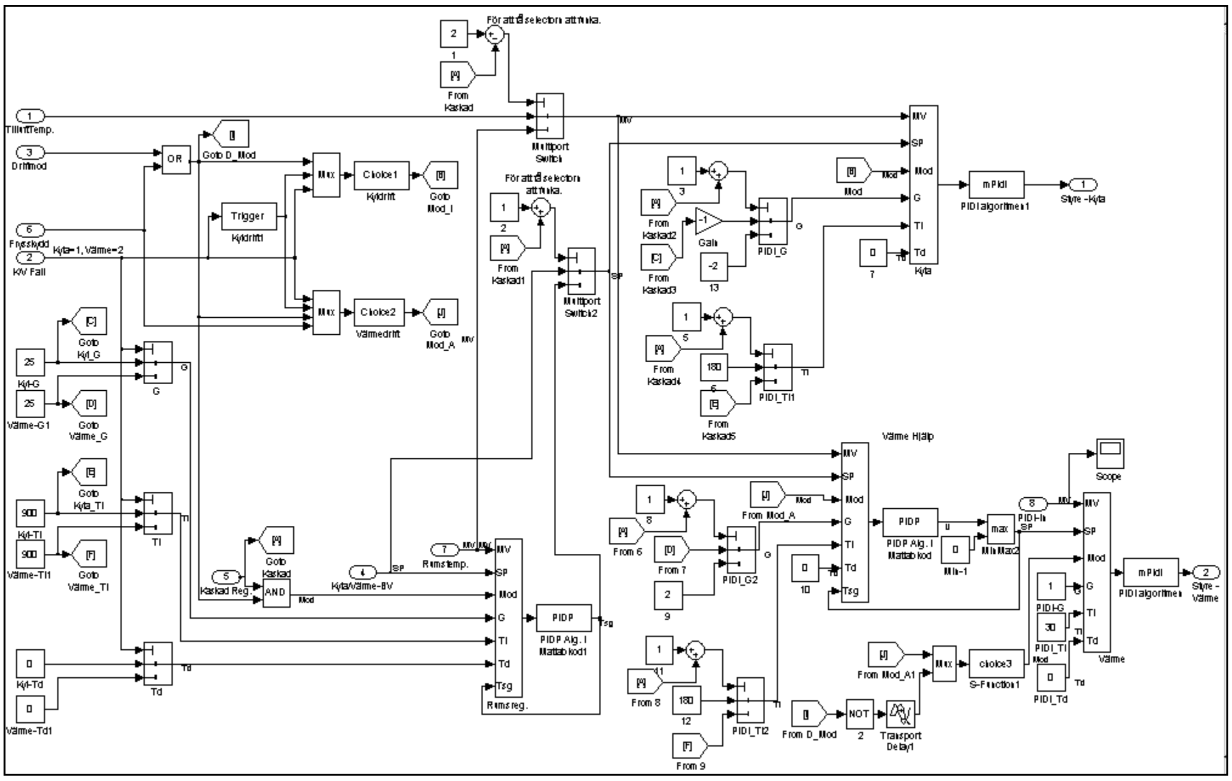
The subsystem Cooling/Heating Case.



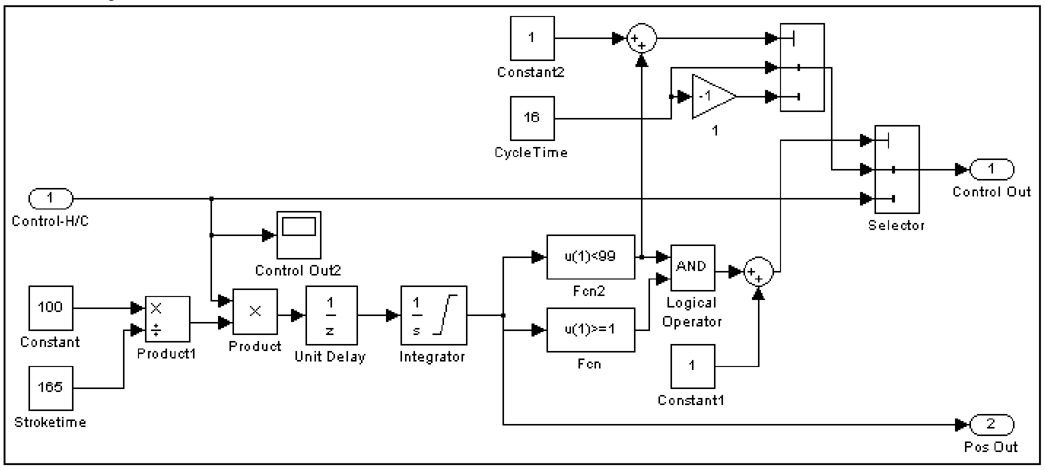
The subsystem FreezeGuard.



The subsystem Heat /Cool Control.

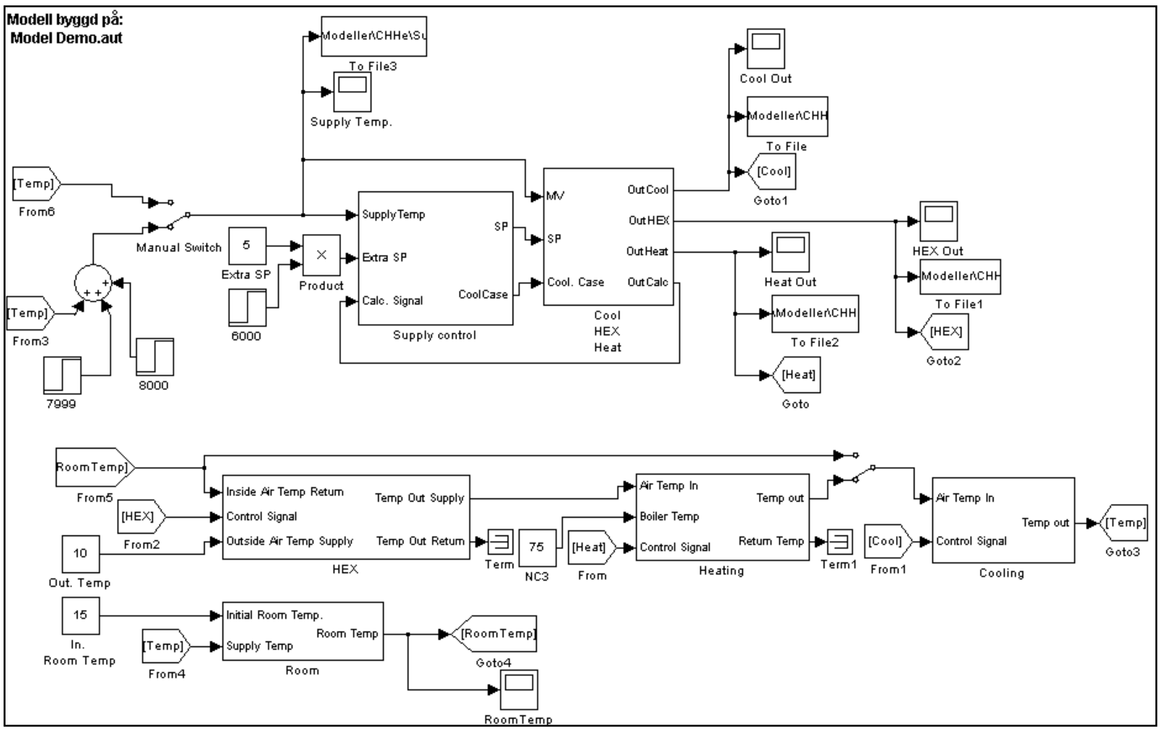


The subsystem Inc/Dec Out

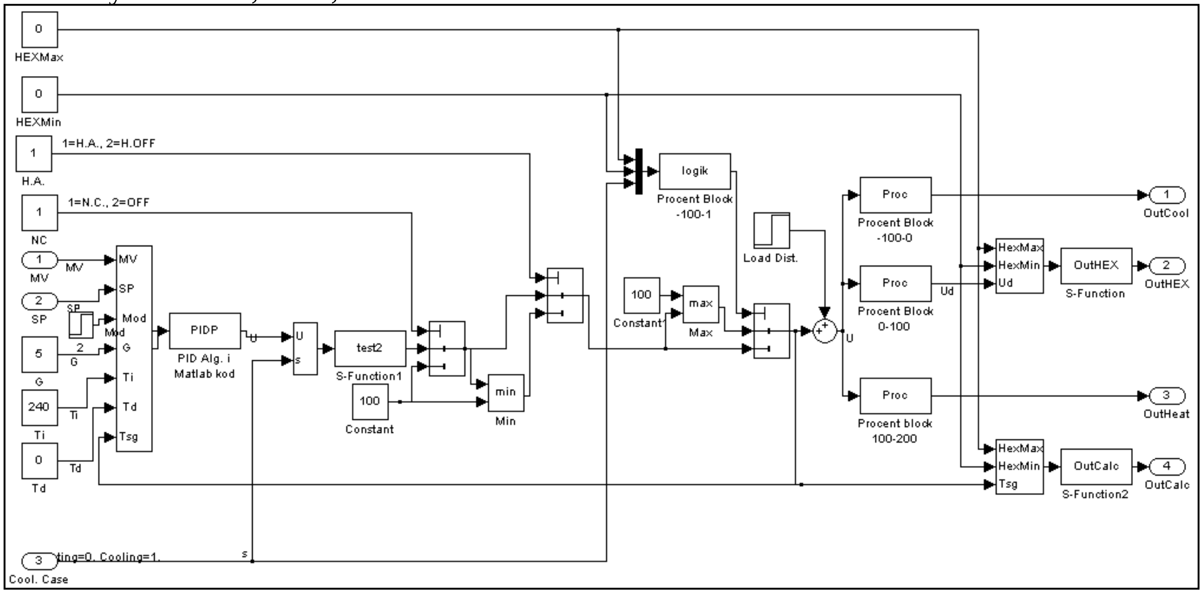


8.2 Appendix B, The Model CHHE

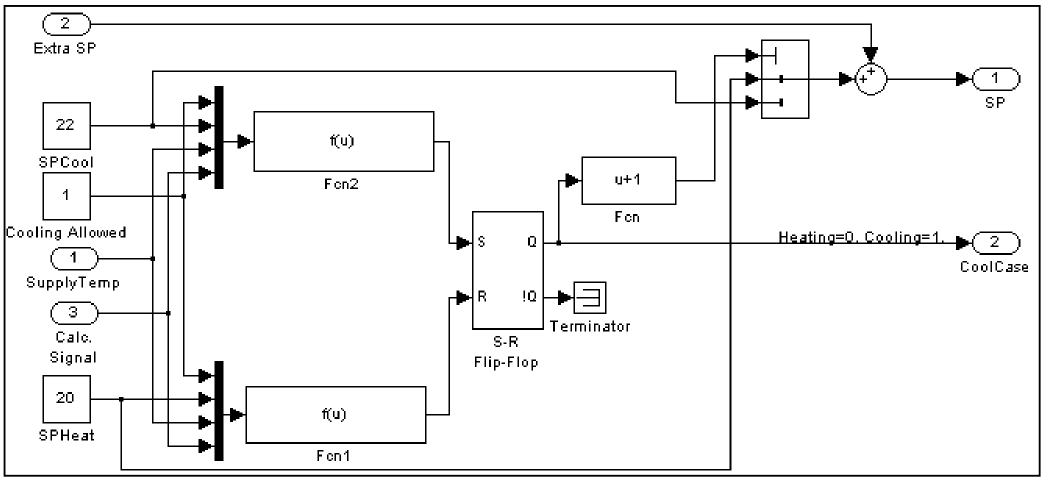
The main system of the model CHHE.



The subsystem Cool, HEX, Heat.

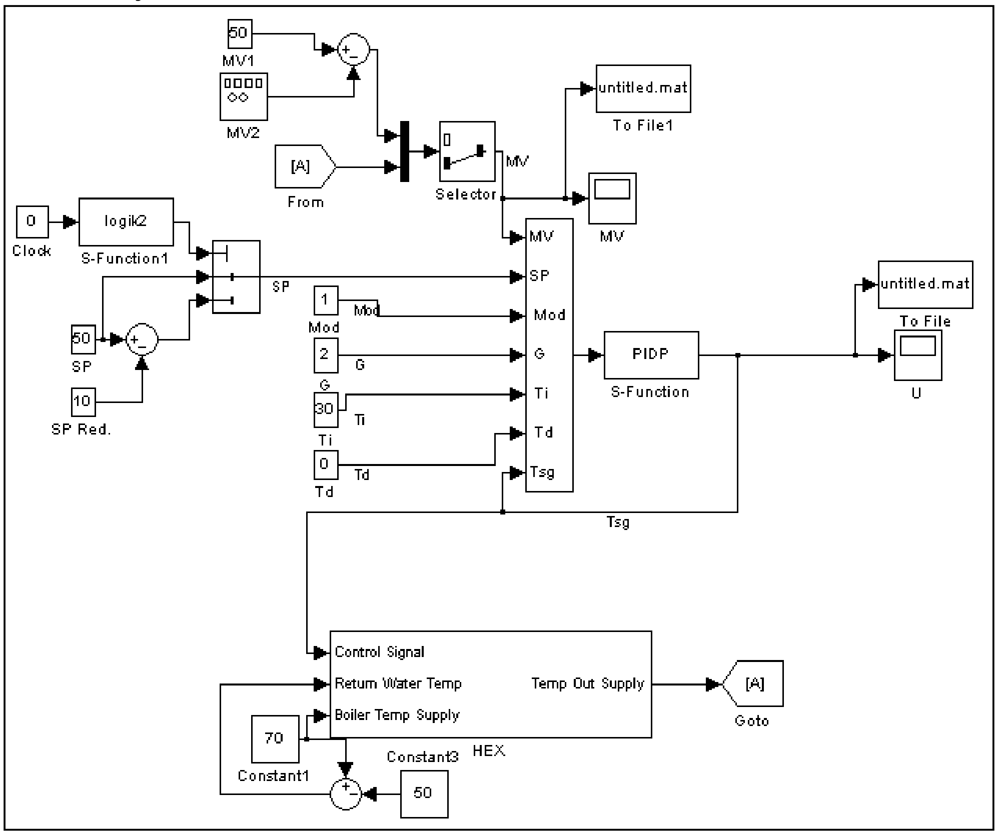


The subsystem Supply Control.



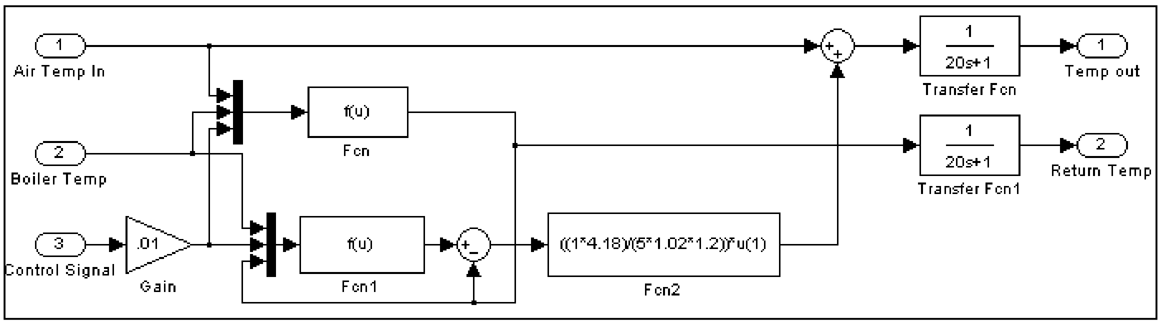
8.3 Appendix C, The Model DHW

The main system of model DHW.



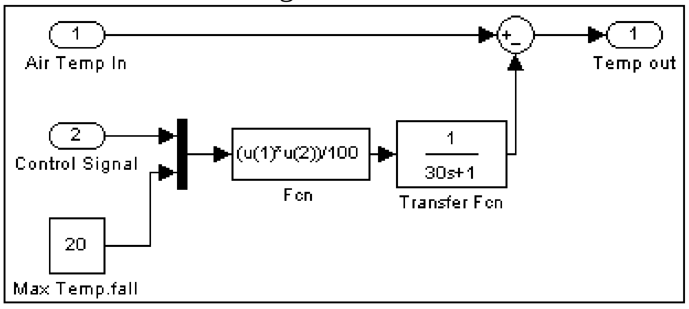
8.4 Appendix D, The Model of a Heating Coil

The model of a Heating Coil



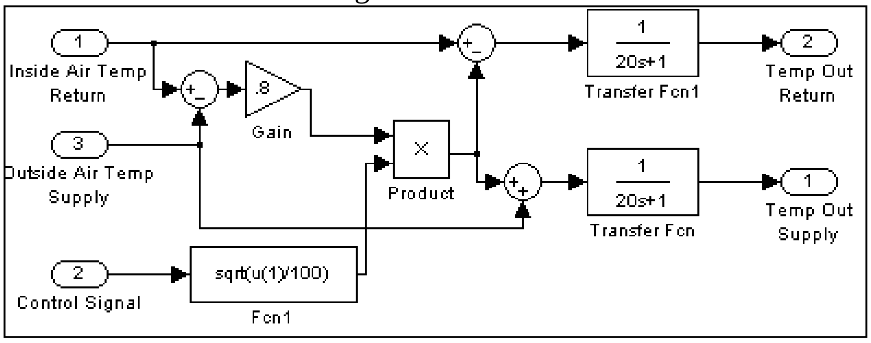
8.5 Appendix E, The Model of a Cooling Coil

The model of a Cooling Coil.



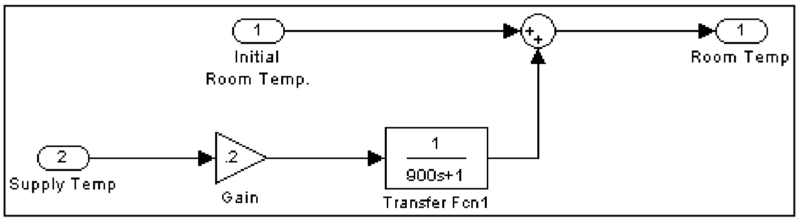
8.6 Appendix F, The Model of a Heat Exchanger

The model of a Heat Exchanger.



8.7 Appendix G, The Model of a Room

The model of a Room


1
Initial
Room Temp.
2
Supply Temp
2
Gain
1
900s+1
Transfer Fcn1
1
Room Temp