SINGLE-PLATFORM STIMULATED SIMULATORS (SPSS)

COREY WHITE

Single-Platform Stimulated Simulators (SPSS)

by

©Corey White

B.Eng., Memorial University of Newfoundland, Canada (2000)

A thesis submitted to the
School of Graduate Studies
in partial fulfillment of the
requirements for the degree of
Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

May 2007

St. John's Newfoundland

Single-Platform Stimulated Simulators (SPSS)

by

Corey White

Abstract

Simulating a process control system during the design phase is a key step in ensuring the system is designed correctly and meets the design specifications. There are several methods that are commonly used in industry to simulate process control systems, each selected based on the level of detail of the simulation and the cost. Many Engineering firms will choose a simulation method based on cost rather than level of detail because cost has a higher priority.

This thesis will look at some of the ways industry currently simulates process control systems and will compare them on Cost, Fidelity (level of detail of the simulation), Implementation and Conversion. As well, an alternate simulation method will be presented that will strike a balance between cost and fidelity.

Acknowledgements

I would like to thank my Supervisor, Dr. Siu O'Young for his guidance and support throughout this project. I would like to thank Mr. Terry Carter and ABB for their technical assistance on the DCS portion of this project. I would like to thank Schneider Electric for their technical support on the PLC portion of this project. Finally, I would like to thank SEA Systems Limited and the Natural Sciences and Engineering Research Council (NSERC) for their financial support of this project.

Contents

List of Figures

Chapter 1

Introduction

Many industries, such as oil and gas and power generation, demand control systems with a high degree of safety and reliability. To meet this demand, many companies use simulation to thoroughly test and debug their process control equipment before plant commissioning. There are many ways to perform these simulations. Methods involving everything from simple tie-back logic to powerful third party software packages have been implemented in industry. Each method has it's benefits and disadvantages for the user in terms of cost and simulation results.

1.1 Problem Statement

The Instrumentation Control and Automation (INCA) research group at Memorial University is looking into the current methods of simulation and investigating alternatives to these methods. The simulator they are trying to develop will serve three main functions. The first and most important function is to enable control engineers and other interested parties to thoroughly test and debug their control equipment, i.e. software, programmable logic controllers (PLCs) and communication networks, without using any process equipment. This simulation will be performed prior to the installation of the system. Secondly, the simulator will provide a means by which the system operators can be trained on how to use the system and how to deal with problems that can occur during the operation of the plant. The third function will be to provide personnel with a means to "recreate" a fault that occurred within the process to determine exactly what caused the fault.

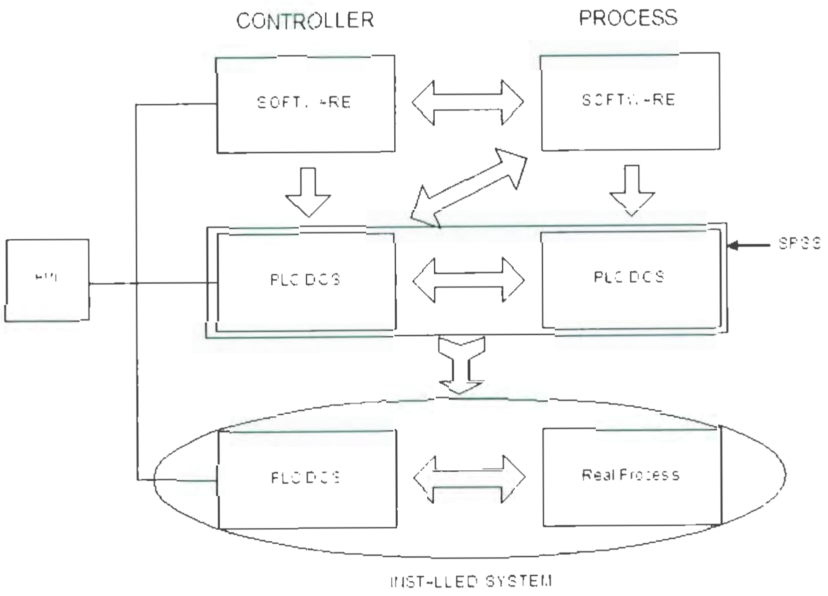


Figure 1-1: Graphic Representation of Simulation Methods

The proposed method assumes that the theoretical analysis of the control system has been completed and that the characteristics of the individual components that make up the process are understood. In order to achieve the above mentioned functions, all of the variables, such as communication loop delays and logic execution time, need to be included in the simulation. This will bring the simulation as close to the real process as possible.

1.2 Analysis of Typical Simulation Methods

There are typically three ways in which control systems can be simulated: Software - Software, Hardware - Software, Hardware - Hardware. A control systems developer can choose any method and use it exclusively during the design process or they may wish to progress to the different modes of simulation as the design process evolves. Figure 1-1 is a graphical representation of these simulation methods and how they relate.

In Chapter 2 these simulation methods will be discussed with examples of how they are used in industry today. These methods will be evaluated on the following criteria:

- Cost
- Fidelity
- Implementation
- Conversion

The cost analysis will look at what equipment (computers, control equipment, software), real estate and human resources are required to implement the simulation. The analysis will not provide an exact value for the cost of each method rather it will rate each method relative to the others. For example, simulation A may get a rate of $1 because it only requires one computer where as simulation B may get a value of $5 because it requires three computers, seven controllers and a large office.

The fidelity analysis will look at how realistic the simulation is. It will look at how closely the simulation matches the real process and assign a percentage to it. A simulation that can provide process trends that match the trends from the real process will get a value of 100%. Simulations that just trigger the inputs individually will get a value of 5%.

The implementation analysis will look at the level of difficulty in setting up the simulation. Each simulation method will be given a rating of one to ten, one being easy and ten being difficult. For example, a simulation that takes one person a couple of hours to set up will be given a score of one and a simulation that takes five people three weeks to set up will be given a score of ten.

The conversion analysis will look at the level of difficulty in converting the control logic from the simulation environment to the plant control system. An easy conversion, e.g. transferring the logic directly from the simulation to the control equipment, will get a value of one. A difficult conversion, e.g. having to rewrite the logic from scratch in the control equipment, will get a value of ten.

Chapters 4, 5 and 6 will provide an alternate method of simulation referred to as Single-Platform Stimulated Simulation (SPSS). Finally, Chapter 7 will summarize the simulation analysis and provide some options for industry.

The main focus of this thesis is process control simulation methods currently being used in industry. Therefore the background information came from company web sites and marketing materials and from my experience from working in the automation industry for the past five years.

Chapter 2

Current Industrial Practice

2.1 Software to Software

The first simulation method mentioned in Chapter 1 is Software to Software simulation. With this method, both the control logic and the process are modelled in PC based software packages. There are numerous software packages on the market that do this type of simulation. Some packages like Hyprotech's Hysys and Kongsberg Simrad's ASSETT are able to simulate both the process and the control logic. These programs are specifically designed to simulate engineering processes, such as petrochemical processes. As a result of this, very detailed mathematical equations are used to model equipment and physical properties, such as chemical reactions, resulting in highly realistic simulations. Other packages like Allen Bradley's Emulate and Modicon's Concept provide a simulation of their PLC's that logic can be loaded into. Inputs can then be manually triggered to simulate process changes. This provides a very quick and easy way to check the logic for errors.

The Software to Software simulation method is strictly a PC based method, meaning that the only capital requirements are a PC and the simulation software. This makes this method very useful during all stages of development for a project because the software can be resident on the design engineer's computer and be accessible at all times. There are six main phases that make up the life of an engineering process:

1. Design
2. Construction
3. Commissioning
4. Start-up
5. Operation
6. Maintenance/Upgrades

Within this *life cycle* there are several key points at which an engineering firm may want to run a process simulation. These points are shown in Figure 2-1 [2]. Many sim-

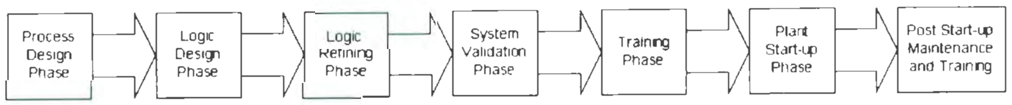


Figure 2-1: Process Simulation Phases

ulation software developers, such as Kongsberg Simrad, have designed their software packages with the goal in mind that one simulation package can be used for every stage of the project life cycle, thus reducing costs during plant start-up and commissioning and during the plant's operational life [2]. Figure 2-2 [3] is a marketing image from Kongsberg showing that ASSETT is a Life Cycle simulator.

Since the Software to Software method is PC based, it is able to take advantage of the computational power of PCs. Even the most basic PC can solve complex mathematical equations that are virtually impossible to solve any other way. This means that process with complex mathematical models, such as three phase piping and reactors, can be simulated on a PC making the overall simulation much more realistic.

Figure 2-2: Life Cycle Simulation

There are several advantages to using the Software to Software method:

1. It is economical. Compared to other methods of simulation, this method is economical because the only capital requirements are the software and a computer.

2. No specialized equipment required. Since this is a PC based simulation method, there is no specialized equipment, such as control equipment, required to run the simulation.

3. Easy to set up. Again, since this is a PC based simulation, the only set up that is required is installing the software and creating the plant model. In some commercially available software packages, creating the plant model is as simple as connecting together pipes and tanks and valves. This can be seen in the HYSYS screen shot in Figure 2-3

The disadvantages to this method of simulation are:

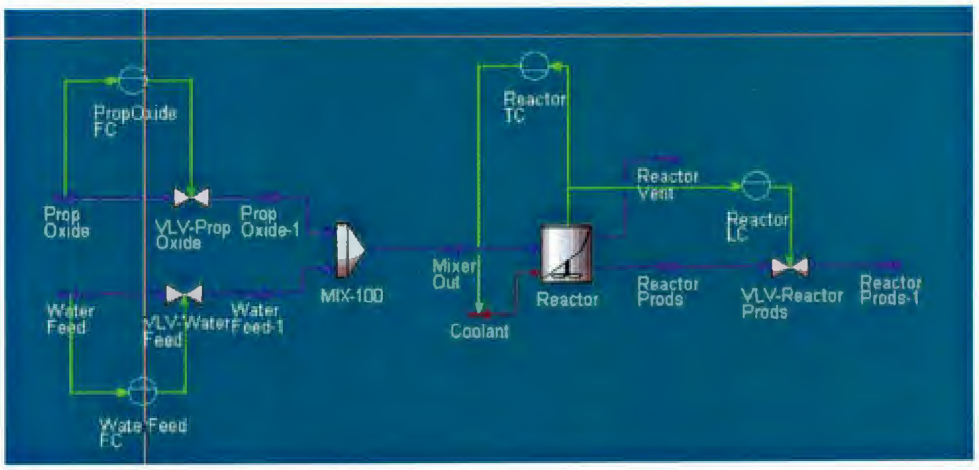


Figure 2-3: HYSYS Screen Shot

1. There are some system performance characteristics that are difficult to model. For example, communication loop timings and logic execution time and priority are hard to model because they depend on the type of control equipment used, size of logic and communication network configuration.
2. Time consuming to convert control logic. Any logic developed for the simulation cannot be directly loaded into the control equipment. It first has to be converted to the language that the equipment understands. Since this language is proprietary, there are no software conversion tools available that convert logic from the simulation software into the control manufacturer's software. This means all logic will have to be created from scratch in the controller software, which is very time consuming and is open to errors.

2.2 Hardware to Software

The second method listed is the Hardware to Software method. With this method, the control logic is executed in the control equipment, such as a PLC or DCS, and the process is modelled in a PC based software program, similar to the type used in the previous method. As with the previous method, the software provides a highly accurate mathematical model of the process. Input Output data is passed between the PC and the control equipment via a custom communication network.

The advantages of the Hardware to Software method are:

1. This method is more realistic because it uses the same type of equipment that will be used to control the actual process. This allows the characteristics such as communication loop delays and logic execution time to be incorporated into the simulation.

2. All control logic created for the simulation is already in the language that the control equipment understands. Therefore no logic conversion is required.

3. Easy to set up. Since the same software used to model the plant in the Software to Software method is used in this method, the ease of set up would be the same.

The method also has some disadvantages:

1. Depending on the equipment being used, there may be some significant real estate required to house the simulation.

2. Establishing communications between these two items can be a difficult and costly process because typically control networks are closed, proprietary networks. They require special equipment or programming to allow devices, other than those used to control the process, to access the network.

Incorporating the control equipment into the simulation makes this method ideal for operator training. It allows the simulator to be set up like the actual plant control room and allows the operators to not only get familiar with the operator consoles, but with all the control equipment and how it all interacts. This method is also useful for post startup logic and HMI updates for systems, such as offshore oil and gas platforms where having someone on site for updates is a very costly process. This simulation method allows updates to be fully tested on shore and then sent to qualified personnel on site to load.

The Hardware to Software simulation method was the method used by the Terra Nova Alliance to train operators for the Terra Nova Floating Production, Storage and Offloading (FPSO) vessel.

2.3 Hardware to Hardware

The final simulation method mentioned above is the Hardware to Hardware method. With this method, both the plant model and the control logic are executed in control equipment. Two of the ways this type of simulation is being done in industry are:

- Slave controller contains a discrete set of data, such as sequence of events data or I/O data from a similar process, that triggers the inputs in the control logic. Resulting outputs determine next set of inputs.
- Inputs replaced in the logic with switch blocks (for digital inputs) and constant blocks (for analog inputs). The switch blocks are then manually turned on and off to simulate changing digital inputs and the values in the constant blocks are changed to simulate changing analog inputs.

This method is a low fidelity simulation method because the data set used to trigger the inputs is discrete. This means that the accuracy of the simulation depends on the sample rate of the data. For example, data that is sampled every millisecond would provide a more accurate simulation than data sampled every second because there is a greater chance of detecting transients and spikes with the higher sample rate and having transients and spikes in the simulation provides greater information on system performance.

Engineers and technicians at Bailey SEA (Nfld.) Limited and SEA Systems Limited use both types of Hardware to Hardware simulation to simulate logic and HMI updates for the Hibernia offshore oil platform and the Terra Nova FPSO. They use the slave controller method to simulate the Fire and Gas (FGS) Emergency Shut Down (ESD) control systems for Hibernia and they use the second type, typically referred to as the "Tie Back" method, for all other system updates.

The main use for the Hardware to Hardware simulation methods is during control logic and operator interface development and upgrading. Since all the input values have to be changed manually in the tie back method, it is too time consuming and inefficient to use this type of simulation for operator training. Also, by manually changing the input values one at a time, you only observe how a small section of the control system reacts to input changes and you are unable to see how the system acts as a whole to input changes. For example, a steam flow control valve for a boiler is being simulated to determine PID tuning values. A constant block is used to simulate the steam flow and is manually adjusted to simulate changes in flow rate and allow tuning values to be determined. This process assumes a constant steam flow. However, in reality steam flow is never constant and changes in steam flow cause changes in pressure upstream of the valve. Since the steam boiler is a control process as well, changes in outlet pressure and flow will cause the controller to adjust combustion and water flow to bring the pressure and flow back to the setpoint. These adjustments cause fluctuations in steam flow which affect the flow control valve. By tuning the valve based on constant flow the valve may not respond properly to fluctuations caused by the boiler control system.

As can be seen in the above discussion and in the examples in Chapter 5, each simulation method is best suited for one particular type of simulation. For example the Software to Software method is ideal for high level process design and optimization simulations but is overkill for basic logic update testing. These types of simulations are considered "Fit For Purpose" simulations, meaning a simulation method is chosen based on what the simulation is to achieve. Figure 2-4 illustrates this idea. If the main purpose of the simulation is operator training, the designer may want to choose

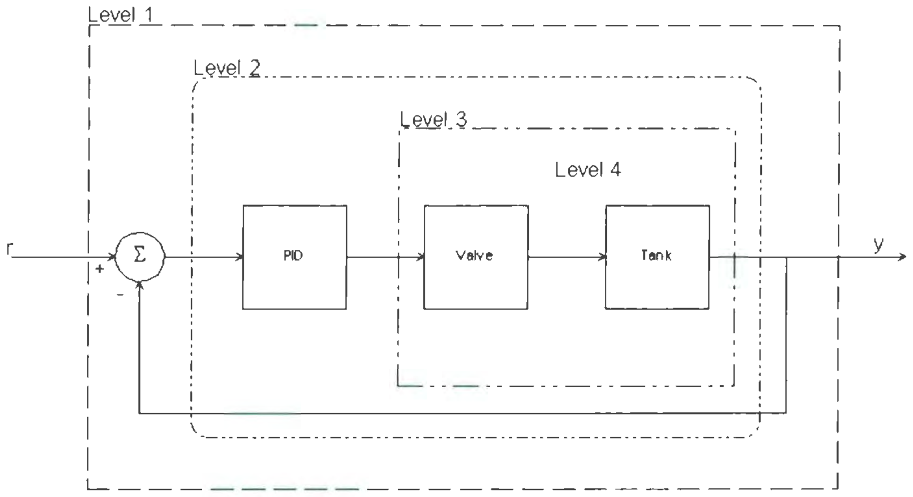


Figure 2-4: Fit for Purpose Simulation

Levels 1 or 2 where the process and control are assumed to be ideal and the output is the reference, or setpoint, multiplied by a constant. If the simulation was to be used for logic update testing, then Level 3, where a non-ideal controller is controlling an ideal process, would be more applicable. Finally, if the purpose of the simulation is to analyze the process to determine areas for optimization, then Level 4 simulation could be used. Here, all process equipment, tanks, valves, pipes, are modelled in the simulation to provide the most realistic simulation possible.

To add another option to the above list of simulation methods, this thesis proposes to use the control equipment to simulate both the process and the control logic. This method would be called Single-Platform Stimulated Simulation and would fall between the Hardware to Software and Hardware to Hardware methods It would implement the mathematical model of the plant within the control equipment, similar to the models used in the Software to Software methods. The idea of this method is to provide a more realistic simulation like in the software to software method while saving the user the cost of purchasing a third party software package and communications link. As well, as with the hardware to hardware method, this method would save significant engineering time because all the logic for the simulation is created in the same programming environment that will be used for the actual plant.

Chapter 3

A Framework for SPSS

In the previous chapter the different methods used to simulate a control system were discussed. This chapter will discuss how a plant model is developed and some of the options available to simulate the model.

3.1 Control System Definition

Any device or group of devices that manipulate one or more variables of a system to achieve a desired result is called a control system [4, p.4]. More specifically, if these devices monitor the actual result and manipulate the variables based on the difference between the actual and desired results, they form a Closed Loop Control System. Feedback is the process of monitoring the actual result and comparing it to the desired result [4, p.10]. One example of a closed loop control system would be driving in a car. The desired result would be car positioned in the middle of the road. The driver visually monitors the current position of the car between the lines on the road and, based on the current position and where he/she wants the car to be, adjusts the steering wheel to change the car's position [4, p.5].

The object or objects being controlled is generally referred to as the plant. The desired result is called the setpoint. The means by which results are monitored are called sensors. The devices that monitor and manipulate system variables are referred to as controllers. The devices used to manipulate variables are called actuators [5, p.2]. In the example of the driver above, the car is the plant, the driver is the controller, the drivers eyes are the sensors, the position of the car in the middle of the road is the setpoint and the drivers arms are the actuators. One control system, or loop, in an industrial boiler maintains the level of water in the tank. The sensor is a level transmitter which converts the level to an electrical signal (input). This signal is then read by the controller, which is typically a Distributed Control System (DCS) or a Programmable Logic Controller (PLC). The controller will subtract the setpoint from the level and apply a control algorithm, such as proportional integral derivative (PID), to the value to generate the actuator adjustment value (output). The actuator is a valve on the water supply side. The output value will either increase or decrease the valve opening to adjust the flow of water to the tank and keep the level constant. Figure 3-1 is a typical graphical representation of a closed loop control system.

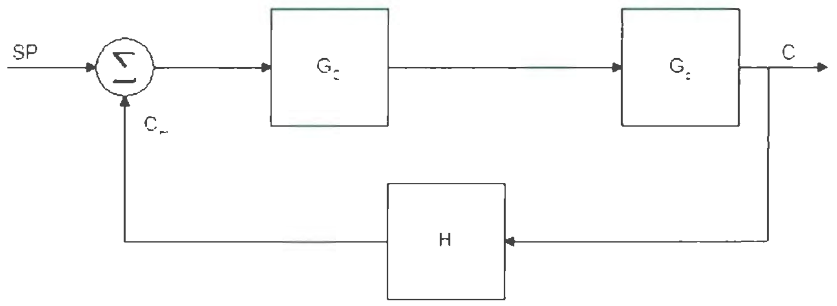


Figure 3-1: Graphic Representation of System Transfer Function

A control system is designed on two levels: abstract and physical. The purpose of the abstract level design is to obtain an appropriate control strategy and ensure the system will perform as outlined in the specifications. The purpose of the physical level design is to obtain the appropriate hardware and software required to implement the control strategy [5, pp.4,5].

In order to implement the abstract level design, a model of the plant must be developed. One way to develop this model would be through fundamental principles. With this method, key components of the plant are identified and mathematical equations that describe these components are written, along with the equations that link the components together [5, p.6]. For example, the level of a liquid in a tank with an inlet pipe and an outlet pipe connected to the bottom is described by the following equations:

$$\begin{aligned} h_1(t) &= \frac{1}{A}\int q_{net}dt + h_1(0) \\ q_{net} &= q_{in} - q_{12} \\ q_{12} &= C_V\sqrt{\rho g h_1 - \rho g h_2} \end{aligned}$$

Where: h(t) = level in tank at time t

h_1 = level in tank 1

h_2 = level in tank 2

q_{in} = flow rate into tank

q_{out} = flow rate out of the tank

q_{12} = flow rate between tanks, in this case between tank 1 and tank 2

A = Cross sectional area of tank

C_V = valve coefficient

ρ = density of water

g = gravitational acceleration

Another way to develop the plant model is to look at the plant as a black box. With this method, the model starts out as a simple first order transfer function and, based on observations of the inputs and outputs of a similar process, adjustments are made to the transfer function to match the observations [5, p.6].

3.2 Simulation Options

With the mathematical model of the plant and control logic determined, there are a number of options that can be used to test and analyze the performance of the control system. Figure 3-2 lists the different options that can be used. They are based on whether the plant and controller are analytical, simulation or real.

Plant \ Controller	Analytical	Simulation	Real
Analytical	1 "Text-book" Design/analysis Bode Plot, Root Locus, State-Space	2 Hysis/Simulink	3 SPSS
Simulation	4 Hysis/Simulink	5 Hysis/Simulink	6 Stimulated Simulation
Real	7 N/A	8 N/A	9 Online

Figure 3-2: Simulation Matrix

Box 1 represents the "text book" analysis of the control system, where both the plant and the controller are analytical. The model for this analysis is usually the system transfer function. Variables such as sensor noise, actuator noise, pump output, etc. are assumed to be some constant value to make the analysis simpler. Examples of typical "text book" methods to analyze these transfer functions are bode plots and root locus for single-input, single-output systems, commonly known as classical design and state-space models for multi-variable systems [5, p.231].

The "text book" analysis is usually done during the design phase of a project. Engineers, using computer programs such as Matlab and Simulink, would be able to generate graphs of the system transfer function and determine the performance characteristics. Also, by varying parameters such as the loop gain or sensor noise, the system response can be observed. This can lead to a set of performance characteristics that can guide the design of the physical control system.

The system analysis for Boxes 2 and 4 in Figure 3-2 has one component analytical and one component simulation. The equations for the analytical component would be similar to the analytical equations used in the "text book" method. The simulation component would be a collection of the mathematical equations that describe each piece of the component being simulated. For example, a plant containing a pipe that feeds liquid to a tank through a control valve is the component being simulated. The model would consist of the result of the mathematical equation that describes the flow rate of the liquid through the pipe, modified by the equation for the control valve, being used in the equation describing the level in the tank.

Simulations that are done to study and optimize the process fall into box 4. As discussed in Section 2.1, programs like Hysis, ASSETT and D-Spice are used to generate detailed mathematical representations of the plant by connecting together graphical representations of the equipment used in the process. The program then solves the mathematical equations to generate process parameters such as flows, pressures and levels which are checked by system designers to ensure the process is functioning within the specifications. At this stage the control equipment characteristics are not critical to the analysis so simple control equations are used.

Simulations performed during the design phase of new control equipment or during control logic design would fall into box 2. Here the controller is simulated and the plant is analytical. Control equipment manufacturers such as Modicon and Rockwell Automation provide a controller simulation program as part of their logic development software packages. These simulation programs execute the control logic the same way and at the same rate as the real controllers do. They also allow the user to change inputs, observe outputs and monitor program execution. These controller simulation programs can also be modified by control equipment manufacturers to test new controller architectures.

Simulations that would fall into box 5 are similar to those that fall into box 4. The main difference is that the particular operating characteristics of the control equipment are taken into account. Most control equipment have a strict way in which to execute logic. First the controller CPU will execute the logic from start to finish then it will take care of data management, such as reading inputs, writing output data and communicating with other controllers on a communication network. This process occurs continuously while the controller is running. The time it takes for a controller to complete one cycle through the logic and data management is called the processor scan time. Figure 3-3 illustrates how an Allen-Bradley PLC 5 performs this process [6]. This scan time is dependant on the amount of control logic and the

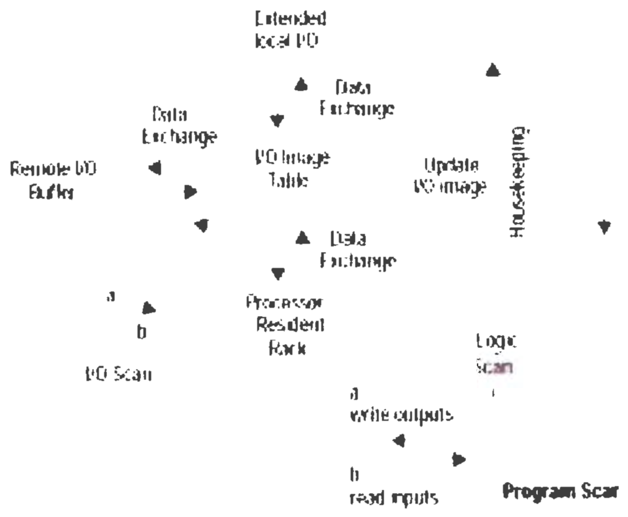


Figure 3-3: Allen-Bradley PLC 5 Program Scan

amount of data that the controller has to manage. If the scan time is known or can be estimated, time delays can be added into the controller simulation to delay when data gets transferred to the plant model or when certain sections of logic get executed.

Box 6 simulations have the real control equipment connected to a PC running the same simulation software that is used in simulations covered by box 4 and box 5. These simulations would come under the Hardware to Software simulation method discussed in Section 2.2.

Single-platform stimulated simulation falls into box 3. Here analytical models of the plant, similar to those used in the "text book" analysis, are loaded into control equipment similar to the equipment being used in the real plant. Depending on the configuration of the control system, several controllers can be networked together with some controllers sharing the plant model and other controllers sharing the control logic. This allows for real controller scan times and communication loop delays to part of the simulation.

The following three chapters will develop SPSS simulations. In Chapter 4 will define a plant model to be simulated and it will show how the model is used in an SPSS simulation in a PLC. Chapter 5 will show how this same simulation can be done in a DCS. Finally, Chapter 6 will show the simulation of a typical pulp and paper mill process.

Chapter 4

PLC Simulation

Over the past number of years, the computing power of the programmable logic controller has made significant advancements. When the first PLC was developed in the late 1960's, early 1970's it contained 1 kilobyte of memory [7]. Today an Allen Bradley PLC can can be purchased with 8 megabytes of memory and a math coprocessor for performing floating-point calculations [8]. The PLC has gone from containing small ladder logic programs that turn motors on and off and open and close valves to large function block based programs that control complex, safety critical systems such as oil and gas processing [9].

In Chapter 2, it was discussed that the most effective simulation method was to use a third party software package that contained a mathematical model of the plant. Since most PLC and DCS programming environments contain all the function blocks required to build mathematical models, the idea for this research project was to develop a Single-Platform Stimulated Simulator (SPSS) using standard industrial control equipment. The hope for this project is that this method will make the simulations easier to develop and more flexible because there is no need for the user to learn a third programming language and it may be easier to modify the plant model to provide a more realistic simulation. For example, this method may make it easier to incorporate multiple controllers into the simulation to bring the timing characteristics of the simulation closer to the timing characteristics of the real system.

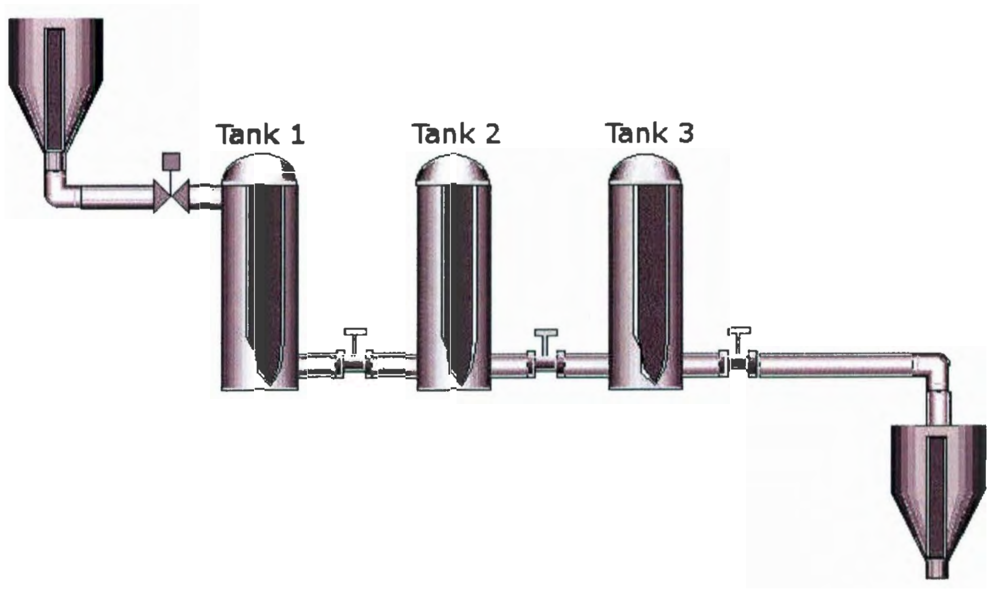


Figure 4-1: Diagram of Three Tank Process

4.1 Plant Model

The first stage of the project was to determine a simple process to simulate and the equations required to simulate it. To make the modelling part of the simulation easier, a process that consisted of three tanks containing water was created. The water is able to flow from tank 1, through tank 2, to tank 3 via piping that connects each of the tanks at the bottom. A controller monitors the level in tank 2. Once the level drops below a set point, the controller opens a control valve allowing water to flow into tank 1. Once the level in tank 2 goes above the set level, the controller closes the control valve allowing the level in tank 2 to drop. A diagram of this system is shown in Figure 4-1. This process can be described by the following equations:

Tank 1:

$$\begin{aligned} h_1(t) &= \frac{1}{A}\int q_{net}dt + h_1(0) \\ q_{net} &= q_{in} - q_{12} \\ q_{12} &= C_V\sqrt{\rho g h_1 - \rho g h_2} \end{aligned}$$

Tank 2:

$$\begin{aligned} h_2(t) &= \frac{1}{A}\int q_{net}dt + h_2(0) \\ q_{net} &= q_{12} - q_{23} \\ q_{23} &= C_V\sqrt{\rho g h_2 - \rho g h_3} \end{aligned}$$

Tank 3:

$$\begin{aligned} h_3(t) &= \frac{1}{A}\int q_{net}dt + h_3(0) \\ q_{net} &= q_{23} - q_{out} \end{aligned}$$

Where: h(t) = level in tank

q_{in} = flow rate into tank

q_{out} = flow rate out of the tank

q_{12} = flow rate between tanks, in this case between tank 1 and tank 2

A = Cross sectional area of tank

C_V = valve coefficient = 55

ρ = density of water = 1000

g = gravitational acceleration

Each tank was set to be 1m high with a radius of 25cm. The pipes joining the tanks were set to be 25cm long with a radius of 1.27cm. This pipe radius meant that the valve coefficient, C_V, for the manual valves between the tanks was 55. The tank cross sectional area (πr^2) was 0.1963495 m^2. This process was controlled such that when the water level in tank 2 fell below 0.45m, the controller would open the source control valve, allowing water to flow into the system and bring the water levels back up. Once the water level in tank 2 rose above 0.55m, the controller would close the source control valve allowing the water levels to drop.

To ensure the results from the PLC simulation are correct, the simulation was run in Simulink. The code for this simulation is shown in Figure 4-2. This simulation provided a graphical plot of the change in tank height with respect to time (Figure 4-3)

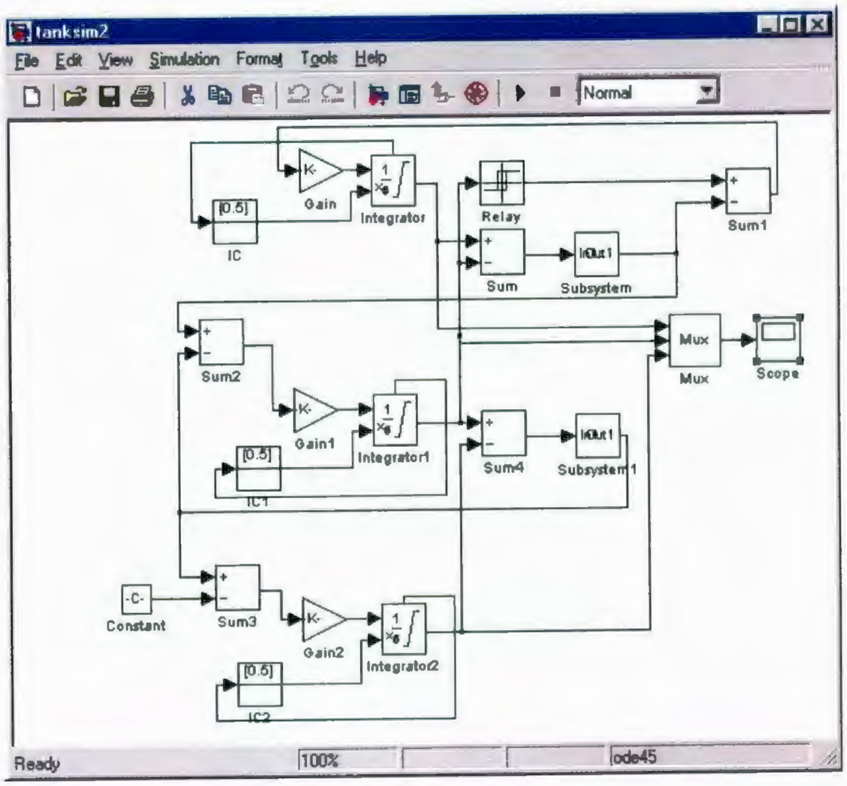


Figure 4-2: Simulink Code

that was used as a comparison for the simulator that was created during this project. If the results from the simulator created for this project matched the Simulink results, then it will be assumed that the new simulator is correct as well.

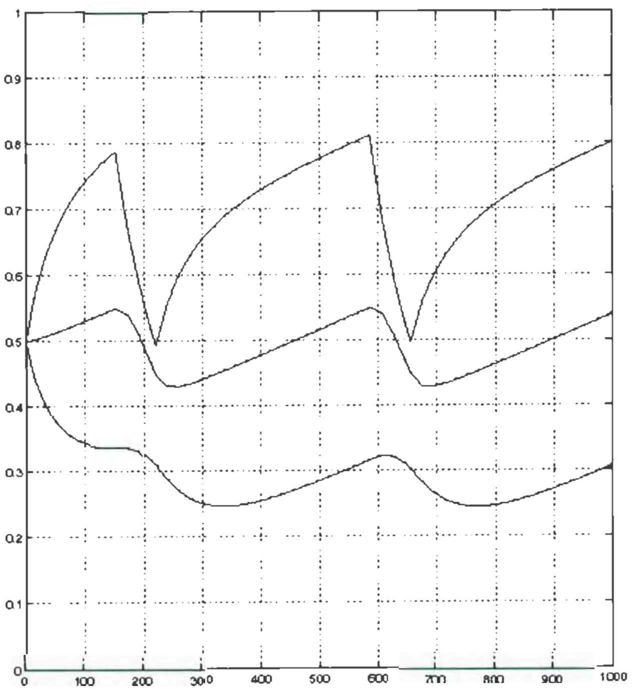


Figure 4-3: Graph of Water Level vs Time

4.2 Control Equipment

Once a reference set of results was established, the plant model was recreated in the PLC programming environment. The control equipment that was used for this part of the project was a Modicon Momentum PLC, Concept programming software and Factory Link operator interface software. All of this equipment was purchased from Schneider Electric. The reason this system was chosen was because of it's compact size and Concept was based on the IEC 1131 standard for PLC programming. IEC 1131 is an international standard that specifies five different languages for programming

PLCs. These languages are Ladder Diagram, Function Block Diagram, Structured Text, Sequential Function Chart and Instruction List [10]. This simulation uses the Function Block Diagram language because to the user, this language is very similar to Simulink and D-SPICE, where mathematical equations are generated by connecting the appropriate functions together. As well, many DCS systems, such as ABB's Infi 90 system, use function block based languages. Using a language that is common to a number of vendors means the simulation is not limited to one vendor.

Figure 4-4 shows the simulation logic that was created in Concept. As this figure shows, the logic was split up into six sections, with each section representing a component of the system, such as a tank. The reason for creating the logic in this fashion was to show that component function blocks can be created to represent a particular piece of equipment. These blocks would mask the actual logic required to model the piece of equipment, so that the simulation logic resembles a process and instrumentation drawing instead of a complex mathematical model. This also makes the Concept simulation more like the commercially available simulation software.

An initial simulation was run using the 16-bit PLC simulator that was included with Concept. This was a very useful first step because it showed how the variable values changed during the simulation. As well, it provided a means to ensure that the code was written correctly and that it did not contain any errors. When the tank simulation was initially created, several function blocks were placed in the wrong order. When this code was run in the simulator, an error message was displayed and some of the variable values were shown as NAN (not a real number). These two pieces of information suggested that the error was due to attempting to take the square root of a negative number in the calculation of the flow rate between the tanks.

After this error was corrected, the simulation ran with no errors. While the simulation was running, it was observed that the variable values were changing as expected. Upon initial startup the source control valve was open and the level values were increasing. When the Level2 variable (water level in tank 2) reached 0.55, the

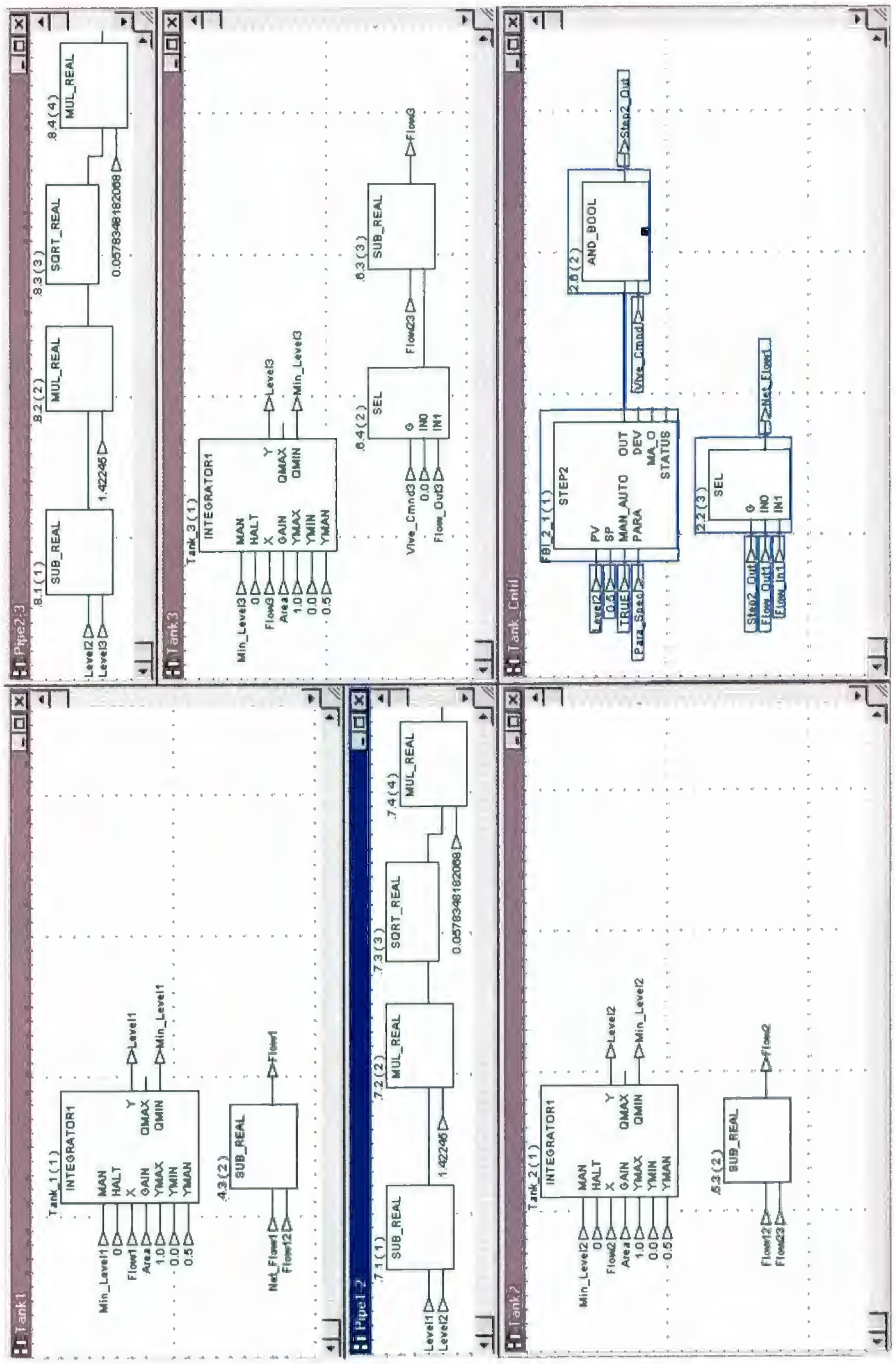


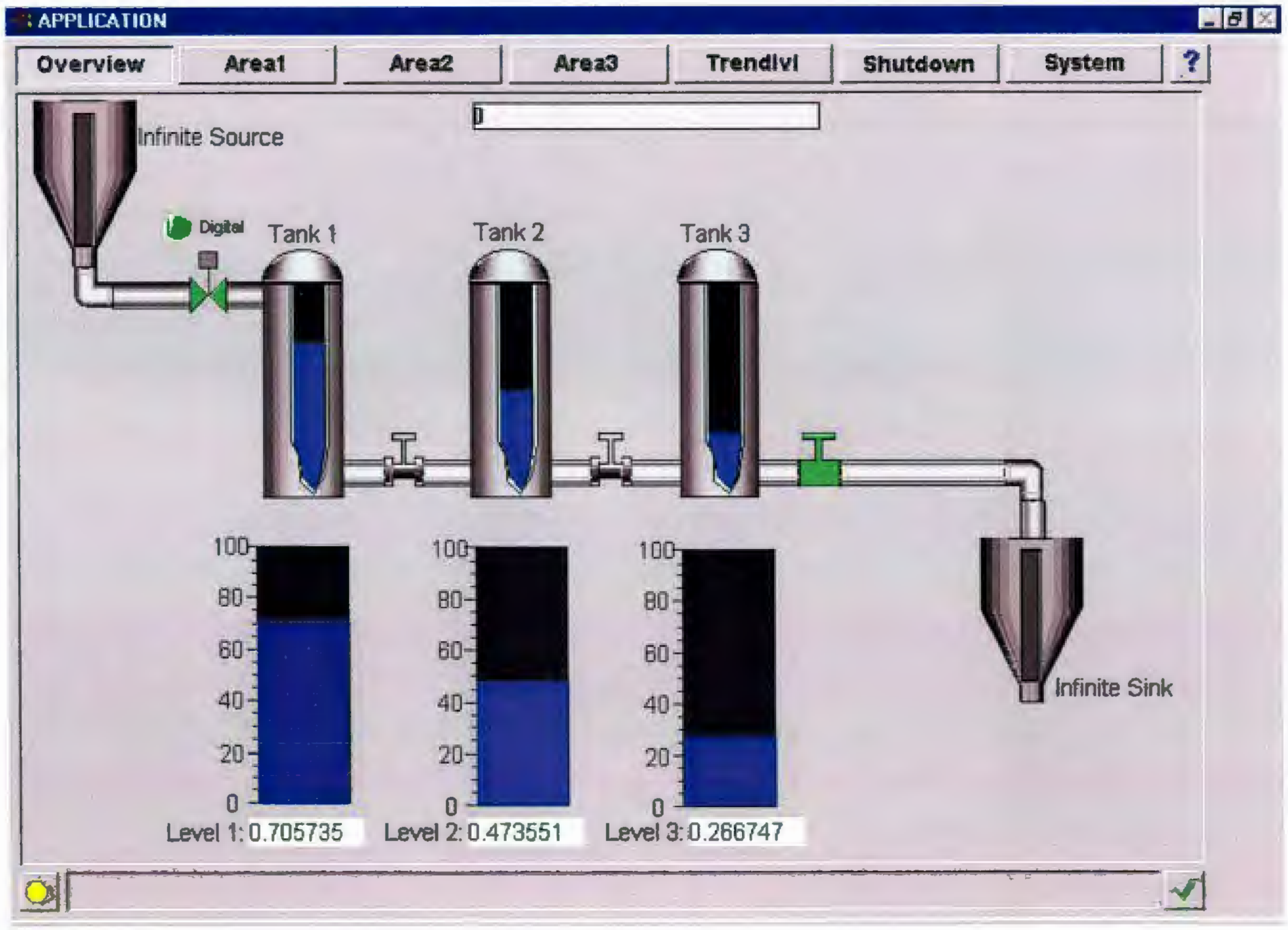
Figure 4-4: Concept Simulation Logic

source control valve closed and the water levels dropped until Level2 reached 0.45 when the control valve opened again and the levels started to increase.

4.3 Graphical Display

Although using the 16-bit PLC simulator was useful in proving that the Concept code was correct and that the variable values changed as expected, i.e. increased to 0.55 and then dropped to 0.45, there was no indication of how closely the simulation matched the Simulink results. Concept does not provide a way to plot variable values on a graph. As well there was no way to test the operator interface. To make the simulation a little more realistic and more useful to the user, the simulation was downloaded and run on a PLC and Factory Link was used to display the results. Figure 4-5 is a screen shot of the Factory Link application created for this project. Factory Link was configured to read the water levels from registers within the PLC, read the controller state (on or off), read the state of the tank 3 outlet valve (open or closed). The water levels were displayed in three different ways. First the changing levels were displayed in the form of the animation of water levels rising and falling in tanks. Secondly, the changing levels were indicated on the bar graphs below the tanks. Thirdly, each level was plotted on a trend chart on another screen. The controller state was indicated by a digital light near the source control valve. When the controller was on, the light was green and when the controller was off the light was red. The tank 3 valve position was indicated in a similar fashion using the manual valve graphic on the outlet of tank 3. Additionally, the source control valve could be turned on and off by clicking the control valve graphic on the Factory Link screen.

The Factory Link application worked exactly as expected. All values were displayed as expected and the configured alarms worked properly. However the level trend did not plot the level values as expected. It was hoped that the trend would produce a plot of water level versus time, for each tank, similar to the Simulink plots.



Level 1: 0.705735 Level 2: 0.473551 Level 3: 0.266747

Figure 4-5: Factory Link Screen for Simulation

This did not occur. Although the level trend did plot the changing water levels, the scale was not big enough to provide an overall picture of how the system was performing. As well there wasn't an easy way to print the trend results. Since Factory Link automatically generates a file containing all the values used in a trend, it was decided that this file would be imported into Microsoft Excel and the values would be plotted there. This plot is shown in Figure 4-6.

A close comparison of the Simulink results and the Factory Link results shows that the two simulations match very closely. Except for a difference in scales, the two simulations are identical. This leads to the conclusion that it is possible to use Concept to create both the simulation and the control and have the code run within a PLC.

4.4 System Timing

One of the most important issues in any control system is timing. Most industrial processes today have a centrally located controller communicating with numerous remote input/output racks located throughout the plant. Figure 4-7 is an example of such a PLC network used at a Pulp and Paper mill [11]. In this particular setup, each major component of the paper making process has it's own controller communicating with several remote I/O racks as well as the DCS system. To start and stop motors, commands come from the operator interface through the DCS to the PLCs. The start command is a pulse from zero to one and the stop is a pulse from one to zero. The PLCs interpret these pulses and turn the appropriate outputs, in the I/O racks, on and off. The question for logic designers is how long should these pulses be. If the pulse is too short, the PLC would not see it or would not have time to energize the output. If it is too long it may interfere with emergency stop logic in the PLC. Being able to simulate the entire system with communication time delays will provide engineers with the ability to determine the appropriate pulse lengths prior to commissioning.

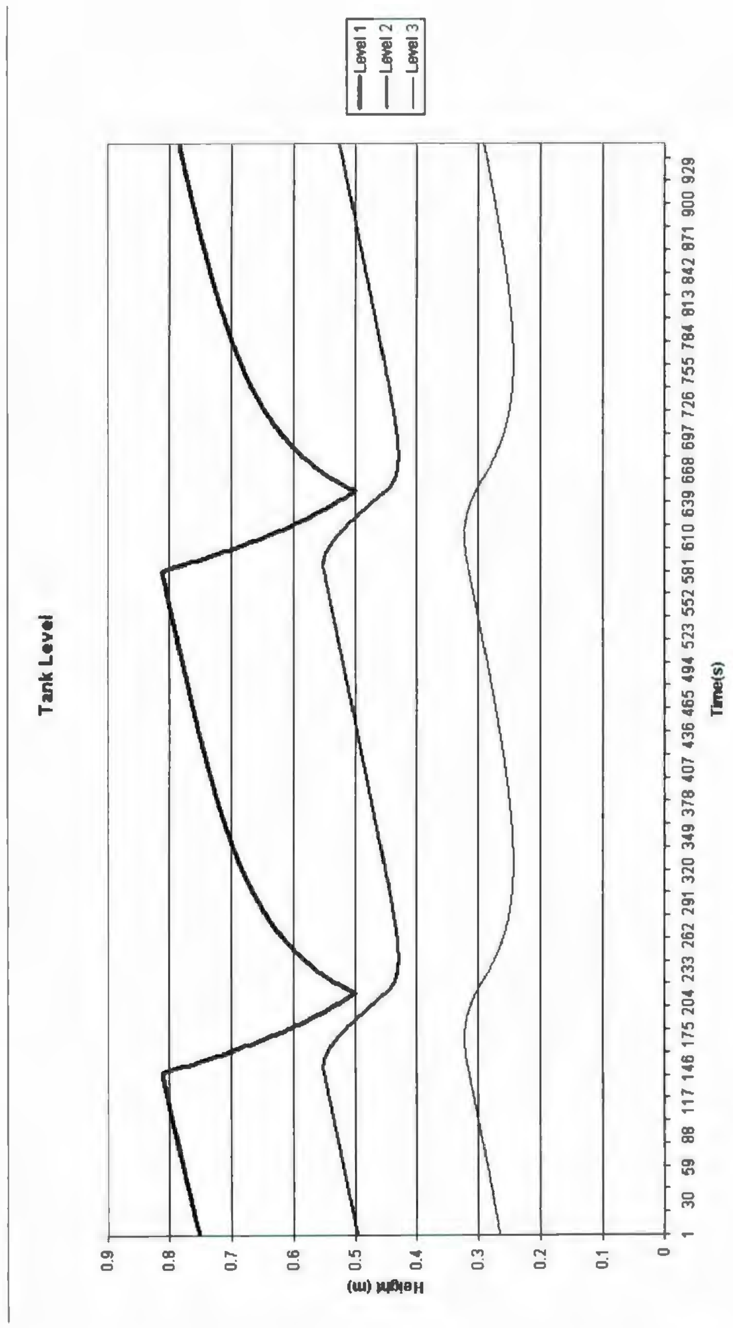


Figure 4-6: Graph of Water Level vs Time For One PLC Simulation

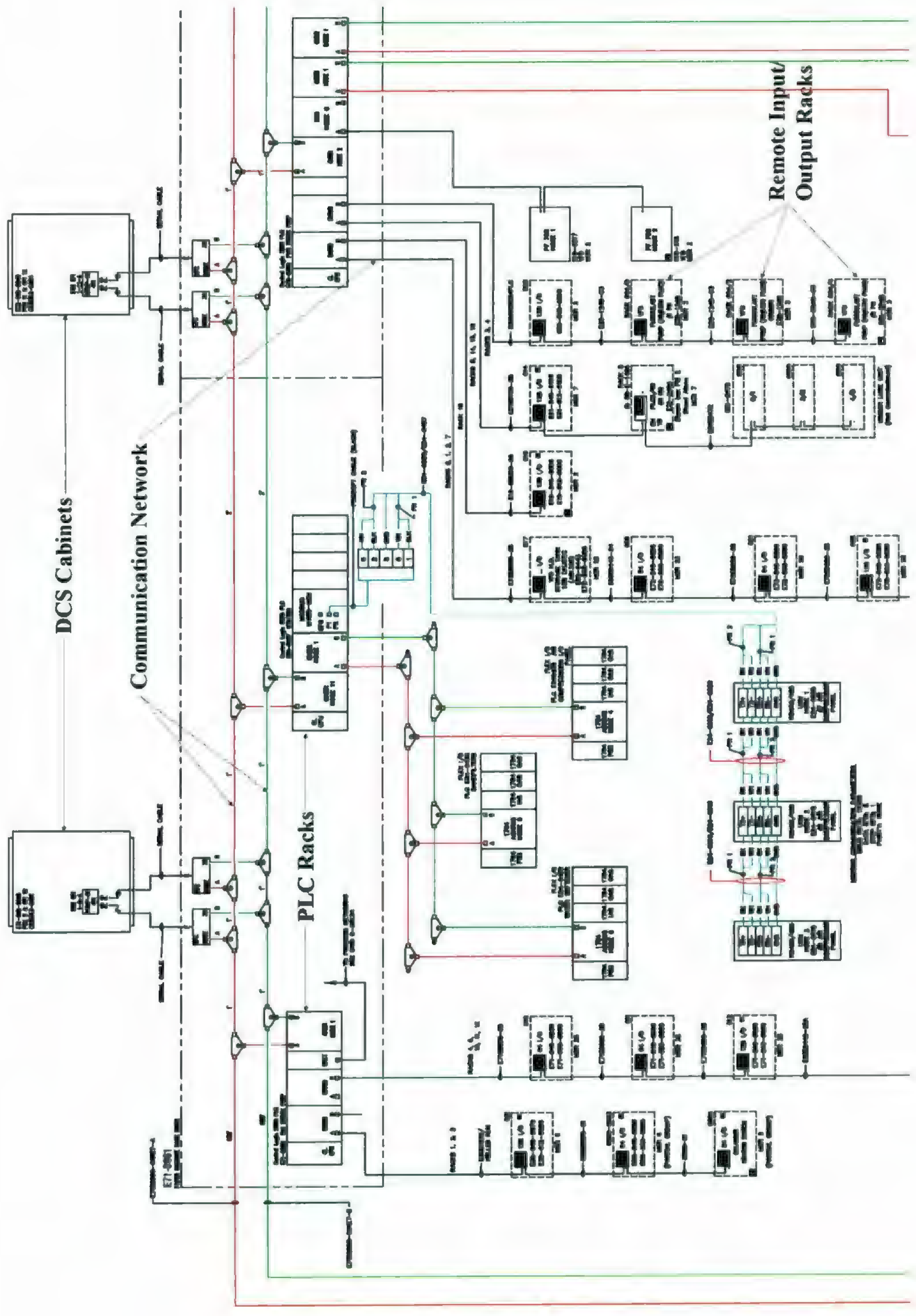


Figure 4-7: Typical Control Network

With several thousand motors in a plant, this would save significant commissioning time. As well, it also gives designers more confidence that the system will work as designed.

In order to incorporate these issues into simulations using PLCs, the control logic and plant model can be split up between several controllers connected to a common communication network. One controller would contain the control logic while the others would contain the plant model, for example one controller for every field I/O rack. This ensures that the control logic will execute at the same rate as it will when controlling the live plant (no plant model logic to execute) and it allows communication network characteristics to be part of the simulation.

The above mentioned three tank example was split between two Momentum PLCs to test this idea. Data was passed back and forth between the PLCs via a Modbus Plus network. Figure 4-8 is a picture of the PLC setup used for this simulation.

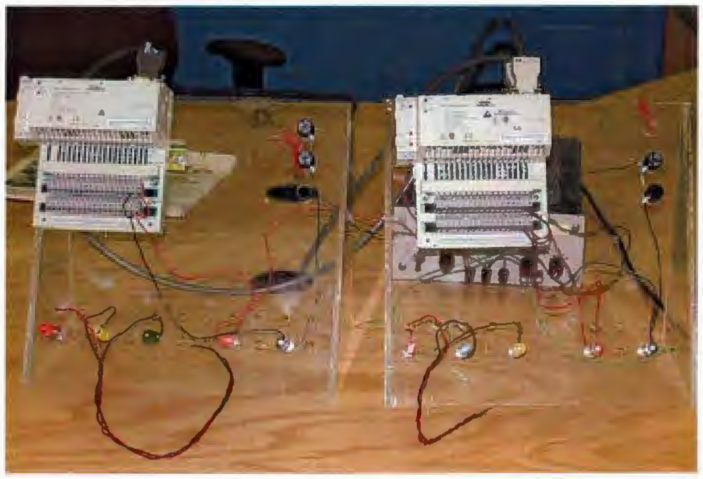


Figure 4-8: PLC Simulation Equipment

Figure 4-9 shows the water level vs time trend for the simulation. This trend has the same basic shape as the one PLC simulation trend, however the time span is longer and the maximum level of Tank 1 is not as high as in the one PLC simulation.

In the one PLC simulation the level reached 0.81129 meters whereas in the two PLC simulation it only reached 0.79931 meters. The increased time span is a result of the time delay introduced by the communication network, it took longer for the operator interface to read the same number of points from the plant model PLC. The difference in tank levels is a result of the control logic executing faster than in the one PLC simulation. Since the control logic PLC only had three function blocks to execute, as opposed to eighteen in the one PLC simulation, it was able to react to tank level changes faster. Figure 4-10 combines all three simulation results.

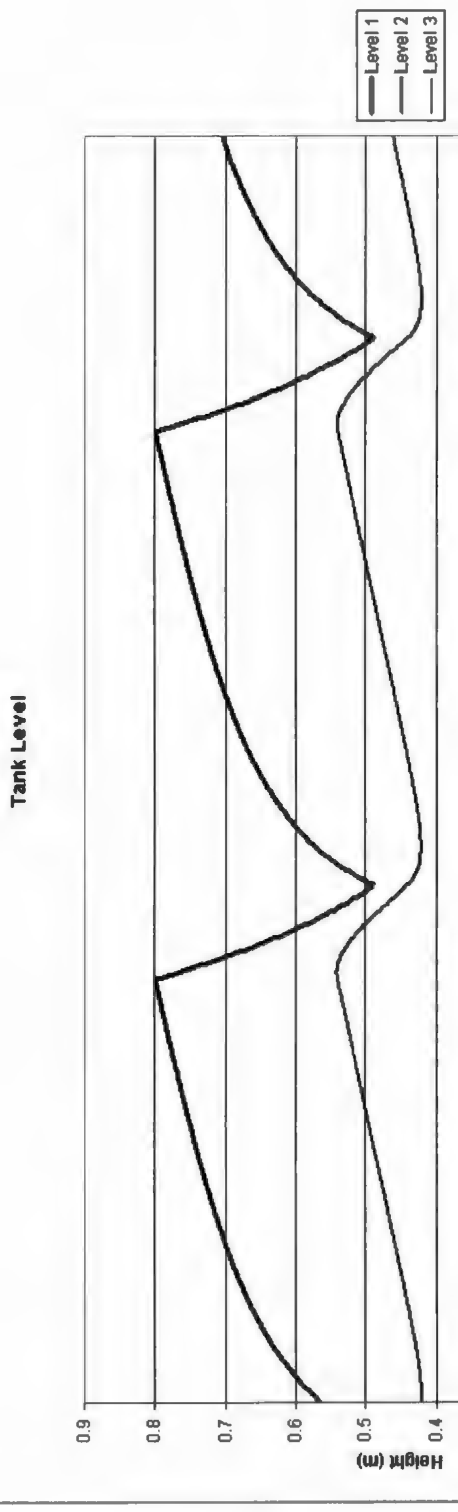


Figure 4-9: Graph of Water Level vs Time For Two PLC Simulation

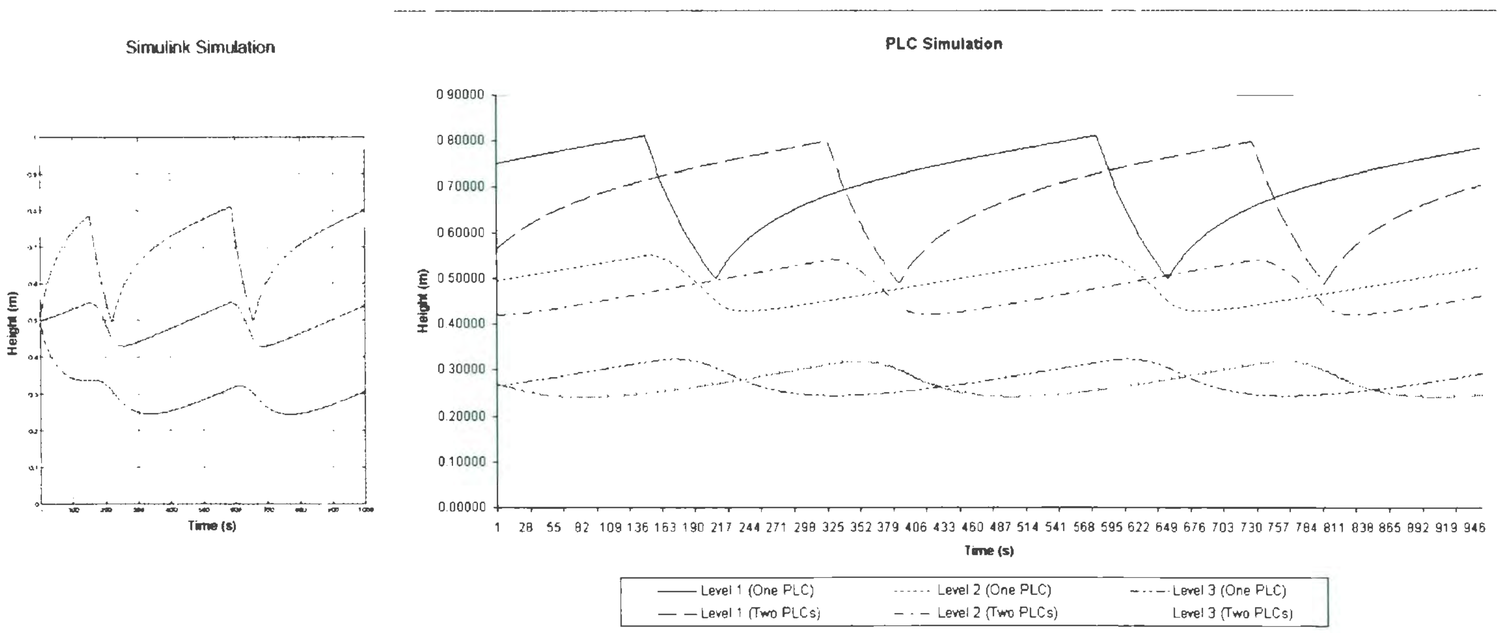


Figure 4-10: Combined Results of All Three Simulations

Chapter 5

DCS Simulation

5.1 Terra Nova Simulator

As mentioned in Chapter 2, the Terra Nova Alliance used the Hardware-Software method to simulate their Floating Production, Storage and Offloading (FPSO) vessel. Developed by Fabcon Canada between September of 1998 and February of 2000, this simulator has four main components:

- Operator Interface Stations (OIS) and Engineering Workstation (EWS)
- Process Control Units (PCU)
- The simulation computer

The OIS, EWS and PCUs are part of the ABB INFI90 Distributed Control System (DCS). The simulation computer is a standard Pentium based PC running specialized simulation software, to be discussed later. The simulation computer communicates with the PCUs via ethernet and Universal Simulation Modules (USM). The USM is specialized add-on to the INFI90 DCS that acts as a bridge between the Ethernet based simulation network and the Controlway, as shown in Figure 5-1. During simulation, the USM redirects the input/output data from the I/O modules in the DCS to the simulation software.

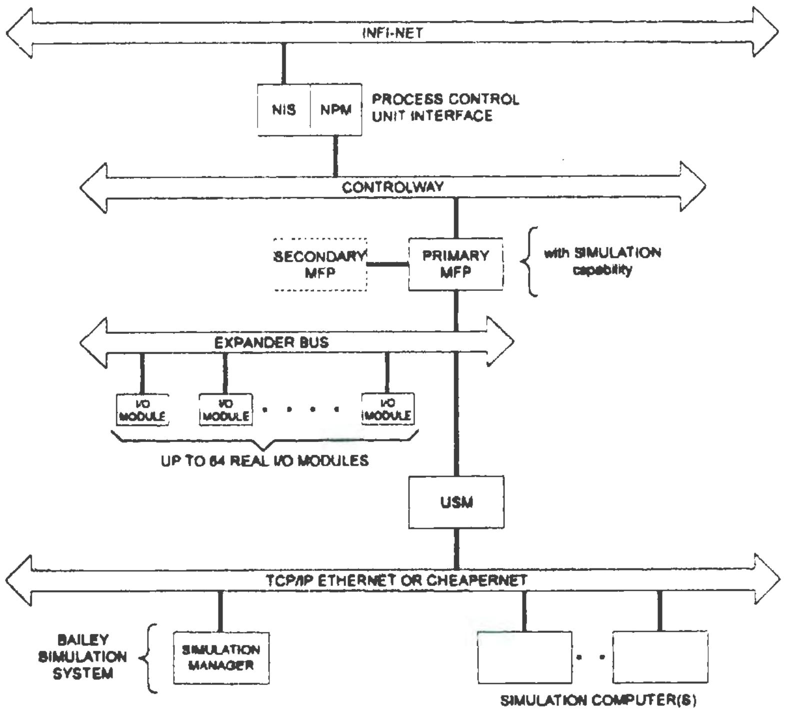


Figure 5-1: Simulator Layout Using ABB Universal Simulation Modules [1]

Physically, these modules are located in the PCUs next to the process controllers. Typically, there is one USM per process controller. Figure 5-2 shows how the modules are laid out in each of the PCUs used for this simulator. The simulation PC runs a software package called Dynamic Simulator for Process Instrumentation and Control Engineering (D-SPICE). D-SPICE was developed by Fantoft Process Technologies AS, parent company of Fabcon Canada, for the purpose of simulating process systems, such as oil and gas production [12, p.2]. Models are created in this software by simply connecting together component function blocks, shown in Figure 5-3. The components themselves are modelled by mathematical equations written in C/C++. This code is hidden from the user, however, D-SPICE does allow for the user to create his or her own function blocks [12, p.4].

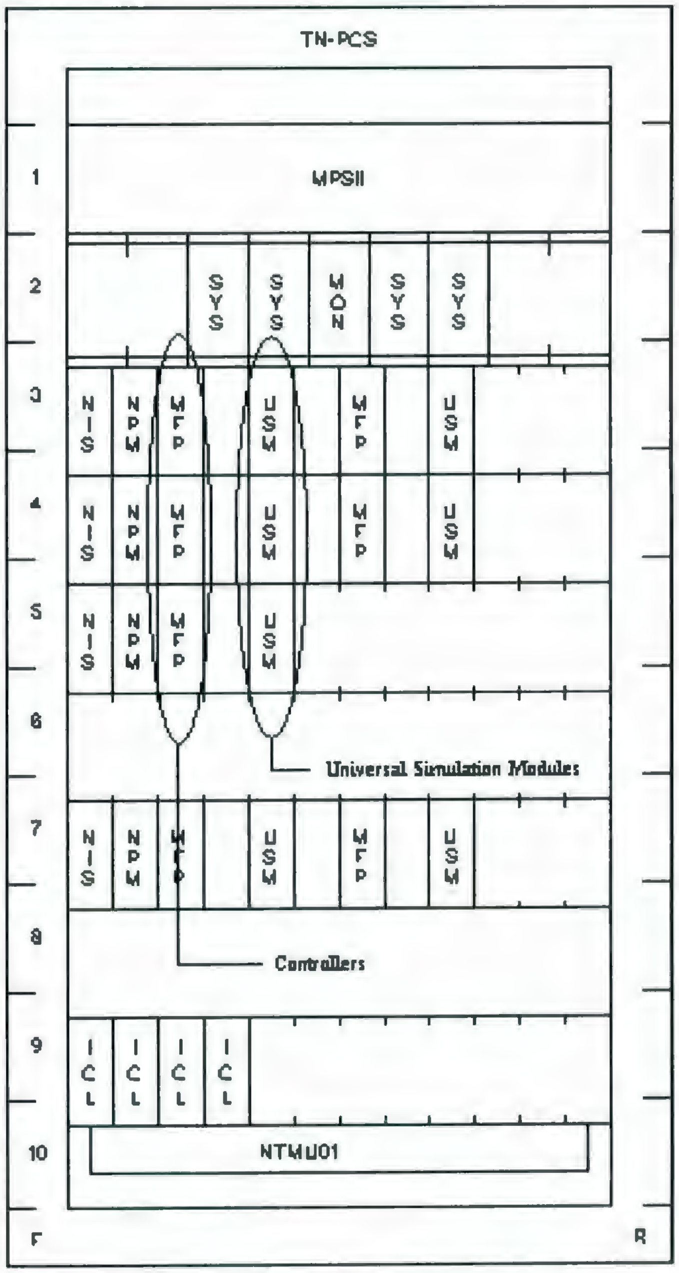


Figure 5-2: Terra Nova PCU Layout

There are certain systems, such as thermodynamic systems, whose mathematical models are complex and require large amounts of computational power. Simulating these systems using their mathematical models would slow down the simulation to the point of making it unusable. In D-SPICE, simplified versions of these systems are modelled using look-up tables and regressions. The data required for these models is usually generated by dedicated software, such as Peng-Robinson for thermodynamic systems, and is specific to the particular system being modelled [12, p.4].

5.2 Hibernia Simulator

Chapter 2 also mentions a Hardware to Hardware simulation that Bailey SEA (Nfld.) Ltd. and SEA Systems Ltd. use to test Hibernia's Fire and Gas (FGS) Emergency Shut Down (ESD) control system. The control system containing the logic to be tested is an ABB INFI 90 DCS, similar to the one used for the Terra Nova simulator mentioned above. The slave controller is a Triconix PLC. I/O data is passed back and forth between the systems via the INFI 90's General Purpose Interface (GPI). The GPI is a component of the ABB DCS that allows the DCS to communicate with a variety of PLC brands. The software is designed such that the GPI can be reconfigured to communicate with a variety of different PLC brands. Figure 5-4 shows the layout for this system.

The PLC has two serial ports. One port is connected to the GPI and the other is connected to simulation computer. The simulation operates as follows:

- The computer feeds input values to the PLC
- Based on these values and the sequence of events logic, the PLC writes appropriate data to it's memory registers
- These registers are then read by the DCS and dealt with accordingly, i.e. alarms are triggered, equipment is turned on or off, etc.

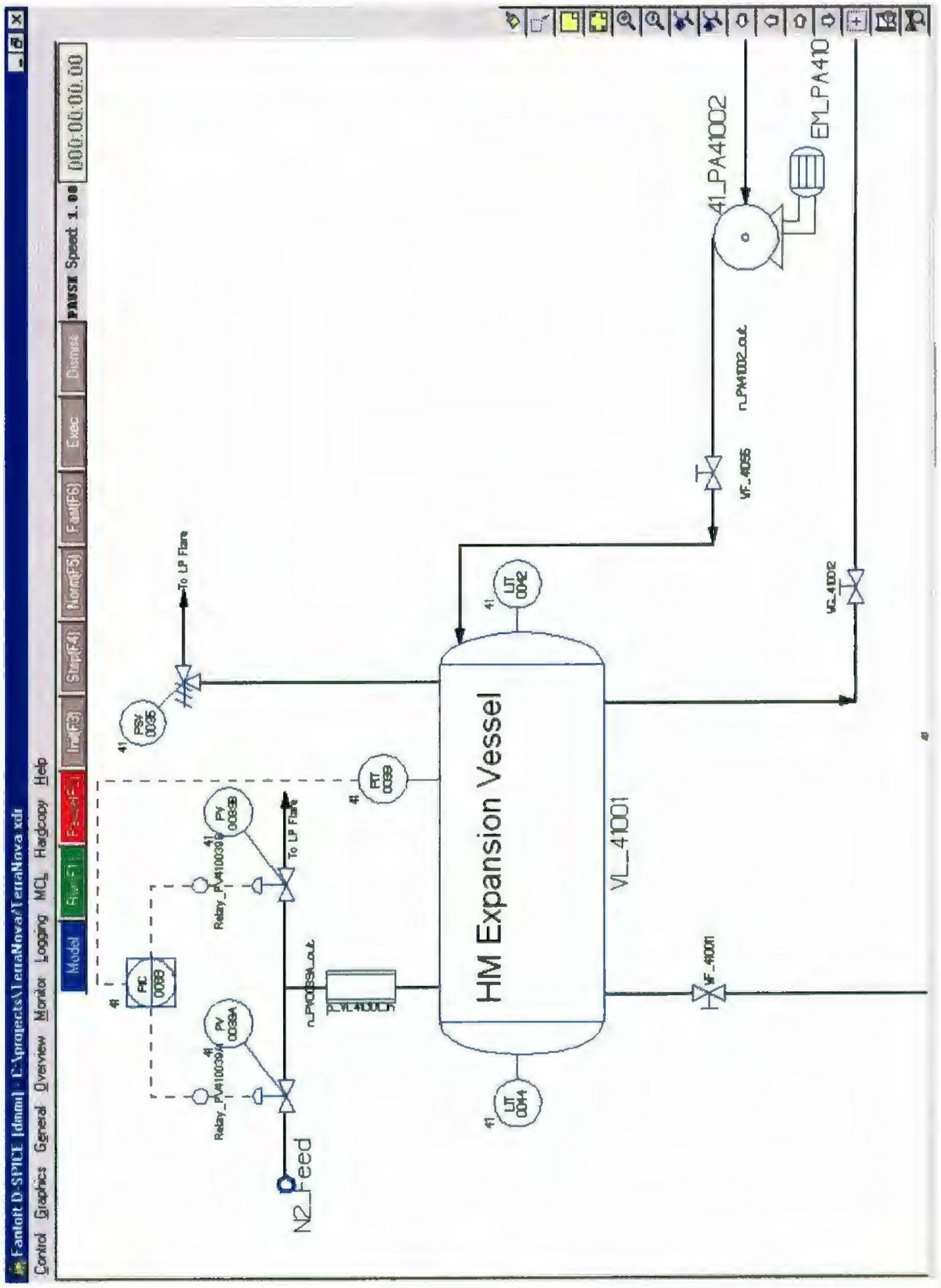


Figure 5-3: Terra Nova Simulation Logic

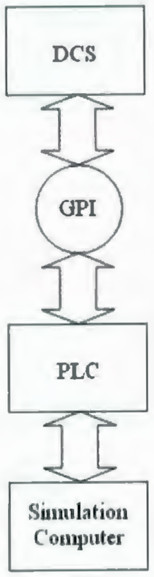


Figure 5-4: Block Diagram Layout for Simulator

- PLC registers are updated with output data
- Output data is sent to computer to determine next set of data

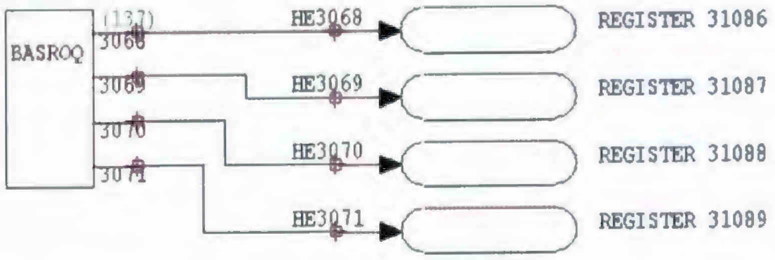


Figure 5-5: Logic Required to Read PLC Memory Registers

Figure 5-5 shows part of the logic required to access external data via the GPI. The BASROQ function block (function code 137) is the block that reads the data from the PLC and makes it available to rest of the control logic. The ovals on the right hand side of Figure 5-5 are cross reference blocks that link this piece of logic to other parts of the control logic. Each BASROQ block can read a maximum of four blocks of PLC memory called registers. These registers can contain analog data such as tank levels or 16 digital data points, such as valve open/close commands.

In order for the DCS logic to access the individual digital data points (bits), they must be separated using the RDEMUX function block. Figure 5-6 shows how this done for register 31087. Each bit is attached to it's own cross reference so that the rest of the logic can use them. Figure 5-7 is an example of how the control system would act on the GPI data and then update the PLC registers.

Staff at Bailey SEA (Nfld.) and SEA Systems have also used the second type of Hardware to Hardware simulation method mentioned above in Chapter 2. Typically referred to as the tie back method, this type of simulation is used to test logic and HMI updates for both Hibernia and Terra Nova. In the ABB system, there are certain function blocks that are linked to input data. Figure 5-8 shows a typical piece of control logic that is used to bring digital input values into the controller. The DIGRP function block (function code 84) is required by the INFI90 system and it acts as the link between the I/O termination blocks and the rest of the control network.

To do simulation using "tie back logic", function block 84 is replaced by the ON/OFF function block (function code 50), as shown in Figure 5-9. These blocks are then turned on and off manually according to a predetermined data set, such as a sequence of events table, and changes in the system states and outputs are observed to ensure correct operation of the logic.

5.3 Simulator Flexibility

Any simulation method developed has to be flexible enough to work on any type of control system made by any manufacturer. There are two restrictions inherent to the proposed simulation method that limit the types of control systems that can be used.

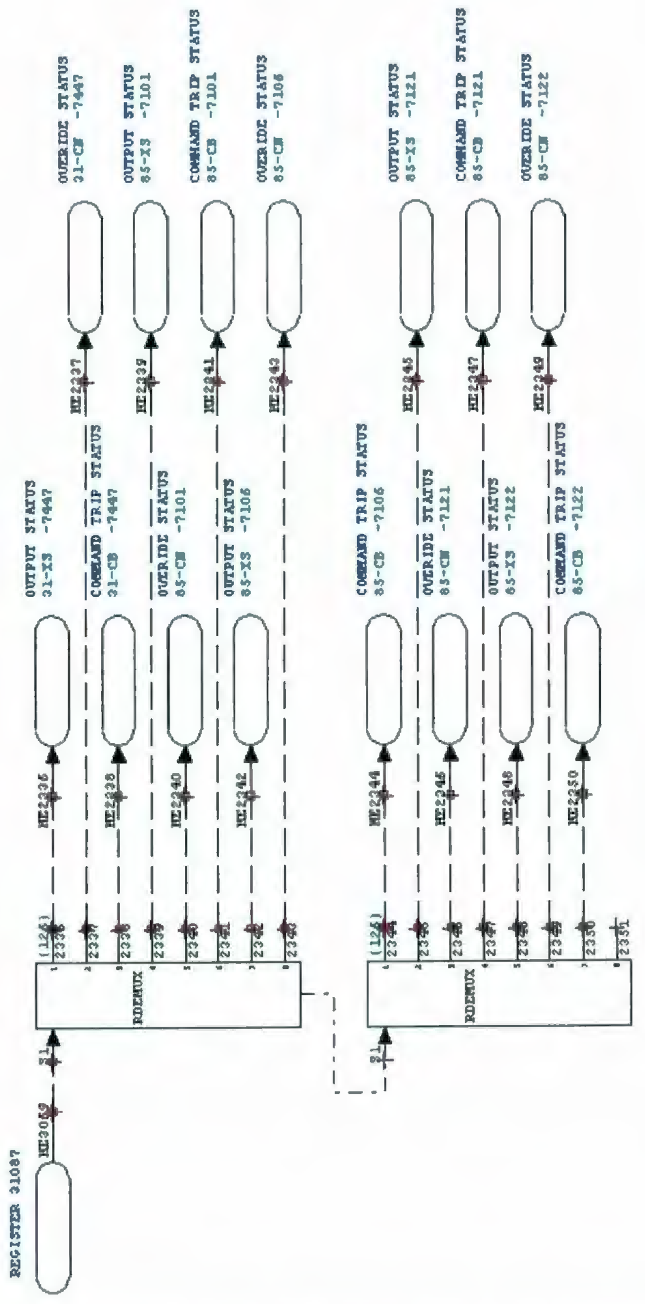

REGISTER 31087
RDEMUX
RDEMUX
OUTPUT STATUS 31-XS -7447
COMMAND TRIP STATUS 31-CB -7447
OVERIDE STATUS 85-CN -7101
OUTPUT STATUS 85-XS -7106
COMMAND TRIP STATUS 85-CB -7106
OVERIDE STATUS 85-CN -7121
OUTPUT STATUS 85-XS -7122
COMMAND TRIP STATUS 85-CB -7122
OVERIDE STATUS 31-CN -7447
OUTPUT STATUS 85-XS -7101
COMMAND TRIP STATUS 85-CB -7101
OVERIDE STATUS 85-CN -7106
OUTPUT STATUS 85-XS -7121
COMMAND TRIP STATUS 85-CB -7121
OVERIDE STATUS 85-CN -7122

Figure 5-6: Logic Used to Extract Bits From GPI Data

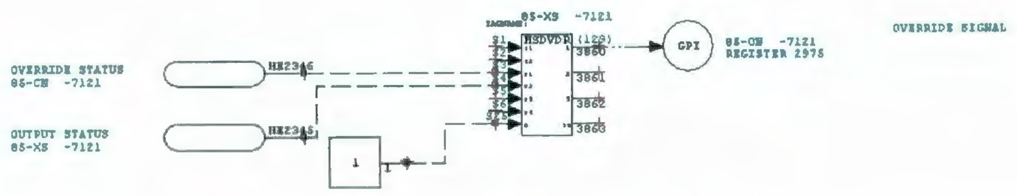


Figure 5-7: Logic That Uses GPI Data

These restrictions are: the system must support function block programming and the system must have the ability to have several controllers communicating together over a common communication network. However, most of the controls equipment used in industry either have this capability now or will have it in the near future.

To test the flexibility of the proposed simulation method, the same three tank process was run in an ABB Infi90 DCS and programmed using Composer Engineering Work Station software. Figure 5-10 shows a typical Infinet layout [13]. This simulation used two PCUs and one EWS. The Control PCU, shown in Figure 5-11, contained an Infi90 Multi Function Processor (MFP02) and executed the control logic. The Plant PCU, shown in Figure 5-11, contained a Harmony Bridge Controller (BRC100) and executed the process logic. Data was passed between the two processors and the EWS via the Infinet network. In Composer, logic is generated on "logic documents", or "CAD sheets". Typically, logic is laid out in such a way that all the logic associated with a particular sub process, e.g. temperature control loop, is contained in one CAD sheet. This simulation had four CAD sheets, one for each tank and connecting pipe and one for the control logic. Figure 5-12 shows the logic generated for Tank 1 and the pipe joining tanks 1 and 2. This logic could be broken down even further to have each pipe on it's own CAD sheet. As with the PLC simulation, having the logic separated by equipment type makes it easier to reuse the logic in future simulations.

Figure 5-13 shows the control logic for this simulation. As with the PLC simulation, as the level in Tank 2 rose above 0.55 the input valve would close. Once the level dropped below 0.45 the input valve would open. In the Infi 90 system data is transferred to the communication network by Exception Report blocks. Block 104 (function code 45) in the control logic transfers the valve command (open/close) over the network and block 121 (function code 30) transfers the Tank 1 level. On the other side, block 200 (function code 122) and block 150 (function code 121) allow data to be read from the network.

Figure 5-14 is the level tend obtained from the simulation. As can be seen, this trend has the same shape as the PLC and Simulink simulations. The only difference, which can be easily seen in the Tank 1 level, is the levels go higher and lower in the DCS simulation than in the other two simulations. The reason for this is the time delay caused by the communication network. The control logic is looking at the communication network for the Tank 2 level data and the plant model logic looks to the network for the valve command. As a result of the network time delay, the level in Tank 2 actually goes up to 0.58 before the valve is commanded to close and the levels start to go down. In comparison, Tank 2's level in the PLC simulation only goes up to 0.55081. Table 5.3 shows a table of the minimum and maximum tank levels for the Simulink, one PLC, two PLC and DCS simulations.

At the beginning of this section it was stated that there are two restrictions inherent to the proposed simulation method that limit the types of control systems that can be used. The first restriction is the system must support function block programming. This is because most industrial control equipment, either PLC or DCS, can be programmed using function block diagram language and it's easier to convert from one platform to another if they are programmed using a common method. For example, it took a couple of days to develop the three tank simulation in the PLC using function blocks. Since the ABB DCS also understands function block programming it only took half a day to develop the simulation in the DCS. This is because the majority of the time taken to develop the PLC simulation was used in determining which function blocks to use, how to lay them out on the screen, how to connect them together and testing to make sure the simulation works properly. When the simulation was created in the DCS, the same function block types and layout could be reused which reduced the time required to develop the simulation.

The second restriction is the system must have the ability to have several controllers communicating together over a common communication network. The reason for this restriction is that in order for the simulation to be as realistic as possible, the control equipment has to be operating at as close to the same rate as it would be in the real process as possible. As mentioned in Section 3.2, the scan time of a controller depends on the amount of logic it has to execute and the amount of data it has to manage. Therefore, in order to make the simulation as real as possible, any controller that would contain control logic in the real process should contain only control logic in the simulation. The plant model would then be contained in other controllers and communicate with the control logic controllers via a communication network.

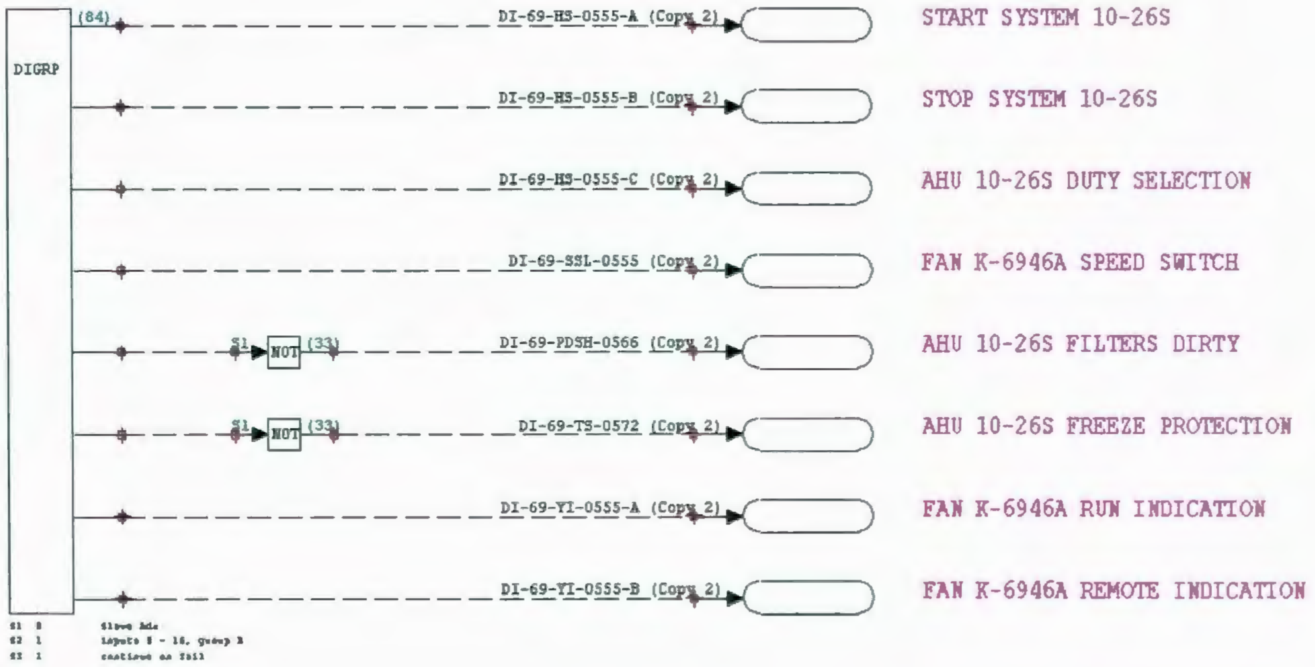


Figure 5-8: Function Code Logic to Input Digital Values

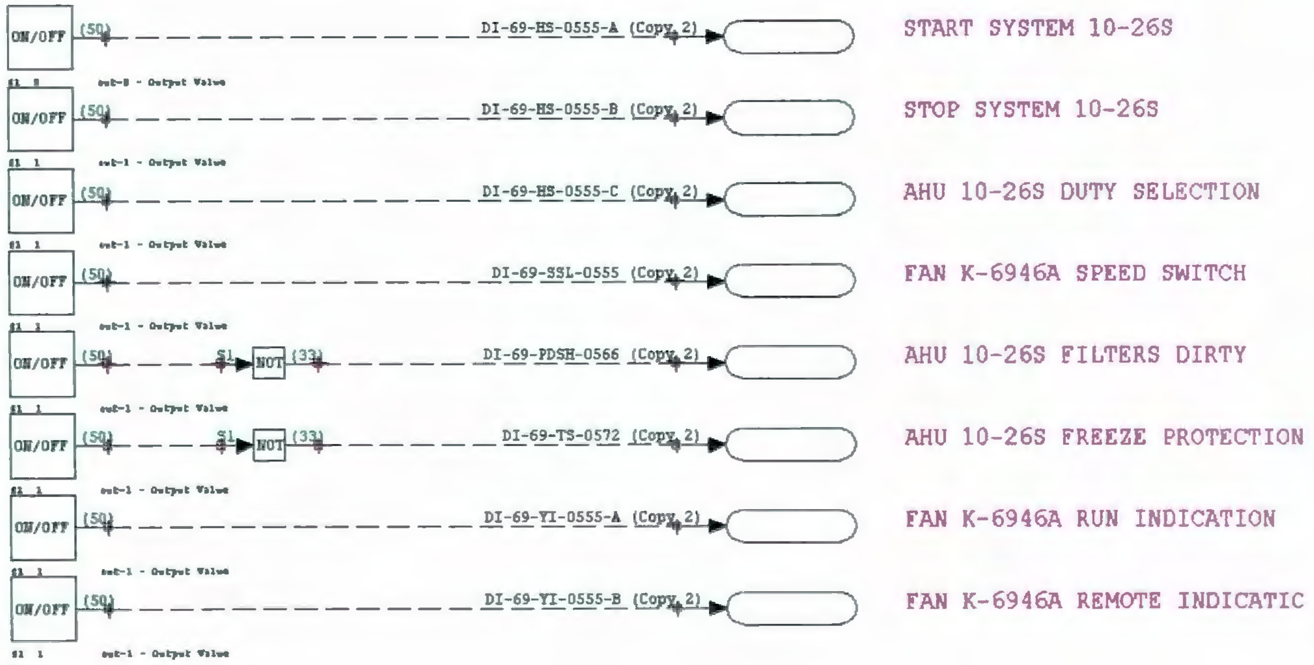


Figure 5-9: Tie Back Logic for Simulation

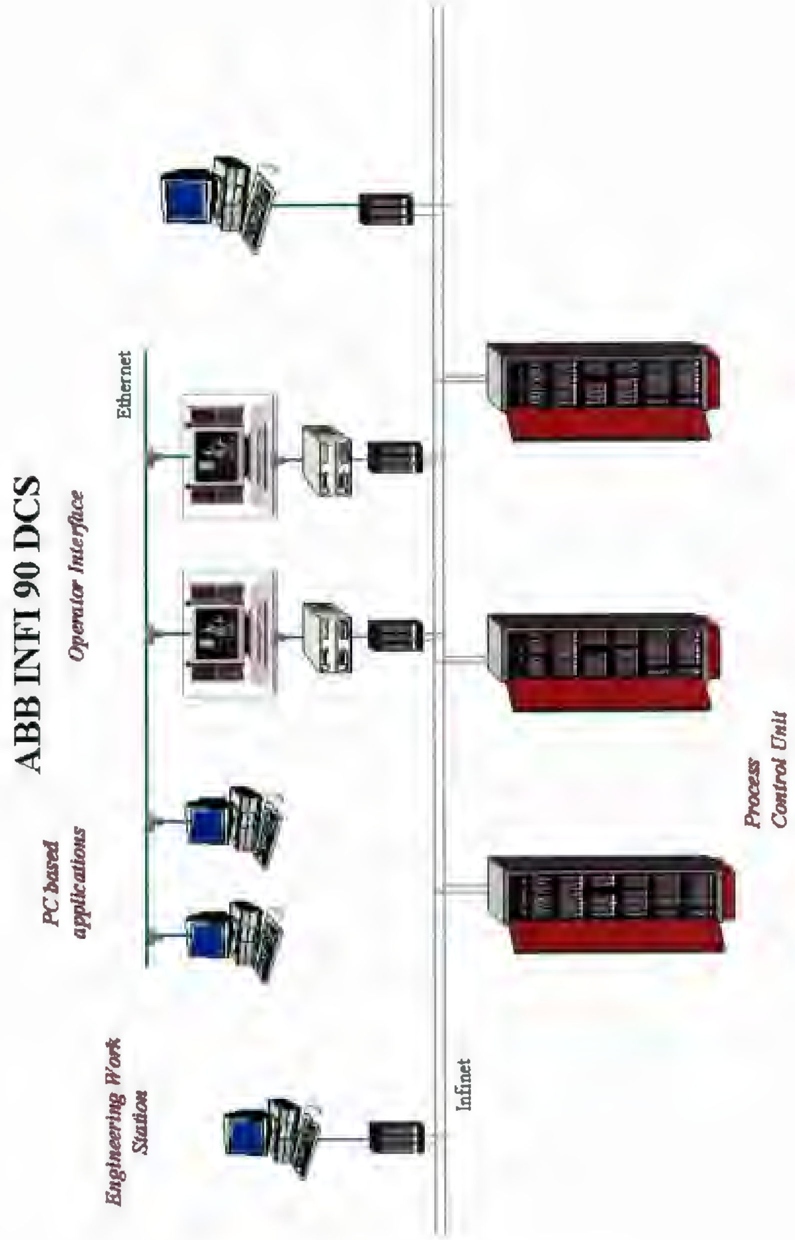


Figure 5-10: Typical Infinet Layout

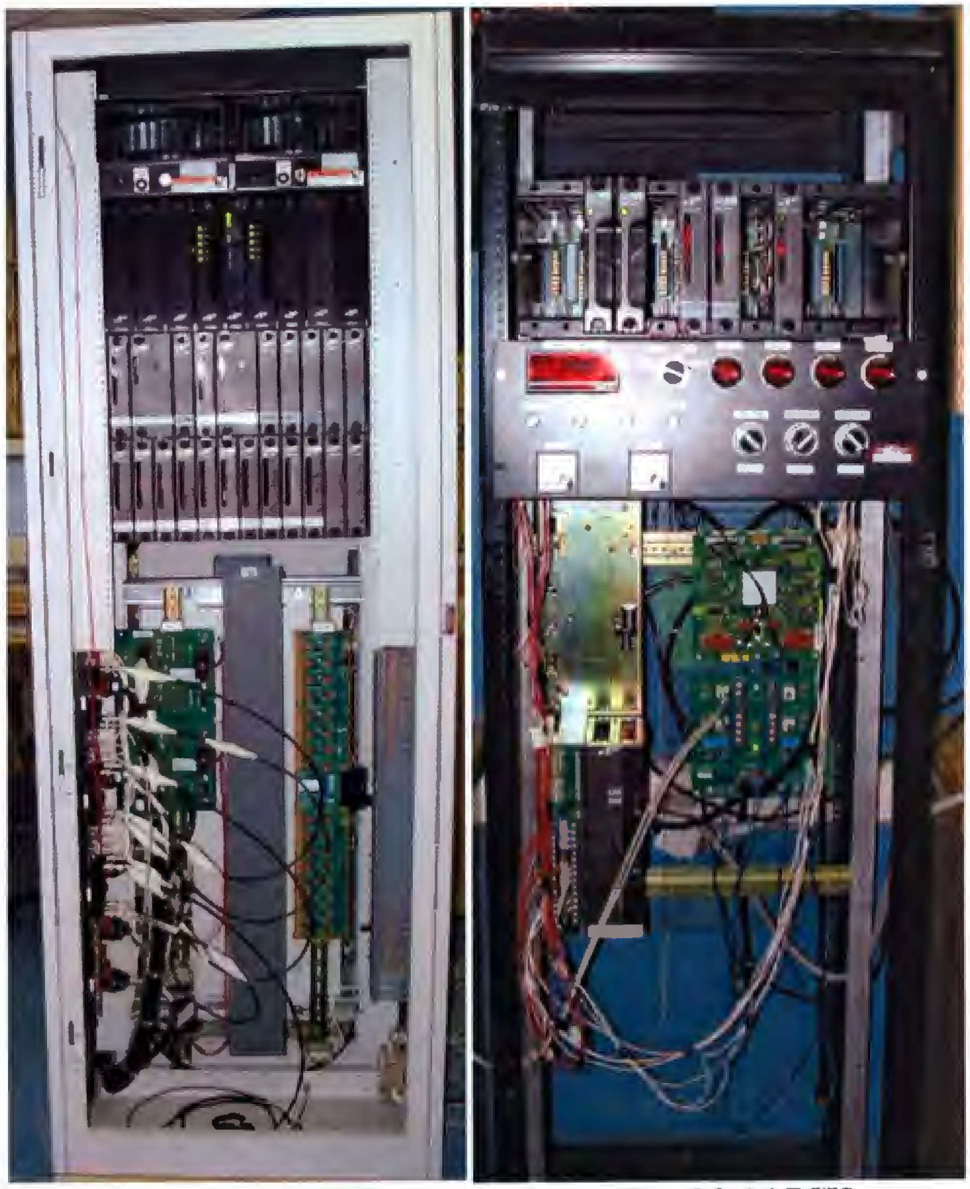


Figure 5-11: Plant and Control Logic PCUs

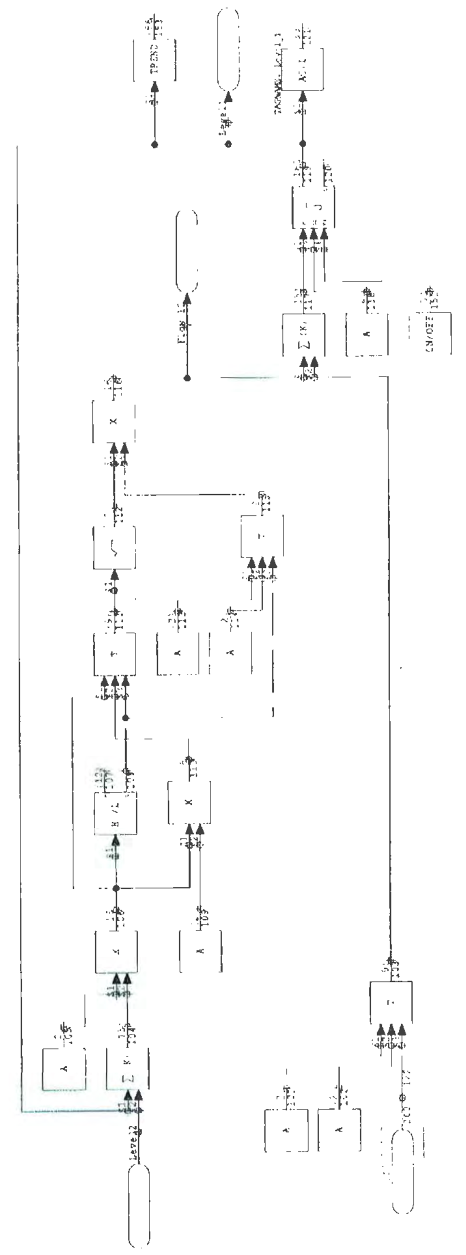


Figure 5-12: DCS Logic for Tank 1

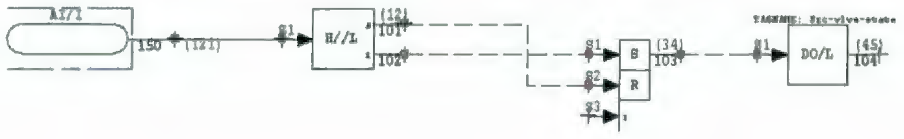


Figure 5-13: Simulation Control Logic

	Tank 1 Level		Tank 2 Level		Tank 3 Level	
Simulation	Min	Max	Min	Max	Min	Max
Simulink	0.5	0.81	0.44	0.55	0.25	0.32
One PLC	0.49873	0.81129	0.42893	0.55081	0.24485	0.32339
Two PLCs	0.48833	0.79931	0.42032	0.54125	0.24132	0.31851
DCS	0.479513	0.822411	0.414521	0.561542	0.23259	0.334067

Table 5.1: Simulation Results Table

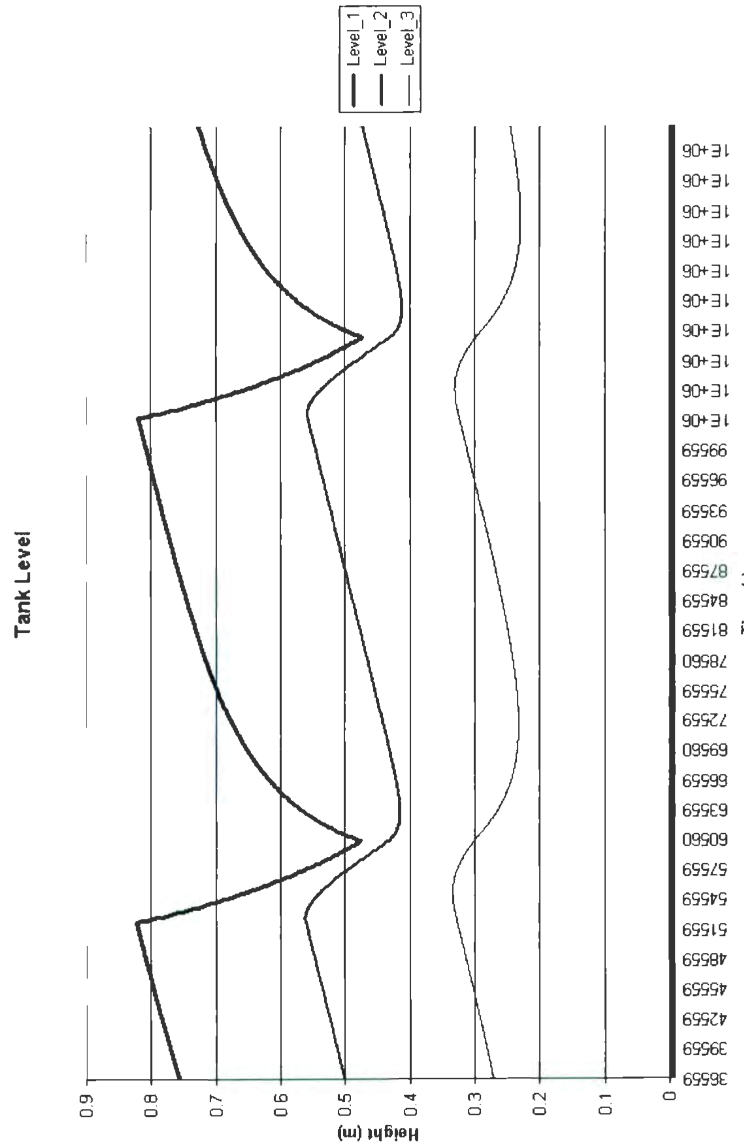


Figure 5-14: DCS Simulation Results

Chapter 6

Real Life Example

To illustrate how SPSS can be used to simulate a typical industrial process, this chapter will take a typical industrial process, a hotwell tank, and develop a simulation of it in the ABB DCS. Figure 6-1 is the P&ID for the hotwell tank used for this simulation [14].

6.1 Process Description

The hotwell tank is an important part of the steam generation process at a pulp and paper mill. Ideally, all steam generation processes are closed loop systems. This means that all the steam that is sent to the various parts of the mill is returned to the boiler as water, called condensate. In reality, however, small amounts of water and steam are lost due to leaks in the system. The hotwell tank acts as a buffer between the boiler feed water system and the various mill processes.

The condensate flows into the hotwell which is kept at a set level, say 50%. From here the condensate is fed to the Condensate Storage tank, which is also kept at a set level. Finally, the condensate enters the Deaerator tank where it gets mixed with steam to remove dissolved gasses, such as carbon dioxide and oxygen, before it is fed to the boiler.

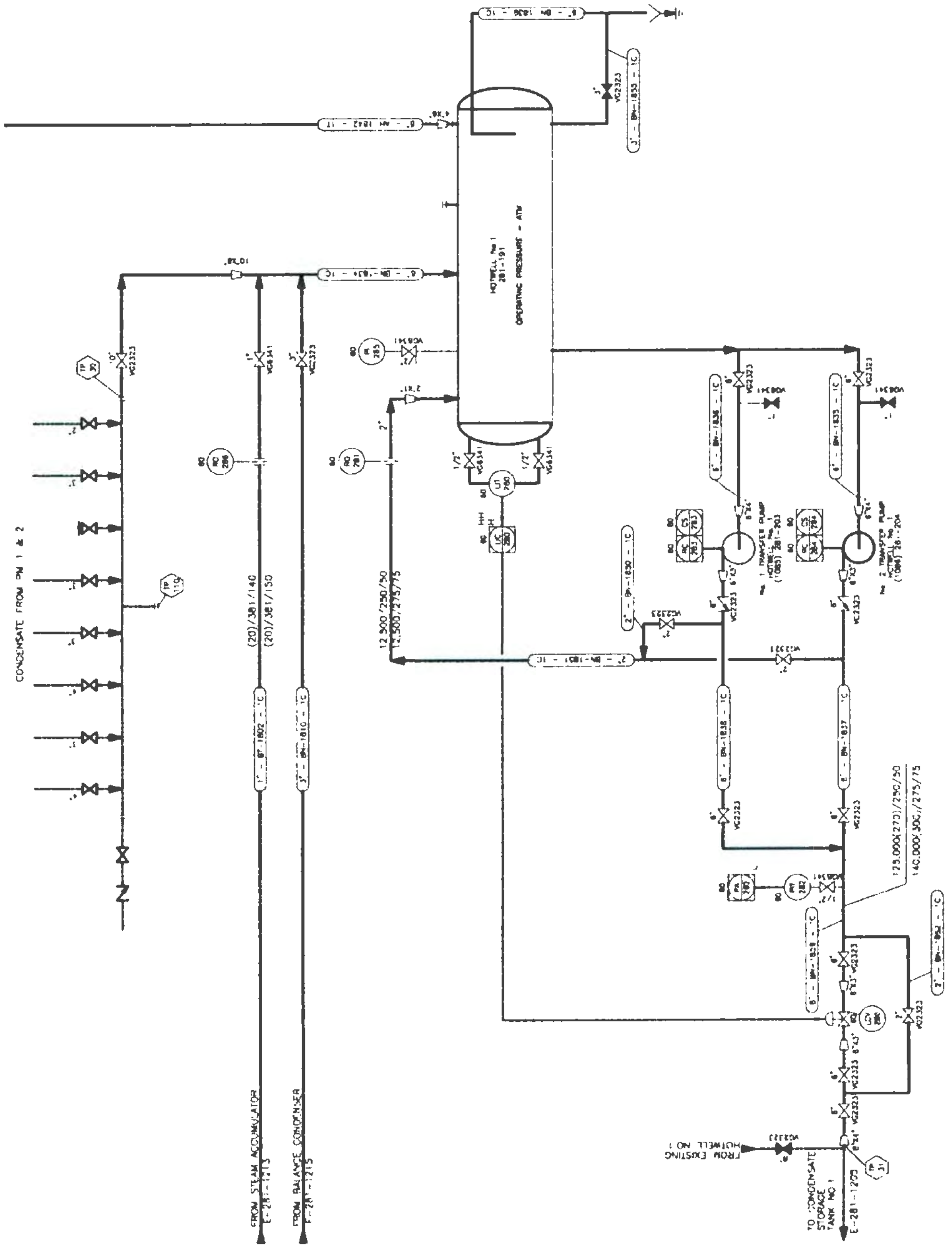


Figure 6-1: Hotwell Tank P&ID

Under steady state operating conditions, the amount of condensate returned is sufficient enough to keep all tank levels constant and require a small amount of fresh, chemically treated water to make up for the losses. However, during paper machine or TMP startups and shutdowns there are large swings in steam demand and condensate return and the three tanks (hotwell, condensate storage and deaerator) serve to smooth out these transitions so that the boiler always sees a steady flow of feedwater. For example, when a paper machine starts up, there is a large amount of steam being generated but not a similarly large amount of condensate being returned. At this point, the water level in the tanks will drop but there is sufficient capacity to provide the required condensate without the tanks going dry. The opposite is true during a paper machine shut down. In this case there is a large amount of condensate being returned but the amount of steam being generated is reduced. The tanks will start to fill above their normal operating level but not to the point where they overflow.

In this system the hotwell level is controlled by the outlet control valve (LCV280) and the redundant pair of transfer pumps. As the level in the tank increases, the control valve opens and as the level decreases the valve closes. One pump is selected to run by the operator and will run as long as the tank level remains above the low low level cutoff point. If the running pump fails, the second pump will start. Figure 6-2 is a trend of the actual hotwell level and valve position during steady state operation.

6.2 Plant Model

As in the example used in chapters 4 and 5 the level in the tank is modelled with the following integral equations:

$$h(t) = \frac{1}{A}\int q_{net}dt + h(0)$$

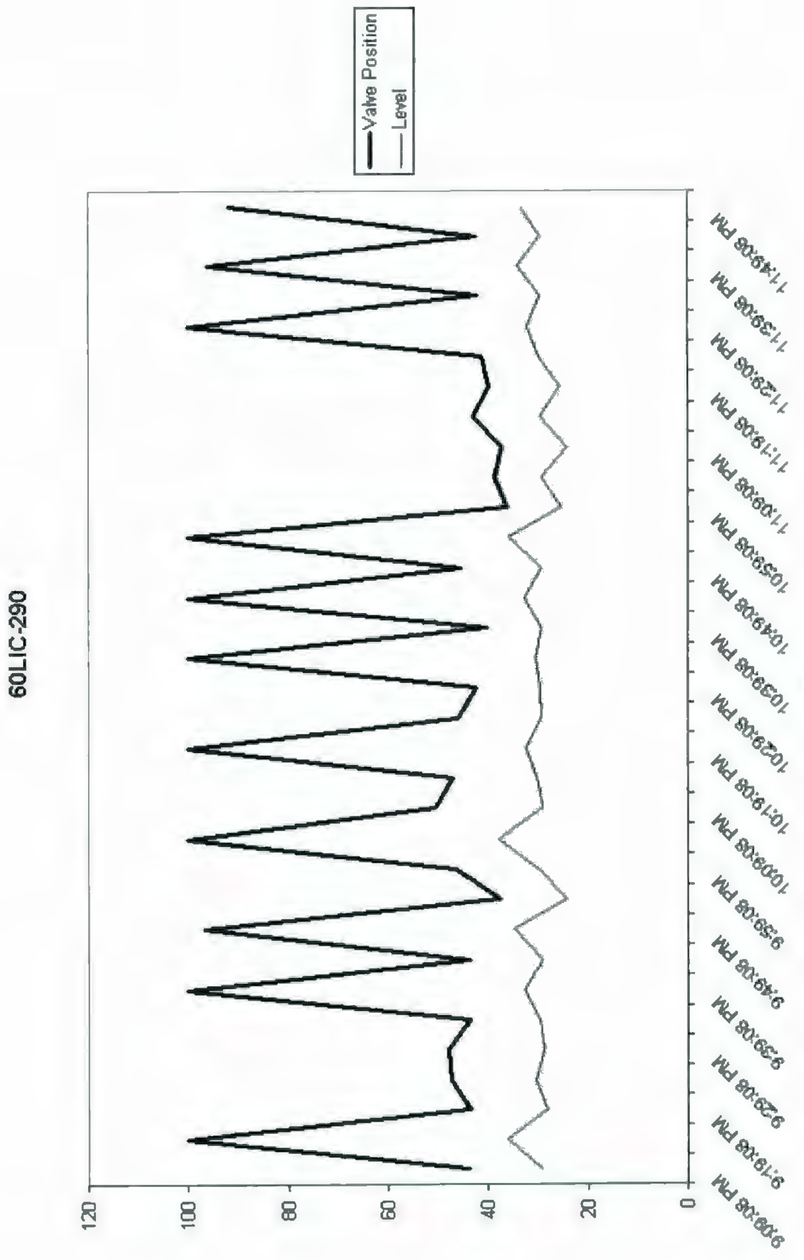


Figure 6-2: Hotwell Tank Level Trend

$$\begin{aligned} q_{net} &= q_{in} - q_{out} \\ q_{out} &= C_V\sqrt{P_2 - P_1} \end{aligned}$$

Where: h(t) = level in tank

q_{in} = flow rate into tank

q_{out} = flow rate out of the tank

A = cross sectional area of tank

C_V = valve coefficient

P_1 = pressure on the outlet side of the valve

P_2 = pressure on the inlet side of the valve

In order to simplify the simulation, several ideal assumptions were made. First, the hotwell tank is cylindrical shaped, laying on it's side, which means the cross sectional area changes as the level changes. For this simulation, the cross sectional area was kept constant at $3.2516m^2$ (cross sectional area for a tank level of 0.763m). This assumption was considered to be acceptable because the water level did not change that much and at the minimum level the area was $3.1887m^2$ and at the maximum level the area was $3.1239m^2$. These small changes in area would not have a huge effect on the simulation results. Secondly, there was no information available for the control valve outlet pressure so it was assumed to be zero. Finally, the control valve was assumed to be a linear valve, meaning that if the control logic output to the valve was 50%, the valve was open 50% and if the output was 75% then the valve was open 75%. As shown in Figure 6-1, the control valve has a 7.62cm diameter which corresponds to a C_V of 108. The pressure of the inlet side of the control valve was kept constant at 41.57574psi.

Figure 6-3 is a trend of the valve position, tank level and inlet flow for the simulation. The bottom line is the inlet flow trend scaled down by 10%. Since the inlet flow was very erratic, it was impossible to model with a mathematical formula. The best way to model the inlet flow would be to feed the data obtained from the actual

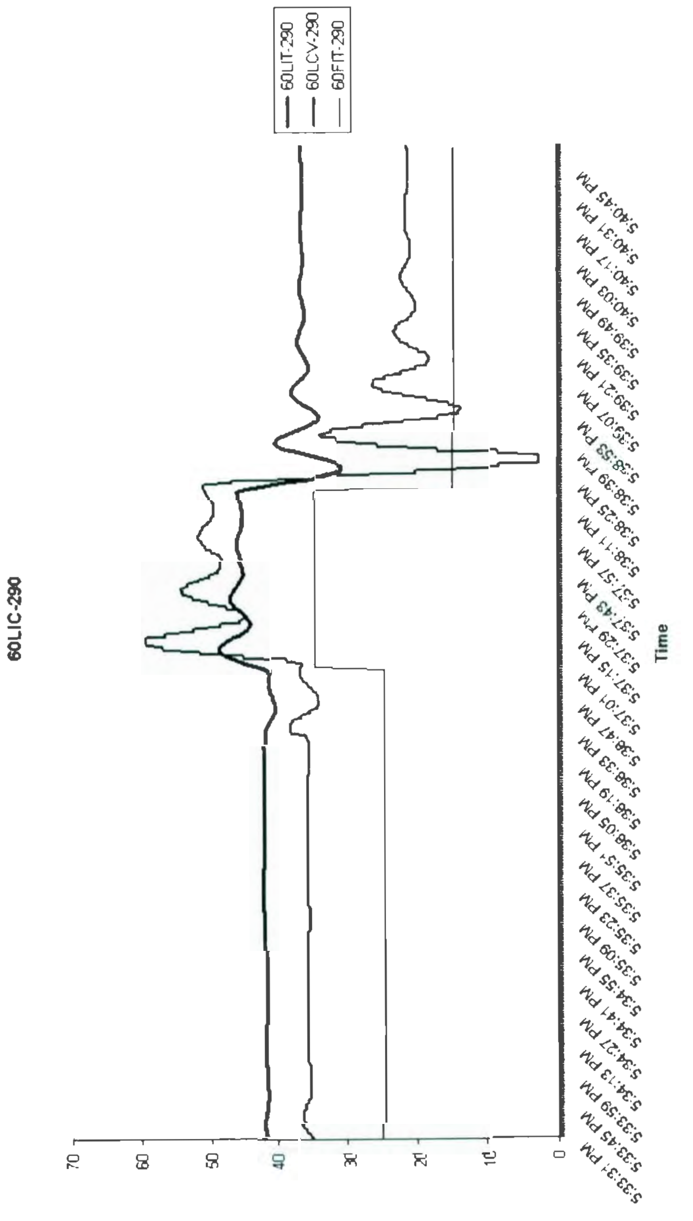


Figure 6-3: Simulated Hotwell Tank Level Trend

hotwell tank trend into the simulation. However, due to hardware limitations this option was not possible. Instead, a Manual Set Constant (function code 2) block provided the inlet flow value. This block provided a constant value for the input flow but could be changed at any time during the simulation, simulating step changes in flow. The other two lines show how the valve position and level reacted to the step changes. As can be seen in the trend, the valve position lags the tank level. This is the expected result for a closed-loop feedback control system, as the lag is caused by the controller sampling the level, performing it's calculations and changing the output valve position.

The control for this tank was provided by a PID block (function code 18) but only the proportional gain was used. The setpoint was set at 0.762m (30 on the trend). As can be seen on the trend, the tank level is always slightly above the setpoint. This is because there is no integral gain in the PID block and with only proportional gain, there is always a large error signal (setpoint minus measured value). Also, the trend shows a high number of oscillations in the valve position after a step change in the inlet flow. This is a result of the controller "hunting" to try and find the new valve position for the new inlet flow. This phenomenon is a result of low or no derivative gain in the PID block. In this case there was no derivative gain. Running this simulation longer and with proper PID tuning, the trend would look more like Figure 6-2.

Chapter 7

Conclusions

7.1 Simulation Method Comparison

Most of today's industries, whether it be pulp and paper, oil and gas, power generation or food manufacturing, have complex control systems consisting of multiple controllers communicating with multiple inputs and outputs and with each other. Having the ability to fully test the control system during all phases of development will save time and money during commissioning and start-up. This thesis looked at some of the methods that industry is currently using to simulate their control systems and introduced an alternate method called Single-Platform Stimulated Simulator (SPSS). These methods were compared based on cost, fidelity, implementation and conversion. The results of this comparison are shown in Table 7.1.

	Cost	Fidelity	Implementation	Conversion
Software-Software	\$2	70%	2	10
Hardware-Software	\$10	95%	7	1
Hardware-Hardware	\$6	10%	3	4
SPSS	\$7	85%	3	1

Table 7.1: Simulation Method Comparison Table

The Software-Software simulation method has the lowest cost of the four methods because all that is required is a computer, the simulation software and small office.

The amount of time required to develop the simulation would vary depending on the process being simulated and the software used. However, with a package like HYSIS, a complex plant model could be generated quite quickly because of it's graphical interface. This is also why it has the lowest implementation score because all that is required to set up the simulation is install the software and build the model. Being able to calculate the theoretical mathematical equations that describe the real process provides the most realistic simulations. However, all of the software packages researched had basic control functions and did not take into account time delays added into the system by control equipment. Therefore the Software-Software simulation method scored a 70% for fidelity. This simulation method is the most difficult to convert because the control logic has to be rewritten from scratch in the control system programming software.

The Hardware-Software method is the most expensive of the four methods because along with the simulation software and computer you also require control equipment and equipment to connect the two. In the case of the Terra Nova simulator where it was used to train operators, twice the amount of control equipment has to be purchased, one for the plant and one for the simulator. If a DCS control system, similar to the ones pictured in Figure 5-11, is used for the simulation then a lot of space will be required to house it. This method provides the most realistic simulation because the control system is added in which allows it's characteristics to be modeled as well. The biggest challenge in implementing this type of simulation is getting the control equipment to talk to the simulation software. Most control equipment will require extra components to make the link between the two. There is very little effort required to convert the control logic to the real process because it is already written for the control equipment. The most time consuming part would be redirecting the input output logic to the appropriate field devices.

The Hardware-Hardware method would be slightly cheaper than the Hardware-Software method because the simulation software and interface equipment are not required. Also, since there is no plant model to develop, the amount of time required to set up the simulation would be less. This method scored the lowest on fidelity because manually forcing individual inputs only indicates how small portions of the logic will react and doesn't show the whole picture. Also, this method doesn't take into account system time delays. The implementation of this type of simulation is a little more difficult than the Software-Software method because all the input points have to be modified to add constant blocks to them. In a system with a lot of inputs, this could be very time consuming. This also means that the conversion is a bit more time consuming then the Hardware-Software method because all the constant blocks will have to be removed and the inputs will have to be redirected to their appropriate field address. This process can introduce errors, such as blocks being missed and inputs pointed at the wrong address, which would have to be dealt with during commissioning, causing unnecessary delays.

SPSS, as expected, falls between the Software-Software and the Hardware-Software methods. The cost is less than the Hardware-Software method because, like the Hardware-Hardware method, there is no third party software package or interface equipment required. However, the cost to develop the plant model is increased because all of the mathematical equations that describe the plant would have to be recreated. The fidelity of SPSS falls in between the Software-Software and Hardware-Software methods because SPSS takes into account the characteristics of the control equipment but it is not able to perform certain complex mathematical operations. An example of this would be the matrix algebra required to do multi-phase flow calculations used in oil and gas processing simulations. Industrial controllers are not designed to handle matrices. This method would be a little more difficult to implement than the Software-Software method because of the fact that each mathematical equation would have to be created in the logic. Having component function blocks available would decrease the implementation time. As with the Hardware-Software method, SPSS is very easy to convert because the logic is already created in the language that the controllers understand. All that is required to convert the logic is readdress the inputs and outputs to their appropriate I/O modules.

7.2 Future Work

As mentioned in Chapter 2, all of the simulation methods discussed are "Fit for Purpose". The Hardware-Hardware method is best suited to logic functionality testing where as the Software-Software method is best suited to process testing and optimization. Single-Platform Stimulated Simulation will not replace any of the existing methods. Rather, it is meant as another simulation option that would allow engineering firms that would normally use only simple simulation methods, like tie back logic, the ability to create high fidelity simulations. In my opinion, the main reason why the Software-Software and Hardware-Software methods are not widely used, especially in the PLC market, is cost. The cost of the software, the cost of the hardware, the cost of the manpower to set up the simulation and the cost of the real estate to house the simulation. Managers find it hard to justify the these costs unless required by a particular project.

This cost can be reduced in a couple of ways. One way would be the control equipment manufacturers create equipment function blocks that contain the mathematical equations so that the model generation would be similar to the Software-Software method. A second way to reduce the development cost would be for engineering firms to develop their own equipment function blocks that they can reuse on other projects. From this work, the best way to achieve the above goal is to have a library of user developed function blocks available in the control system programming software that would contain the theoretical equations that describe a piece of equipment, like a tank or a piece of pipe. With this library, plant models would be created by connecting together equipment function blocks, similar to the Software-Software method. There are a couple of ways that this library can be created. First, the research community can work with local engineering firms to help them develop simulations for projects they are working on and build up their function block library. The research community has access to the theoretical mathematical equations and they can bring this knowledge and equations to the engineering firms. Secondly, the control equipment manufacturers, again working with the research community, can develop the function blocks and offer them to their customers as an add-on package to their software.

For further research in the area of process control simulation please refer to the thesis by Paul Handrigan titled "Distributed Systems, Hardware-in-the-Loop Simulation, and Applications in Control Systems". This thesis focuses on simulations using the Hardware-Software method.

Bibliography

[1] Elsag Bailey Company. *Instruction Universal Simulation Module (LMUSM01/02/03)*. Elsag Bailey Company, 1993.

[2] Kongsberg Maritime AS. Assett The Dynamic Life-Cycle Simulation Concept.

[3] Kongsberg Maritime AS. Snorre B Project Brochure.

[4] Robert N. Bateson. *Introduction to Control System Technology*. Prentice-Hall, Inc., 1996.

[5] Pierre R. Bélanger. *Control Engineering: A Modern Approach*. International Programmable Controls, Inc., 1995.

[6] Rockwell Automation. Enhanced and Ethernet PLC-5 Programmable Controllers User Manual, November 1998. Publication Number 1785-6.5.12, Page 5-2.

[7] Richard E. Morley. The History of the PLC. www.barn.org/FILES/historyofplc.html.

[8] Rockwell Automation. Automation Systems Catalog. www.ab.com/en/epub/catalogs.

[9] C.T. Jones; L.A. Bryan. *Programmable Controllers Concepts and Aplications*, chapter 1, page 4. International Programmable Controls, Inc., 1983.

[10] Schneider Electric. Concept 2.6 User Manual, October 2006. Publication Number 33002204.07, Page 8.

[11] ABGS. Final Phase 1 DCS/PLC Communications Network. Bowater Mersey Paper Company Ltd. Drawing Number E-22797.

[12] Fantoft Process Technologies AS. D-SPICE Simulator Overview. June 2000.

[13] ABB. Infi 90 Training Manual.

[14] Sandwell-HMI. Piping & Instrumentation Diagram, Paper Machines Hotwells. Corner Brook Pulp and Paper Drawing Number 67150.