ISSN 0280–5316
ISRN LUTFD2/TFRT--5650--SE

Modeling and Control in Matlab for ABB's Control Builder

Mathias Larsson

Department of Automatic Control
Lund Institute of Technology
October 2000

Department of Automatic Control
Lund Institute of Technology
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
October 2000

Document Number
ISRN LUTFD2/TFRT--5650--SE

Author(s)
Mathias Larsson

Supervisor
Patrik Svensson, Per Olofsson, ABB
Tore Hägglund

Sponsoring organisation

Title and subtitle
Modeling and control in Matlab for ABB's control builder
(Modellering och simulering i Matlab för ABB's control builder)

Abstract

One problem when new controllers or control systems are built is the test of the controllers or the control systems. One way to test the controller is to build a real process to test the controller against. Another way to test the controller is to build the process-model in some kind of computer simulation tool. The advantages of using a simulation tool in a computer are quite many. For example it is often quicker to build the process-model in a computer than to build a real process. It is also often quicker and easier to change some parameters in a model built in a computer simulation tool, than to change the parameters in a real process.

In this master thesis a lot of verifications are made to confirm if it is possible to make real-time simulations of process-models build in Matlab/Simulink, and controllers implemented in ABB's Soft Controller. The connection between the process-models in Matlab/Simulink and the controller, implemented in the Soft Controller, is made via ABB's commercial OPC-MMS Server, connected to the Soft Controller, and a Gateway application, connected between the OPC-MMS Server and Matlab/Simulink.

The simulation work as it should if the user of this simulation tool is aware of the delays that will appear due to the different sampling stages of the system. The possibilities with this kind of simulations are that test of function blocks (FB) and control modules (CM) used in ABB's Control Builder can be made in an easy way. Another possibility is to show customers improvements that can be made in their control system.

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
84

Recipient's notes

Security classification

Abstract

One problem when new controllers or control systems are built is the test of the controllers or the control systems. One way to test the controller is to build a real process to test the controller against. Another way to test the controller is to build the process-model in some kind of computer simulation tool. The advantages of using a simulation tool in a computer are quite many. For example it is often quicker to build the process-model in a computer than to build a real process. It is also often quicker and easier to change some parameters in a model built in a computer simulation tool, than to change the parameters in a real process.

In this master thesis a lot of verifications are made to confirm if it is possible to make real-time simulations of process-models build in Matlab/Simulink, and controllers implemented in ABB’s Soft Controller. The connection between the process-models in Matlab/Simulink and the controller, implemented in the Soft Controller, is made via ABB’s commercial OPC-MMS Server, connected to the Soft Controller, and a Gateway application, connected between the OPC-MMS Server and Matlab/Simulink.

The simulation work as it should if the user of this simulation tool is aware of the delays that will appear due to the different sampling stages of the system. The possibilities with this kind of simulations are that test of function blocks (FB) and control modules (CM) used in ABB’s Control Builder can be made in an easy way. Another possibility is to show customers improvements that can be made in their control system.

Index

Acknowledgments

I would like to thank my supervisors Patrik Svensson at ABB Automation Product, Per Olofsson at ABB Automation Systems and Tore Hägglund at the Department of Automatic Control in Lund Institute of Technology. I would also like to specially thank Stefan Sällberg and Niklas Giheden at ABB Automation Product for their useful help with some of my programming problems. Lot of other people at ABB Automation has also been helpful during this thesis. Without all those nice people this master thesis work would have been less inspiring to do.

1 Introduction

This Master Thesis is a continuation of the master thesis “Communication between Advant Control Builder and Matlab/Simulink”[2]. The problem with the result in that thesis is that the communication does not work so good for real-time simulations.

The main idea with this master thesis has been to make the simulation of the control system to work in real-time. There has also been a desire to check how the process model can be build, and also to build some easy models.

1.1 Thesis outline

The outline of this thesis is described here. In the beginning of this thesis some theoretical studies are described that have been made to understand more about real-time programming and ABB’s control system. Since much of the work has been to verify that the simulation will work in real-time, there are in some chapters a lot of diagrams shown.

Descriptions of each chapter are here shown.

- Chapter two describes the ABB products used in this thesis. The description is like a small overview over how, and what the products can perform.

- Chapter three describes the main idea about how the simulated process-model is connected to the Soft Controller, instead of a real process.

- Chapter four is a small description about problems like delays in sampled system.

- Chapter five is a theoretical chapter that describes some parts that are used to understand how real-time applications in Windows works. There is also a theoretical part that describes how Matlab/Simulink and C++ work.

- Chapter six describes how the performance in the real-time simulation can be increased.

- Chapter seven describes the performance of the communication if the simulation of a control system is made on two computers. The simulation is made with simulation system developed in the master thesis “Communication between Advant Control Builder and Matlab/Simulink”[2].

- Chapter eight describes a test of the performance when two computers are used, and the improvements described in Chapter six is used.

- Chapter nine describes the simulation performance if one computer is used, and what have to be changed if one computer is used for all processes.

- Chapter ten is about the conclusions.
- Chapter eleven is about future possibilities.
- Appendix A is a description about how to use the Gateway application.
- Appendix B is a description about how to build process-models in Matlab/Simulink.

2 ABB:s distributed control system

In this chapter the ABB products used in this thesis are described. The described applications are Control Builder, Soft Controller and OPC-MMS Server[1].

2.1 Control Builder

Control Builder (CB) is a software development tool. It is a fully integrated Windows application for configuration of the ABB products: AC 800C, AC 800M, AC 250, and Soft Controller. The communication between the Control Builder and the controllers are made with serial line or Ethernet network. The communication connection is shown in figure 2.1.

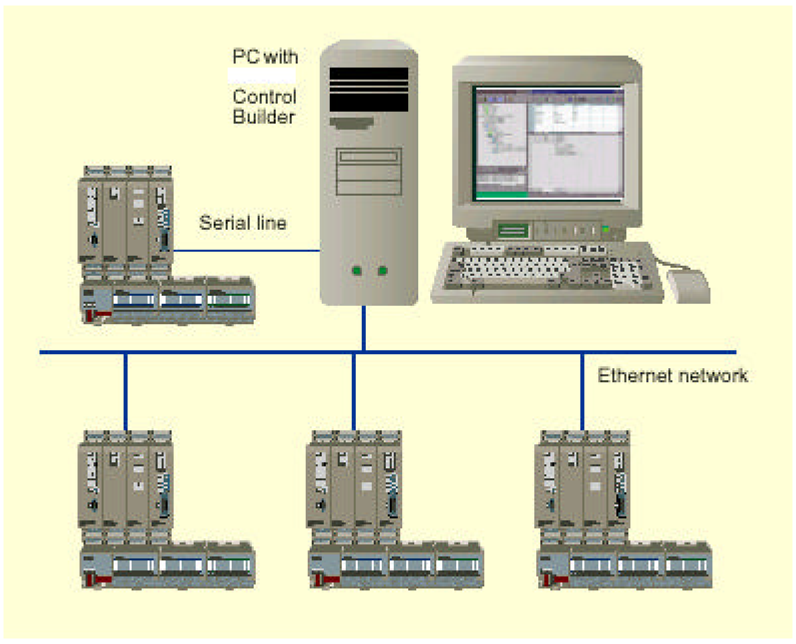


Figure 2.1. Communication between the Control Builder and the controllers.

A project in Control Builder is built up by an application, which can be divided into different programs. In the programs the program code, function blocks and functions are placed. All the programs are connected to a task. In the task the task time and the priority for the program are placed.

Programming of controllers is made with the Control Builder in off-line mode. This means that the controller is not in direct contact with the Control Builder, and changes in the code for a control loop cannot be seen at the controller in off-line mode. The Control Builder supports the languages Structured Text (ST), Instruction List (IL), Function Block Diagram (FBD), Sequence programming (SFC) and Ladder Diagram (LD).

Structured Text is a high level programming language. In Structured Text it is possible to write advanced and compact code in a logical and structured way.

Instruction List is a language where the instructions are listed in a column with one instruction at each row. The structure of the Instruction List is similar to simple machine assembler code.

Function Block Diagram can be described as function blocks that are connected together with lines. This looks very much like models that are built in Simulink. The function block can be for example AND-or OR gates etc.

Sequence programming is a way of programming that looks like Grafcet programming. The programming is built up with something that looks like state-graphs with states events and transitions between states. In Sequence programming it is also possible to have parallel branches of the execution lines.

Ladder Diagram is a graphical programming language that is made to look like relays, connection terminals and connections between relays. This programming language is made to make it easier for electricians that has used the old way of ladder programming, with real relays and connection terminals, to use new PLC's with computer instead of real relays.

With the Control Builder there are possibilities to use libraries with predefined functions, function blocks and control modules. Some of the libraries are:

- The System library that contains all the basic data types and functions e.g. type conversion, math and time.
- The Logic function library that contains flip-flop, timer and counter function block.
- The Communication library that contains client function blocks for protocols like MMS, SattBus etc.
- The Control libraries that contain control blocks like for example PID function blocks.
- The Alarm library that contains function blocks for alarm and event detection.

When the program is made it is compiled to machine code and optimized for the specific controller before it is downloaded into the controller. Before the program is downloaded to the controller there are possibilities to test the program in a simulate mode inside the Control Builder. This makes it possible to find some of the possible errors that can appear before tests are made with real controllers and processes. All errors that can appear when a real process is connected to the controller are not possible to find in simulate mode, but some of the wrong code can be found.

When the program is loaded to the controller, the Control Builder can be used as an online tool to check and change status of variables in the control loops. This means that for example parameters in a PID controller can be changed while the controller is running and controlling a real process.

2.2 Soft Controller

Soft Controller is a real-time controller that runs under Windows NT in a PC. The programming of the controller is made via Ethernet or serial COM port. The communication between the Control Builder and the Soft Controller is achieved with the MMS protocol.

The I/O communication is made with Central I/O via serial I/O Bus or with Remote I/O via PROFIBUS. The main difference between Central I/O and Remote I/O is the possible length of the communication cable. Central I/O can have a maximum length of 2.5 meters, and remote I/O with PROFIBUS can have cable length of 100 to 1200 meters depending on the transmission rate.

Communications with other control systems are done with the standards MMS, Sattlink, COMLI, SattBus, Data Highway Plus or 3964R. The system with the Soft Controller and possible communication links are shown in figure 2.2

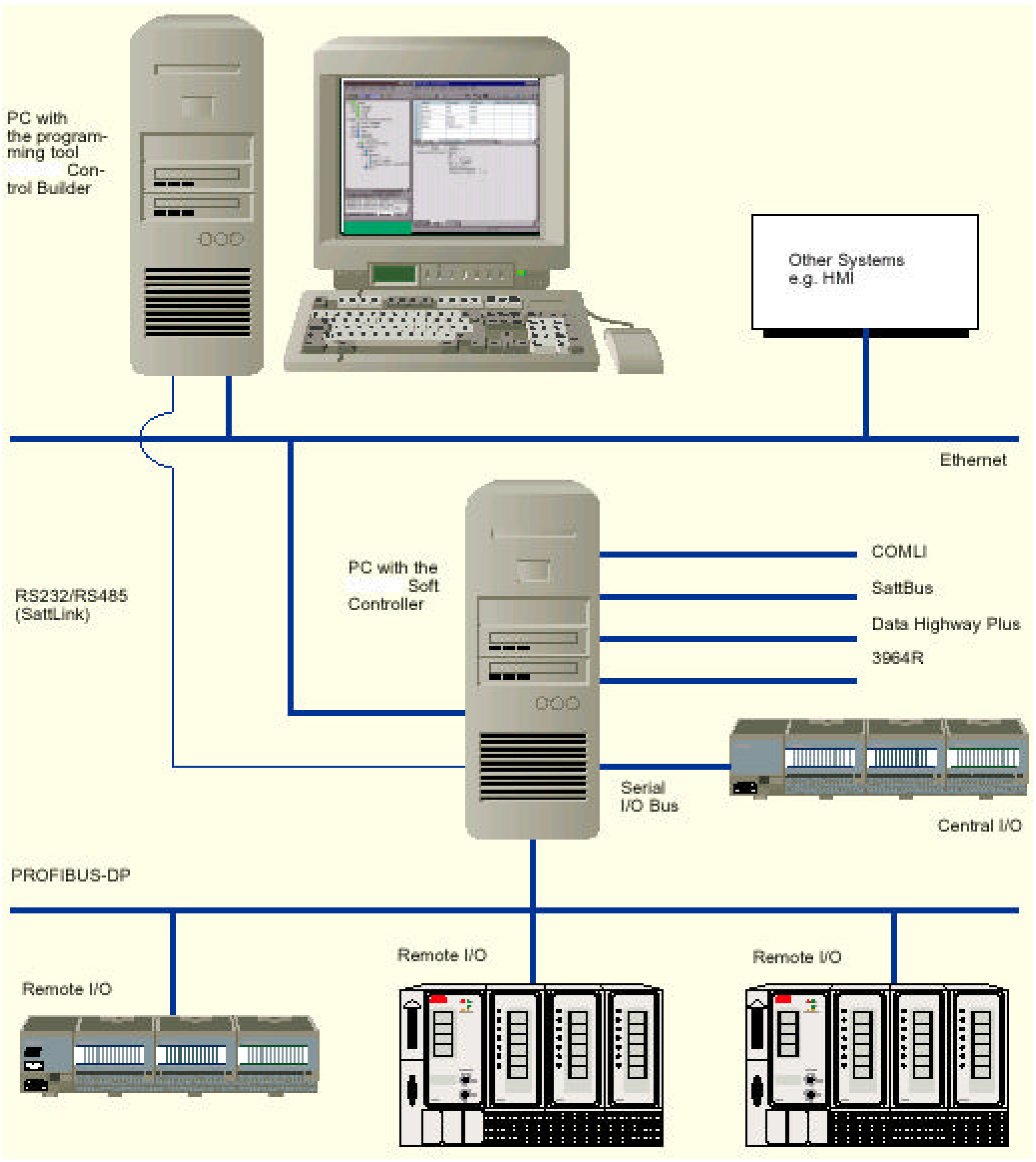


Figure 2.2. Control system with Soft Controller, Control Builder and communication included.

2.3 OPC-MMS Server

ABB has an own OPC-MMS server that makes it possible to write and read global MMS-variables from the controller. The OPC-MMS server acts as a MMS client to the controller that it is connected to. Then the OPC-MMS server converts the global MMS-variables from the controller to OPC items that can be read/write from OPC-clients.

The OPC-MMS Server is primary made for use in communication with operator- and supervisor system. It is therefore not built for high precession communication between controllers and processes. This means that it for example has no absolute sample time.

3 Simulation idea

The main idea with simulation in this thesis has been to check if it is possible to simulate processes in Matlab/Simulink in real-time, and also control these processes with ABB's Soft Controller. The desirable way of doing the simulation is to have everything that is needed for the simulation and control in one computer. It is also desirable to use standard components of ABB's commercial products when that is needed.

3.1 Real process

To understand how real processes and controllers work, a block diagram over the real control system including the process is drawn in figure 3.1. The Soft Controller, with for example a PID controller, is used to control a real process via some kind of fieldbus, I/O, actuators and sensors. The set-point to the PID controller in the Soft Controller comes from the Control Builder.

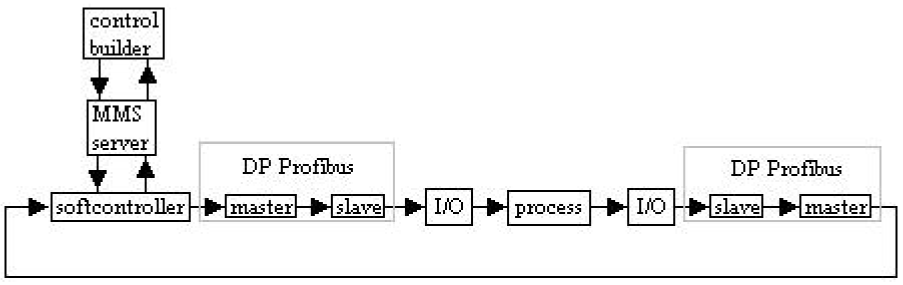


Figure 3.1. Block diagram for control system with a real process.

3.2 Simulated process

One way to make it possible to simulate the process is to build a process model in Matlab/Simulink and connect this process-model to the Soft Controller via Matlab Engine, a Gateway application and the OPC-MMS Server. This connection is shown in figure 3.2. Instead of taking the in- and output signals via a fieldbus from the Soft Controller the OPC-MMS Server together with the Gateway take care of the communication. The OPC-MMS Server, Control Builder and Soft Controller are described in chapter 2. The Gateway is an application that has to be used to make the communication between Matlab Engine and the OPC-MMS Server possible [2].

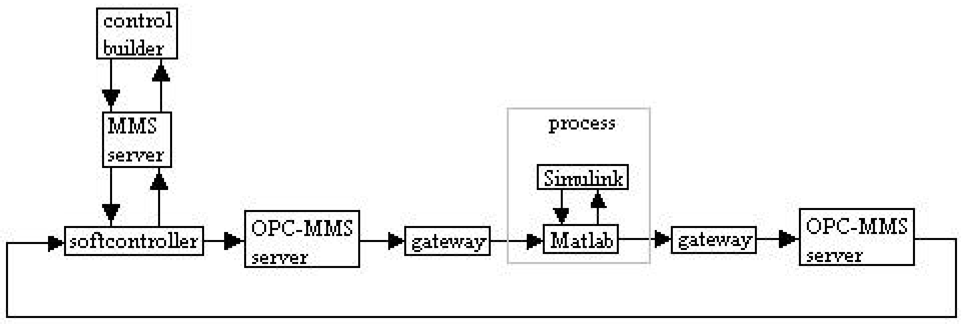


Figure 3.2. Block diagram for control system with the process simulated in Matlab/Simulink.

4 Evaluation of problem with the simulation idea

There are some problems with the way of solving the simulation idea described in chapter 3. The biggest problem is that the signal is sampled six times in loop from the Soft Controller and back again. Another problem is that the sample times are not synchronized, which means that big variable delays can occur. In this chapter problems with delay and sample times are described.

4.1 Delays

A delay in the system can be written as f(t-T). Laplace transform of f(t-T) is:

$$L(f(t-T))(s) = \int_0^\infty f(t-T)e^{-st}dt = e^{-sT}L(f(t))(s)$$

The delay with T can therefore be written as a multiplication with $G(s)=e^{-sT}$. The amplification of the signal is $|G(i\omega)|=1$ and arg $G(i\omega)=-\omega T$. This means that there is a phase lag. If the delay is big it will therefore be difficult to get a stable closed loop system. The phase margin will be smaller with bigger delays. In figure 4.1 a phase plot for e^{-s} is shown.

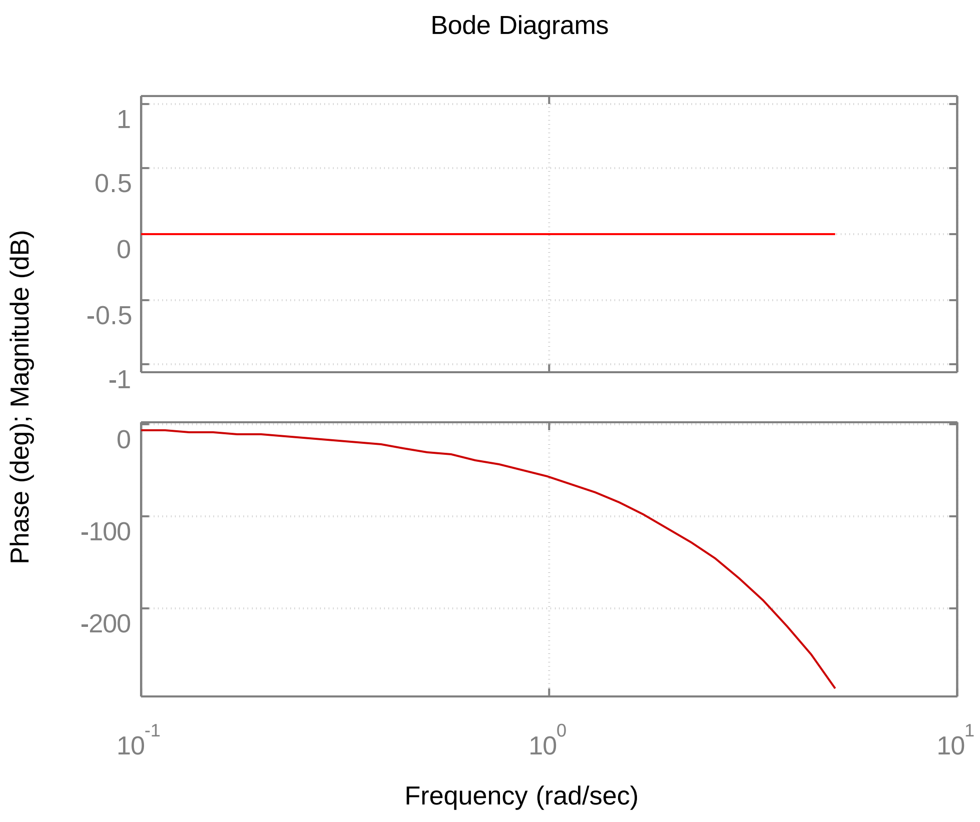


Figure 4.1. Bode diagram for e^{-s}. If the delay is big compared to the desired cut frequency it can be difficult to get a stable system.

In sampled system there will always be delays related to the sample and hold circuit. For small sample interval T the delay can be written as a time delay e^{-sT}. Sampled system always has lower stability margins than a time continuous system.

If PI controllers are used to control systems with time delays, the behavior of the system may not be good. The integrator part of the controller can make the system unstable if the gain is not kept low. If the gain is kept low, the step response will be quite slow.

4.2 Selection of sampling rate for controllers

The theoretical lowest sampling rate, for controllers, to avoid aliasing problem is twice the highest frequency component in the signal that is sampled. For controllers a reasonable choice of sample rate for closed loop systems are 4 to 10 per rise time [3].

5 Study of important subjects needed to understand the different parts in the simulation idea

In this chapter some terms and definitions that are needed to understand and solve problems that arrived when the simulation problem was going to be solved are described. The subjects that are described are Windows 2000, some real-time programming terms, Visual C++ and Matlab with some of its toolboxes.

5.1 Windows 2000

An introduction to some important details about Windows, as a real time operating system, that is good to know when making real time applications for Windows environments.

5.1.1 Real-Time operating system

The definition of a real-time system is that it does not only depend on the logical correctness of the computation, but also upon the time at which the result is produced. This means that all data have a time component included.

Real-time applications can be divided into two groups: hard real-time and soft real-time. Hard real-time applications must provide a response to some kind of event within a specified time. This response must be predictable and independent of other activities in the operating system. Soft real-time operating system operates in high speed to get as low response times as possible, but the response time is not predictable and not independent of other activities in the operating system.

Windows 2000 is not a hard real-time operating system. It is more a general-purpose operating system that has capabilities of fast response time.

5.1.2 Programming Windows applications

Traditionally applications execute on top of an operating system, which take care of file management and communications with other unit around. The application has a sequential structure where the execution starts in a main function and from this main function, other functions in the application are called. When the application needs something from the operating system it uses systemcalls. A systemcall is a call for a system function that can manage a favor like for example open a file, writes something on the screen etc. In a traditional operating system the program asks for services from the operating system [4]. Figure 5.1 shows a traditional operating system.

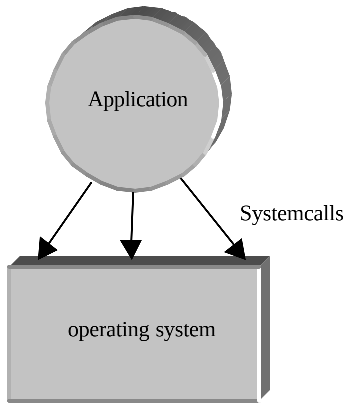


Figure 5.1. An application in a traditionally operating system executes sequential and requests for services from the operating system, so called systemcalls.

Windows operating system uses events from the user and sends all these events to the application. The application has then probabilities to choose that of the events it would like to use. Therefore the application in Windows has a lot of messages to take care of. In this case it is more like Windows is the overhead function and the program runs on Window rules. Windows operating system is shown in figure 5.2.

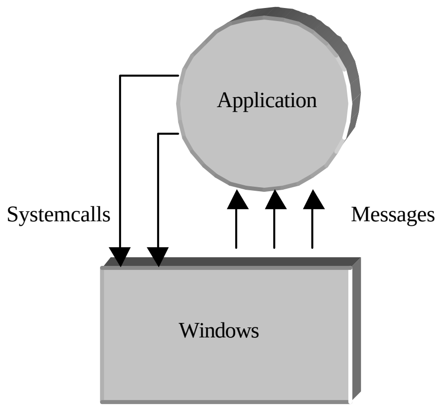


Figure 5.2. Windows applications are controlled by the operating system messages.

5.1.3 Win32 API

In Windows NT the Win32 is the name of the application programming interface (API). This interface defines a set of functions that an application can call, and it also defines how these functions behave.

5.1.4 Processes

A process can be defined as an instance of a running program. Processes built with win32 are inert. This means that the process can not do anything if it is not owning a thread (thread is described later). The thread is then responsible for executing the code in the processes address space. There can be a several threads inside one process. All the threads are executed at the "same time". The meaning of "same time" is not really true if the computer that executes the treads do not have one CPU for each thread. This is of course not the usual way of executing threads, instead the operating system schedules the time that each thread can use the CPU [5].

5.1.5 Threads

A thread is a path of execution within a process. If the process has more than one thread within it, it becomes a multithread process. In multithread processes it becomes more difficult to know how long time each task uses. This problem is related to that the threads are constantly preempted. Preempted means that it is time for other threads to execute, and executing thread has to be suspended for a time [5].

In Windows there are two kinds of threads, worker threads and user interface threads. The differences between these threads are that the user interface thread has windows and its own message loop [6].

5.1.6 Priority

The threads and processes that are executed can have a priority given from the user. The priority can be between 1 and 31. The priority is not written directly, it is instead given as in the table 5.1 and table 5.2. A thread with higher priority is always executed before one with lower priority. Therefore it is not a god idea to make a thread with an infinite loop and then give it the highest priority in the process. If there is an infinite loop in the thread there shall be some function that suspend the thread for some time, to let other threads execute. Functions that suspend threads are for example:sleep, WaitForSingleObject and WaitForMultipleObject. If there is no priority settled for the threads or the processes they are all going to get the same priority, the normal priority [4].

Table 5.1. Priorities for processes. In the third row there is a priority value 7/9. This means that the process gets the value 7 if the process executes in the background and 9 if it executes in the foreground.

priority class	**priority value**
IDLE_PRIORITY_CLASS	4
NORMAL_PRIORITY_CLASS	7/9
HIGH_PRIORITY_CLASS	13
REALTIME_PRIORITY_CLASS	24

Table 5.2. Priorities for threads.

priorityconstants for threads	**priority value**
THREAD_PRIORITY_LOWEST	two lower than priority class value
THREAD_PRIORITY_BELOW_NORMAL	one lower than priority class value
THREAD_PRIORITY_NORMAL	same as the priority class
THREAD_PRIORITY_ABOVE_NORMAL	one higher than priority class value
THREAD_PRIORITY_HIGHEST	two higher than priority class value
THREAD_PRIORITY_IDLE	1, if not realtime class, then 15
THREAD_PRIORITY_TIME_CRITICAL	15, if not realtime class, then 31

5.1.7 Scheduler

Most of the facts in this section comes from MSDN Library Visual[7]. The scheduling routines of the operating system run at the highest priority. Processes that run at highest priority can not be interrupted during a quantum. A quantum is the maximum amount of time that a thread can run before the system checks for another ready thread of the same priority to run. This prevents a CPU-bound process from monopolizing the processor. Currently in Windows 2000, 3 quantum are equal to either 10 milliseconds with single processor or 15 milliseconds with multiple processors.

The switches between the threads are called context switches. A context switch consists of tree main parts. The first part is to save the states of the processor immediately before the context switch. The second part is the actual context switch. And the third part is to resume the states of the restored process.

When a thread has been running for one quantum the kernel preempts it and moves it to the end of the ready queue for its priority. The scheduler is based on the priority of the processes and the threads. The scheduler always lets the thread with the highest priority run, even if a thread with lower priority is in the middle of its quantum when this happens. To allow all threads to execute sometimes it is important that some function, that suspends the thread, is written in the code for the threads. Otherwise the thread with highest priority will always has access to the CPU. Functions that make threads suspended are those functions that wait for messages from other threads. Some of these functions are explained under in chapter 5.1.9.

In the Win32_OperationSystem class information about currently running Win32 operating system is represented. Two of the properties in the Win32_Operating class are QuantumLength and Quantumtype.

The QuantumLength is an 8-bit number that specifies the number of time ticks before an application is swapped out. QuantumLength can be set to 0, 1 or 2. The number represents quantum length of the threads between 10-30 ms.

The QuantumType is an 8-bit number that specifies fixed or variable length of the quantum. It is possible to use longer quantum for foreground applications than for background applications.

5.1.8 Multitasking

Windows NT has possibilities to real multitasking. The Win32 module takes care of the scheduling of the processes and the threads. A thread is the part that is executed, and a process can have one or more threads. It is possible to perform multitasking either by making more processes or by more threads. The advantages by having more threads in a process are that they can share variables, memory etc. This makes threads better than processes if you look for as small usage of CPU power as possible. To execute a thread more often than the other threads, there are possibilities to set priorities of the threads.

The problem with multitasking is that you do not know when the processes or the threads execute. Therefore is it quit important to synchronize the processes and threads. This can be read about in chapter 5.1.9.

5.1.9 Synchronization of multitasking threads

When the CPU has more than one thread to execute a new problem will be introduced. The problem is that it is impossible to know when the switches between the threads are going to happen. To solve this problem there are some synchronization tools that can be used. The tools are critical sections, mutexes, semaphores and events. These synchronization objects are all kernel objects except the critical sections, which only can be used inside one process [4].

5.1.10 Critical section

A critical section protects availability of a variable in the process. This means that a thread can use the variable but another thread can not use the variable before the first thread has left the critical section. If a thread try to use the critical variable while another thread already uses it, the trying thread will be suspended until the first thread is finished in the critical section. Critical section works only on threads inside the same process [5].

5.1.11 Mutex (Mutual Exclusion)

A mutex is almost the same as a critical section, but it can also be used between processes [5].

5.1.12 Semaphores

A semaphore has a functionality that looks like the critical section and the mutex. But the semaphore can include a counter instead of a flag. A process can also wait for a semaphore that has been released by an other process [5].

5.1.13 Events

The event is used to report to other processes. Mutexes and semaphores are usually used to control access to data, but semaphores are used to signal that some operation has completed. For example a process can be dependent on that another process is finished with its work before it self can execute. This can be handled with an event [5].

5.1.14 DLL-files (Dynamic Link Library)

In Windows there are two kinds of files that are executable, files with the suffix .exe and .dll. The difference is that a DLL file can not execute by it self, it needs an application that loads and calls the library. A DLL is a program module that is linked to the application when the application is compiled. If there are more than one application that uses the same DLL, the DLL is only loaded once, and the data are shared between the applications. This means that a DLL also can be used as a communication module or to handle coordination between applications. In figure 5.3 the difference between static linking and dynamic linking is shown.

A DLL exports functions that can be used by all applications that have links to that DLL. The data of the library are shared between the applications that use the library. A DLL can also be seen as a client/server architecture, where the DLL module serves some client applications [4].

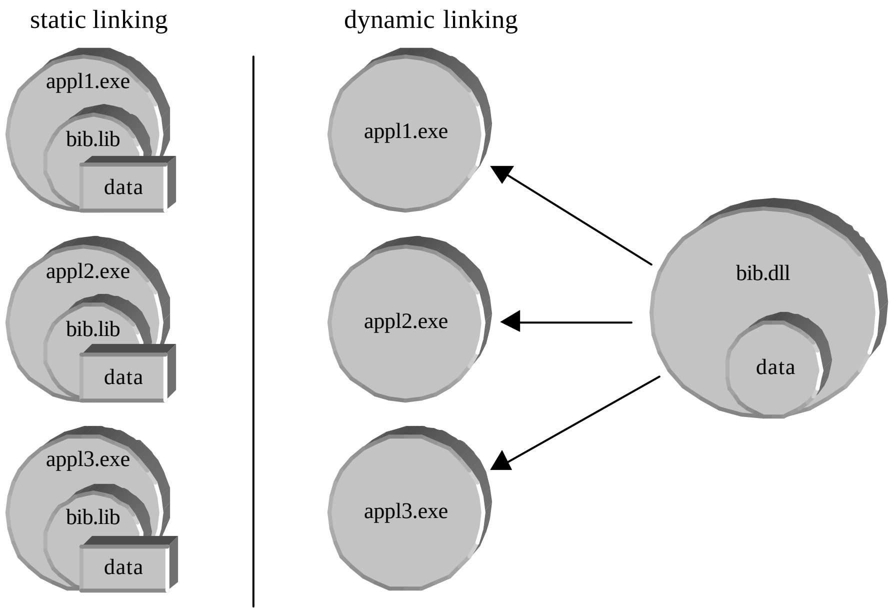


Figure 5.3. Static linking compared to dynamic linking.

5.1.15 ActiveX

ActiveX is a Microsoft Windows protocol for component integration. ActiveX is a family of related object-oriented technologies that have a common root, called the COM (*Component Object Model)* [5].

5.1.16 Component Object Model (COM)

COM is a specification on how to build reusable components that can be used with different computer languages. The specification standard says how components and clients should be build to ensure that they can work together. To hide the language that the components are written in, the components has to be compiled to binary form before they are ready to use. There are more to read about COM in [8] and some specific about COM in this kind of project can be read about in [2], [9], [10].

5.1.17 Manufacturing Message Specification (MMS)

MMS is a collection of abstract commands for monitoring and control of industrial equipment. The basic idea of MMS is that it works as client-server model. The client requests for a service from the server. The server executes the service and gives an answer to the client as acknowledgment and specification of the operation [11]. In figure 5.4 the described MMS client-server model is shown.

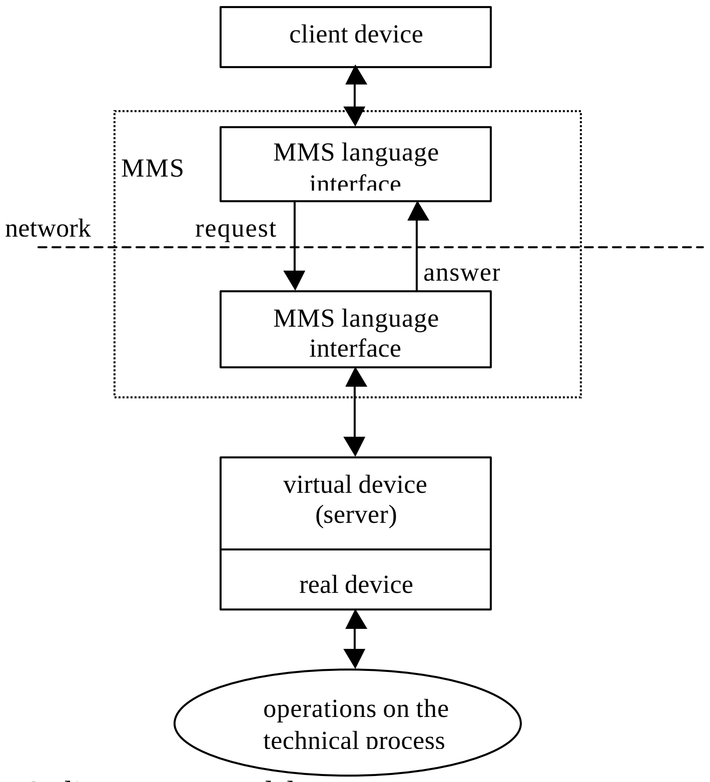


Figure 5.4. MMS client-server model.

5.2 Timers in Windows

There are some different timers that can be used in Windows. Here three of those timers are described.

5.2.1 System Timer

The system timer is a timer that is included in Windows NT and Windows 2000. GetTickCount is a function that can be used to retrieve the number of milliseconds elapsed since Windows was started. The limitation with the system timer is the resolution. The resolution is approximately 10ms. As can be seen in chapter 8 this is not so good if measurements of small time differences are made. To avoid this problem multimedia timer or high-resolution timer can be used [7].

5.2.2 Multimedia Timer

An application, that uses the multimedia timer, requests and receives timer messages at specified intervals. The resolutions for multimedia timers are set by the functions timeBeginPeriod and timeEndPeriod. The function timeBeginPeriod should be called immediately before multimedia timer functions are used, and the function timeEndPeriod should be called immediately after the multimedia timer functions are used. The desired resolution of the multimedia timer is the only input parameter to the above described functions. The desired resolution must be the same in both the timeBeginPeriod and the timeEndPeriod. The minimum resolution that can be set is 1 ms. One problem with the multimedia timer is that the interrupt handler does not make context switch possible during the time when the timer messages arrives [7].

There are some functions that can be used between the timeBeginPeriod and timeEndPeriod. The function that is used in this thesis is timeGetTime which gives the time, with the desired resolution, elapsed since Windows was started.

5.2.3 High-Resolution Timer

If higher resolution than 1ms is needed the high-resolution timer shall be used. The problem with using high-resolution timer is that not all systems have a high-performance-counter, which means that if applications are made dependent on high-resolution timer the application is not system independent. Therefore the high-resolution timer is not used in any of the simulation tests [7].

5.3 Visual C++

The Gateway application that is used in the communication link is written in C++. Therefore it is necessary to have knowledge about the language C++. To be able to write Visual C++ applications it is good to know a little about the build process. Figure 5.5 shows an overview of the Visual C++ build process. Not all details in this figure are here going to be explained. For more details of the build process the recommendation is to look in the book [5]. But the major parts of the figure 5.5 are explained here.

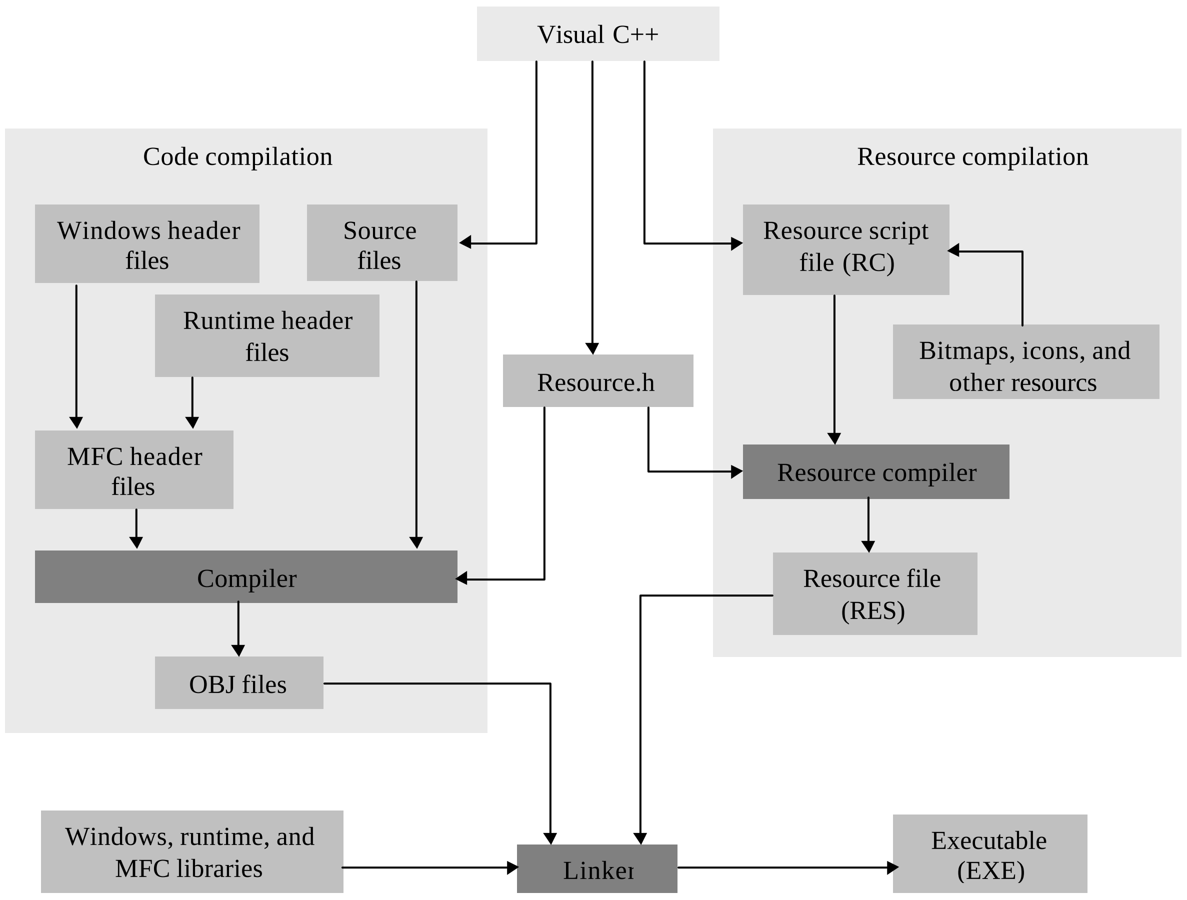


Figure 5.5. Picture of the Visual C++ application build process.

5.3.1 The Resource Compiler block

In this block are for example the windows, dialog boxes, icons etc. built and compiled. The resource script file (RC) is a file in text format that describes the project’s menu, dialog etc. The Resource compiler reads the RC file and then compiles this file in to a binary file for the linker [6].

5.3.2 Code compilation block

This block includes the source C++ code that is written and header files for the application. It also includes the C++ compiler that compiles these files into an OBJ file for the linker [6].

5.3.3 The Linker

The linker reads the OBJ and RES files produced by the C++ compiler and the resource compiler. It also accesses the LIB files for MFC (Microsoft Foundation Class) code, and Windows code. It then creates the project's EXE file [6].

5.3.4 Microsoft Foundation Class library (MFC)

The MFC library is a class library that makes it easier for the programmer to make powerful applications. It includes a lot of features, for example: File Open, Save, Save As, Print Preview, Print support, Dialog windows etc. There are a lot more features than those that have been shown here and those can be read more about in [6].

5.4 Matlab

Matlab is a computation tool for advanced mathematical calculation. It is built to handle big matrix calculations. There are also great possibilities to use Matlab solver for calculations needed in other programs. Appendix B describes how to use Matlab/Simulink when a small simulation model is built.

There are a lot of toolboxes that can be used together with Matlab. Some of the toolboxes that can be of interest when designing and testing control systems are in this chapter briefly described. There are more to read about those toolboxes at Mathworks homepage [12].

5.4.1 Simulink

Simulink is a toolbox to Matlab that makes it possible to build advanced simulation models with a graphical user interface. Figure 5.6 shows how a model may look like in Simulink. The blocks in the model can be a mixture of continuous and discrete blocks.

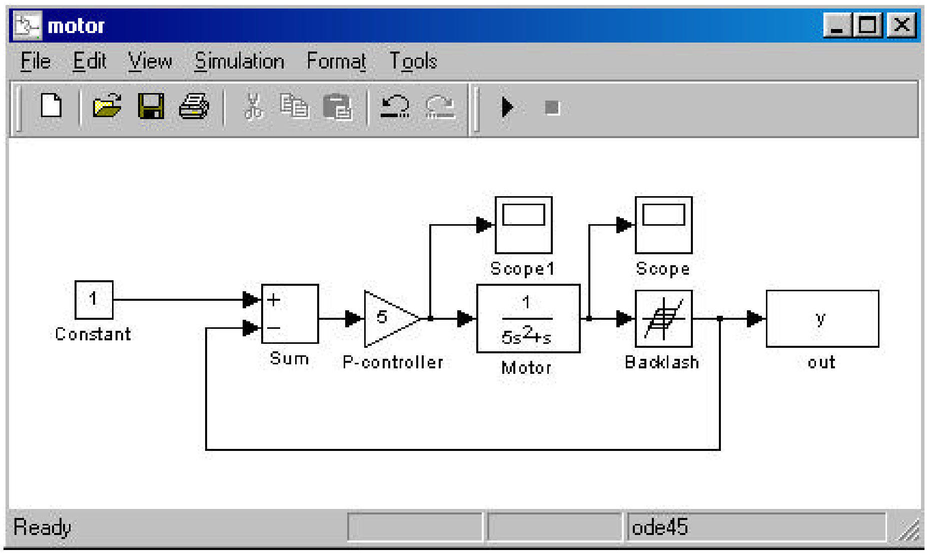


Figure 5.6. Example of how Simulink models may look like. This model is a motor with backlash and a P-controller. There are also two scopes that show the signals at the point where they are connected.

To be able to construct Simulink models that are working good, even in real time simulation tasks, it is a good idea to first look a little bit into how simulation in Simulink works. In figure 5.7 a block diagram over the simulation loop is shown [13].

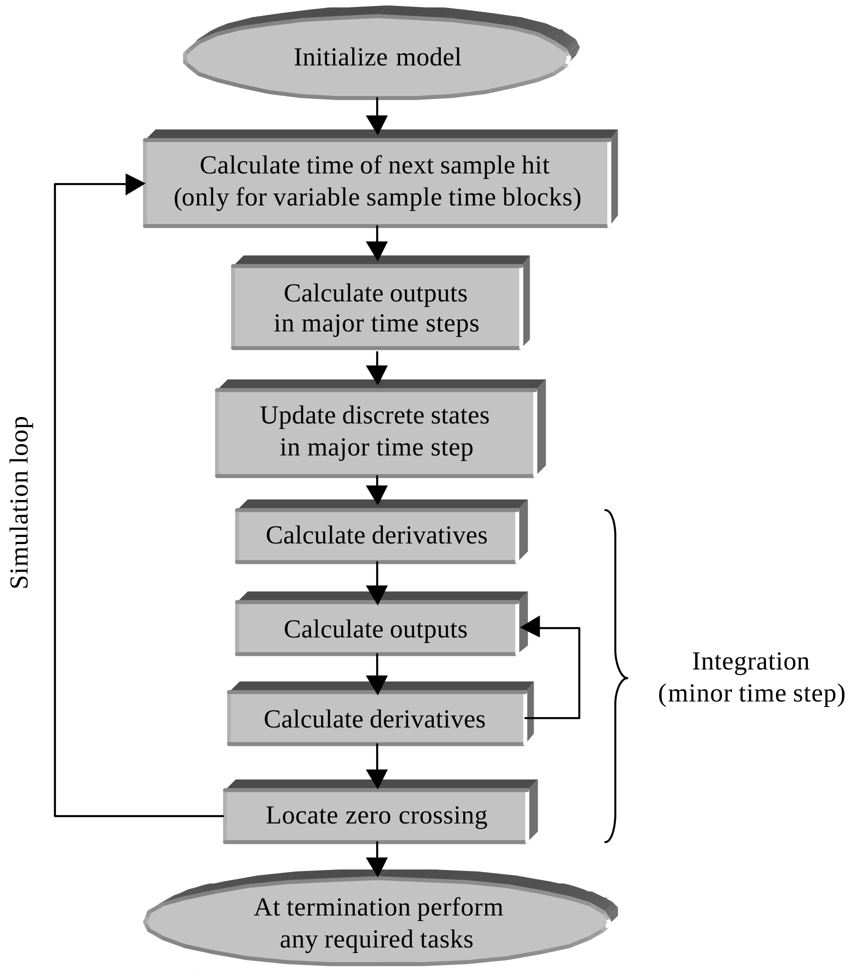

Initialize model
Calculate time of next sample hit (only for variable sample time blocks)
Calculate outputs in major time steps
Update discrete states in major time step
Calculate derivatives
Calculate outputs
Calculate derivatives
Locate zero crossing
At termination perform any required tasks
Simulation loop
Integration (minor time step)

Figure 5.7. How Simulink perform simulation.

Initialize model.
Here are all initialization of all S-functions and other blocks in Simulink made.

Calculate time of next sample hit (only for variable sample time blocks).
If variable step is selected, the next time for the simulation loop is calculated in this step.

Calculate outputs in major time steps.
After this step is all output ports valid for the current time step.

Update discrete states in major time step.
In this step is all discrete steps updated once every sample time.

Integration (minor time step).
These steps are called if the model contains continuous states that need to be updated with minor time steps.

5.4.2 Stateflow

The Stateflow toolbox can be used to make graphical models of event driven systems using finite state machine theory, statechart formalisms, and flow chart notation. Together with Simulink the Stateflow toolbox gives an opportunity to design embedded systems with complex control and supervisory logic within the Simulink model.

To create a Stateflow diagram in Simulink, a Stateflow Diagram block from Simulink Library has to be put into the Simulink model. By double clicking on the Stateflow Diagram block a new window appears where the states and the transition can be drawn.

5.4.3 Real-Time Workshop

The Real-Time Workshop toolbox is used as a complement to the Simulink and the Stateflow toolboxes. It can be used for rapid prototyping, embedded real-time control and real-time simulation [12].

As a rapid prototyping tool, Real-Time Workshop helps the developer to speed up the development of new ideas. There is no need for hand coding, the code is automatically real-time optimized and compiled into C code from the graphical picture of the model in Simulink. The compiler takes care of all different kinds of Simulink blocks. This means that it is possible to use both continuous and discrete functions in the Simulink model that is compiled.

Embedded Real-Time Control systems are easy to build when the system is designed in Simulink or Stateflow. Code for real-time control systems or Digital Signal Processing (DSP) applications can be generated, compiled and loaded into the selected processor.

Real-Time Simulation with entire system or hardware in the loop simulation can be done. Real-time simulations are typically used in applications as: training simulators, real-time validation and prototype testing.

5.4.4 Real-Time Windows Target

Real-Time Windows Target is a toolbox made for optimized simulations of real time system [12]. It has a special library with blocks that makes it easy to run true hardware in the loop. Real-Time Windows Target consists of two major components, real-time kernel and a Simulink library with blocks that makes it easy to connect true hardware into the Simulink model.

The real-time kernel is a highly optimized real-time kernel that runs in real-time priority under Windows NT.

The Simulink block library in Real-Time Windows Target makes it possible to connect I/O signals from a lot of standard I/O boards connected to the computer.

5.4.5 System Identification Toolbox

The System Identification Toolbox is a toolbox that creates mathematical models of dynamical systems based on observed in- and output data [12]. This toolbox includes Matlab functions for:

- Parametric estimation
- Spectral and correlation analysis
- Simulation
- Model validation
- Model order selection
- Presentation
- Model conversion

5.4.6 The Control System Toolbox

The Control System Toolbox can be used for modeling, analyzing and designing of control system. Lots of classic control tools are included in this toolbox, for example plot functions for Nyquist-, Bode- and root-locus diagram etc [12].

5.4.7 The Robust Control Toolbox

The Robust Control Toolbox provides tools for the design and analyses of multivariable control systems where robustness is a concern. This includes systems where there may be modeling errors, dynamics that are not completely known, or parameters that can vary during the life span of the product [12].

5.4.8 M-files

M-files are files written in a special language that is used in Matlab. The M-files are written as a function that can be called from Matlab.

5.4.9 MEX-Files

Mex-files are subroutines with the code written in C or Fortran. The MEX-files are dynamically linked into Matlab. This means that there are possibilities to call existing C or Fortran programs without rewriting them as M-files. It is also more efficient to write the code in C or Fortran if the code includes computation loops as for example for-loops [14].

5.4.10 S-Functions

An S-Function is a special function that the user can write to interact with Simulink´s equation solver [15]. The S-function may include both continuous and discrete systems. It is also possible to have different sampling rates inside the same S-Function. There are two ways to write an S-Function. The ways are like an m-file or as C-file that is compiled to an MEX-file. In Windows the MEX-file gets the suffix .dll. Adding the S-Function block from the Functions & Tables sublibrary into Simulink and write in the name of the S-Function in the dialog box makes the implementation of the S-Function into Simulink.

In this thesis the C MEX-file has been used and this is the way of writing S-Function that here is going to be explained more in detail. S-functions that are written as a C MEX-file provide more functionality than S-functions that are written as M-files. Another difference is that S-functions written as C MEX-files have a shorter execution time related to that Simulink do not need to call the whole S-function in every step of the simulation. By using C MEX-files Simulink directly calls the right part of the S-function for the state that the simulation is in. Figure 5.8 shows how the different functions in the S-Function are placed in simulation order. There are a lot of other functions that also can be placed in the simulation loop, but here only the functions used in this project are described.

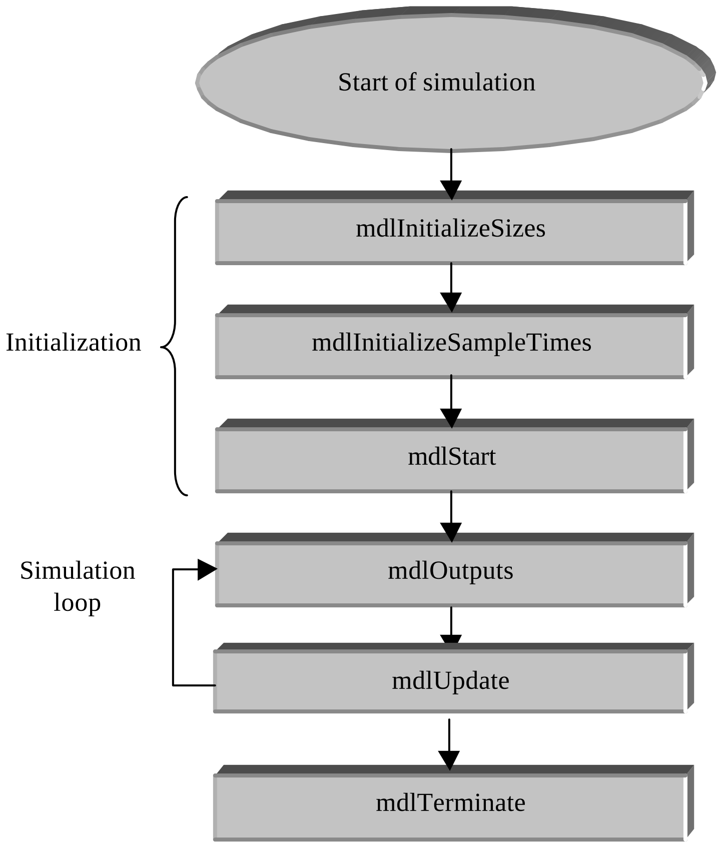


Figure 5.8. Block diagram over how S-Function Works.

mdlInitializeSize
In this function the number of inputs, outputs, sample times, function parameters etc. are initialized.

mdlInitializeSampleTimes
This function initializes the sample time.

mdlStart
This function initializes vectors and variables

mdlOutputs
This function is called once every time step to calculate and update the outputs.

mdlTerminate
Termination of the S-Function at the end of the simulation.

5.4.11 Performance of simulation in Simulink

The performance of the simulation depends on a lot of different factors. This makes it important, especially in real time simulations to choose those factors right [13]. Some important factors that can slow down the simulation in Simulink are:

- The model includes a Matlab function block. This makes that Simulink need to call this function in every time step. To get rid of this problem it is better to use the built in function block or elementary math block if it is possible.

- The model includes an M-file S-function. M-file S-function also forces Simulink to call this function in every time step. Converting the M-file S-function to a C-MEX file S-function can solve this problem.

- The accuracy for the solver is set too high. If it is possible without losing to much information in the simulation, the accuracy of the solver should be as low as possible.

- The solver may not be the right for this simulation problem. The solvers that are available are described in [13].

- The model uses sample times that are not multiples of each other. The solver then has to take small enough steps to ensure sample time hits for all sample times.

5.4.12 Matlab Engine

Matlab engine makes it possible for Matlab to communicate with other program [14]. In this thesis Matlab is communicating with the Gateway program that is written in C++. On Windows system the communication is made thought ActiveX. In the Matlab engine library there are sets of routines that allow other programs to access Matlab engine. These routines can be found in table 5.3.

Table 5.3. Matlab Engine functions.

function	**purpose**
engOpen	Start up Matlab engine
engClose	Shut down Matlab engine
engGetArray	Get a Matlab array from the Matlab engine
engPutArray	Send a Matlab array to the Matlab engine
engEvalString	Execute a Matlab command
engOutputBuffer	Create a buffer to store Matlab text output

6 Minimizing delays seen by the controller

In this chapter improvements made to minimize delays seen from the controller are described. Here are also some theories about CPU time usage in Control Builder, Soft Controller and OPC-MMS Server described. The method of how to measure the variable delay is also described in this chapter.

6.1 CPU time usage

Soft Controller is executed under Windows 2000 with real time priority. Sine Soft Controller executes under Windows 2000, and other processes also shall have access to the CPU time, the part of the CPU that the Soft Controller can use is adjustable. In normal case the Soft Controller uses 30% of the CPU time. Of these 30%, 50% is used to the precision task and 20% to normal tasks. The CPU time split in the normal case is shown in figure 6.1. To change the time allocated to the Soft Controller a system variable CpuTimeQuota can be set to values between 30-100% [16].

The actual CPU usage is measured by the subsystem called Non Satt-Line Activities (NSLA). This subsystem calculates the sleep time for the Soft Controller process.

The same idea, which is described for Soft Controller, with the system variable CpuTimeQuota is also used in OPC-MMS Server and Control Builder, but with the difference that the OPC-MMS Server and the Control Builder executes under normal priority.

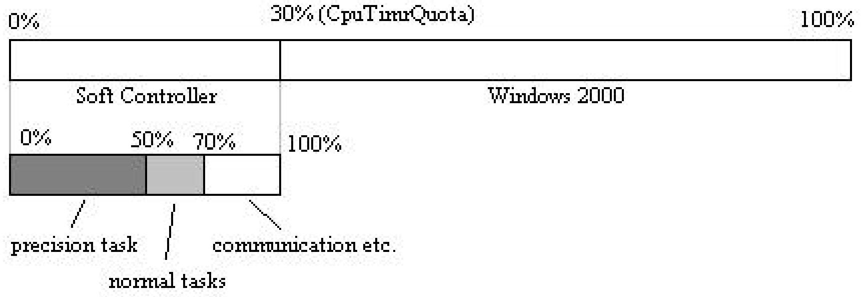


Figure 6.1 CPU assignment for Soft Controller in Windows 2000 based control system.

6.2 Delay test

In this section the tests made to verify delays in the communication between the controller and the models in Simulink are described. As can be seen in chapter 3 there are lots of steps where the signal can be delayed. In these tests all delays, except big computation delays that can appear with big projects in the Soft Controller, or with big process-models in Matlab/Simulink are included.

The main idea of this test is to send a sawtooth signal from the signal generator in figure 6.2 and then let the signal pass through, in the following order: Gateway, OPC Server, Soft Controller, OPC Server, Gateway and back again to the Simulink model, where the sawtooth signal was sent out from. In this way the phase of the outgoing signal can be compared to the phase of the incoming signal. The difference in phase between these two signals is approximately the delay in the communication link. If the result should be reliable, it is important that the jitter measured in the real-time block is under 1. This is due to that the diagram plotted in Simulink uses the time from the real-time block.

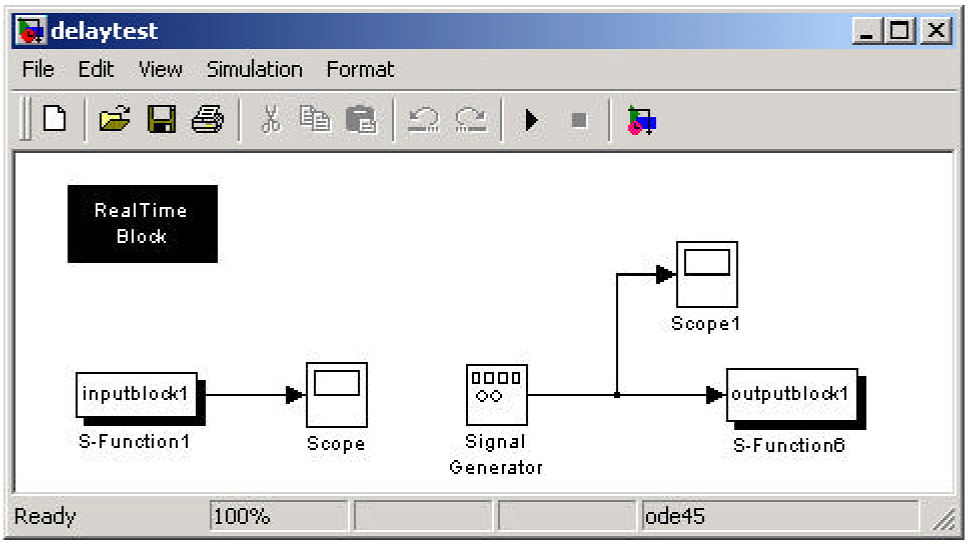


Figure 6.2. Model in Simulink for delay test of the communication between the controller and Matlab/Simulink.

To make the signal pass through the Soft Controller in quickest possible way, for this test, the controller is built up as in figure 6.3. The sample time for the OPCIn and OPCOut block should be as low as possible. The sample time in the Soft Controller has to be tested so that the Soft Controller does not interrupt the other processes too often. If long sample time is used in the Soft Controller, it is important to note that the delay that is measured is not only the delay in the communication link, but also the delay in the Soft Controller. The choice of sample time in the Soft Controller is dependent on how many computers that are used, and how fast each computer is.

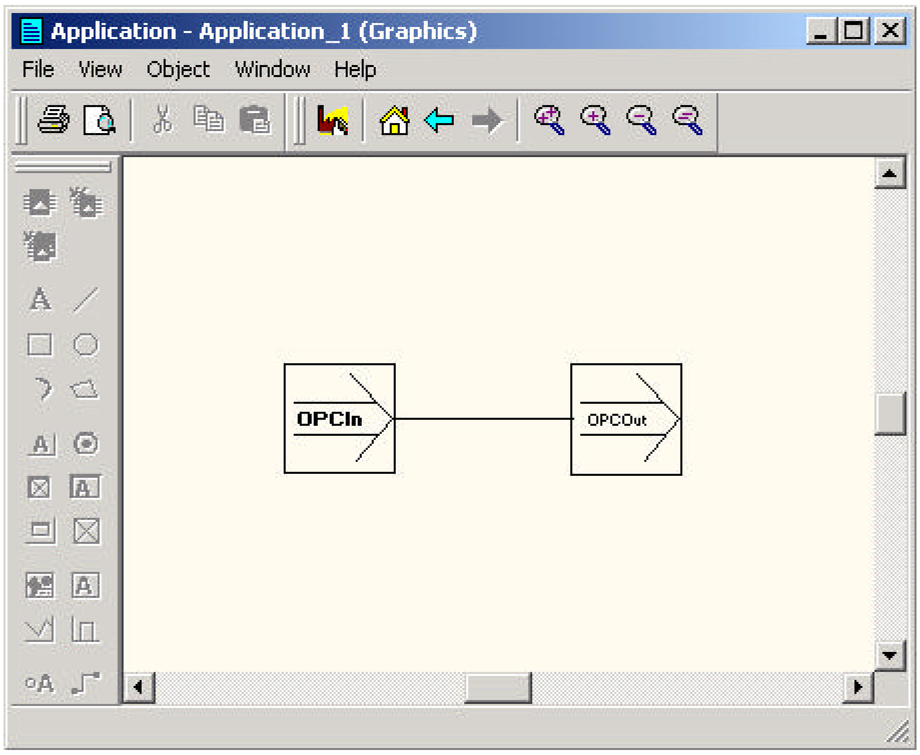

Application - Application_1 (Graphics)
File View Object Window Help
OPCIn
OPCOut

Figure 6.3. Function chart in Control Builder. The OPC variables are sent directly from OPCIn to OPCOut. The only possible delay is the sample time for the application.

The delay tests are used in the chapters 7, 8 and 9.

6.3 Improvement of the real-time performance in Matlab

If the system timer is used for measurement of the time in the real-time block, the lowest sample time that can be used is limited by the resolution of the system timer. To make it possible to use lower sample times a multimedia timer can be used. The timers are described in chapter 5.2. In figure 6.4 a very simple model is shown that has been used to compare the performance of system timer and multimedia timer. Later a little bit more complex models are used for test of the timers.

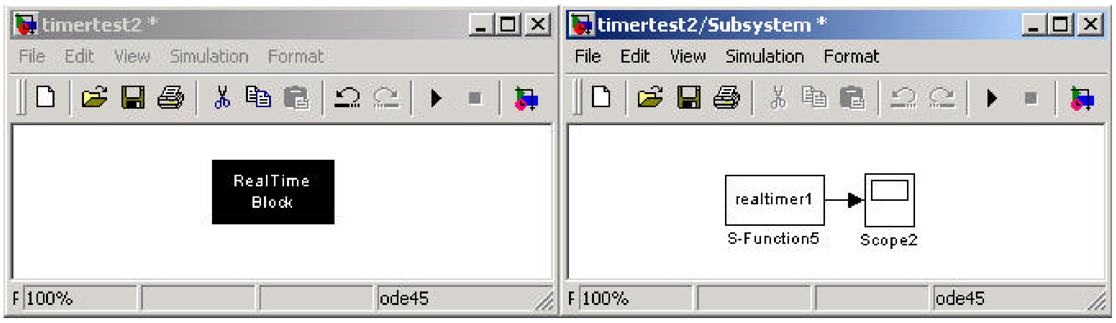


Figure 6.4. Simple model for timer test. Only the real-time S-function is used. Left figure shows the real-time block used in all the simulation tests. The right figure shows how the real-time block looks inside.

The first test is made with a system timer and a multimedia timer with the sample time 10ms. In figure 6.5 the jitter using system timer is shown and in figure 6.6 the jitter using multimedia timer is shown. When the system timer is used the resolution of 10ms in the timer makes the jitter vary from 0-1, which is shown in figure 6.5. When the multimedia timer with resolution 1ms is used, instead of the system timer, the jitter is getting lower.

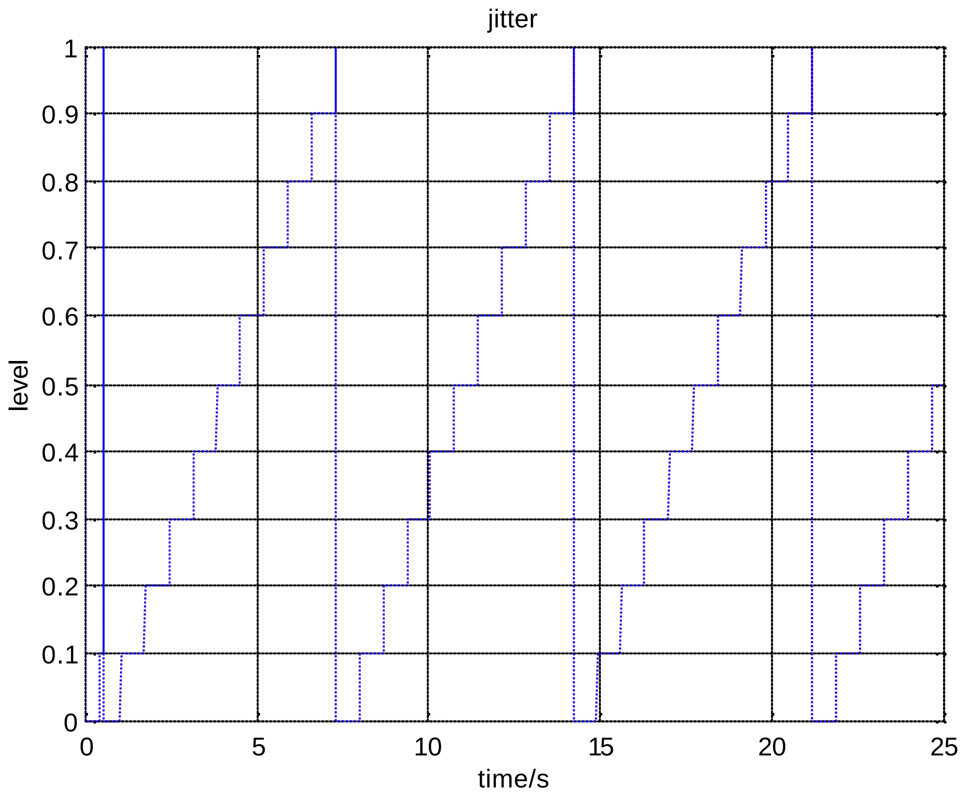


Figure 6.5. Jitter while system timer with resolution 10ms and a sample time of 10ms is used. The diagram comes from the scope block in figure 6.4.

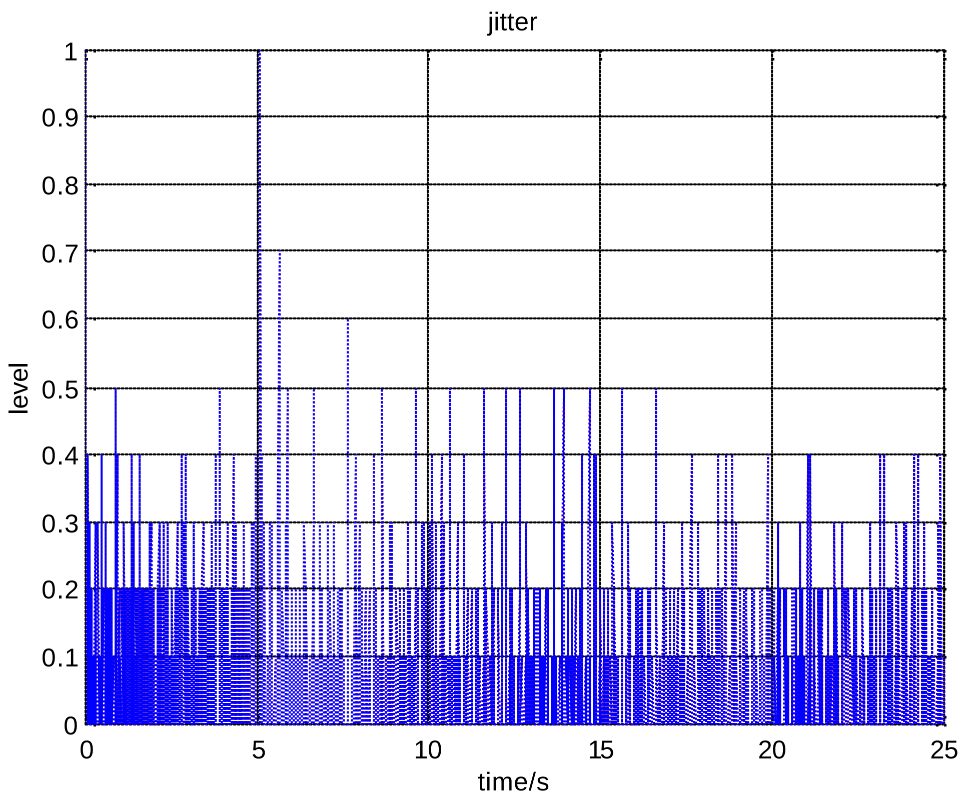


Figure 6.6. Jitter while multimedia timer with resolution 1ms and a sample time of 10ms is used.

In the following test the Soft Controller and the communication between the Soft Controller and Matlab/Simulink is also included. The model used in Simulink is the one shown in figure 7.3. This test is made to see how the system-timer and the multimedia-timer perform in the environment it will be used in. In figure 6.7 the jitter from a real-timer function that uses the system timer is shown. In figure 6.8 the same simulation is made, but a multimedia timer is used instead of the system-timer.

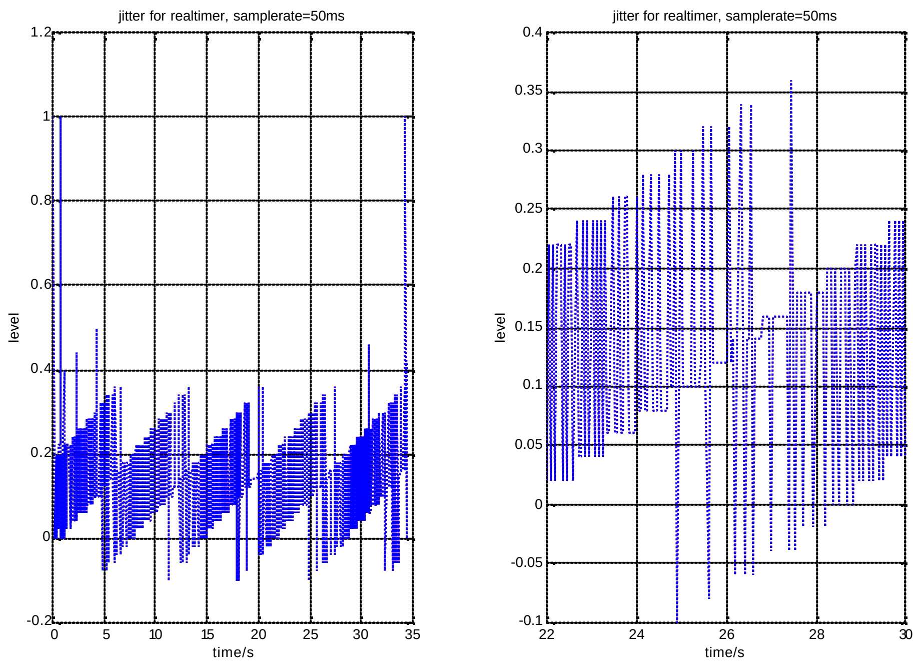


Figure 6.7. Jitter from a real-timer function that uses the Windows API function GetTickCount(). The sample time is 50ms.

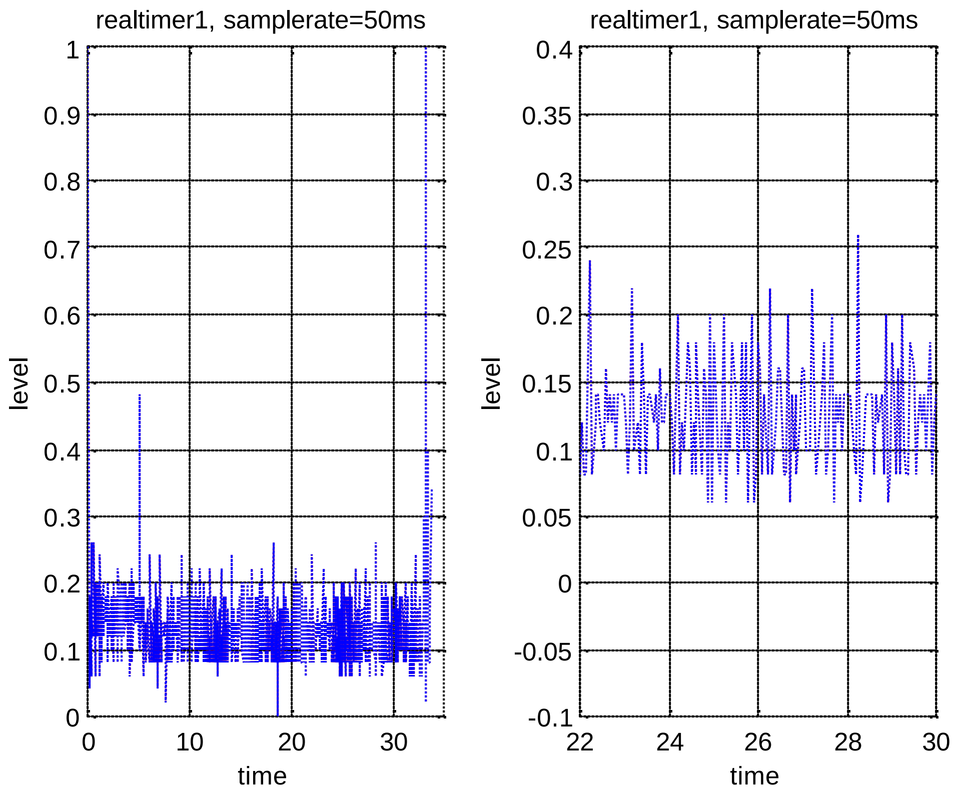


Figure 6.8. Jitter from a real-timer function that uses the Windows API multimedia timer with 1ms resolution. The sample time is 50ms.

When all the processes for control, communication and computation of the process model uses the same CPU, the priority of the processes can make a big difference in the performance of the simulation. Therefore some tests with other priority than the default priority have been made. It is of course not easy know what is going to happen if the priorities for some of the processes are changed, and the priorities for some other processes that use the same CPU are not changed. The choices of priorities are very complex in this simulation system since the processes are not synchronized. The diagram from the tests, shown in the figure 6.9 and 6.10, are made as the latest described simulation, with the differences that the sample time is decreased to 10ms and that the base priority of Matlab is raised from normal to high priority. In figure 6.9 the system timer is used and in figure 6.10 the multimedia timer is used. In the left diagram of figure 6.9 and figure 6.10 the jitter is shown when Matlab runs in normal priority, and in the right diagram of those figures the jitter is shown when Matlab runs in high priority.

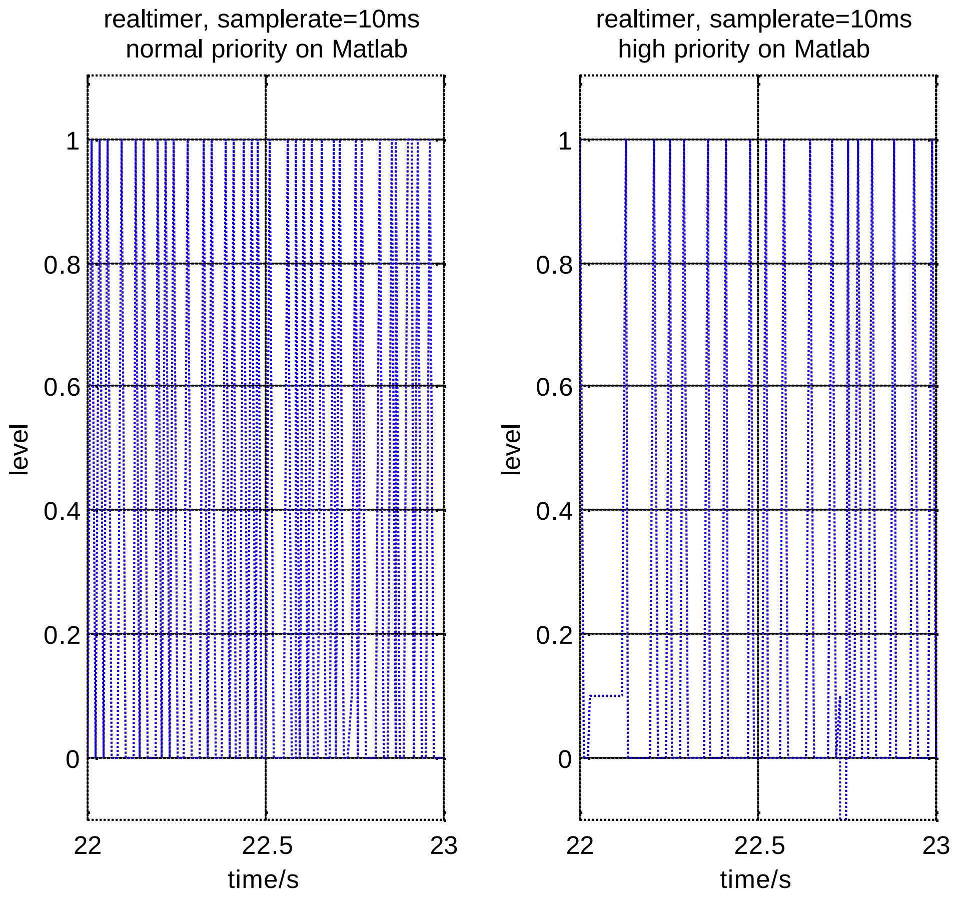


Figure 6.9. Jitter from a real-timer function that uses the Windows API function GetTickCount(). The sample time is 10ms. The left diagram is for Matlab in normal priority. In the right diagram is the process priority of Matlab raised to high priority.

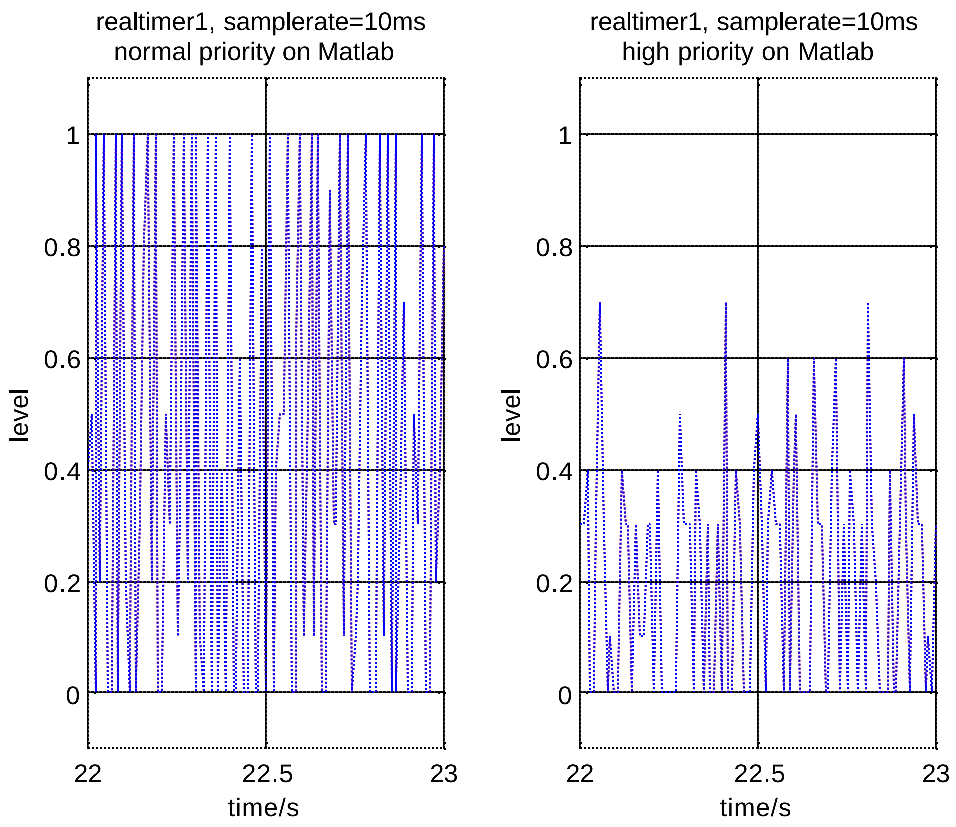


Figure 6.10. Jitter from a real-timer function that uses the Windows API multimedia timer. The sample time is 10ms. The left diagram is for Matlab in normal priority. In the right diagram is the process priority of Matlab raised to high priority.

It is not easy to make some conclusions about what is happening when processes get a higher priority than it has from the beginning. But with some well-known controllers and processes some practical tests of the simulation performance with different priorities of the processes can be made. This will of course not give maximized performance, but probably better performance than when base priorities are used. The reasons for the better real-time performance is that processes like Control Builder, OPC Server panel and some Windows processes that do not need to execute in real-time, only execute when there is some free CPU-time.

7 Control of function and performance of a existing Gateway program

In this chapter tests and verifications of a Gateway described in “Communication between Advant Control Builder and Matlab/Simulink”[2] is made and explained. The tests have been made on two computers connected via a 100MB Ethernet network. Computer1 was a Pentium II 300MHz with 128MB RAM. In this computer the Control Builder, the Gateway and Matlab were running. Computer 2 was a Pentium 133MHz with 64MB RAM, and here Soft Controller and OPC-MMS Server were running. The first part of this chapter is about the timer function that has to be placed in Simulink to get the process-model to work in real time.

7.1 Timer function

The timer function is an S-function that suspends the execution of the simulation model in Simulink. The suspension time is computed so that the chosen sample time follows the real time. This function has to be implemented because Simulink is not made for real-time simulations. The suspension is made with the Windows command sleep. This command suspends the thread in which the sleep function is in for the number of milliseconds that is written as the input parameter to the function. To get the sleep function to suspend the right number of milliseconds some time calculations have to be made. In figure 7.1 a simplified calculation loop for the timer function is shown and in figure 7.2 the indices used in figure 7.1 are shown.

A function used to get the time in the calculation is the Windows function GetTickCount described in chapter 5.2. This function retrieves the number of milliseconds that have elapsed since the system was started. The limitation of this function is the resolution of the system timer. In Windows NT the resolution of the system timer is approximately 10ms. The effect of resolution is clearly seen in figure 6.5. If higher resolution of the timer is needed, the recommendation is to use a multimedia timer or a high-resolution timer. More about this can be read about in chapter 5.2.

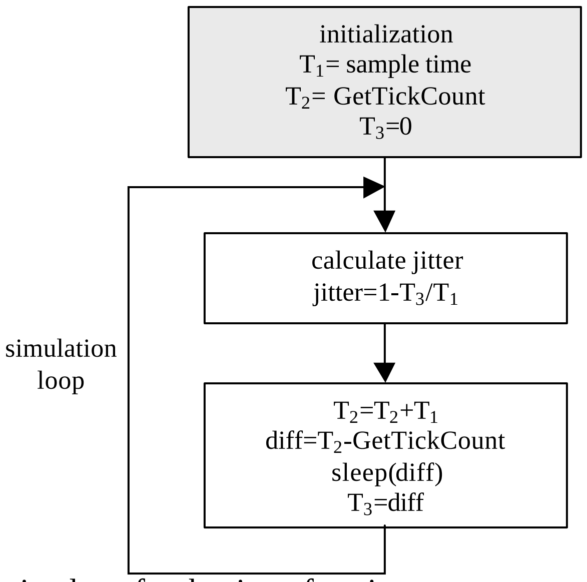


Figure 7.1. Simulation loop for the timer function.

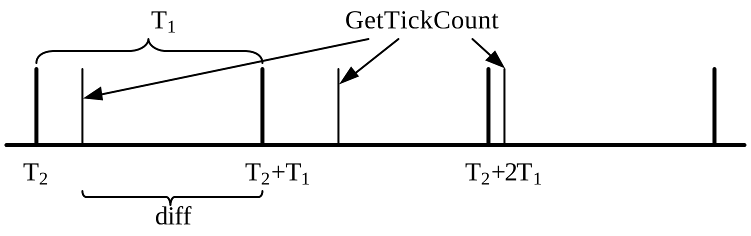


Figure 7.2. Explaination of indices used in figure 7.1.

7.2 Verification of functionality in the timer function

The verification model is built to verify the functionality when the controller, OPC-MMS Server and the process work together. To make the system work hard with different process models at the same time, the models are built in three different ways. The models are built as three S-functions that models three equal tanks, one model that looks like a continuous transfer function of a DC servo and one model that looks like a discrete state space description of a DC-servo.

There is a desire to choose the number of in-and outputs independently, but when this test was made the maximum of in-and outputs are five. This number is limited by the Gateway when this test was performed. In the test all the inputs and all the outputs were connected at the same time to verify that it would not cause any problem when more than one input and one output are connected at the same time.

The process models are completely separated from each other in the test. Each process model has one PID controller each implemented in the Soft Controller. It would have been nice if the models were linked together in some way, but in that case it is difficult to make some good conclusions from the simulation. In figure 7.3 the process-models for the tests are shown in a picture from Simulink.

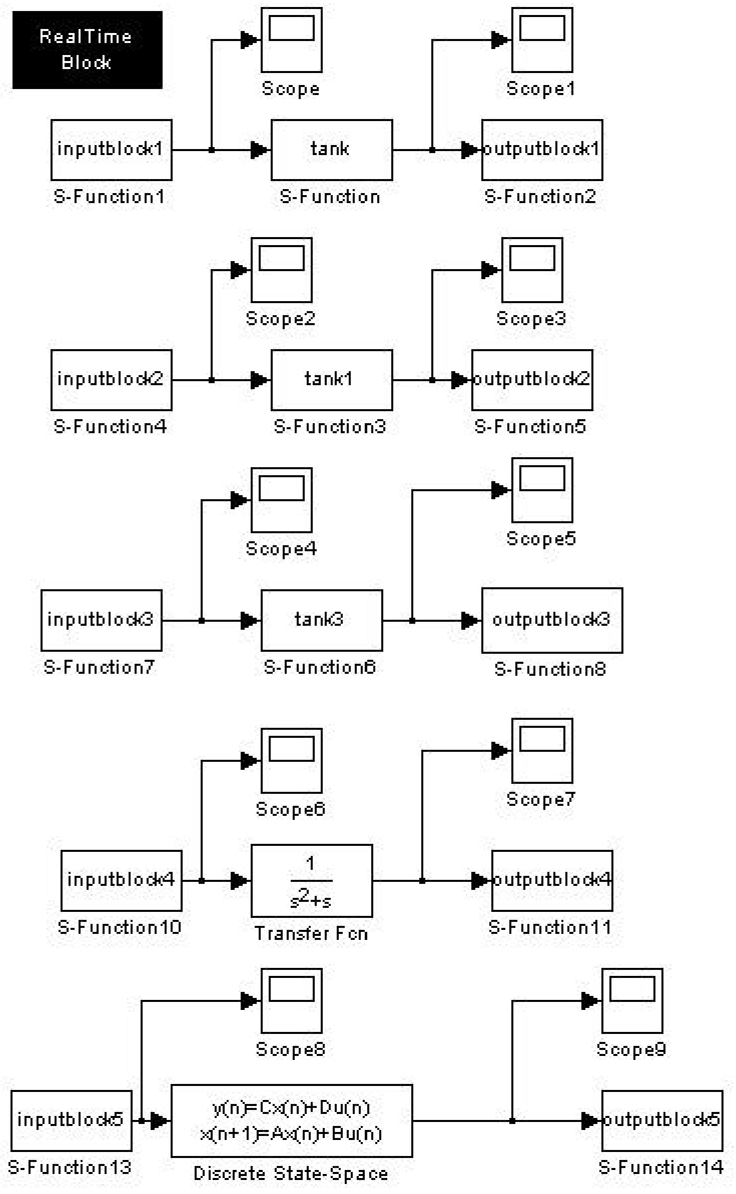


Figure 7.3. Test system made to verify the functionality of the Gateway program.

The three process models on the top of figure 7.3 are exactly the same process models, built as S-functions. Also the three controllers for these process models are exactly the same. The reason for this is just that it was a desire that all in- and outputs should be used in the test.

To have some signals to compare, plots have been made from the signals in Matlab and in Control Builder. In figure 7.4 to figure 7.13 the control signal and the process values for the tree different process models are plotted. The difference between the plots in Control Builder and Matlab are the places were the signals are measured for the plots. The differences of measurement points are due to that the OPC-MMS Server and the Gateway are connected between the Soft Controller and Matlab. This communication causes a delay between the measurement points.

Data for the tree tank models in figure 7.3 are here shown.

height=0.4 [m]
radius tank=0.03 [m]
sample time=10 [ms]
flow in=control signal from controller [m^3/h]
flow out=(area out)*(tank level)$^{1/2}$ [m^3/h]

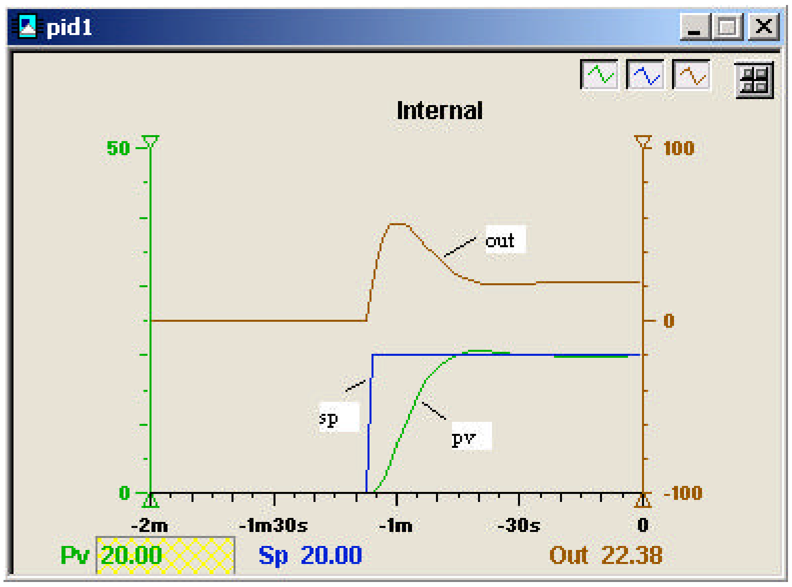


Figure 7.4. Diagram from Control Builder that shows the signals at the PI-controller when a tank made as an S-function is the process model. The controller parameters are: Gain=5.85, Ti(s)=7.96.

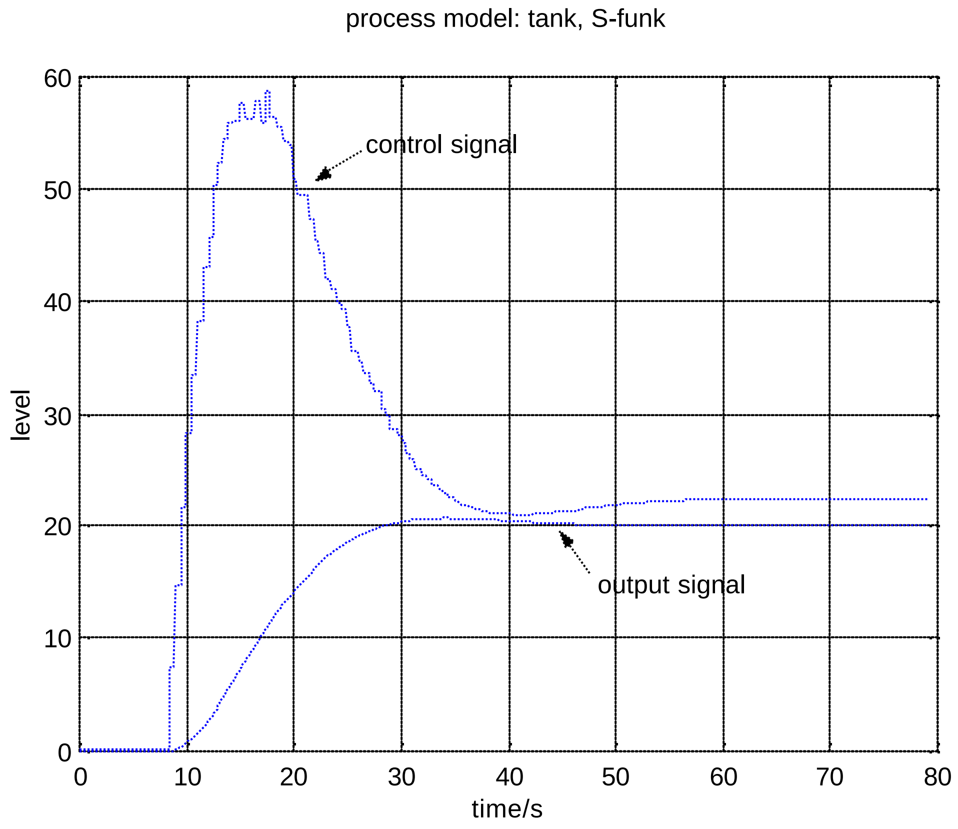


Figure 7.5. Diagram from Matlab that shows the signals at the process model of a tank, when a PI-controller is used as controller. The difference between this figure and figure 7.4 is that the Gateway and the OPC-MMS Server are placed between the controller and process model, which gives different measurement points.

To be able to make some conclusion about what is happening, the control signals from the figures 7.5, 7.9 and 7.12 have been zoomed, and are shown in the figures 7.6, 7.10 and 7.13. In those figures the various sample time, described later in chapter 7.3 can be seen.

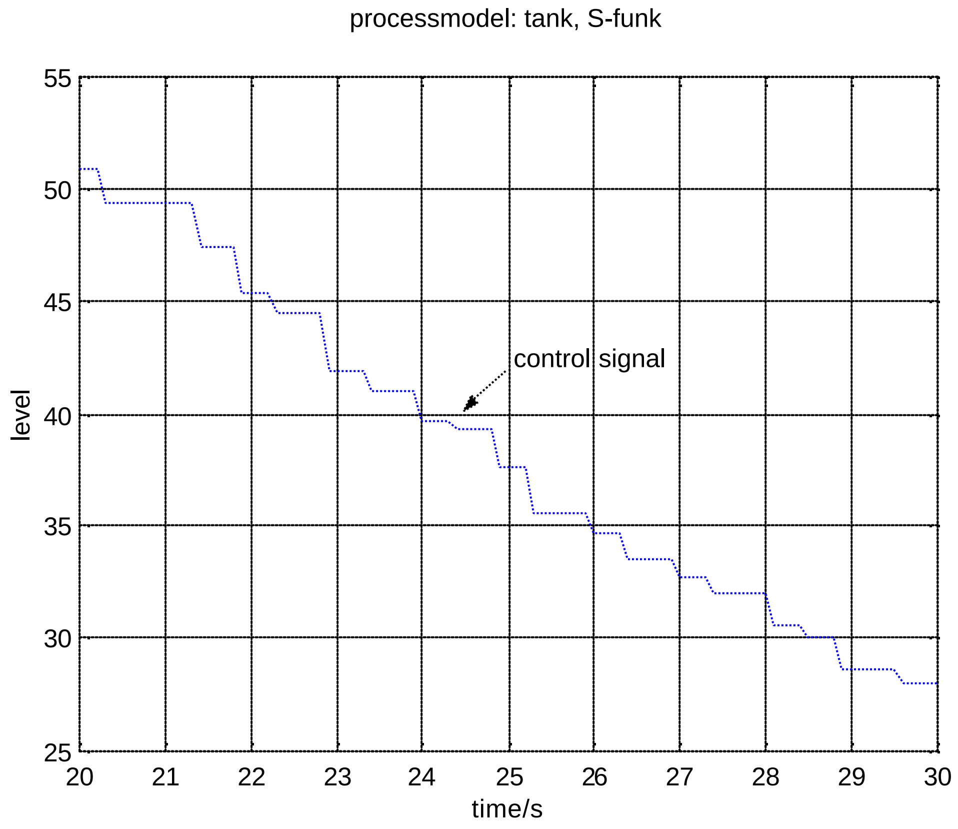


Figure 7.6. Zoomed part of the control signal from figure 7.5. In this figure the level of different steps and the variable sample time are shown clearer than in figure 7.5.

A DC-servo is used both as the continuous- and discrete process model in the Simulink test model shown in figure 7.3. The result is shown in figure 7.8 to figure 7.13. The model of the normalized DC- servo process is shown in figure 7.7 and described with the equations (6.1) to (6.8).

Torque balance for a general rotating system:

$$J\ddot{q} + D\dot{q} + Kq = M \tag{6.1}$$

J is the moment of inertia, D is the damping coefficient and K is the spring coefficient. θ is the angle position and M is the total torque acting on the system. In these tests the spring coefficient is set to zero. The torque equations for the electrical motor is:

$$M = k_e \cdot u(t) \tag{6.2}$$

where u(t) is the voltage to the motor and k_e is a motor constant. The continuous state space description is then given by:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -\frac{D}{J} & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} \frac{k_e}{J} \\ 0 \end{bmatrix} u(t) \tag{6.3}$$

In the tests the coefficients D, J and k_e are chosen in such way that the continuous transfer function becomes:

$$G(s) = \frac{1}{s^2 + s} \tag{6.4}$$

A block diagram over the DC servo is shown in figure 7.7.

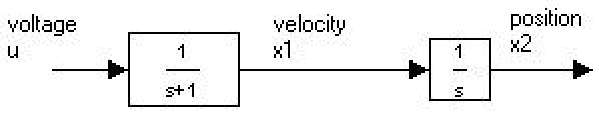


Figure 7.7. Block diagram over the DC motor process used in the verification tests.

Sampling the normalized continuous state space model makes the discrete state space model. The sampling is made with:

$$\begin{aligned} \Phi &= e^{Ah} \\ \Gamma &= \int_0^h e^{As} ds B \end{aligned} \tag{6.4}$$

$$\begin{aligned} A &= \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \\ B &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ C &= \begin{bmatrix} 0 & 1 \end{bmatrix} \\ D &= 0 \end{aligned} \tag{6.5}$$

Laplace transform of e^{Ah} gives:

$$\begin{aligned} L(e^{Ah}) &= (sI - A)^{-1} = \begin{pmatrix} \frac{1}{s+1} & 0 \\ \frac{1}{s(s+1)} & \frac{1}{s} \end{pmatrix} \\ \Rightarrow \Phi &= L^{-1}(e^{Ah}) = \begin{pmatrix} e^{h} & 0 \\ 1-e^{h} & 1 \end{pmatrix} \end{aligned} \tag{6.6}$$

Γ then becomes:

$$\Gamma = \int_0^h e^{As} ds \cdot B = \int_0^h \begin{bmatrix} e^{-s} \\ 1-e^{-s} \end{bmatrix} ds = \begin{pmatrix} 1-e^{-h} \\ h-1+e^{-h} \end{pmatrix} \tag{6.7}$$

The discrete state space model is then:

$$\begin{aligned} x(kh+h) &= \begin{bmatrix} e^{-h} & 0 \\ 1-e^{-h} & 1 \end{bmatrix} x(kh) + \begin{bmatrix} 1-e^{-h} \\ h-1+e^{-h} \end{bmatrix} u(kh) \\ y(kh) &= \begin{bmatrix} 0 & 1 \end{bmatrix} x(kh) \end{aligned} \tag{6.8}$$

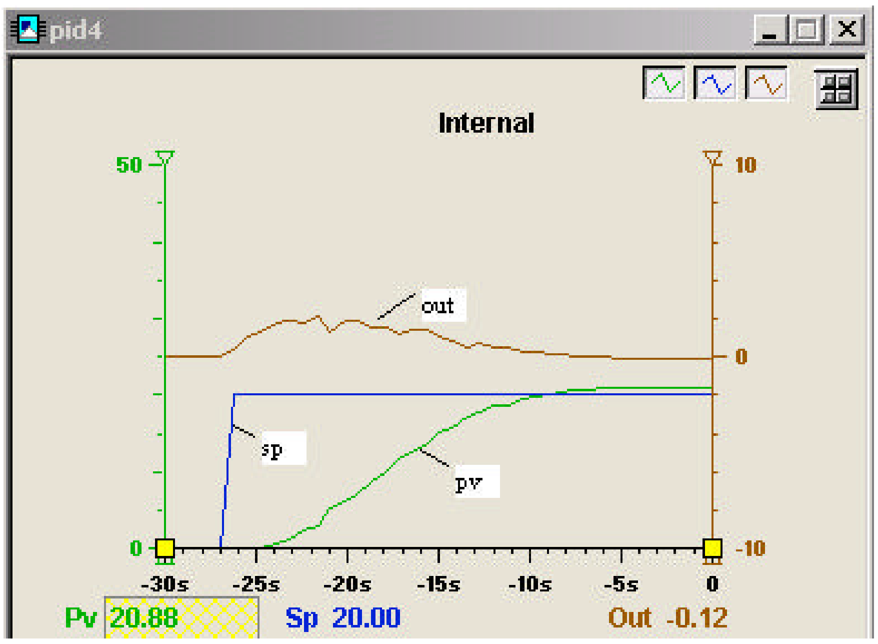


Figure 7.8. Diagram from Control Builder that shows the signals at the PID-controller when a continuous transfer function of a DC-Servo with TF=1/(s^2+s) is used as the process model. The controller parameters are: Gain=0.15, Ti(s)=8.10, Td=1.30, Der filter time=0.16.

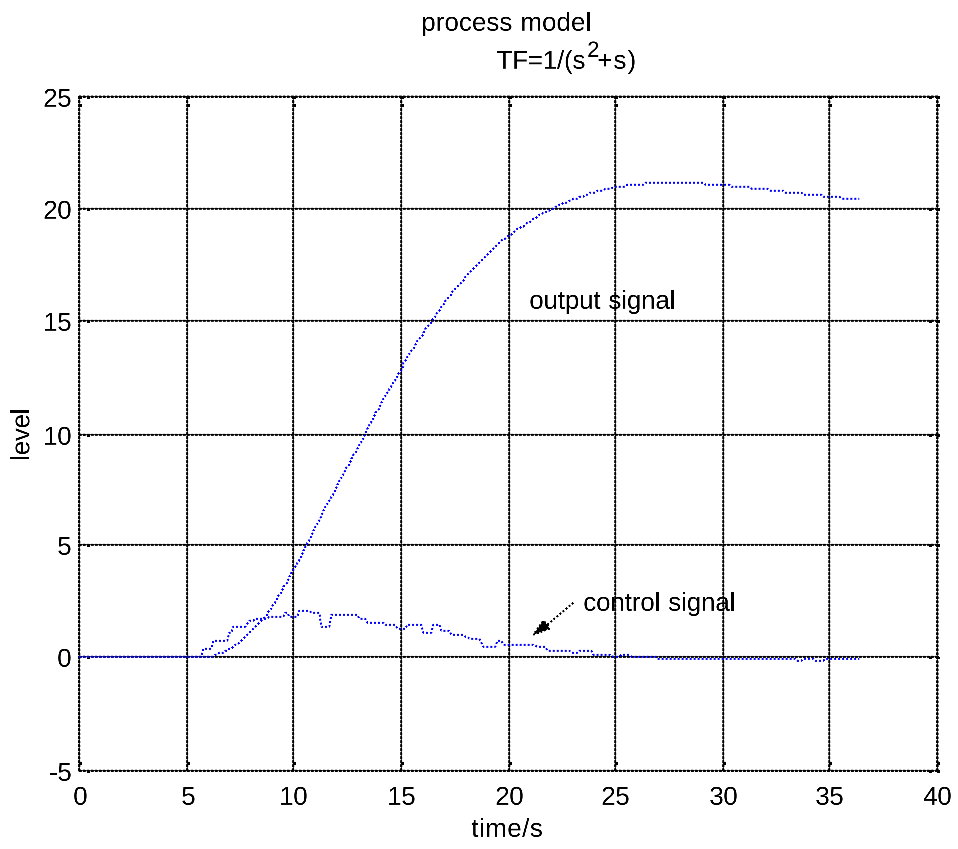


Figure 7.9. Diagram from Matlab that shows the signals at the continuous process model of a DC-Servo when a PID-controller is used as controller. The difference between this figure and figure 7.8 is that the Gateway and the OPC-MMS Server are placed between the controller and process model, which gives different measurement points.

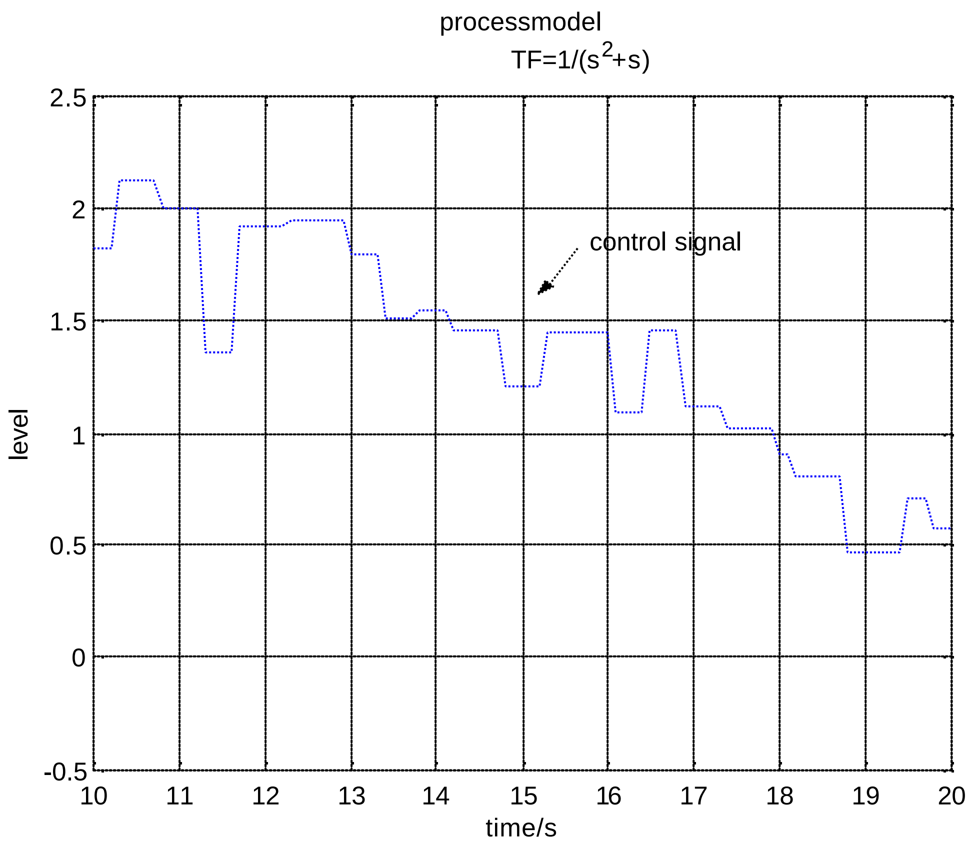
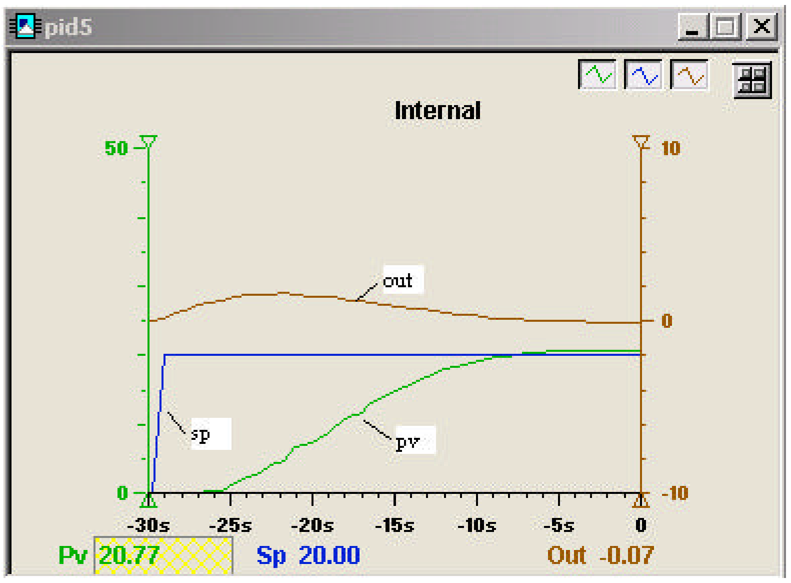


Figure 7.10. Zoomed part of the control signal from figure 7.9. In this figure the level of different steps are shown clearer than in figure 7.9.



Pv 20.77 Sp 20.00 Out -0.07

Figure 7.11. Diagram from Control Builder that shows the signals at the PID-controller when a discrete state space description with sample time h=100ms is used as the process model. The controller parameters are: Gain=0.15, Ti(s)=8.10, Td=1.30, Der filter time=0.16.

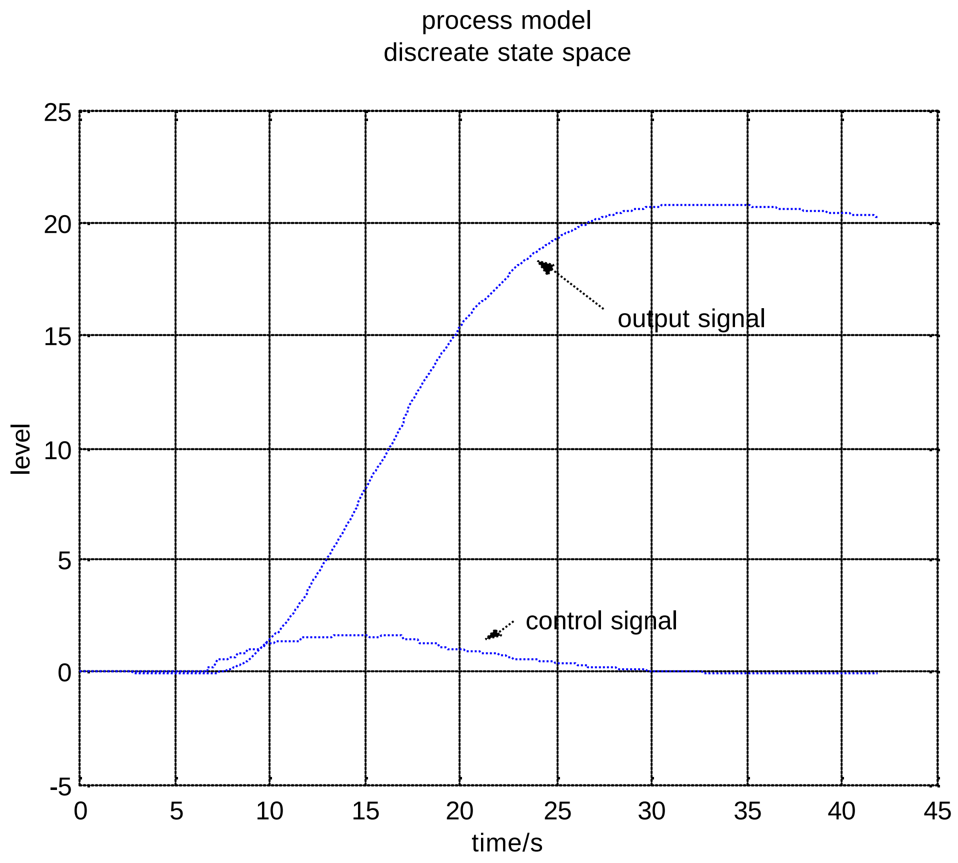


Figure 7.12. Diagram from Matlab that shows the signals at a discrete process model of a DC-Servo when a PID-controller is used as controller. The difference between this figure and figure 7.11 is that the Gateway and the OPC-MMS Server are placed between the controller, which gives different measurement points.

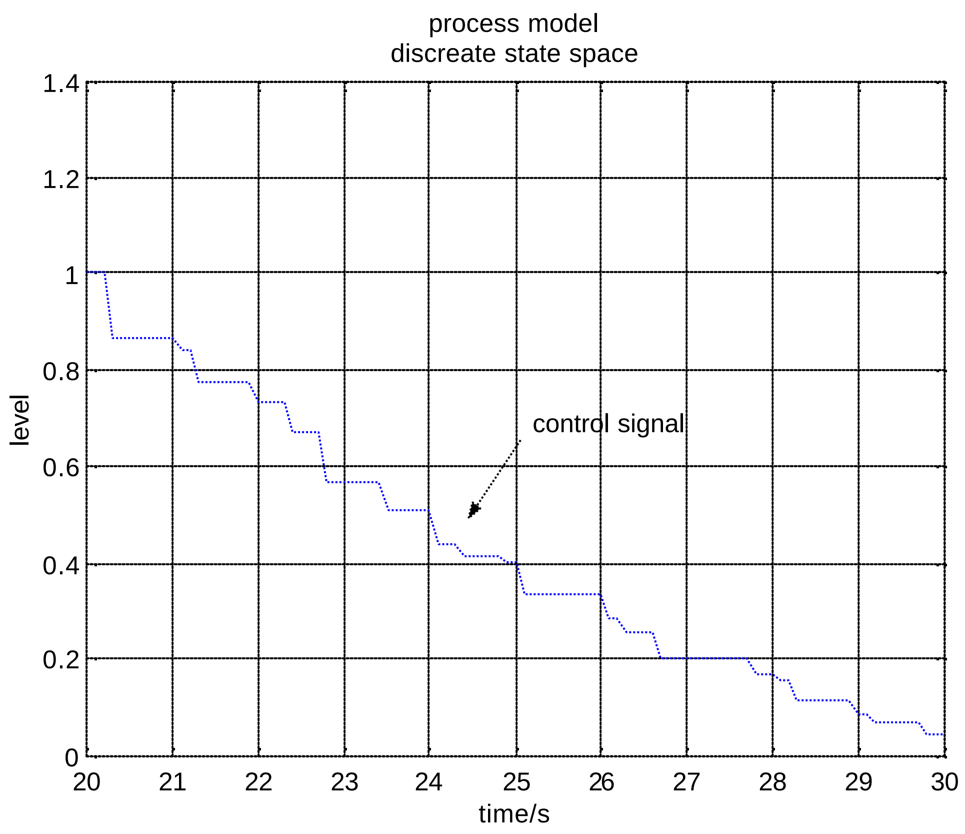


Figure 7.13. Zoomed part of the control signal from figure 7.12. In this figure the level of different steps and the variable sample time are shown clearer than in figure 7.12.

In the following two figures 7.14 and 7.15 the jitter during the simulation is shown. The jitter is a measurement of how well the simulation works with the chosen sample time. More about jitter is described in the chapter 6.3.

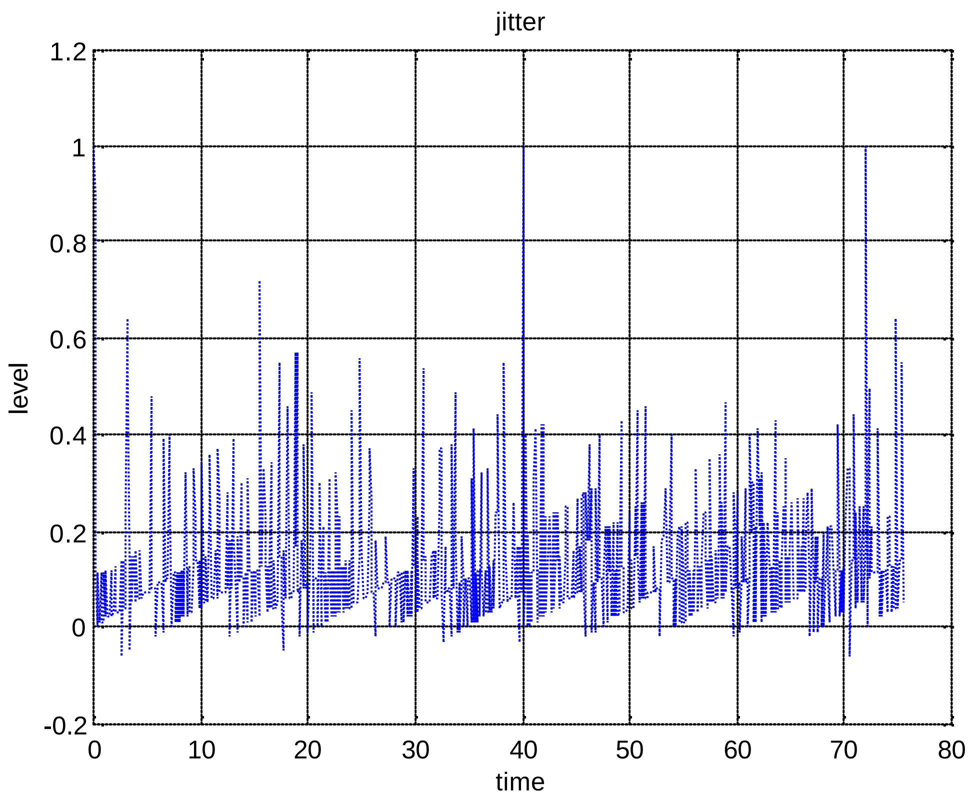


Figure 7.14. This figure shows the jitter when the model is the one in figure 7.3. The sample time is 100ms.

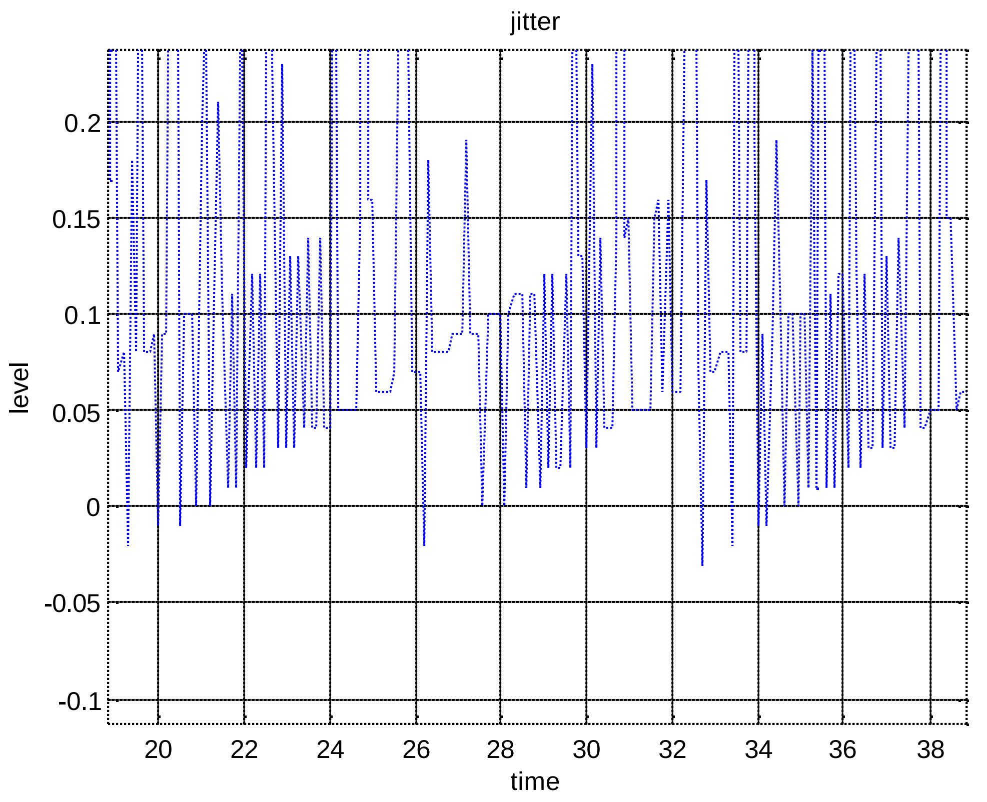


Figure 7.15. Zoomed part of figure 7.14. In this figure a trend can be seen. The jitter locks like a sawtooth signal with the amplitude of 0.1.

7.3 Sample time

There are five steps where the signals are sampled outside the controller. The steps are the one in the process model, two in the OPC-MMS server and two in the Gateway. A block diagram over the steps, where the signals are sampled, is shown in figure 3.2. The bottleneck of this system is the update rate for the OPC-MMS server. Since ABB’s OPC-MMS server is not made for use in fast time critical applications, it has a minimum update rate that is set to 250ms. This sample time is not exact, it depends on the CPU-load and the number of variables that the OPC-MMS Server serves. The update rate for the OPC-MMS server is not easy to check continuously, but it is possible to see the actually update rate as snapshots in the dialog window for the connection of the OPC-MMS server. In the test made for this application the actual update rate oscillates between 200-300ms when the update rate is set to 250ms.

The sample time of the different parts in the system have to be chosen so that the controller has a sample time slower than the slowest update rate in the OPC-MMS Server. This selection of sample rate is used to avoid that there is the same value on two samples in the controller, when it not should be that. The best choice of sample time in the controller is to choose the controller to be as slow as possible. This is because the quotient delay/(sample time) between the model and the controller is then at its minimum. The choice of sample time for the Simulink process model is best if the sample time is chosen as low as possible. The limitation of the sample time in Simulink is the jitter described in chapter 6.3.

In figure 7.16 the main delays in the system are shown. The ‘delay 1’ is the delay related to the difference in sample time between the OPC-MMS Server and the process model in Simulink. The ‘delay 2’ is related to the difference in sample time between the OPC-MMS Server and the controller. According to that the OPC-MMS Server has not completely stable sample time and that the six steps where sampling are made, including the sampling in the Soft Controller, are not synchronized, there will be differences in the delay. The ‘delay 1’ will have a delay that moves between zero and the sample time of the process model in Simulink. Delay 2 will have a delay that moves between zero and the slowest sample time of the OPC-MMS Server.

In the verification test the sample times that are used are 100ms for the process model, 250ms for the OPC-MMS Server and 500ms for the PID controller in the Soft Controller. According to that the sample time of the OPC-MMS Server varies between 200 and 300 ms, the maximum delay in this test should be twice the maximum ‘delay 1’=100ms plus twice the maximum ‘delay 2’=300ms. This gives the maximum total delay for this test to be 800ms i.e. more than one sample of the controller. There are more delays that have not been taken into consideration when the theoretical delay was computed. One of these delays is the one in the Gateway. The delay in the Gateway is not easy to check, since the update rate for the Gateway is set as fast as possible.

A delay is of course never a god behavior, and a varying delay between 0-800ms is of course not better. In tests with controllers that do not take into consideration delays in the process the conclusions of the simulations can be completely wrong. In this simulation system there is no easy way to check the delay in any of the simulation steps. This also makes it difficult to use controllers that can control processes with delays.

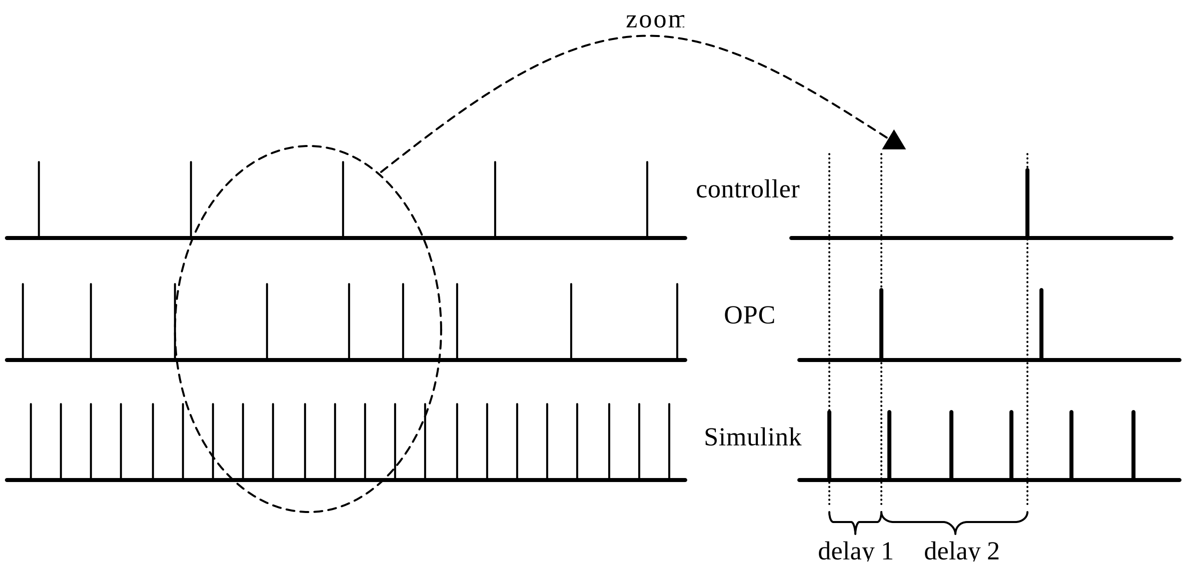


Figure 7.16. The figure shows two delays in the system with a controller, OPC-MMS Server and Simulink.

7.4 Measured delay

The measurement of the delay is made with the method described in chapter 6.2. The left diagram of figure 7.17 shows the in- and output signals from Simulink over some period of the sawtooth signal, and the right diagram shows the jitter over the same period. In figure 7.18 a shorter time span of the signals from the left diagram in figure 7.17 is shown. In figure 7.18 the delay of the system is clearly shown. The delay has been graphically measured in a diagram that looks like the right diagram in figure 7.17, but with the difference that the diagram was made much larger than the diagram shown in this thesis. The delay is measured with one low and one high limit. This measurement gives a delay that varies between approximately 600-1200ms.

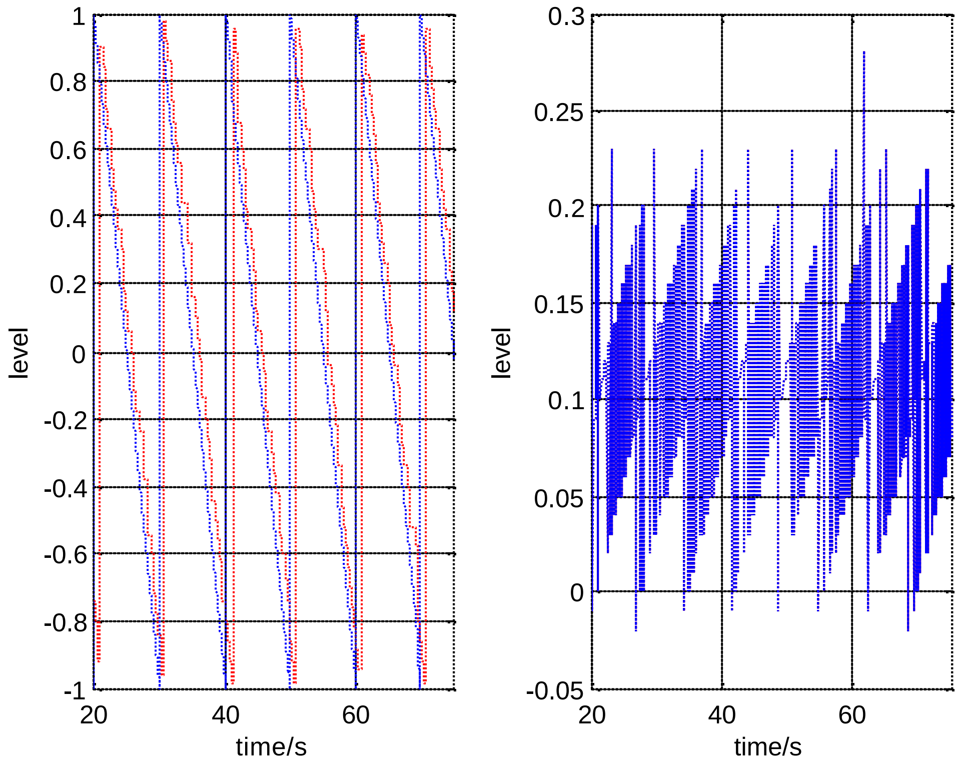


Figure 7.17. The left diagram shows the difference between in- and output signals in the delay test. The right diagram shows the jitter during the same delay test.

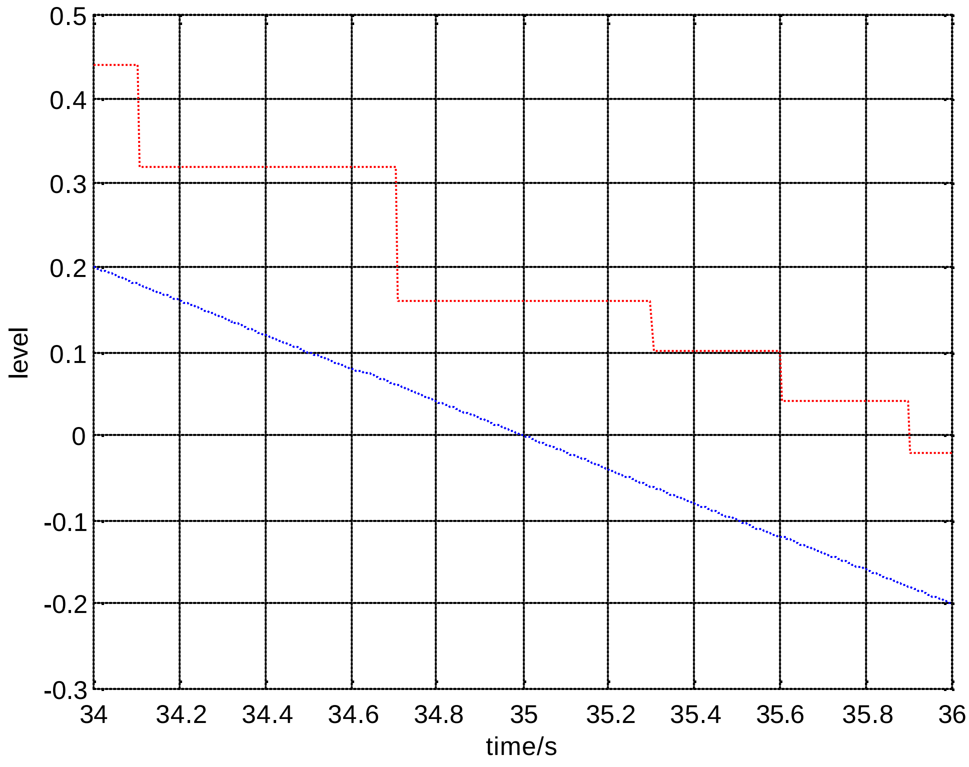


Figure 7.18. Part of the left diagram in figure 7.17 plotted with a smaller part of the time axis. The delay between the curves is clearly seen. The bottom curve is a part of the sawtooth

signal measured at the point where the signal is sent out, and the top curve is the curve of the signal when it comes back.

7.5 Conclusions about this test

The delay in the communication is very high in this test. The reasons for this big delay and that the delay varies are mainly due to the long sample times in the OPC-MMS Server and in Matlab/Simulink, and the fact that the sampling in the different steps in the simulation are not synchronized. The reason for the high sample time in the OPC-MMS Server is that in the commercial OPC-MMS Server the choice of sample time is low limited to 250ms. In Matlab the sample time is limited by the resolution of the timer function. In this case a system timer.

The conclusion about this test is that simulations should not be made as in this chapter, unless a delay over 1s is acceptable during the simulation.

To make the delay smaller the sample time in the OPC-MMS Server and in Matlab should be made smaller. This problem is described more in chapter 6,8 and 9. The variation of the sample time is more difficult to fix, but the variation will also be smaller if the sample time decreases.

8 Improved test on two computers

In this test a special OPC-MMS Server only made for this test is used. In this OPC-MMS Server there is a possibility to set smaller values of the update rate. The improved real-timer function described in chapter 7 is also used. Also some improvements of the computers are used. The computer improvements are that computer 1 has got more RAM, and computer 2 now is a computer with a Pentium Pro 200MHz processor and 128MB RAM. In this test the diagrams from the Control Builder are not shown. Because those diagrams only give the information that the signals at the Control Builder looks like the signals in Matlab/Simulink.

8.1 Data

In the tables 8.1 and 8.2 the data for the computers used in the simulation are described.

Table 8.1. Computer 1, Pentium II 300MHz, 224MB RAM

process	**priority**	**sample time/ms**
Control Builder	normal (default)	-
Gateway	high	high as possible
Matlab/Simulink	high	10

The Gateway is made to read values from Matlab/Simulink as fast as possible when there are some new values to read. This is the explanation to the sample time for the Gateway in table 8.1. The sample time for the Control Builder is not of interest in the tests made in this thesis.

Table 8.2 Computer 2, Pentium Pro 200MHz, 128MB RAM

process	**priority**	**sample time/ms**
Soft Controller (with PID)	Realtime (default)	100
OPC Server:	Normal	10

8.2 Test

The simulation shown here is a step response on the continuous DC-servo, shown in figure 7.3, made with a PID controller and a PI controller. In the left diagram of figure 8.1 the step response using a PID controller is shown and in the right diagram the jitter for the same simulation is shown.

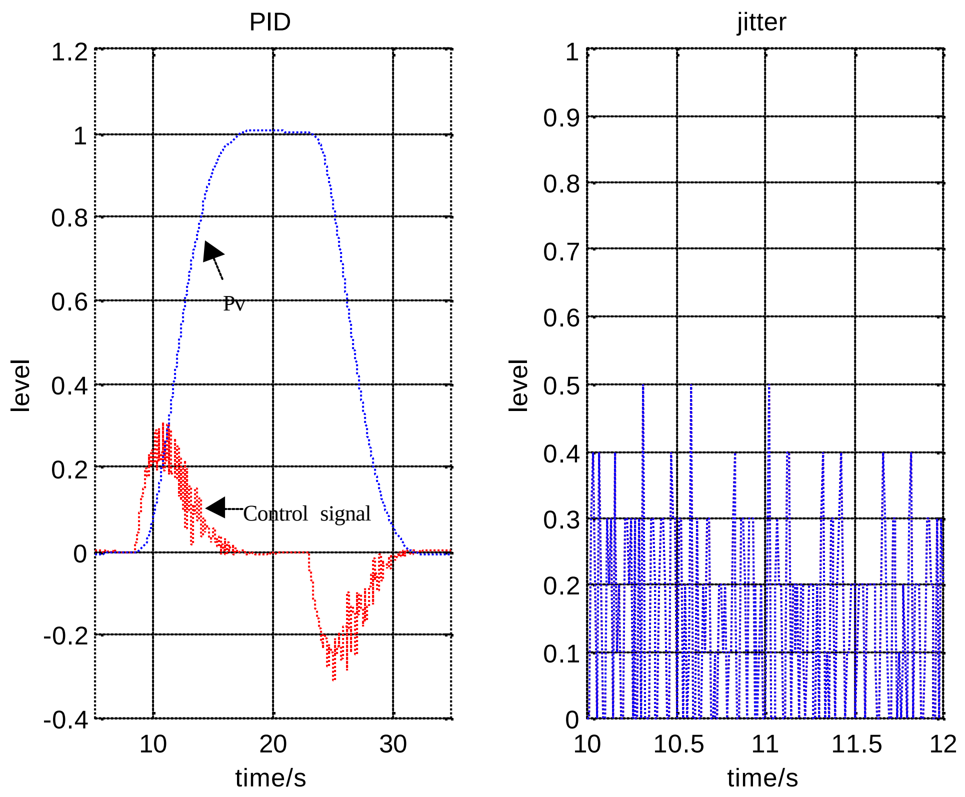


Figure 8.1. Left diagram shows the step response using a PID controller with gain=0.48, Ti=4.00, Td=0.64 and filter time=0.08s. Right diagram shows the jitter for a part of the time when the control signal is high.

It is interesting to look at the control signal while it is raising to see the sample time it has and how much variation in the sample time there is. It is easier to look at the same simulation, but without the derivative part in the controller i.e. a PI controller. Figure 8.2 and 8.3 show the same simulation as in figure 8.1, but without the derivative part in the controller.

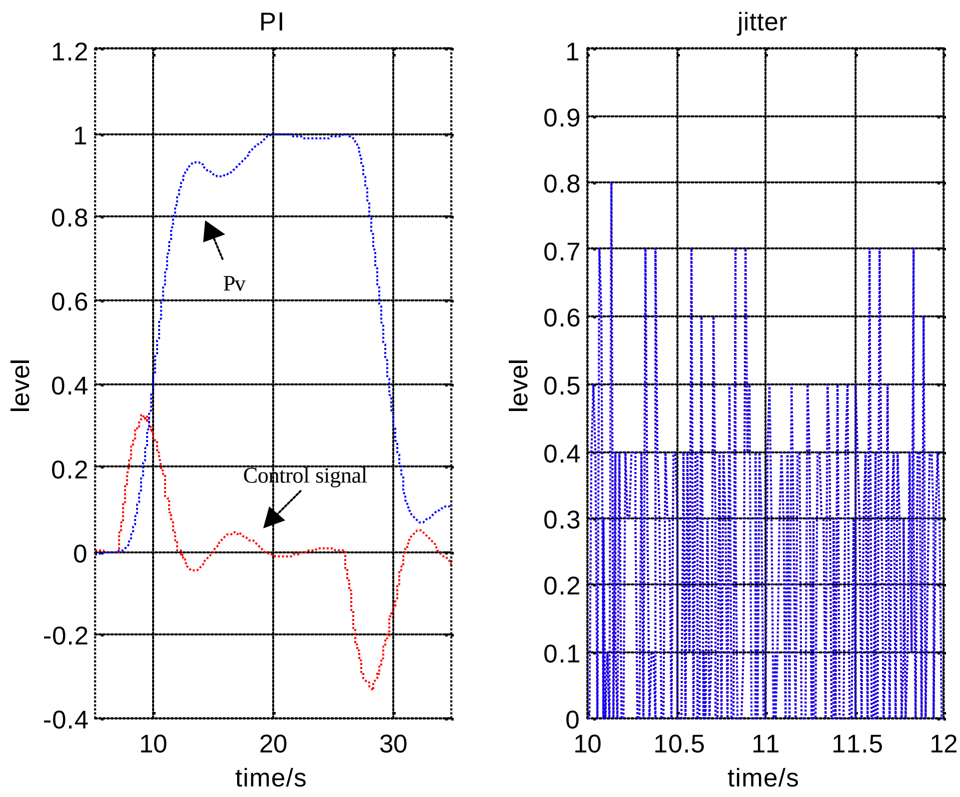


Figure 8.2. Left diagram shows the step response using a PI controller with gain=0.48, Ti=4.00. Right diagram shows the jitter for the some of the time when the control signal is high.

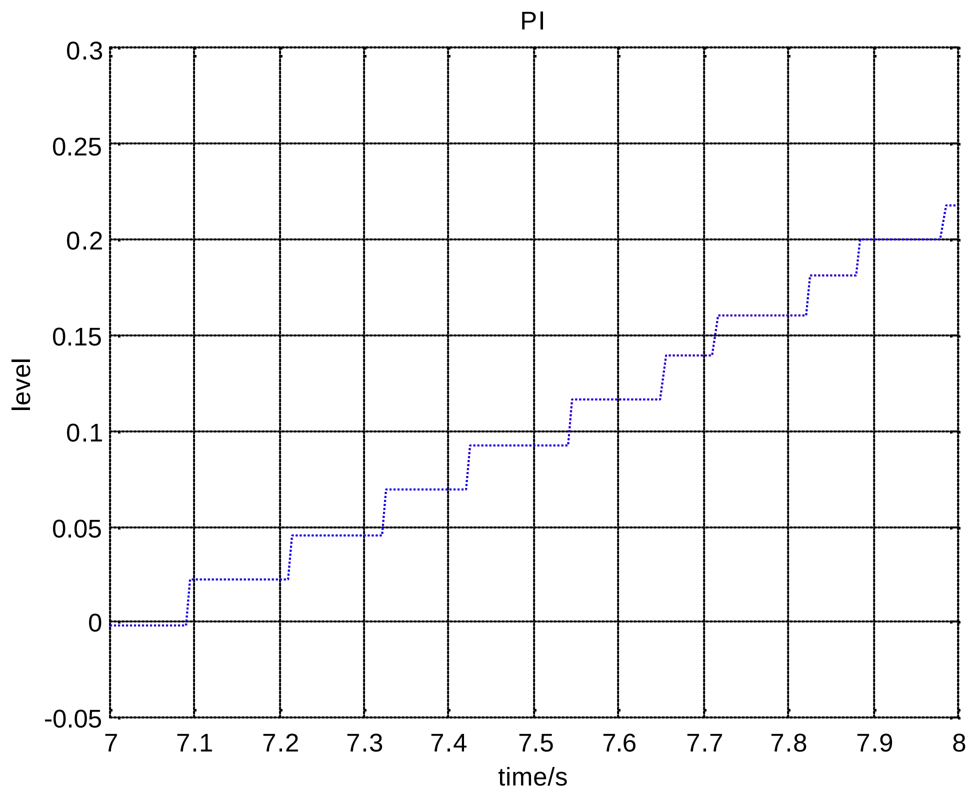


Figure 8.3. Zoomed part of the control signal from figure 8.2.

In figure 8.3 it can be seen that the sample time is around 100ms and that the variation of the sample time is between 50-130ms.

8.3 Delay

The delay test is made with the method described in chapter 6.2. In this test, which is made with two computers, the advantage is that the processes that have to run in real-time can be divided between two CPU's, and two processes can run at the exact same time. This means that less context switches has to be made at each CPU, and therefore higher sample times can be used in the OPC-MMS Server, Soft Controller and in the process model in Simulink without loosing information. The in- and output signals during the delay test are shown in figure 8.4 and figure 8.5.

The disadvantage by using two computers connected via an Ethernet connection on the network is the definition of Ethernet. The definition of Ethernet says nothing about real time reliability of the transmission of the signals that are sent between the computers in this test. In figure 8.4 some undeterministic delays can be seen that are probably due to the Ethernet connection. The delays are up to a few seconds, which is not a good behavior if the big delay appears for example when some step changes in the process signals are present.

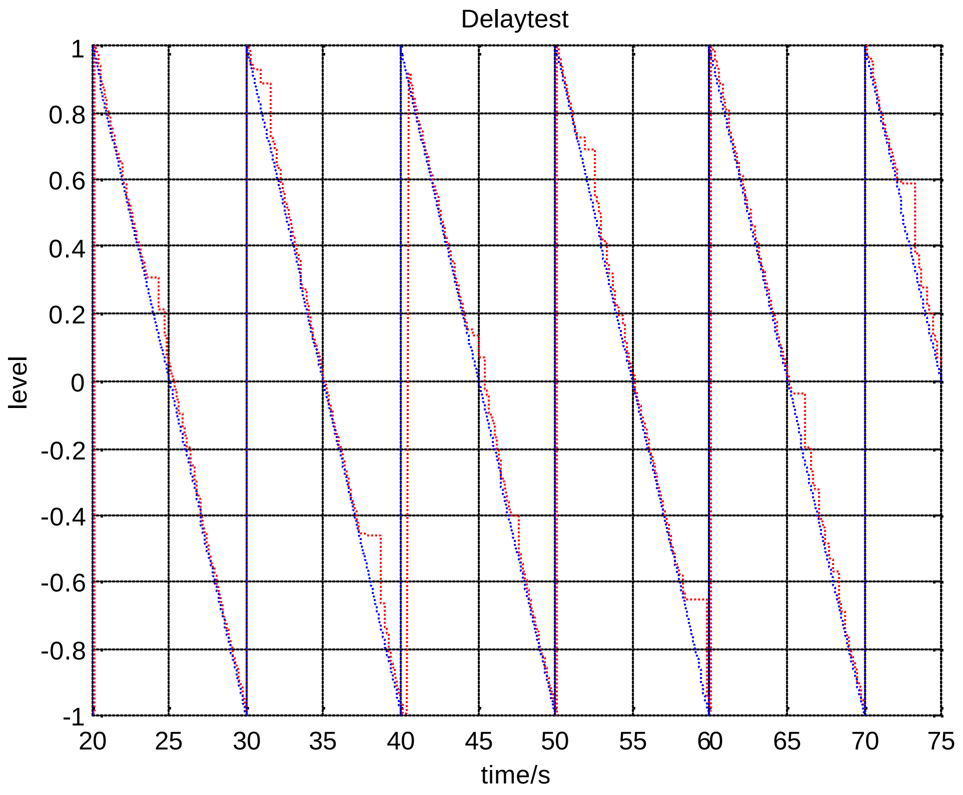


Figure 8.4. Diagram that shows the in-and output signals when the delay test is made on two computers. The deterministic longer delay in the diagram is due to the communication link between the computers.

The figure 8.5 shows the same diagram as in figure 8.4, but with a smaller time-slice in the time axis. In this diagram the delay over an ordinary part from the figure 8.4 can be measured. The delay varies approximately between 80-130ms

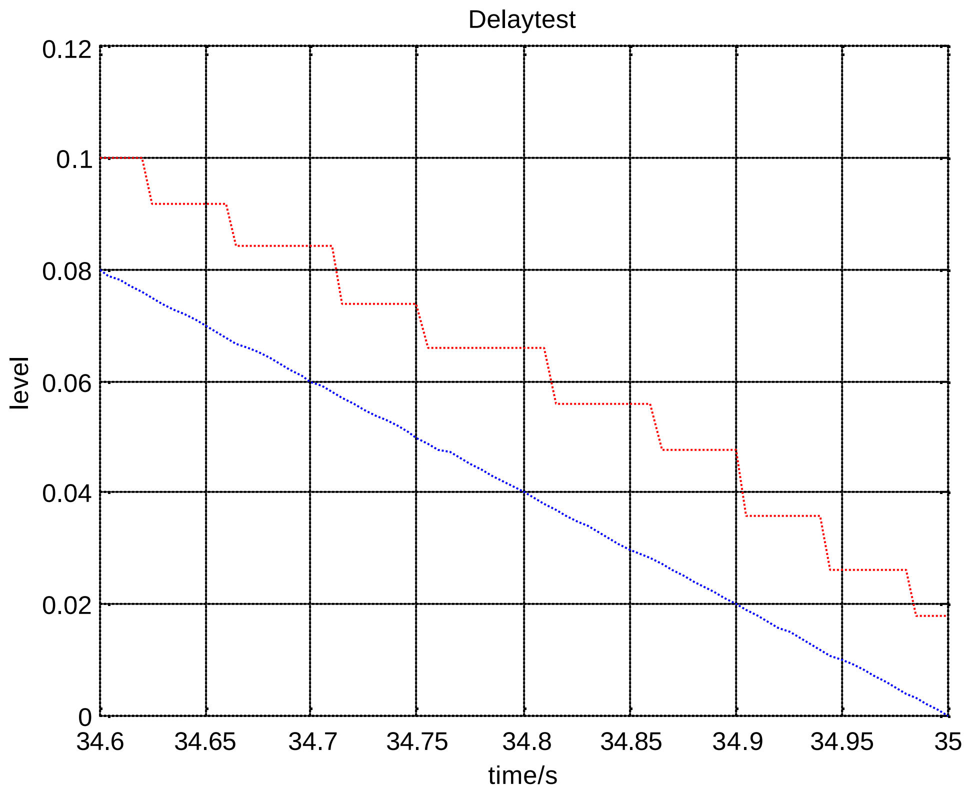


Figure 8.5. Diagram that shows the in-and output signals when the delay test is made on two computers. The chosen time interval is where there are no extremely big delays as shown in figure 8.4.

The jitter for this simulation is shown in figure 8.6. As can be seen from the diagram the jitter varies, but it is not 0.5 in so many places. This means that the delay of the chosen sample time in Simulink is almost always under half the chosen sample time.

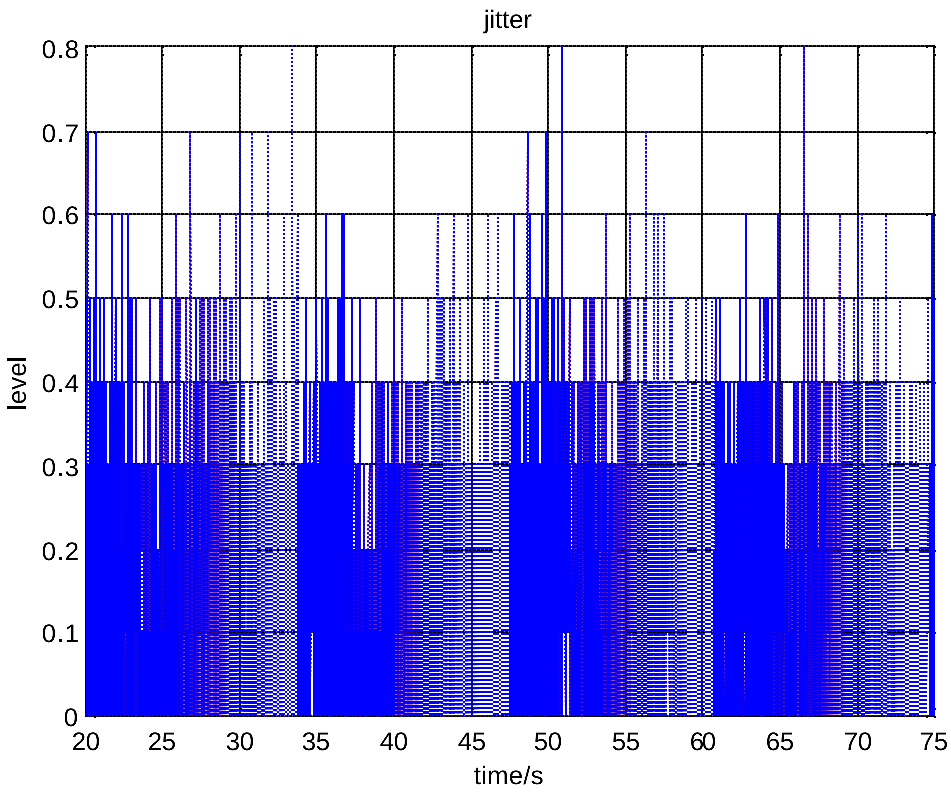


Figure 8.6. Jitter during the delay test with two computers.

8.4 Conclusions about this test

The improvements made in the OPC-MMS Server and in the real-time function in Simulink make a big difference in the real-time performance. Compared to the test in chapter 7 the delay in this test is decreased approximately 10 times. The delay, according to the delay test, in this test is approximately between 80-130 ms.

The only thing that is not so good in this test is the long undeterministic delays, which can be some seconds long. One possible solution to the undeterministic delay is that an Ethernet network is used for communication between the computers. Ethernet is not defined for real-time communication.

9 Tests on one computer

To make it possible to use one computer for the tests the computer has to be quite fast. Computer with lower performance than the one described in chapter 9.1 is not recommended to use. Other things that are quite important, are the CPU time that the processes use before they let other processes access the CPU and the difference between the priority levels among the processes.

9.1 Data

The sample times for the OPC-MMS Server and Matlab/Simulink are in this test increased from 10ms to 30ms compared to the test with two computers described in chapter 8. The increase of the sample times are made because of the jitter in Matlab/Simulink is too high if 10ms sample time is used in the OPC-MMS Server and in Matlab/Simulink. In table 9.1 data for this simulation are shown.

Table 9.1. Computer 1, 400MHz Pentium II with 256MB RAM

process	**priority**	**sample time/ms**
Control Builder	normal (default)	-
Soft Controller (with PID)	realtime (default)	100
OPC Server	high	30
Gateway	high	high as possible
Matlab/Simulink	high	30

9.2 Test

The simulation shown here is a step response on the continuous DC-servo in figure 7.3 made with a PID controller. In the left diagram of figure 9.1 the step response using a PID controller is shown and in the right diagram the jitter for the same simulation is shown.

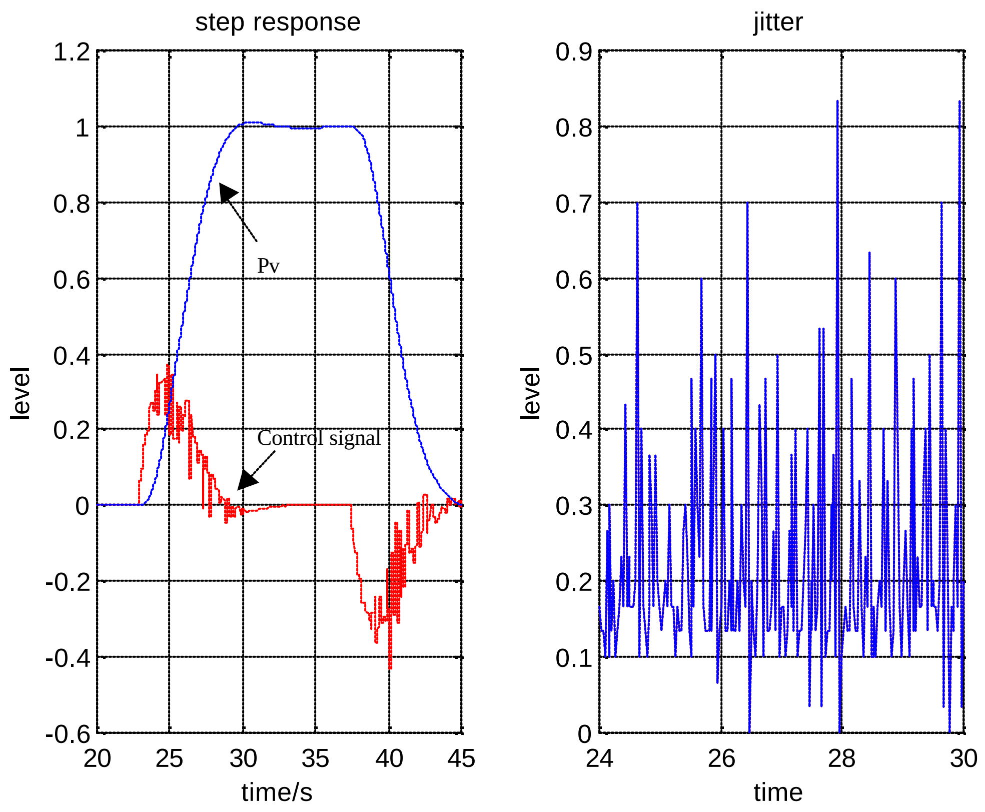


Figure 9.1. Left diagram shows a step response in test with one computer. The right diagram shows the jitter for the same simulation.

This step response test is only made to visually see that the curve of the process value and the control signal behave in a proper way, with the chosen parameters. There are no strange behavior seen in the diagrams.

9.3 CPU time used for the test

In figure 9.2 and 9.3 the CPU times for the processes used are shown. The diagrams are not so easy to look at, but it is clearly seen that the total CPU time in figure 9.2 is at 100% when the CpuTimeQuota for Soft Controller, OPC-MMS Server and Control Builder are set to 30% (default). CpuTimQuota is described in chapter 7.1. In figure 9.3 the CpuTimeQuota is put down to 10% and the total CPU time is clearly under 100%.

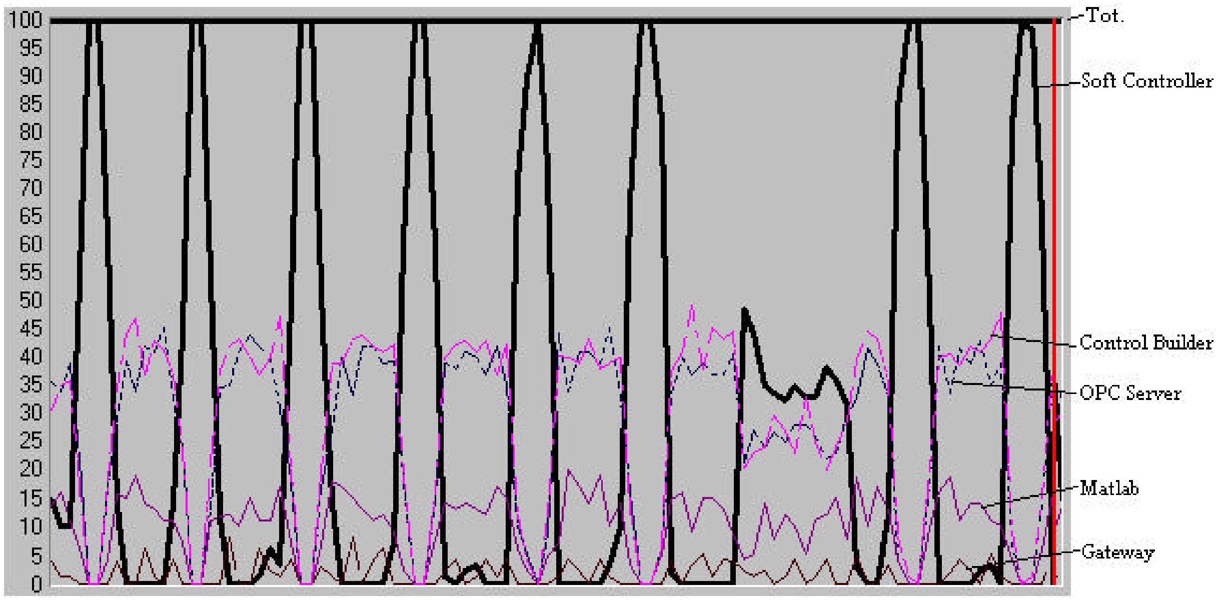


Figure 9.2. Total CPU time and CPU times for Soft controller, OPC-MMS Server, Control Builder, Gateway and Matlab. CpuTimeQuota=30%. The time axis is 100 s.

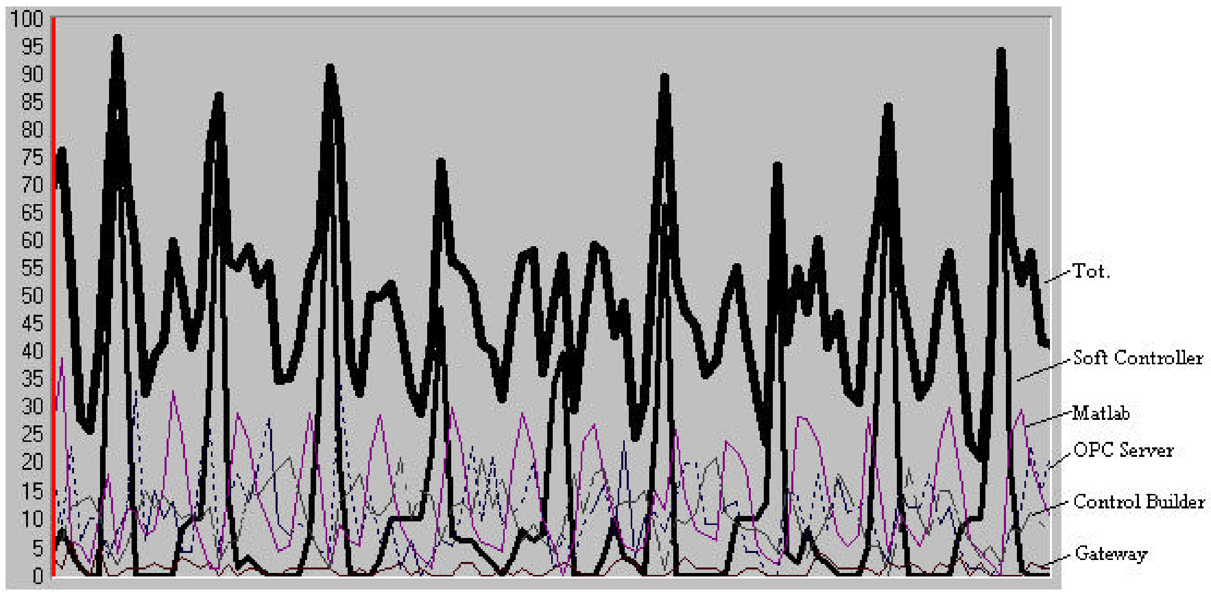


Figure 9.3. Total CPU time and CPU times for Soft controller, OPC-MMS Server, Control Builder, Gateway and Matlab. CpuTimeQuota=10%. The time axis is 100 s.

When a simulation is made it is good to know that you do not use all the CPU time. Therefore the usage of lower CpuTimeQuota than is ordinary used in Control Builder, Soft Controller and OPC-MMS Server has been tested.

9.4 Delay (CpuTimeQuota=10%)

The delay test has been divided into two parts, one with CpuTimeQuota=30% and one test with CpuTimeQuota=10%. In this first test the CpuTimeQuotas for Control Builder, Soft Controller and the OPC-MMS Server are set to 10%. In figure 9.4 and 9.5 the delay test described in chapter 6.2 is used. As can be seen the undeterministic behavior that appears when two computers were used is no longer present. One solution to this is that no Ethernet network is involved in the communication inside one computer.

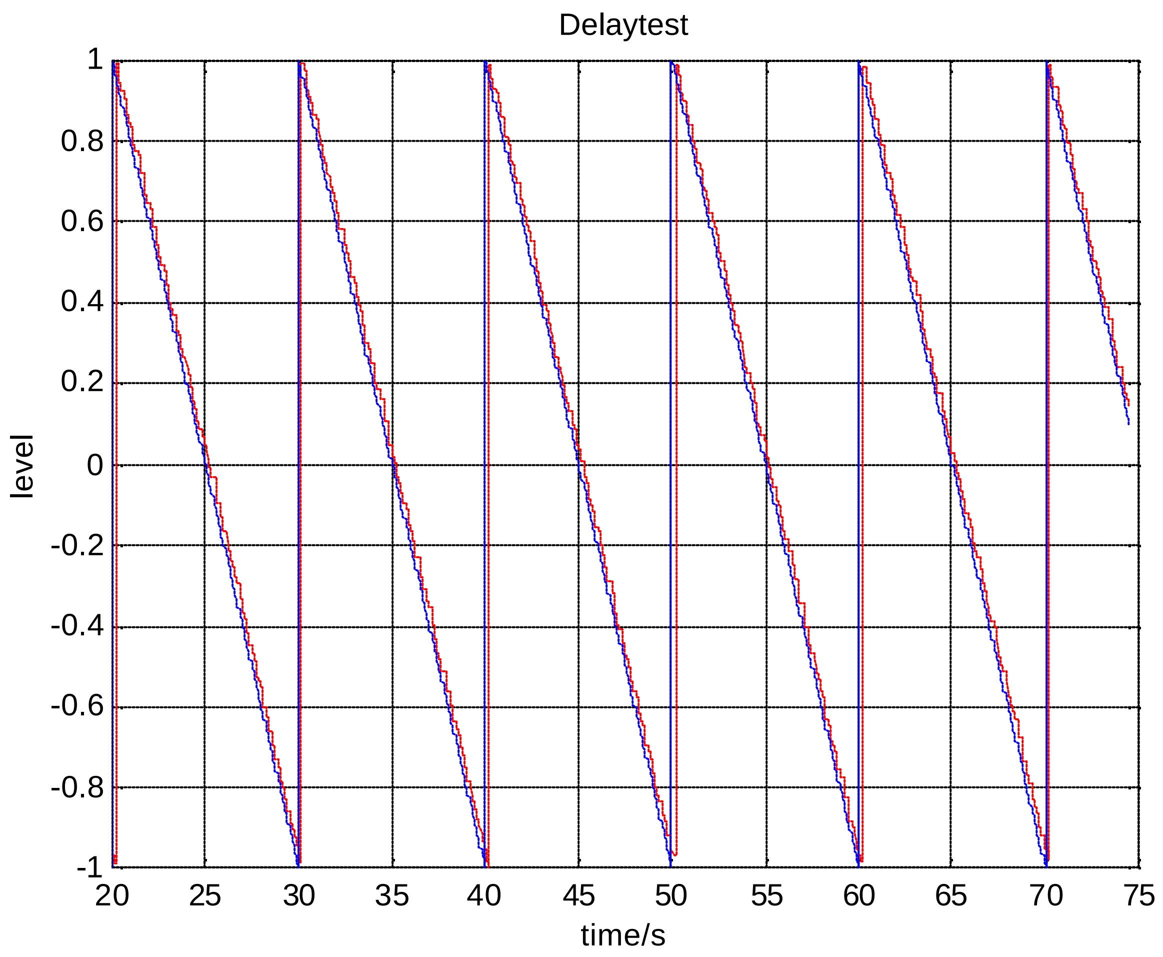


Figure 9.4 Diagram that shows the in-and output signals when the delay test is made on one computer and the CpuTimeQuota is 10%.

The figure 9.5 shows the same diagram as in figure 9.4, but with a smaller time-slice of the time-axis. In this diagram the delay over an ordinary part from the figure 9.4 can be measured. The delay varies approximately between 150-300ms

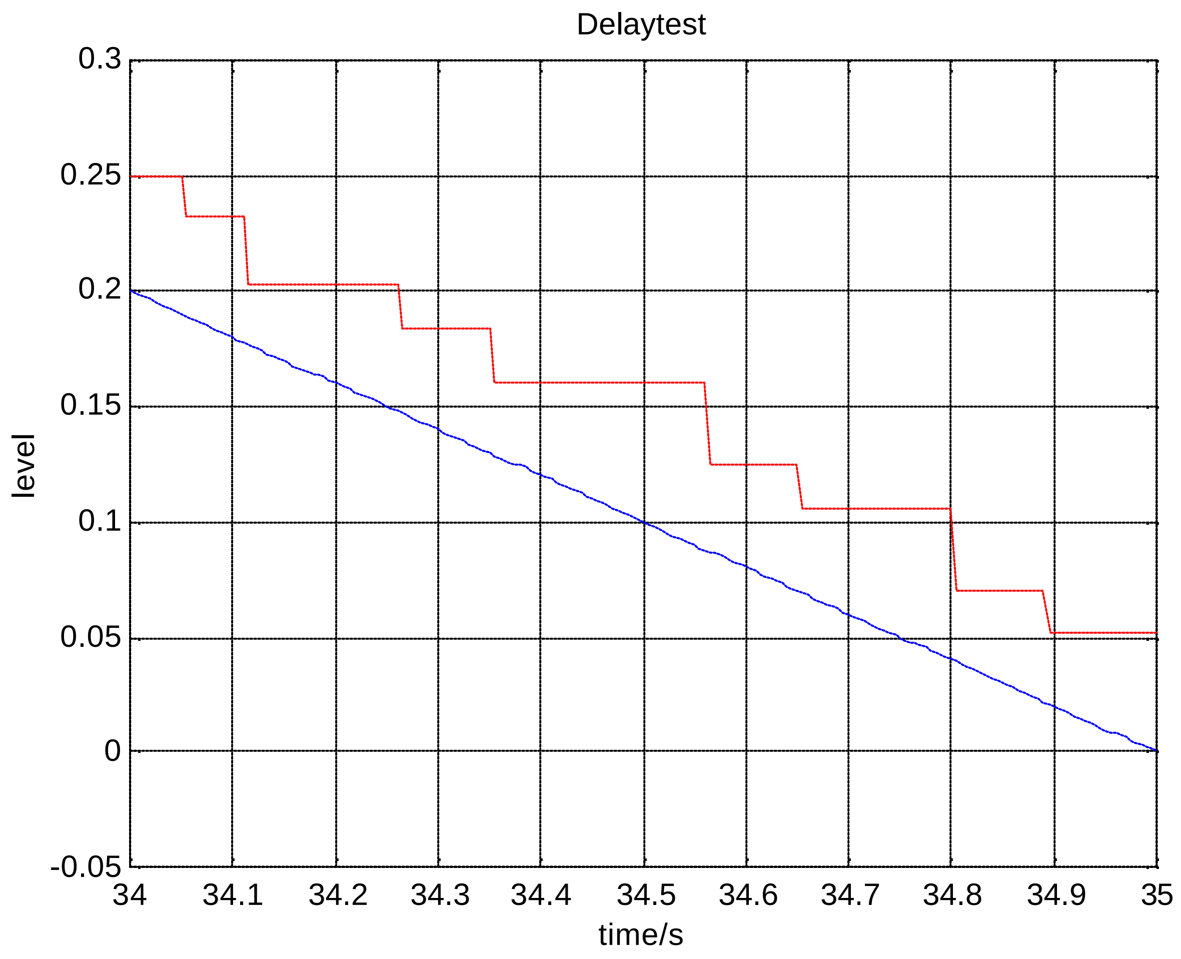


Figure 9.5. Diagram that shows a smaller time-slice than figure 9.4 of the in-and output signals, when the delay test is made on one computer and the CpuTimeQuota is 10%.

The jitter for this simulation is shown in figure 9.6. As can be seen from the diagram the jitter varies a lot, but it is not so many peaks that is up to 1 in the jitter diagram. This means that the delay in Simulink is less than one sample time of the chosen sample time in the Simulink model.

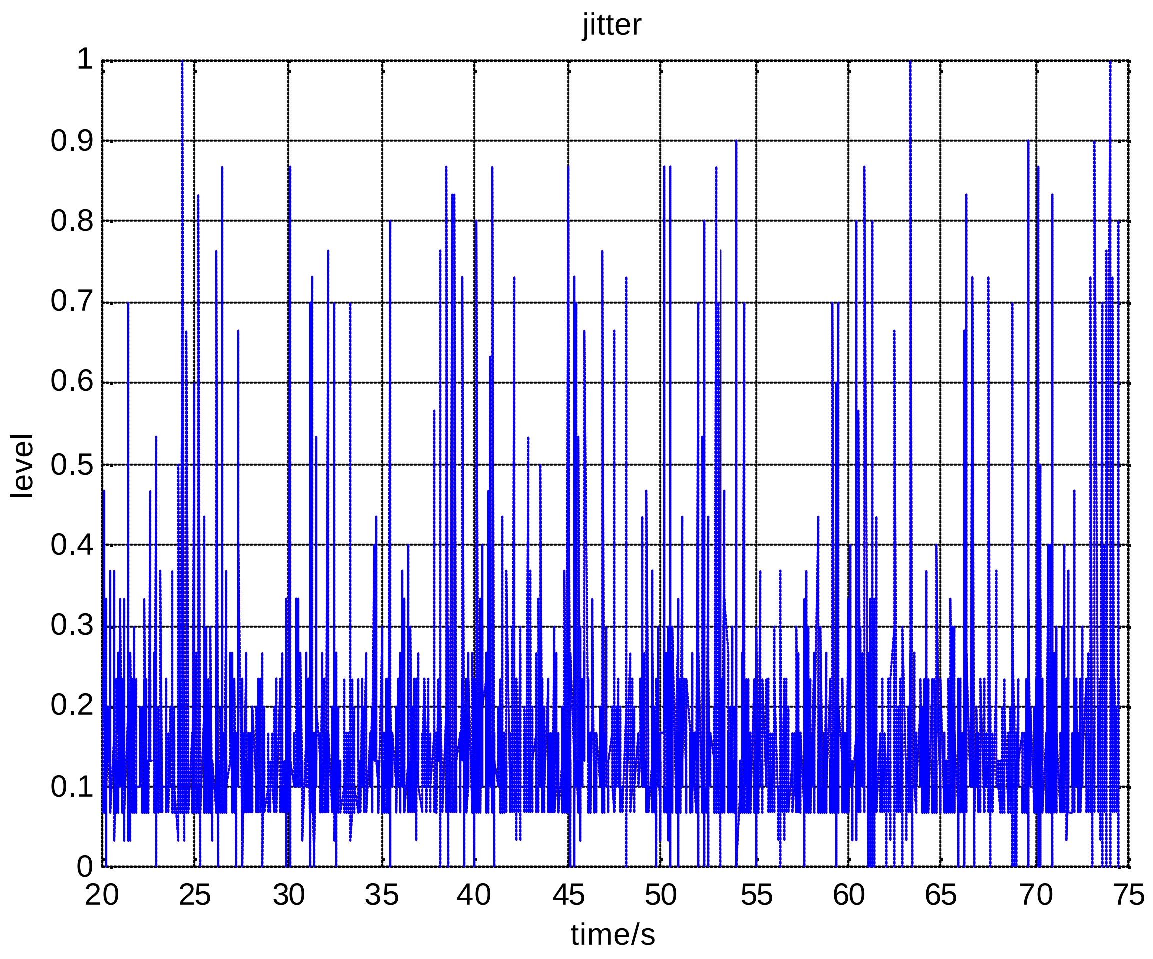


Figure 9.6. Jitter during the delay test with one computer and CpuTimeQuota=10%.

9.5 Delay (CpuTimeQuota=30%)

In this second test the CpuTimeQuota for Control Builder, Soft Controller and the OPC Server are set to default 30%. The figures 9.7 and 9.8 show the signals from the delay test.

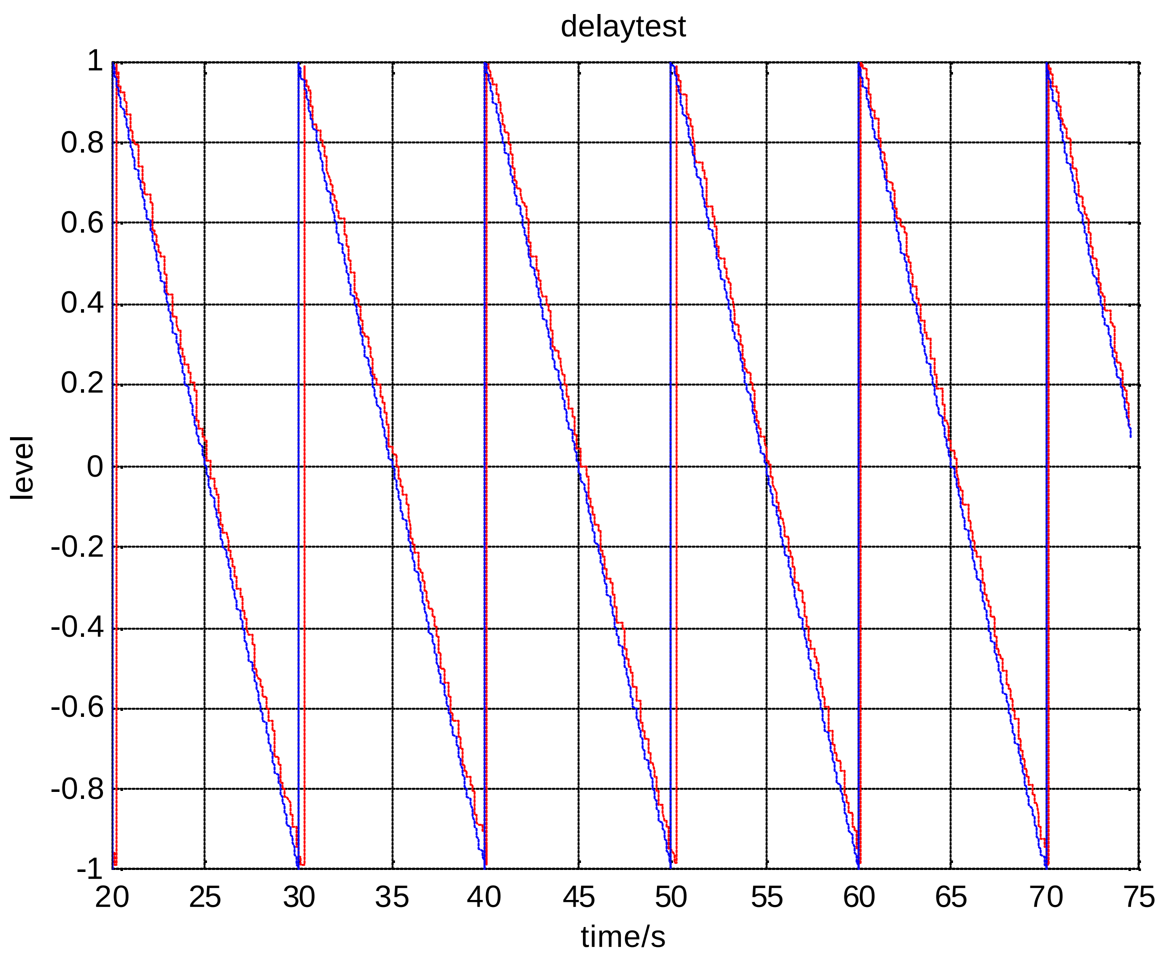


Figure 9.7. Diagram that shows the in-and output signals when the delay test is made on one computer and the CpuTimeQuota is 30% (default).

Figure 9.8 shows the same diagram as in figure 9.7, but with a smaller time-slice of the time-axis. In this diagram the delay over an ordinary part from the figure 9.7 can be measured. The delay varies approximately between 150-400ms i.e. not so much different from the test made with CpuTimeQuota=10%. The difference between the simulations is instead shown if the jitter from the simulation with CpuTimeQuota=30% is compared with the jitter from the simulation with CpuTimeQuota=10%. The jitters from the two simulations are shown in the figures 9.6 and 9.9. As can be seen from the diagram in figure 9.9 the jitter is much too high when CpuTimeQuota=30% is used. When the jitter is as high as in this test, the result of the delay test not usable. The reason is described in chapter 6.2.

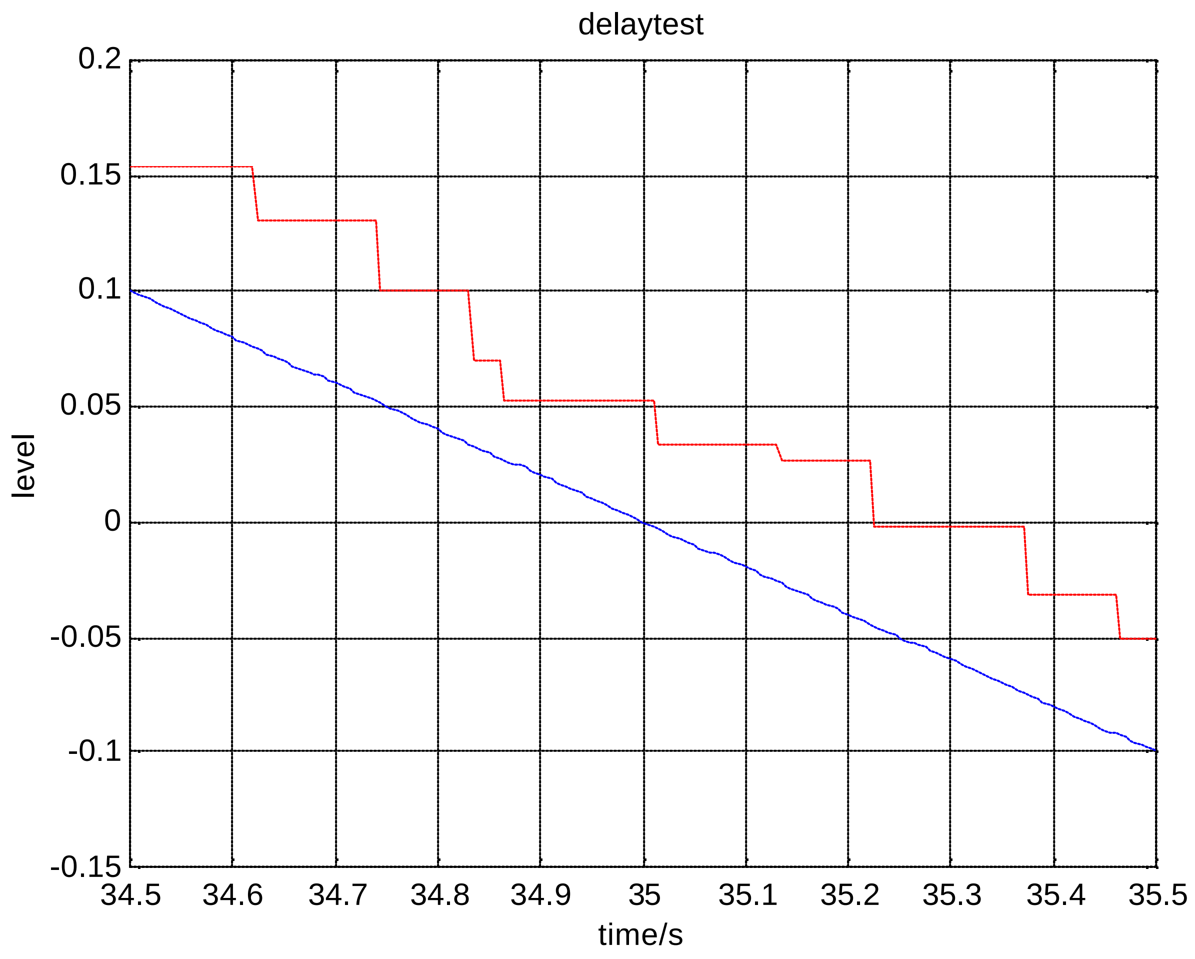


Figure 9.8. Diagram that shows a smaller time slice than figure 9.7 of the in-and output signals when the delay test is made on one computer and the CpuTimeQuota is 30% (default).

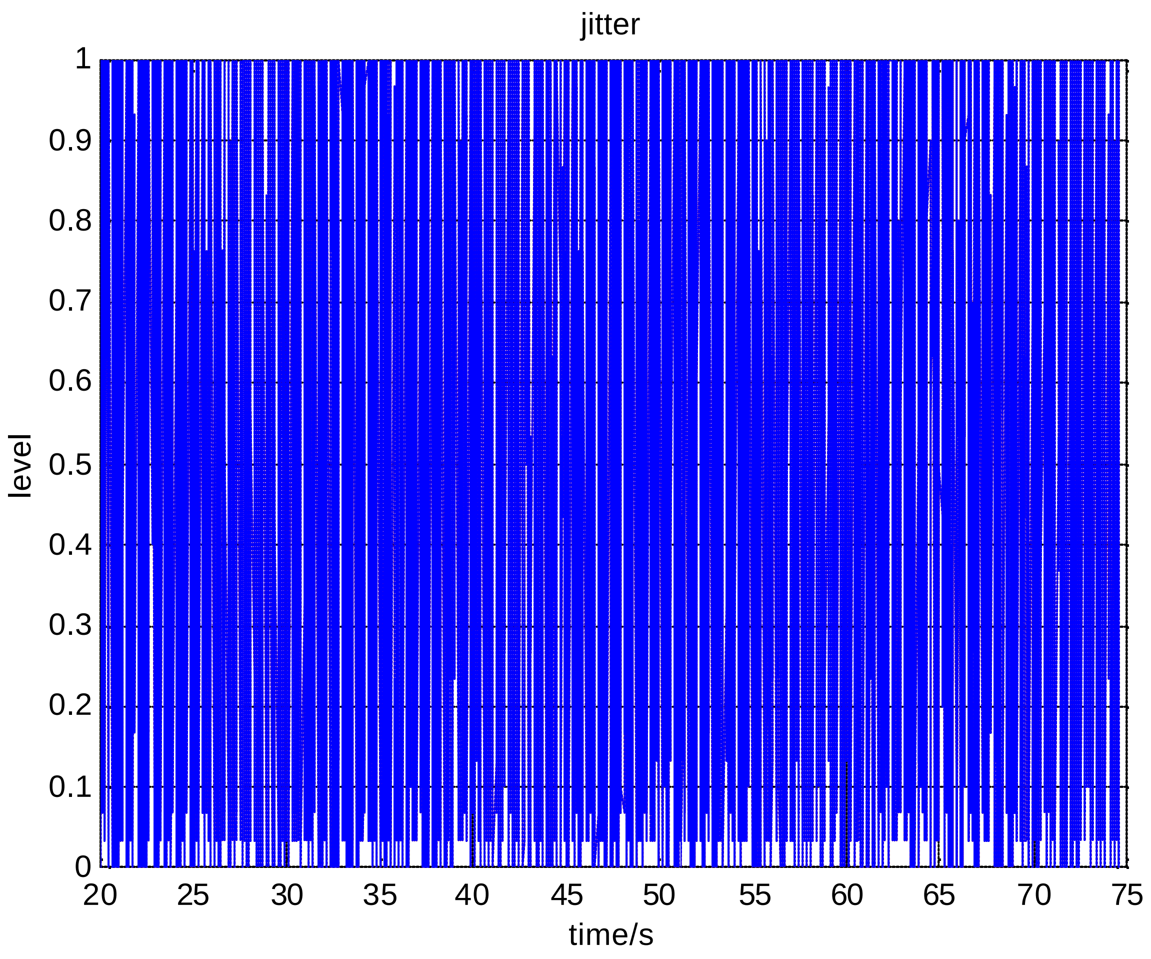


Figure 9.9. Jitter during the delay test with one computer and CpuTimeQuota=30%.

9.6 Conclusions about this test

To make it possible to do simulations in one computer the CpuTimeQuota for the Control Builder, OPC-MMS Server and Soft Controller have to be chosen lower than the low limit for those products. This has been tested in this chapter, and the result is good. The lower CpuTimeQuota that has been tested is 10%. The delay according to the delay test is then around 150-300ms.

10 Conclusions

After all the tests that have been made some conclusions and recommendations about how, and what kind of systems that can be used in the simulations to get reliable result is described. Since it is desirable to only use one computer for the whole simulated system, but two computers were necessary in the beginning of the tests, there are a lot of tests made on two computers.

Some main changes in the Control Builder, Soft Controller and the OPC-MMS Server have to be made to get the simulation to work well in real-time. There has also been changes made in the timer function used in Matlab/Simulink.

The change that has to be made in the Control Builder, Soft Controller and the OPC-MMS Server is the usage of CPU-time for each of these processes. The default usage of CPU-time, called CpuTimeQuota, for the processes are set to 30%. In those commercial applications the CpuTimeQuota can be adjusted between 30-100%. If one computer will be used for all the applications i.e. Control Builder, Soft Controller, OPC-MMS Server Gateway and Matlab/Simulink, the possibilities to adjust the CpuTimeQuota lower than 30% is needed. This means that Control Builder, Soft Controller and OPC-MMS Server have to be recompiled with the low limit of the CpuTimeQuota taken away. This makes it possible to tune the usage of the CPU-time. In this thesis a test has been made with CpuTimeQuota set to 10% for all the processes. If fine tune is desired it is possible to use different CpuTimeQuota for the different processes.

There is also another change in the OPC-MMS Server that has to be made to get the simulation to work with smaller sample times in the controller. The change is to remove the low limit of the update rate for the server. The reason is that the low limit in the default case is set to 250ms, and in the simulation it is desirable to use as low update-rate as possible. When one computer is used, tests have shown that an update-rate at 30ms is a good choice.

Since the process model should look as much as possible like a continuous real process, the time in the Matlab/Simulik solver should follow the real time, and the sample time should be as low as possible. To make this possible a real-time function has been implemented that sample the process model in Simulink. The real-time function uses a multimedia timer to make it possible to use low sample times. When one computer is used, tests have shown that a sample time at 30ms is working well.

The main conclusion is that the simulation works fine if the delays that occur are taken into consideration when you design your simulation system. If one computer, with the data described in chapter 9, is used the delay is varying between 150-300ms. If two computers, with the data described in chapter 8, are used the delay is varying between 80-130ms. If faster computers are used, the delay will probably be smaller.

11 Future Possibilities

In this chapter things that have not been time to investigate are mentioned. Also some parts that is not finished developed in the simulation system are here described.

The first thing that should be solved if simulation should be made as in this thesis is to implement two system-variables. One of the system-variables should make it possible to choose the update rate in the OPC-Server lower than the default 250ms. The other system-variable should make it possible to use a CpuTimeQuota for the Control Builder, the Soft Controller and the OPC-Server that is lower than the default 30%. The reasons for these system-variables are that the update rate in the OPC-Server has a big influence in the delay during the simulation, and a low CpuTimeQuota makes it possible to do simulations in one computer.

The Gateway application has in the end of this master thesis been debugged using a program named Numega BoundsChecker. The reason for this was that memory leakage's were found in the communication loop inside the Gateway. The memory leakage's inside the communication loop have been corrected. Numega BoundsChecker also found some other errors in the Gateway code, but those errors have not been time to investigate and correct. The Gateway is working without those errors corrected, but it is a good idea to investigate and eventually correct those errors. The errors can be seen if the Gateway application is executed with Numega BoundsChecker.

When the first two parts mentioned are corrected, there are some parts in the user interface of the Gateway that can be fixed. The biggest change is to make it possible to choose the number of OPCIn and OPCOut as many as the user wants. To make this change possible almost the whole Gateway application has to be rewritten. A smaller change that can be made is to make it possible to browse when the application from Control Builder should be chosen in the Gateway application. Other changes are to make it possible to print and save the chosen configurations. If it is possible to make changes in the priorities for the Control Builder, the Soft Controller, the OPC-MMS Server and Matlab in a new dialog inside the Gateway this will simplify the start-up of a simulation.

Some studies have been made to see how models can be built in Simulink. The result is that it is quite difficult to build detailed process-models of small elements as valves, tanks etc, and then connect those small elements together as a big process-model. Only some simple not that detailed process-models has been built in this thesis. There are a lot more to do in this area. Some process identification of real processes, and test with those processes implemented in Simulink would be interesting to make.

12 References

[1]http://cwwba.malmo.abb.se/x/homepage/prodkat/html/produkte.htm (2000-07-18), Internal ABB.

[2] Davidsson, Peter-Hansson, Fredrik (2000), *Communication between Advant Control Builder and Matlab/Simulink*, Master thesis at the Department of Automatic Control, Lund Institute of Technology.

[3] Asrom, Karl J-Wittenmark, Bjorn (1997), *Computer Controlled System*, Prentice Hall.

[4] Eriksson, Hans-Erik (1998). *Programutveckling för Windows och Windows NT* , Student literatur.

[5] Richter, Jeffrey (1995), *Advanced Windows*, Microsoft Press.

[6] Kruglinski, David J.- Shepard, George- Wingo, Scot (1998), *Programming Microsoft Visual C++ Fifth Edition*, Microsoft Press.

[7] MSDN Library Visual

[8] Rogerson, Dale (1997), *Inside COM,* Microsoft Press.

[9] Nilsson, Johan (1998), *Verification of control applications, A communication interface for simulation,* Master thesis at Lund Institute of Technology.

[10] Neskovic, Nelica (2000), *OLE for process control in Advant Soft Controller as comunication interface for simulation,* Master thesis at Lund Institute of Technology.

[11] Olsson, Gustaf- Piani, Gianguido (1998), *Computer Systems for Automation and Control, Second Edition,* Prentice Hall.

[12] www.mathwork.com (2000-07-26)

[13] Mathworks (1998), *Using Simulink Version 3*

[14] Mathwork (1998). *Matlab Application Program Interface Guide Version 5*

[15] Mathworks (1998), *Writing S-Functions, Version 3*

[16] Olsson, Jan E (2000), *Funktionspecifikation för Programexekvering,* Internal ABB.

Appendix A

In appendix A the installation and the user manual for the Gateway application is described.

Installation of the Gateway application

The execution file ACB_Simulink.exe is the exe-file that needs to be installed. There are also some files from the Matlab11\bin map that have to be copied into winnt\system32. Those files are libeng.dll, libmx.dll, libut.dll and libmi.dll.

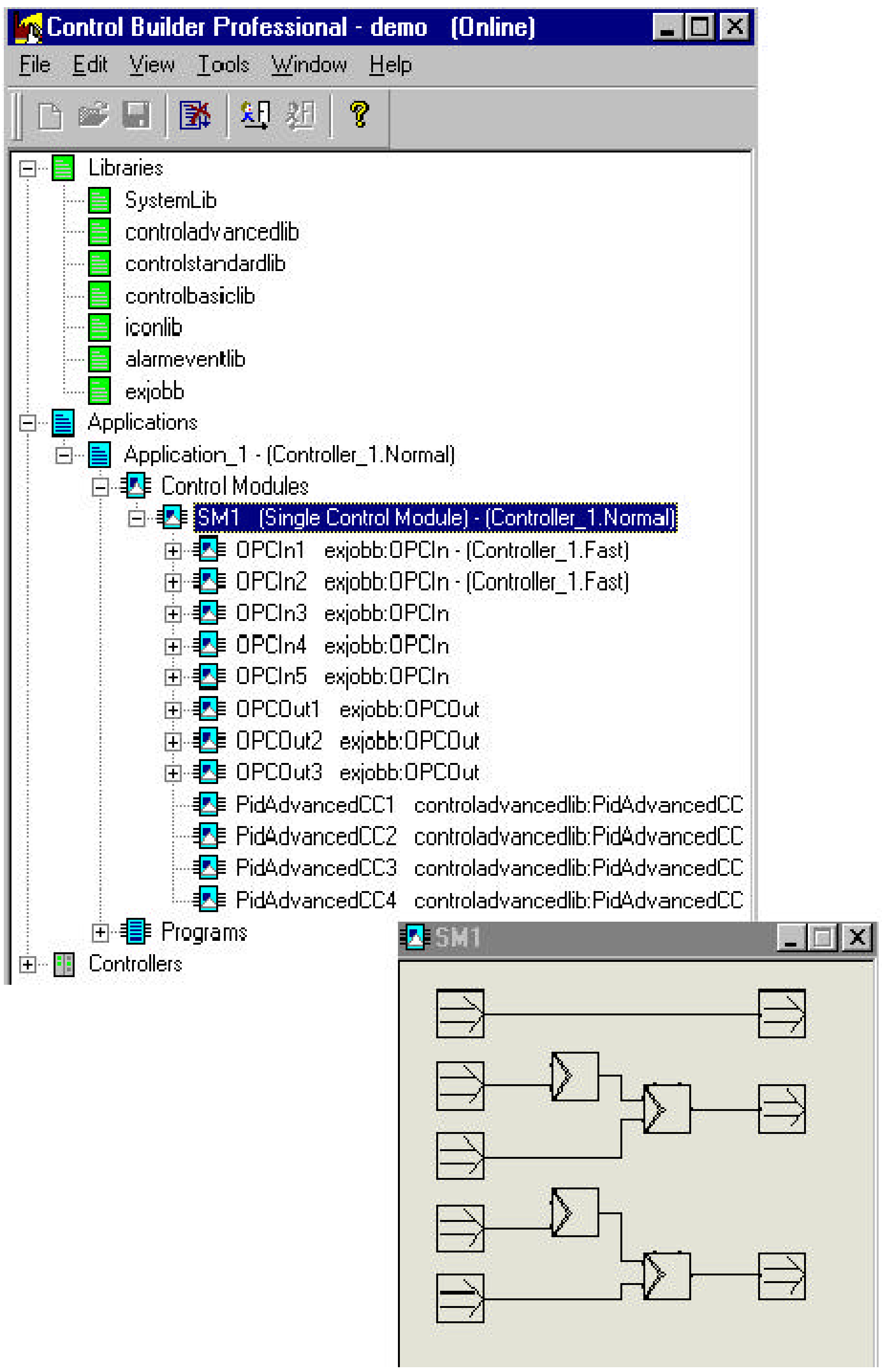
User manual for the Gateway application

The user manual can be used as a slide-show with some important details described. The slide-show is made for simulations made in one computer. The differences if two computers are used are described in the text that follows next to the pictures.

Slide Show

The figures in the following slide-show are placed in the order they should be seen when a simulation is started. The pictures from the Control Builder are taken from the version α-7.

The first figure A.1 shows a simple project in the Control Builder with some controllers and the connections, which are needed to get the communication with the OPC-MMS Server to work. There are only a few differences in this project compared to usual projects. The differences are the OPCIn and OPCOut blocks that are included in the library exjobb. To make the simulation work no changes in the OPCIn and OPCOut block should be made. The project and the interactive window for the application, when it is loaded to the controller are shown in figure A.1. No description about how to use Control Builder is included in this thesis.

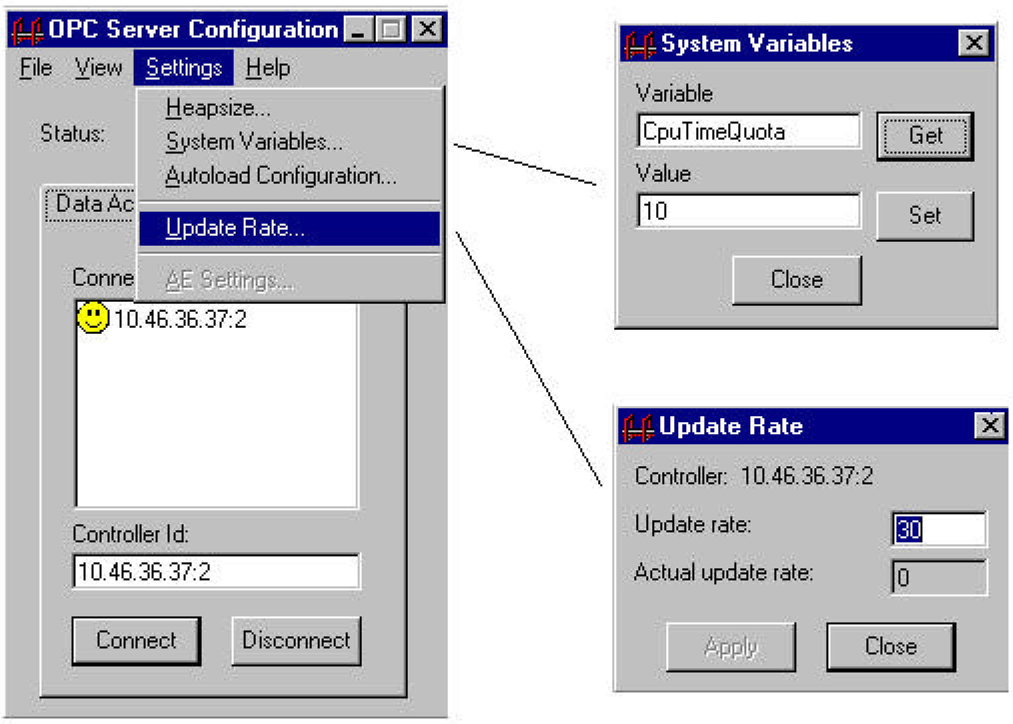

Control Builder Professional - demo (Online)
File Edit View Tools Window Help
SM1

```
Libraries
    SystemLib
    controladvancedlib
    controlstandardlib
    controlbasiclib
    iconlib
    alarmeventlib
    exjobb
Applications
    Application_1 - (Controller_1.Normal)
        Control Modules
            SM1   (Single Control Module) - (Controller_1.Normal)
                OPCIn1   exjobb:OPCIn - (Controller_1.Fast)
                OPCIn2   exjobb:OPCIn - (Controller_1.Fast)
                OPCIn3   exjobb:OPCIn
                OPCIn4   exjobb:OPCIn
                OPCIn5   exjobb:OPCIn
                OPCOut1   exjobb:OPCOut
                OPCOut2   exjobb:OPCOut
                OPCOut3   exjobb:OPCOut
                PidAdvancedCC1   controladvancedlib:PidAdvancedCC
                PidAdvancedCC2   controladvancedlib:PidAdvancedCC
                PidAdvancedCC3   controladvancedlib:PidAdvancedCC
                PidAdvancedCC4   controladvancedlib:PidAdvancedCC
        Programs
Controllers
```

Figure A.1.Project and interactive windows for an application in Control Builder.

Figure A.2 shows the OPC Server Configuration dialog window. The IP number is for the place where the controller is placed. In this window the system variable CpuTimeQuota can be changed. If one computer is used it can be a good idea to set the CpuTimeQuota lower than default 30%. The CpuTimeQuota can also be changed for the Control Builder and Soft Controller shown in figure A.3 and A.4. A recommendation is to set CpuTimeQuota to 10%. Look in chapter 9 to see more about the choice of CpuTimeQuota.

The Update Rate in the OPC-MMS Server can also be changed in the dialog window shown in figure A.2. A recommendation is to use 30ms if one computer is used, and down to 10ms if two computers are used.



Controller Id: 10.46.36.37:2

Variable: CpuTimeQuota

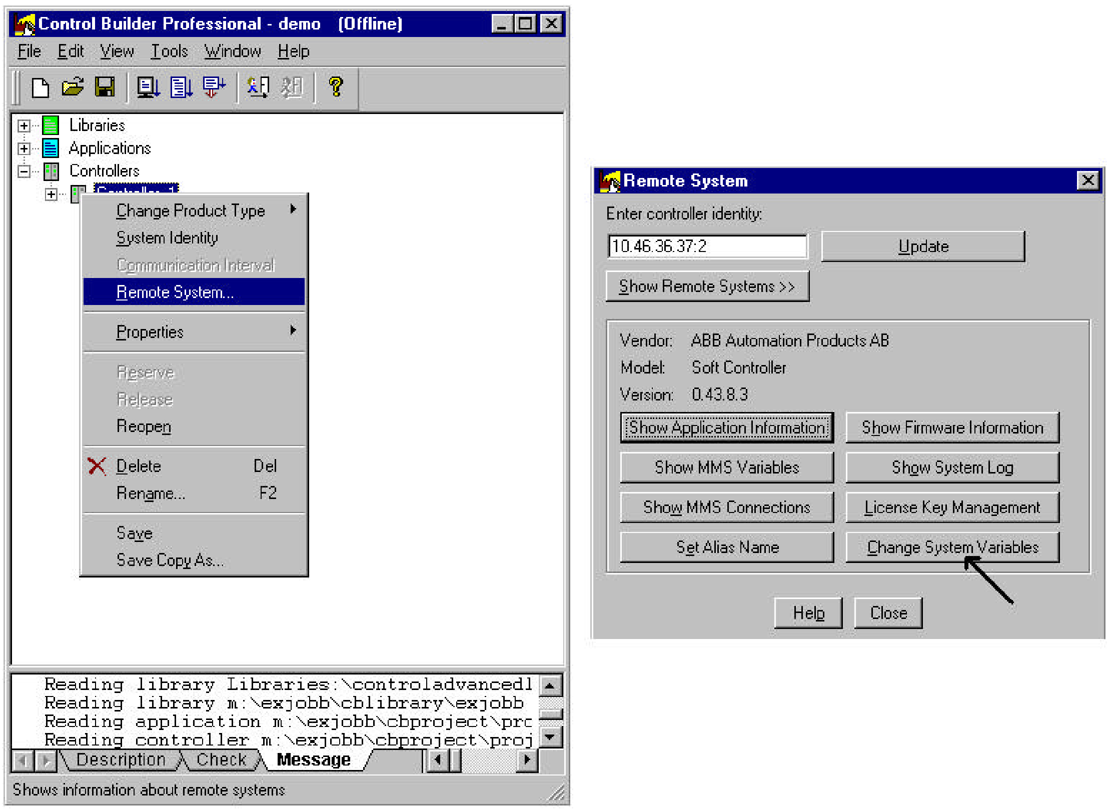
Value: 10

Controller: 10.46.36.37:2

Update rate: 30

Actual update rate: 0

Figure A.2. OPC server Configuration dialog window.

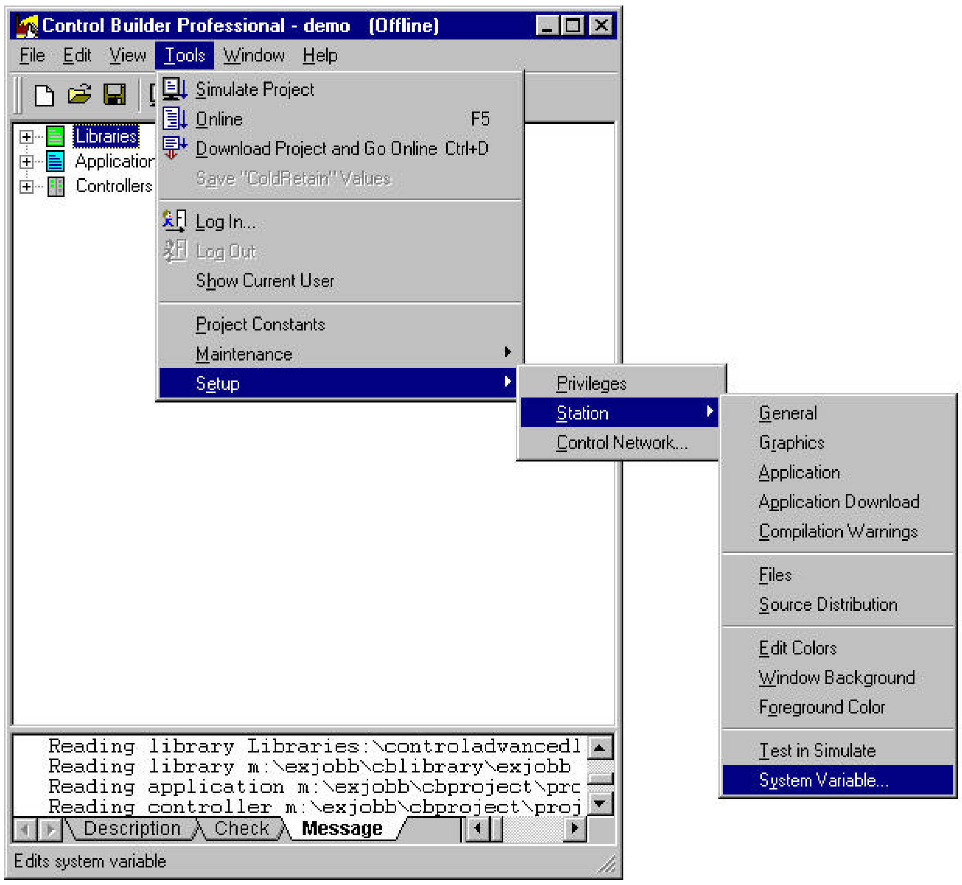


```
Reading library Libraries:\controladvancedl
Reading library m:\exjobb\cblibrary\exjobb
Reading application m:\exjobb\cbproject\prc
Reading controller m:\exjobb\cbproject\proj
```

Enter controller identity: 10.46.36.37:2

Vendor: ABB Automation Products AB
Model: Soft Controller
Version: 0.43.8.3

Figure A.3. Dialog where the system variable CpuTimeQuota for Soft Controller can be changed.



```
Reading library Libraries:\controladvancedl
Reading library m:\exjobb\cblibrary\exjobb
Reading application m:\exjobb\cbproject\prc
Reading controller m:\exjobb\cbproject\proj
```

Edits system variable

Figure A.4. Dialog where the system variable CpuTimeQuota for Control Builder can be changed.

The dialog window shown in figure A.5 is the first window that appears when the Gateway application is started. In the Configure drop-down menu Connect Server should be chosen.

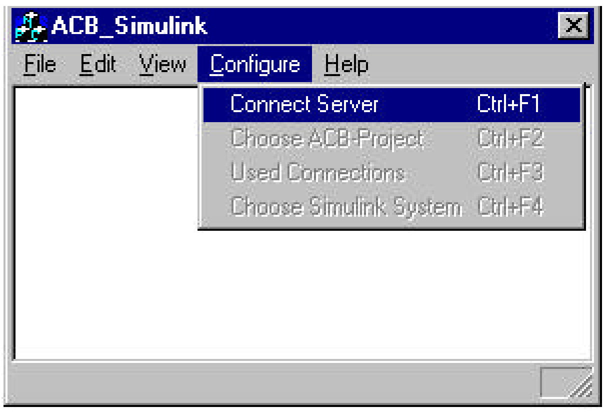
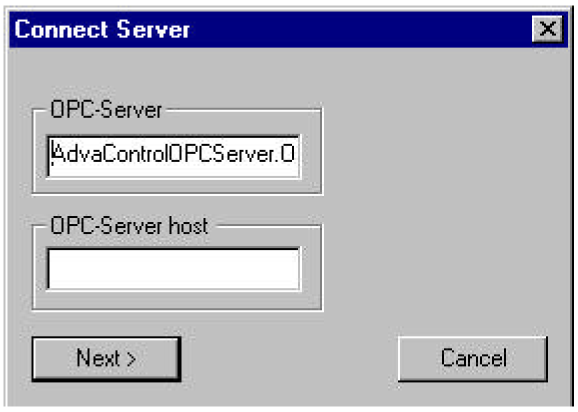


Figure A.5. Main window for the Gateway application.

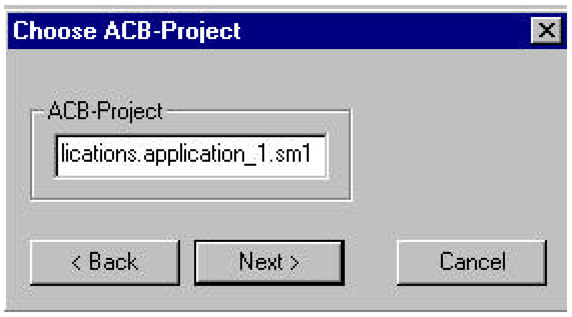
In figure A.6 the OPC Server that will be used is chosen. If one computer is used, it is not necessary to write anything in the edit box OPC-Server host. If the OPC Server is running on another computer, the name of the computer should be written in the edit box OPC-Server host.



OPC-Server: AdvaControlOPCServer.O

Figure A.6. Connect Server dialog in the Gateway application. In this example the name in the OPC-Server edit box is ‘AdvantControlOPCServer.OPCDAServer.1’.

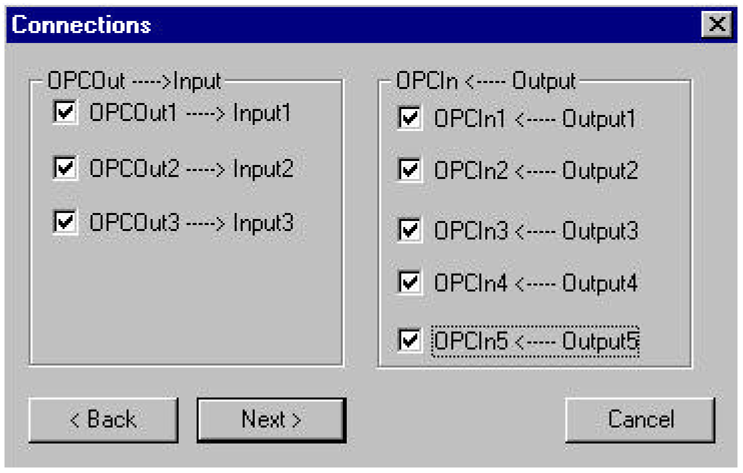
In figure A.7 the project loaded to the Soft Controller shall be typed in.



ACB-Project: lications.application_1.sm1

Figure A.7. Dialog used to type in project name. In this example the name in the ACB-Project edit box is ‘applications.application_1.sm1’.

Figure A.8 shows the dialog, which is used to choose connections. Only the possible choice of connection is shown in the dialog window.



- ☑ OPCOut1 -----> Input1
- ☑ OPCOut2 -----> Input2
- ☑ OPCOut3 -----> Input3
- ☑ OPCIn1 <----- Output1
- ☑ OPCIn2 <----- Output2
- ☑ OPCIn3 <----- Output3
- ☑ OPCIn4 <----- Output4
- ☑ OPCIn5 <----- Output5

Figure A.8. Dialog used to choose which connections that shall be used.

In figure A.9 the dialog for selection of which Simulink model to use is shown. The Browse function can be used to choose Simulink model.

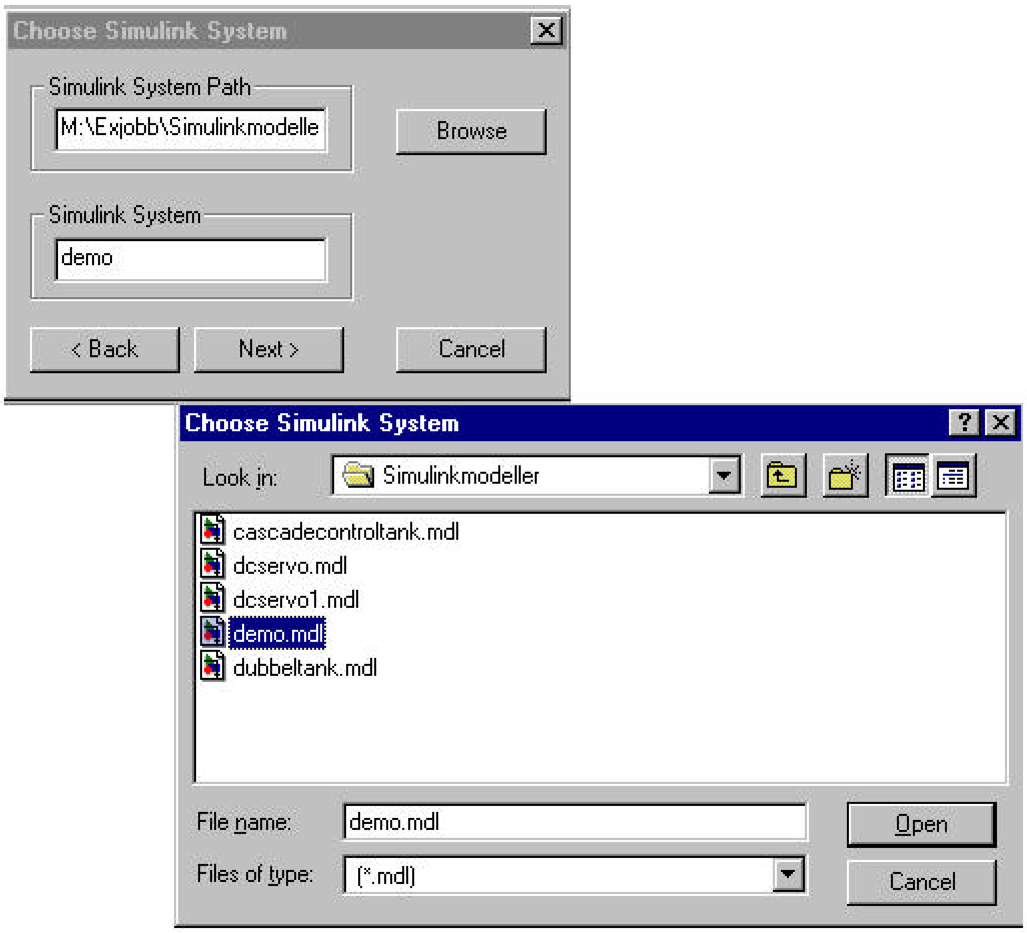


Figure A.9. Dialogs to select Simulink model.

When the finish button in the following dialog is pushed all connections are made, and Matlab/Simulink starts. Push the play button in Simulink to start the process model. The controllers then can be used as they are with a real process connected to Soft Controller.

In table A.1 priorities and sample times for the different processes used during simulations in one computer are shown. The selection of those parameters is free to the user of this simulation tool, but the parameter choice in table A.1 is good to start from. The parameter value in the column priority can be changed in the Windows task manager. If two computers are used, recommendations about priorities and sample times can be found in chapter 8.

Table A.1. Priorities and sample times during simulation in one computer.

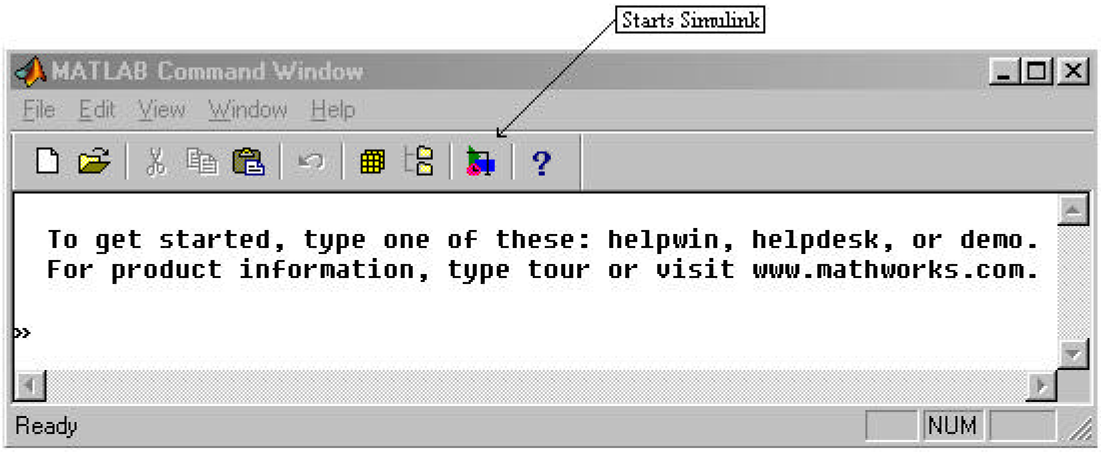
process	**priority**	**sample time /ms**
Control Builder	normal (default)	-
Soft Controller	realtime (default)	100
OPC-MMS Server	high	30
Gateway	high	-
Matlab/Simulink	high	30

Appendix B

In appendix B a short description, from the beginning, about how to build models in Simulink is placed. Matlab version 5.3 (R11.1) has been used in this thesis.

Starting Simulink

When Matlab has been started, Simulink shall be started. The button that starts Simulink is shown in figure B.1.



```
To get started, type one of these: helpwin, helpdesk, or demo.
For product information, type tour or visit www.mathworks.com.

>>
```

Figure B.1 Matlab Command window.

The dialog window that appears is called Simulink Library Browser. In this dialog, shown in figure B.2, a new window can be started. In this new window, shown in figure B.3, the model in Simulink is build.

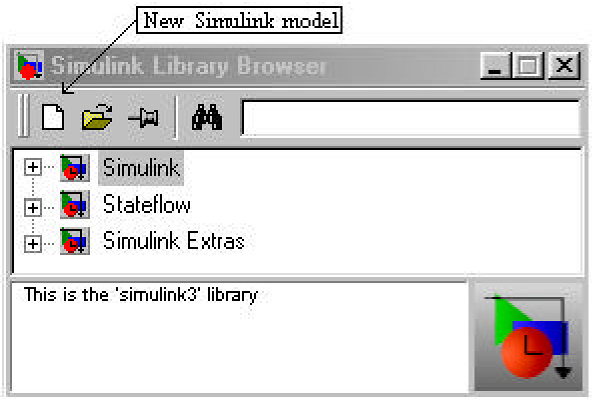


Figure B.2. Simulink Library Browser window.

In to figure B.3 is then function blocks moved from the Simulink Library Browser, in figure B.2.

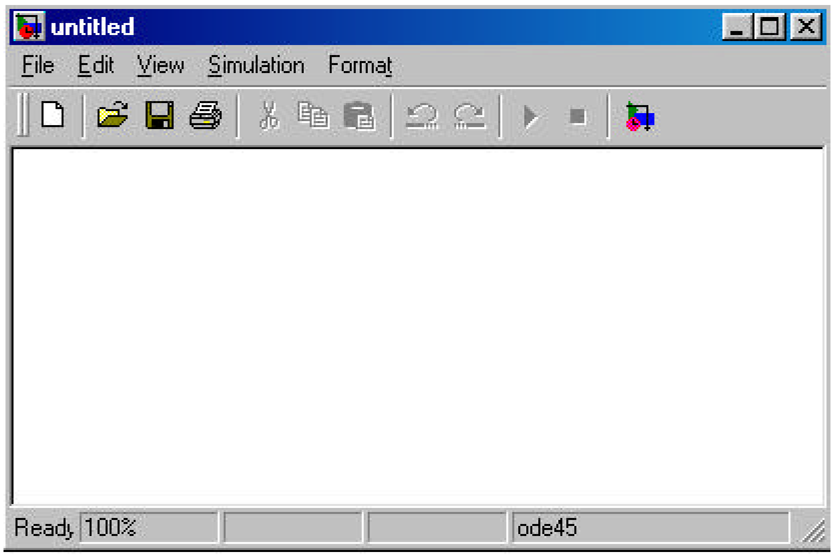


Figure B.3. Window that is used to place function blocks in.

Using the library browser to build a model

In this chapter it is shown how a simple model of a tank, shown in figure B.4, with free flow out is built. The equations for the tank are shown in eq B.1 and B.2.

$$\frac{dh}{dt} = \frac{1}{A}(q_{in} - q_{out}) \tag{B.1}$$

$$q_{out} = a\sqrt{2gh} \tag{B.2}$$

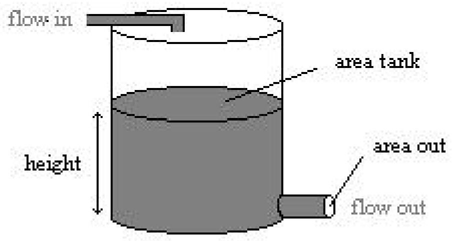


Figure B.4. Tank model.

The used indices are h that is the level in the tank, A that is the area of the tank, q_{in} and q_{out} that are the flow in and out of the tank and a that is the area at the output from the tank.

To build the tank you need a gain block, integrator block and a function block. They are located under the Simulink map in the library browser. Those blocks are moved from the library browser into a window like the one shown in figure B.3. The movement of the blocks is made with “drag and drop”.

To include the special blocks shown in figure B.6, the process-model library shown in figure B.5 should be used. This library is located in Exjobb\simulinklib, and is opened with the open button in the Simulink Library Browser in figure B.2. More about Simulink library can be found in “Using Simulink Version 3”[13]

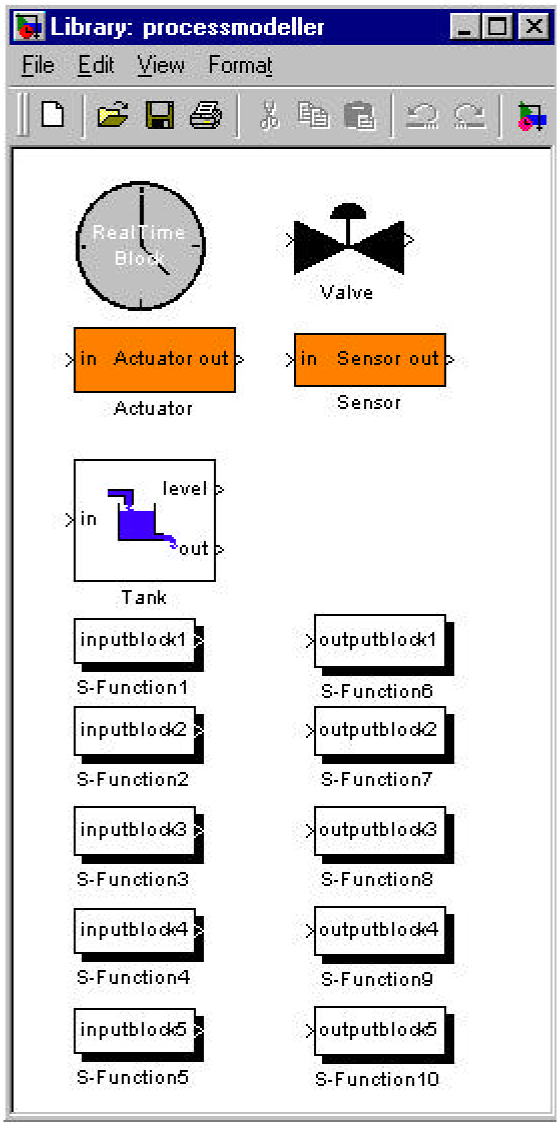


Figure B.5. Library made with blocks that are needed for real time simulations together with Soft Controller.

When all the needed blocks are collected, lines between the blocks can be drawn. The lines between the blocks are drawn as usual in drawing programs. The finished model looks like the one shown in figure B.6.

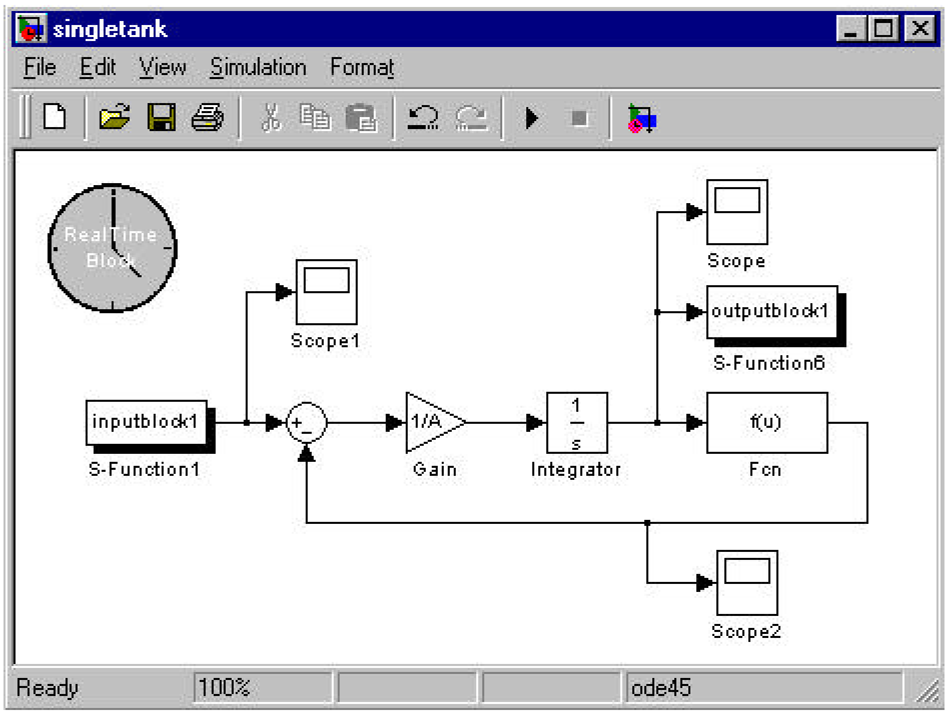


Figure B.6. Tank model in Simulink.

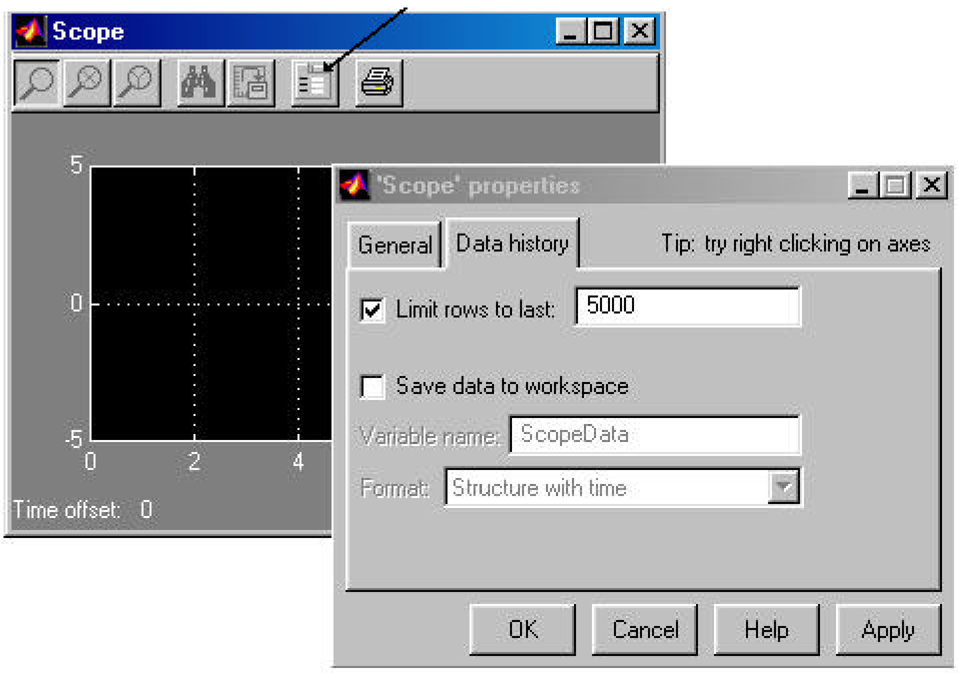
The clock in the left top corner must be included if the simulation shall run in real time. The input-and output blocks are used to communicate with the controller via the Gateway and the OPC-MMS Server. Inside the Fcn block the eq. B.2 is written.

If new S-functions should be used in the simulation, the file sfuntmpl.c in MATLAB11\simulink\src is a template file to start from. It is also a good idea to look in chapter 5.4.10 that describes more about S-functions.

Important things that improve the simulation result in real-time.

To make the simulation work in a good way some details in the scope have to be checked. The figure B.7 appears when you clicked on the scope block in figure B.7. When the button with an arrow on is pushed a new dialog appears. In this dialog a few things have to be checked. The first thing is that the check box “limit row to last ” is activated, and that the number in the edit box is not too high. The test that has been made in this thesis have used the 5000 in the edit box, and with this number the simulation works fine. The second thing that has to be checked is that the check box “save data to workspace” is not activated.

The reasons for that those things have to be checked are that the memory area, used in Simulink, to save values for the plot function is limited. If too many values have to be stored Matlab has to move saved values to other memory areas, which takes time and decreases the performance of the real-time simulation.



[✓] Limit rows to last: 5000

[ ] Save data to workspace

Variable name: ScopeData

Format: Structure with time

Figure B.7. Scope plot and a properties dialog for the scope.

Start the simulation

To make the simulation run for infinite time, the simulation stop time should be set to inf. In figure B.8 it is shown how to come to the dialog where the simulation parameters are chosen.

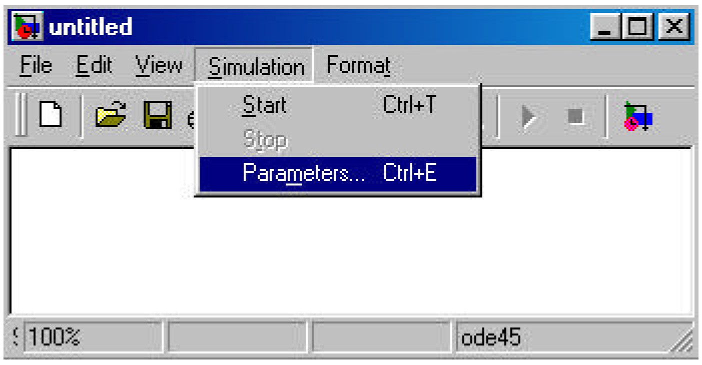
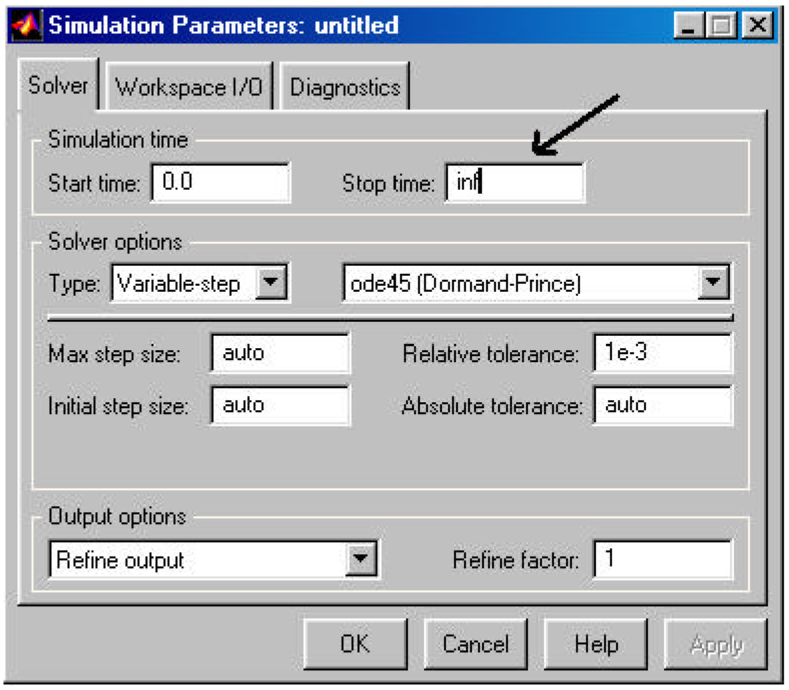


Figure B.8. Drop-down menu for parameter selection.

The parameter dialog is shown in figure B.9.


Simulation Parameters: untitled

Solver | Workspace I/O | Diagnostics

Simulation time

Start time: 0.0 Stop time: inf

Solver options

Type: Variable-step ode45 (Dormand-Prince)

Max step size: auto Relative tolerance: 1e-3

Initial step size: auto Absolute tolerance: auto

Output options

Refine output Refine factor: 1

Figure B.9. Dialog for simulation parameters. The arrow shows how it should look like if the simulation shall run in an infinite loop.

Now a simple model of a tank is made. The model is now ready for simulation with a controller build in Control Builder. There are lots of other things that can be made in Simulink. To read about those things the recommendation is to look at Mathworks homepage [12].

How to use the Simulink model as a process model for controllers is shown in appendix A.