INTEGRATING REAL-TIME SIMULATION MODELS INTO A SCADA ENVIRONMENT

A thesis presented in partial fulfillment of the requirements
for the degree of
Master of Technology
at
The Department of Production Technology
Massey University
Palmerston North
New Zealand

Submitted by
Rangaraju Srinivas Rao BE, DipTech (Massey)
October 1997

Supervised by
Dr. Huub Bakker
Dr. Clive Marsh

ACKNOWLEDGMENTS

I would like express my sincere appreciation to my chief supervisor Dr. Huub Bakker, Senior Lecturer for making this project available and for being instrumental in getting the financial assistance from the Foundation for Research in Science and Technology. My thanks for his help, advice and patience without which this research would not have been possible.

Thanks to my second supervisor Dr. Clive Marsh, Lecturer, for his time and attending to me whenever needed.

Thanks to Mr. Tony Wiltshire, Director, Industrial Interface Company, Auckland, for allowing me to carry on this research at his company's premises.

Thanks to Dean and Phil at the Industrial Interface Company for providing the help whenever required.

Thanks to the Foundation for Research in Science and Technology for providing the financial assistance to this project.

My thanks to technical staff especially John for his help in providing me with the required software/software manuals and posting the software from Massey to Auckland whenever required.

Thanks to my fellow postgrads especially Nigel Russel, Ph.D. student for his help throughout this project.

Thanks to Dr. Wyatt Page, Lecturer for his help and time at the beginning of this project.

I would also like to thank Dr. Bob Hodgson, Head of the Department for giving me an opportunity to gain the coveted postgraduate qualification from the Department of Production Technology.

I would like to thank my wife for helping me in proof reading.

Finally I would like to thank my parents; my dad for his patience, for his financial assistance through out my college days and university days, my mum, an academician in her own way, for her strong belief in education and providing all of us with good education. A *special thank you* dad and mum, I will be eternally grateful to you.

We are what we repeatedly do.
Excellence, then is not an act, but a habit.

ARISTOTLE

Table of Contents Page No

SUMMARY

Due to their low cost and reliability SCADA systems have found great favour in industrial control/monitoring operations. Control system engineers have long wanted to incorporate complex real-time simulations into SCADA systems but have been put off by the long development times. This project deals with a method for automatically generating background simulation/control tasks directly from a development environment to run in a SCADA system. The Matlab/Simulink simulation environment is used in conjunction with the Real-Time Workshop (both from The Math Works) which can build real-time simulation tasks directly from the Simulink model. Data is passed between the stand alone simulation model and the real world by the use of specific Real-Time Workshop (RTW) device drivers which are written in C. These device drivers can be used to communicate with the FIX SCADA database using a library of EDA (Easy Database Access) functions. The device drivers currently exchange data between the simulation model and the SCADA database via disk files. This restriction may be removed with the future release of the Real-Time Workshop software.

This document describes:

- A method for automatically generating background simulation tasks from Simulink graphical models that can access a SCADA database in real-time.

- The application of this method to a simulation model of a falling-film evaporator.

1 INTRODUCTION

Control system engineers have always envisaged the prospect of using the real-time models in an industrial setting.

The inclusion of the real-time models can benefit industry in the following ways.

1. Operator Training - The operator can learn about how the various process react to control actions with the help of simulation models without affecting the real process itself.

2. Control Systems testing - The simulation models can be helpful in testing the control system software prior to trialing it on the real process.

3. Process Monitoring - Operators can compare the real process outputs with the simulation model outputs. This helps them in stopping the process when unusual conditions occur.

4. Testing for optimum operating conditions - Simulation models can be used to test for optimum operating conditions or for testing a certain operation at a new operating condition without affecting the real process.

5. Implementation of advanced control strategies - Advanced control strategies such as multivariable control, model predictive control and non linear control can be implemented as a real-time model without the development of separate real-time software.

Even though using the real-time models can benefit the industry as mentioned modeling and real-time models have not found much favour in the industry. The reasons for this may be as follows:

1. Lack of awareness - Most of the plant managers/operators fail to understand what modeling results in and how it can improve the overall plant operation.

2. Lack of expertise - There is no expertise and/or tools in the company to develop the simulation models and implement it.

3. Cost of modeling - Producing a simulation model incurs significant costs.

4. Cost of implementation - Once the model is developed in the development environment it has to be transferred to the industrial platform. The cost of this transfer is high as the model software has to be more robust than the general purpose software.

In order to produce real-time simulation models for an industrial setting there are two significant environments required. These are the development environment where the model is developed and secondly the implementation environment, where the model is used.

1.1 Development Environment

There are many tools available for development of a model and for simulation. Many of these tools are specific to particular areas and in general are not portable. Examples of these are:

- Hysim, a steady state process modeling package.

- SimFactory, a discrete event factory simulation package.

- Comnet, a communications network simulation package.

A second category of development tools for modeling and simulation are also available. These allow for the direct solution of differential equations, the direct entry of discrete and continuous transfer functions and calculations which make them suitable for the development of models. These tools can be characterised by the following attributes:

- Flexible - The development tool must be flexible so as to handle any type of model that might be constructed including analytical and black box models.

- Fast development - The models have to be developed and altered quickly and simply.

- Nonreal-time - The ability to run at nonreal-time rates so as to allow maximum use of the computing power of the platform system. That is to say, faster than real-time for simpler models and slower than real-time for the models which are too complex to be simulated at real-time speed.

- Mathematical basis - The development tools must be rich in mathematical functions and operations so as to handle both analytical and black box models.

- Graphing Interface - The developments tools must be able to display graphs.

1.2 Implementation Environment - Industrial control/monitoring systems

SCADA (Supervisory Control And Data Acquisition) systems have found great favor in industrial control/monitoring operations due to their low cost and reliability. The SCADA packages can be characterized by the following attributes:

- Robustness - The SCADA packages should be reliable and tolerant of failures in communications, instrumentation and equipment.

- Real-time operation - The SCADA systems must run in real-time to provide information and control actions when required.

- Communications - These systems should be able to communicate with plant devices and other computers.

- Graphical Interface - The SCADA systems should be able to present information in a wide variety of forms, in real-time and to accept operator input in real-time.

- Predictability - The SCADA packages must be able to predict the performance of any given configuration.

These attributes that characterize the SCADA packages must also hold for any real-time simulation running in the same environment.

1.3 Dichotomy

The attributes that make a suitable development environment are not those of a suitable implementation environment. They are sharply opposed to each other for instance as regards real-time versus nonreal-time operation or robustness versus flexibility. This leads to a conflict which can only be resolved by using separate development environment and implementation environment or by compromising with one or other attributes.

If we use separate environments the problem then is how to transfer the model completed, tested from the development environment to be implemented on the industrial platform .

This can be done by converting the developed model into a programming language like C, that can be compiled into a stand alone application. Another way of implementation method is to redefine the model in an implementation language like Cicode.

The development of a simulation in C is often time consuming, especially since the development of such simulations are carried out in a development environment with the final model being converted to C code afterward. One of the major limitations of writing simulations in C is the cost of programming, while the robustness of the resulting code depends upon the programmer producing it and does not encourage an in-house solution. Besides having to recost the simulation into any other language is time wasteful.

1.4 Possible Solution

Once this gap between the development and implementation environments is identified the next question would be whether the process of transfer from the development environment to implementation environment can be automated.

The aims of this project were:

1. To produce software that would allow simulation models to be automatically compiled into a background task that can access a SCADA database in real-time.

2. To produce an example simulation model that would demonstrate the practicality of the software.

3. To document the software, methods and procedures developed.

This report includes the following:

1. A brief introduction to evaporators.

2. Methodology adapted in completing this project.

3. An overview of the products available for the development and implementation environments.

4. A brief discussion of the model development and its testing.

This project was developed for the company and was supported by the Foundation for Research in Science and Technology. As a part of the project the software developed, and its application to a pilot simulation model, was documented for the company. The details of these are given in Appendix 1.

Appendix 1 contains the following details:

- The development of the chosen mathematical model.
- The description of the chosen development environment.
- Implementation of the simulation model in the development environment.
- Results of validation test on the model.
- The description of the software developed.
- Application of the software developed to the simulation model.

The main thesis body contains the important aspects of the topics involved. For the more detailed information the reader will be referred to Appendix 1.

2 PRODUCT OVERVIEW

2.1 Development Environment

There are large number of possible development environments available. Of the simulation environments found in universities and industries perhaps the most commonly used is Matlab/Simulink from The Mathworks (Natick, Massachusetts, USA). This can be used in conjunction with the Real-Time Workshop (also from The MathWorks) to produce standalone real-time applications.

2.2 Reasons for using Matlab/Simulink as Development Platform

Matlab is widely used in industries for research and solving engineering and mathematical problems. Matlab is easy to use and available at low cost; reasons which have also made it popular in universities. Matlab comes with application specific solutions called toolboxes that are comprehensive collections of Matlab functions that extend the Matlab environment in order to solve particular classes of problem. Toolboxes include a Control Systems Design Toolbox, a System Identification Toolbox, a Neural Network Toolbox and more.

An additional layer on top of Matlab is Simulink, a dynamic simulation environment with a graphical interface. This uses the Matlab core numerical engine and can also access Matlab data structures and M-files.

Another product from the Math Works, the Real-Time Workshop, can convert Simulink simulations into real-time, standalone executables by generating C source code that can be compiled and linked via a commercial C compiler such as the Watcom compiler.

Simulations can also be developed directly by writing the applications in the C programming language. One of the major limitations in doing so is the cost of programming, while the robustness of the resulting code depends upon the programmer and does not encourage an in-house solution. Being specifically a simulation development environment Matlab/Simulink requires a fraction of the time for development that it would take to write a program in C. Once the mathematical model is developed the simulation model can be constructed using the Simulink function blocks, the Matlab set of numerical tools as well as the analysis and display tools.

2.2.1 Introduction

The Matlab/Simulink environment consists of a numerical (Matlab) environment and a simulation environment (Simulink) which are closely linked. Models are developed in the graphical simulation environment, Simulink by inserting and linking blocks. There are a large number of blocks in the Simulink block library which provide for flexible simulation options.

Matlab is a technical computing environment for high performance numeric computation. The name Matlab stands for *Mat*rix *Lab*oratory. It is an interactive system whose basic data element is a matrix. In industrial settings Matlab is used for research and solving practical engineering and mathematical problems.

Simulink adds a dynamic simulation layer to Matlab that features a graphical programming language while retaining all of Matlab's general functionality. Simulink is a graphical mouse-driven environment that allows the user to model systems via block diagrams. It can handle linear, non-linear, continuous time, discrete time and multivariable systems. Simulink models can incorporate blocks containing Matlab commands or functions and Matlab's complete set of numerical and visualization tools that can be used to analyze and display the results of Simulink simulation.

The Simulink Real-Time Workshop (RTW) is an automatic C language code generation environment for Simulink. It produces C code directly from Simulink graphical models then compiles and links this code with other application modules and user written modules to produce a stand alone program using one of the available compilers. The compiler is called from Simulink so that the entire sequence can be run from within Simulink.
Figure 2.1 illustrates the functionality of Matlab/Simulink and Real-Time Workshop.

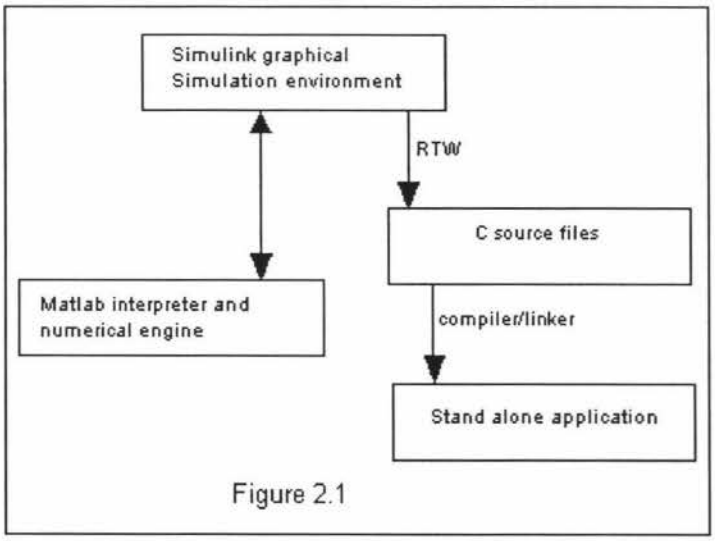


Figure 2.1

Appendix 1, Chapter 4 provides a more detailed information about Matlab/Simulink and Real-Time Workshop.

2.3 Implementation Environment

2.3.1 Introduction

The SCADA (Supervisory Control and Data Acquisition) packages are among the best candidates for an implementation environment because of their suitability to small and medium scale industries within New Zealand. SCADA packages, running on single, generic PCs, have grown from small single-computer packages to distributed computer systems. They are noted for their compatibility with most PLC types. This is in contrast to the Distributed Control Systems (DCS) which were originally large, expensive and proprietary systems that have become smaller and less expensive.

SCADA packages are industrial automation software. These packages provide real-time data to plant personnel and other software applications throughout the plant. The SCADA software performs the basic functions of data acquisition and data management. Data acquisition is the ability to retrieve data from the plant floor and to process the data in the reusable form. Data can also be written back to the plant floor, thereby establishing the critical two way link that control and application software require. Once the data is acquired, it is manipulated and channeled according to the requests of the software applications. This process is called data management. These two functions form the core of industrial automation software and thus maintain the integrity of the data.
The SCADA software perform the following functions:

* Monitoring
* Supervisory control
* Alarming
* Control

Figure 2.2 illustrates these functions:

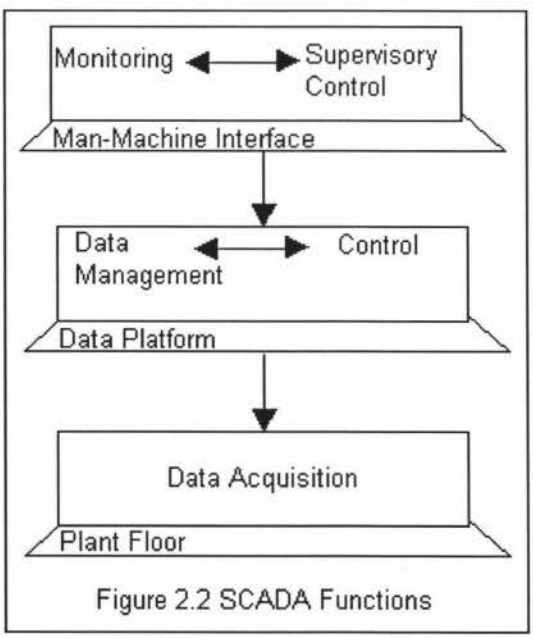


Figure 2.2 SCADA Functions

The ability to display real-time data to operators is called *Monitoring*.

The ability to monitor real-time data coupled with the ability of the operators to change the set points and other key values as on when required directly from the computer is *called Supervisory Control.*

The ability to recognize exceptional events within the process and immediately report these events is called *Alarming*.

The ability to automatically apply algorithms that adjust process values and thereby maintain these values within set limits is called *Control.*

2.3.2 Desirable Features of SCADA Packages

Some of the desirable features of SCADA packages are as follows:

- Robustness: SCADA packages must be robust. They should be reliable and tolerant to failures in communications, instrumentation and equipment.

- Real-time: SCADA packages must run in real-time to provide information and control actions when required.

- Speed: SCADA packages should process the data immediately and able to guarantee a response in case of eventuality within milliseconds.

- Communications: SCADA package must be able to communicate with different types of PLC and other input/output devices.

- Graphical Interface: SCADA packages must be able to provide information in a wide variety of forms and to accept the operator input both in real-time.

- Security: These must able to provide a sophisticated security system.

- Ease of Use: These systems must be easy to use and program.

- Easy Access to Data: All SCADA packages should have a means of accessing the data (reading and writing to data).

- Price: SCADA packages must have an affordable price tag on them and should be available at low cost to the customers.

There are various SCADA packages available as implementation environments. Some of them are:

- FIX from Intellution, USA.
- InTouch from Wonderware, USA.
- Citect from Ci Technologies, Australia.

- Bridge View from National Instruments, USA.
- RealFlex from BJS, USA.

As regards to this project the main features that were looked for in a SCADA package were as follows:

- Easy Access to the SCADA database.
- Good graphics capabilities.
- Availability of the package at low cost.
- Widespread use in industry.

Summaries of the features, advantages and disadvantages of some of these packages are given in the following sections.

2.3.3 FIX – Product Description and Features

The FIX (Intellution) is 32 bit SCADA software running on Windows 95 and Windows NT and written to Microsoft's Win-32 specification. This means that, unlike 16-bit software systems, FIX provide pre-preemptive multitasking, multithreading, plug and play hardware compatibility and symmetric multiprocessing.

The main function of any SCADA software is to make raw data available to plant personnel and managers in formats that are easy to understand. FIX acquires raw data from the process hardware, like PLC's, through a software interface called an I/O driver. The I/O driver software reads data from the I/O device and transfers that data to addresses in the Driver Image Table. The Scan, Alarm and Control (SAC) program reads the data from the Driver Image Table, processes it and transfers the data to the process database. Collectively, the I/O drivers, SAC and the process database make up the data acquisition and management functions of FIX software. The heart of the FIX software is the process database. FIX applications read and write data to a process database. The database reads and writes information to the Driver Image Table.

The database is a representation of the process created by looping together blocks of process control logic. The process database is created with the Database Builder which consists of blocks and chains. A block or tag is a coded set of process control instructions that perform a specified task. Each block has to be provided with several parameters through dialog boxes in the Database Builder.

The Graphics application provides the one that provide the plant personnel with the process information by using high resolution color computer screens and easy to use graphic and data formats.

All the computer displays that allow the operator to interact with the real-time data are developed with graphics application. FIX Graphics application consists of two main programs: Draw and View. Draw provides the display

designer with graphic text, data, animation and charting tools. View provides the operators with commands and ways to interact with the Draw displays in real-time. For direct display of data, Graphics application provides a variety of links that display system or process data in number of formats and contain many configurable options. Links can also be used to write values to the database.

The Draw desktop includes tool box, color box and a cursor position indicator. The items which are placed in the tool box appear as buttons. The toolbox can be customized so that only the development tools which are used most such as snap to grid, align, group, text etc., appear as buttons. The toolbox can be sized and moved like any other window. The color box displays up to 256 different colors. Drawing objects can scale, rotate, change color and move based on process values.

FIX Draw comes with graphic wizards called Dynamos. A Dynamo is an object, link or group of objects that can be stored in a Dynamo set. Draw also provides pre-built dynamo sets containing common process dynamos such as pumps, valves, pipes, meters, face plates etc., that can be pasted into drawing application.

The Historical Trending application provides ability to sample real-time data at operator specified rates. This data can be used for regulatory reporting, process analysis and optimization.
FIX comes with Easy Database Access (EDA). EDA is a library of subroutines that provide easy access to FIX data. The data may be of different types, for different tags and reside in different nodes. Information within FIX system is referenced by Node, Tag and Field. A node contains process database which contains many tags. Each tag refers to a database block which contains many fields. In order an application to read or write to a particular field it must refer to the field by Node-Tag-Field (NTF) identifier. FIX provides sophisticated, user based security system that enables user to precisely control all access to data and system applications. FIX also provides with vast I/O driver library. The features of I/O drivers include sophisticated redundant connections, communications failure detection and recovery.

2.3.4 CITECT – Product Description and Features

Citect is a SCADA package from Ci Technologies, Australia. Citect is 32 bit automation software available for Windows 95 and Windows NT. It is sold with client server architecture, Cicode (a complete control and monitoring language) and communication software. Citect is flexible and can be tailored for small or large applications accordingly.

A Citect project consists of two major elements: Graphic pages and Databases. Graphic pages provide windows into the process. Objects are the basic building blocks of a graphics page. These can be moved, sized, reshaped and copied. The graphic pages are created and edited in the graphics builder.

Databases store configuration information that is used in the Citect run time system to control and monitor the plant.

Citect provides templates for all common page types to ease the creation of graphic images. Templates are tried and tested page designs that user can adapt to his own environment. Examples are Page Menu template, Trend templates, Alarm templates etc. These templates are grouped into several styles, in a variety of page resolutions. A particular object or group of objects, including animated objects and bit mapped objects, which are regularly used can be stored in a library as a library object. These can be recalled and repositioned on any location in a graphics page.

Citect also provides several predefined library objects and a single intelligent object called a genie. A genie is comprised of a group of objects that are related to each other. Citect has two types of genies. Genies that are added to the graphics pages when a system is configured. Super Genies that are accessed with a command at runtime. A super genie is an individual page to which user can pass information when the page displays in the run time system.

Citect comes with Cicode. This is an easy to use computer programming language designed for plant monitoring and control applications. Cicode is a structured language similar to languages such as C or Pascal. It can be used to access all real-time plant data, historical data, operator displays, external databases etc. With Cicode several tasks can be executed simultaneously. Tasks can execute either in foreground or background mode. Cicode is a compiled language so it runs fast and efficiently and includes over 500 in built functions. Functions are reusable and transportable. Users can write their own functions using the standard functions as a base.

Citect trends provide a visual representation of past and current plant floor activity. Any plant floor variable can be logged and trended. Historical data collection continues even when the display is not active. Templates can be used to create multiple trend displays or a command can be used for a single trend display.

Citect protects the plant equipment through alarming. The Citect alarm facility constantly monitors equipment and alerts operators of any equipment fault or alarm condition. Security can be added to restrict runtime access.

2.3.5 INTOUCH – Product Description and Features

InTouch from Wonderware is a graphical application generator for industrial automation, process control and supervisory monitoring. It allows the user to create, on a computer display screen, graphical representations of the physical devices (gauges, meters, knobs, switches, etc.) used to control equipment in a factory or process control system and also to animate them with a wide range of properties to replicate the real process. Types of applications include discrete, process, DCS, SCADA and other types of manufacturing

environments. InTouch is a graphics program and has a proprietary user interface that does not necessarily conform to Microsoft Common User Access Standard.

InTouch does not have preemptive multitasking. The application scripts and data collection run only when called by Windows. For example when re-sizing a window other activities, like alarming, stop.

InTouch graphics tools are available in a configurable toolbox from standard pull down menus. InTouch design tools make it easy to draw, arrange align, layer, space, rotate etc. InTouch graphics support any resolution. InTouch includes pre-configured complex graphic objects called Wizards that can be duplicated and modified. Frequently used objects can be added to the InTouch toolbox for easier access.

InTouch comes with Extensibility Toolkit. This toolkit provides quick and easy access to the InTouch database from programs outside the InTouch. Included is a set of APIs that can be combined and integrated within selected programming languages to read and write from the InTouch database. This toolkit supports programs written in Microsoft and Borland C, Turbo Pascal and Visual Basic for Windows.

2.3.6 Advantages and Disadvantages of FIX, InTouch and Citect

Advantages of FIX:

- GUI is well integrated both in terms of the various packages (Draw, View and Database Builder) and in terms of networking (One can open a picture drawn on another node without leaving View).
- Preemptive multitasking through standard windows multimedia calls.
- Robust and powerful architecture and is user friendly.
- Comes with Easy Database Access Toolkit.
- QuickView is a very helpful tool during the development of interface screens. FIX has got good zooming features.
- Has widespread use in industry.
- Available at low cost especially in mid size systems.

Disadvantages of FIX:

- Inaccuracies in the re-sizing and zoom tools along with the lack of typical re-sizing arrows makes manipulation of objects a bit difficult.
- Cannot wrap text or a symbol on a button created in Draw.

Advantages of Citect:

- Comes with Cicode language which is flexible and almost embedded. Cicode has much in common with C and supports the passing of arguments and a hierarchy of variable types.

- Good integration of Cicode with Citect internals and windows environment.
- Good library of useful bit maps which can be imported.

Disadvantages of Citect:

- Can be used for real-time simulations. But it is not easy to write development code and is time consuming.
- Relatively new entry into the market.
- Cicode is required for many functions that should be standard.

Advantages of InTouch:

- Good graphics tools.
- Recently released extensibility toolkit. This includes a set of Application Programming Interfaces, APIs, that can be combined and integrated within selected programming languages to read and write from the InTouch database.

Disadvantages of InTouch:

- Proprietary distributed control system. The user interface, networking, file and script language are all proprietary. They may not necessarily conform to Microsoft's common user access standard. This increases startup time and training.
- More expensive, especially in mid size systems.
- User must write C - like code for most advanced functions.

2.3.7 Reasons for selecting FIX as the Implementation Environment.

The main reasons for selecting FIX as the implementation platform were its; more widespread use in industry than any of the other SCADA packages, availability at low cost, better integration in smaller networked systems, Easy Database Access whose library of subroutines provide easy access to FIX database.

3 METHODOLOGY

3.1 Software Development

This project aims at automating the process of transferring a simulation model from a development environment to an implementation environment. The solution for this might come in two parts, automating the process of creating program code from the development environment and linking this code to a SCADA package which provides the required graphical and data collection functions. The second problem was simplified by the fact that most SCADA packages provide a library of calls to access the SCADA database. The problem then was to incorporate these library calls into the program code produced by the development environment.

Once this stage was completed the next stage would be to test and demonstrate process by identifying a model and converting it for a target implementation environment.

3.2 Development of the Analytical Model of the Falling Film Evaporator

The Department of Production Technology has installed a pilot scale, three effect, falling film evaporator as a research and teaching platform in the field of control systems engineering.

Quaak and Gerritsen (1990) developed a dynamic model for the standard configuration (straight-through flow) of a falling film evaporator. The method adopted by them was to split the effect into four sub-systems:

- distribution plate sub-system
- evaporation tube sub-system
- product transport sub-system
- energy balance sub-system

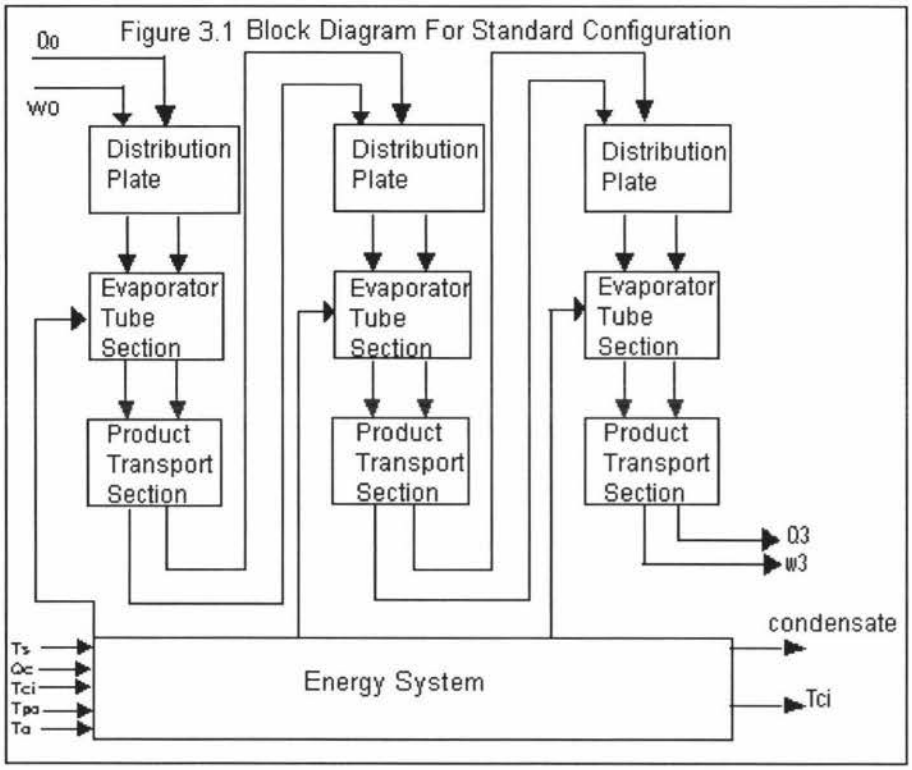
Figure 3.1 shows the evaporator as a system block with all the inputs entering the system from the left side and all the outputs leaving the system from the right side.
Inputs are:

- flow of feed, Qo
- dry matter content of feed, wo
- temperature of feed ,Tpo
- temperature of steam, Ts
- cooling water temperature, Tci
- flow of cooling water, Qc
- ambient temperature, To

Outputs are:

- product outflow, Q3
- product dry matter content, w3

- temperature of cooling water that is used, Tci
- flow of condensate



The product from the top of the effect is sprayed by the nozzle onto the distribution plate subsystem and flows through the evaporation tube subsystem. The product from the evaporation tube sub system enters the product transport sub-system where the evaporated liquor is pumped to the top of the next effect. The distribution plate and evaporation tube sub-system and product transport sub-system contain equations that deal with flow rates and dry matter contents.
The energy balance sub-system is the one where heat and water vapour flows in the evaporator, energy losses to the surroundings are taken care of. The heat supplied to the evaporation tube section is prescribed by the energy balance .

3.3 Building and Testing the Analytical Model

Quaak and Gerritsen's (1990) model was based on the assumption that the level in the separator tank was constant .Assuming that the level on the suction side of the pump is constant the volumetric flows in and out of the sub-system can be considered equal. The author's contribution was to develop new equations for the product transport sub-system which took into account the variable liquid head in the separator.

For the dynamic description of the sub-systems along with the developed mathematical equations refer to Appendix 1, Chapter 3.

At the beginning of the project it was decided to convert a model of the first effect this falling film evaporator into an implementation environment.

Once the mathematical modeling of the first effect of the evaporator was completed it was encoded in the Simulink development environment by the author. Qualitative assessments of the performance of the simulated system were then to be carried out. The model supplied with the operating point inputs was to be tested to check for the outputs from the system using the various simulation algorithms and to see that the simulation of the model was correct.

3.4 Using the Model in a Case Study

Once the simulation model was developed and tested in a development environment it was ready to be implemented on the industrial platform. With the selection of an appropriate implementation environment software had to be developed to link the development and implementation environments.

When the software is developed then the simulation model can be used as an example case study to test the software and to build a real-time application.

4 LINKING THE DEVELOPMENT AND IMPLEMENTATION ENVIRONMENTS

This project initially was aimed at producing a real-time application to run under Windows 3.11 which is a 16 bit operating system. The Real-Time Workshop (RTW) is designed to produce real-time applications that run under 32 bit operating systems. A Windows 3.11 solution was not, therefore, possible.

Applications could be produced to run under extended DOS via one of a number of 32 bit DOS extenders but this imposed some hefty performance penalties. The RTW can also produce Windows NT character mode executables that are essentially DOS applications that use the 32 bit DOS emulation native to Windows NT. Since FIX also runs as a 32 bit application this was the selected platform.

The suggested process is shown in the figure 4.1. The simulation model is connected to the input and output drivers after testing. The Real-Time Workshop is then invoked to produce a C source code file of the model. This is then compiled and linked with the supported C-code from the Real-Time Workshop to produce a 32-bit DOS application which runs under Windows NT as an NT 'character-mode' application. This application contains the real-time model and talks directly to the Fix database. The input driver and the output driver would need to be developed to provide the required link while the 'model' only would be rewritten for each new model.

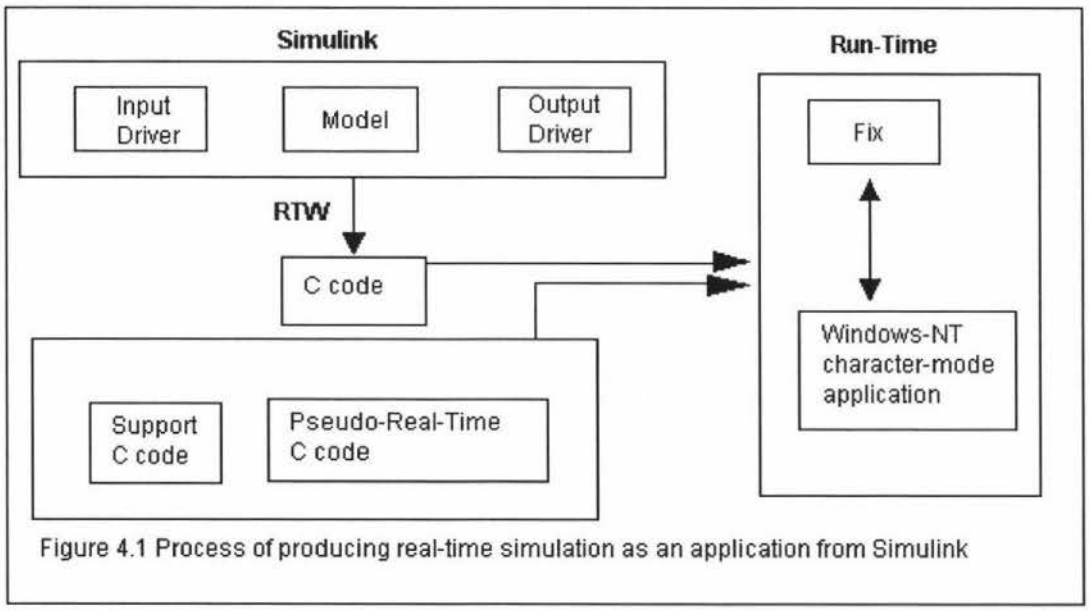


Figure 4.1 Process of producing real-time simulation as an application from Simulink

Two major complications required changes to this solution.

The Real-Time Workshop was written to allow real-time simulation and control of fast mechanical systems. Using the real-time model code imposes a lower limit on the model sample rate of 18Hz or 0.056 seconds due to the hardware timer used. This is much faster than required to run a real-time

application for the process industries. The real-time model code also has the unwanted behavior of stopping the simulation if any overrun occurs. For these reasons the nonreal-time code (nrt-main.c) was used as a basis and rewritten as pseudo real-time application to provide more suitable sampling rates and overrun handling (prt-main.c). This pseudo-real time application is accurate to roughly 0.01 seconds but can run at sample rates ranging from fractions of a second to many hours. If overruns occur the code simply runs as fast as possible and warnings are written to the output screen.

Secondly, the compiler required by the Real-Time Workshop and the Easy Database Access (EDA) library were not compatible. EDA is a library of static ally-linked subroutines that provide easy access to FIX data. The EDA library for FIX is built using the Microsoft Visual C compiler/linker. The RTW, however, does not support Microsoft C and must use the Watcom or Metaware compilers to produce real-time applications. The Watcom compiler cannot read the Microsoft library format. In order to use the EDA library the compiler needs to be able to read the Microsoft library format; the Microsoft Visual C compiler/linker is therefore the best option.

Because of this incompatibility between compilers, the device drivers could not communicate directly with the SCADA database. Instead the real-time model (through its device drivers) and the EDA library must exchange data through disk files. This is shown in the figure 4.2. The Watcom compiler is used to produce the model application with its device drivers and the Visual C compiler is used to produce the SCADA database communication program.

Note that writing to disk files is not as disk intensive as it might seem. The Windows NT operating system uses a disk caching mechanism which can reduce disk access by fraction of 3 or 4.

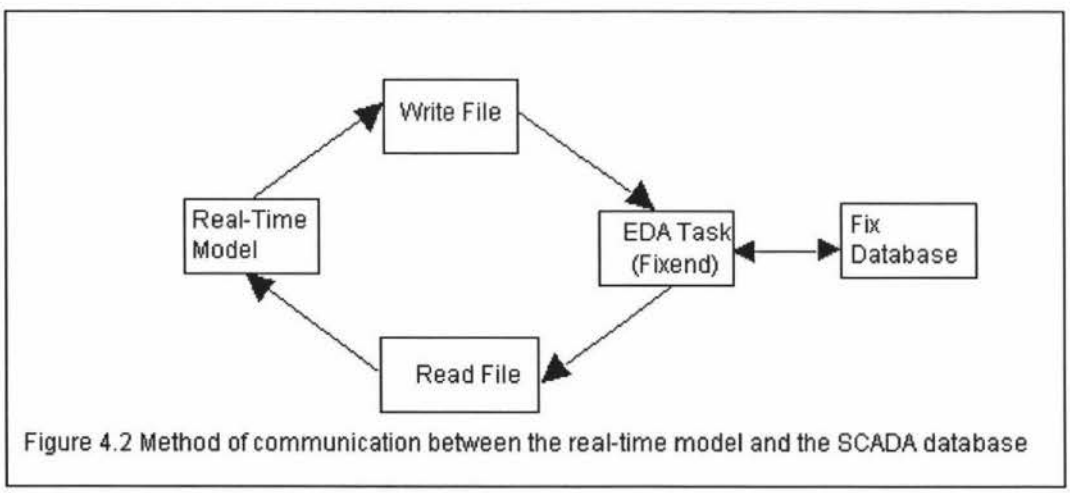


Figure 4.2 Method of communication between the real-time model and the SCADA database

The simulation model is compiled and linked into a real-time application which reads inputs from a read disk file and writes outputs to a write disk file. A standard application called Fixend reads to/writes from these files and transfers the information to/from the SCADA database.

The device drivers and Fixend application were written as part of the project. Their functions are given in more detail in the following sections.

4.1 Functions of the Device Drivers developed:

The simulation model and the FIX database communicate through the device driver blocks. The two device drivers developed are the input device driver, Fixlink In, and the output device driver, Fixlink Out. These drivers are added to the model as normal blocks.

The main functions of the device drivers are as follows:

- On initialisation the device drivers perform a number of validation tests on parameters specified in the dialogue box. The drivers read the configuration parameters from the dialogue box and write them to the configuration file specified where they can be read by the Fixend program.

- During execution the input device driver reads the variables from the read file and stores them as a vector of outputs from the block. The output device driver reads the vector of inputs to the block and stores them in the write file.

4.2 Functions of Fixend (EDA task):

The functions of Fixend are:

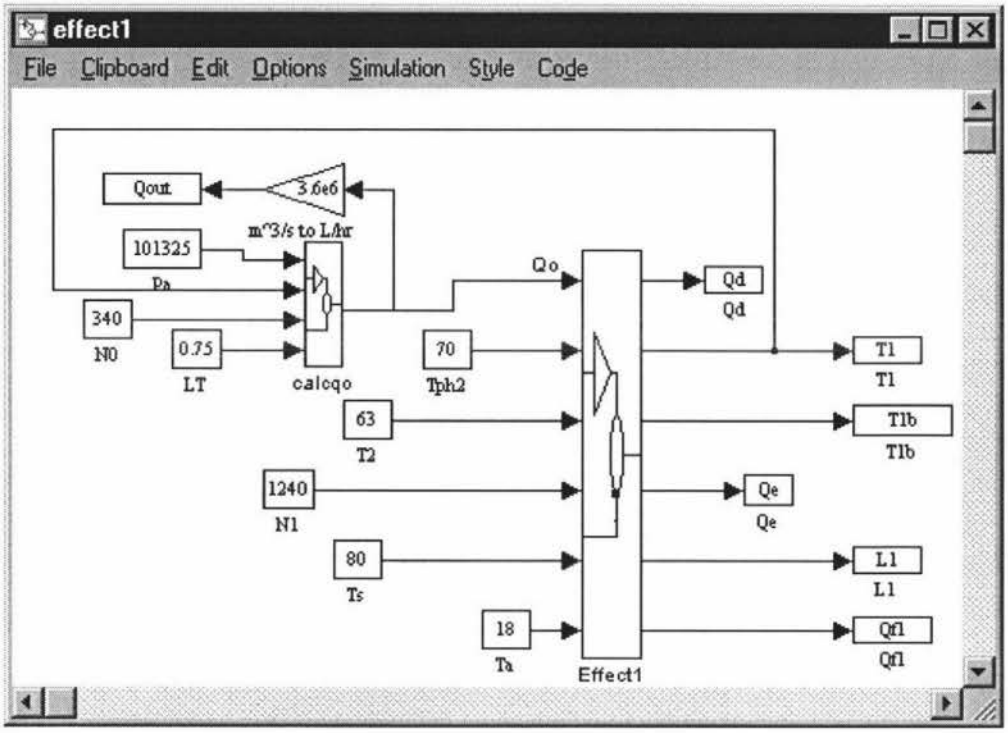
- To read configuration information from the configuration files. If the configuration files are not found it signals an error and continues to look for them.

- To open the read file, get the data from the FIX database and write to the read file.

- To open the write file, read the data from the write file and send the values to the FIX database.

5 BUILDING THE EVAPORATOR CASE STUDY

The equations for the first effect of the evaporator model were implemented in Simulink by the author. The Simulink block diagram below shows the model.



The simulation is an implementation of all the state equations and related algebraic equations developed for the first effect, including the feed system.

NB: The operating point values are taken from the earlier works and are derived through various experiments on the pilot evaporator model installed in the Department of Production Technology at Massey University.

The steady state inputs to the model are:

Tph2 Temperature of the preheater two at 70°C.
T2 Temperature in the second effect which is disturbance input at 63°C.
N1 Speed of the pump at 1240 rpm.
Ta Atmospheric temperature at 18°C.
Ts Steam temperature.
Qo Feed flow rate is a function of atmospheric pressure Pa, feed pump speed N0, feed level LT and pressure in the first effect P1.

Pa 101325 Pascals.
N0 340 rpm.
LT 0.75m.

The outputs from the model are:

Qd Volumetric flow out of the distribution plate.

T1 Temperature in the first effect.
T1b Temperature at the base of the effect.
Qe Volumetric flow out of evaporation tube.
L1 Liquid level in the separator tank.
Qf1 Outflow to the next effect.

The feed flow rate Qout is also set as an output to observe the amount of feed that is being fed to the model.

The top level model has two subsystems; *calcqo,* which calculates the amount of feed flow to the first effect and *Effect1,* which calculates all flows and temperatures.

The following sections explain the subsystem *calcqo* in detail by exploding it.

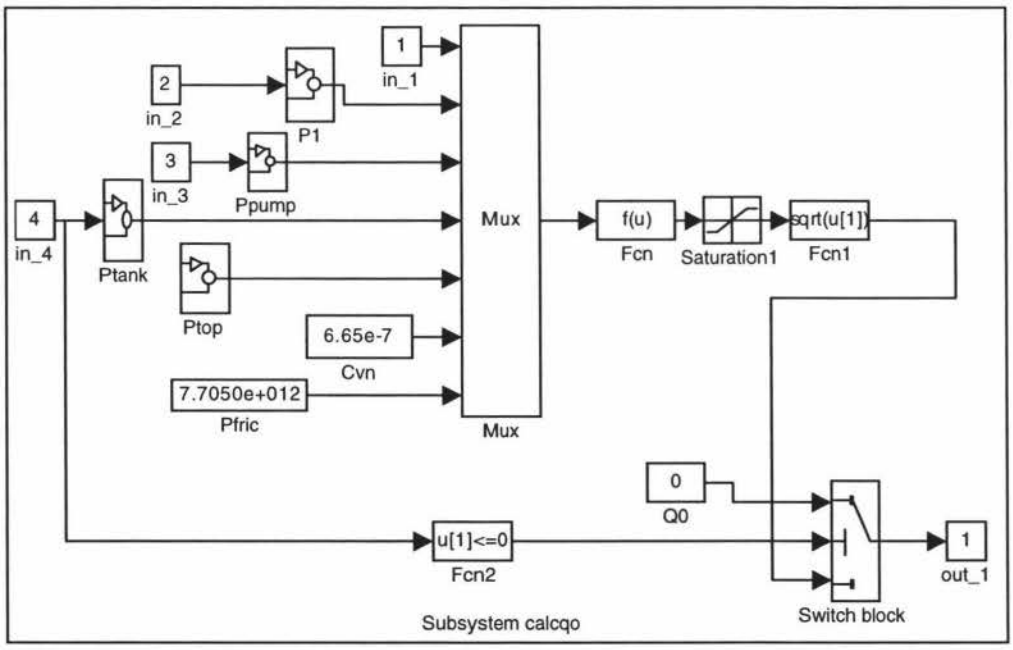
Explosion of Subsystem *calcqo*:

This Subsystem calculates feed flow rate,Qo, to the model.
Q_0 is calculated using the equation:

$$Q_0 = \sqrt{\frac{(Pa - P1 - Ppump + P\tan k - Ptop)}{1/Cvn^2 + Pfric}}$$

Where Pfric is given by

$$Pfric = \frac{32 * \rho 3 * f}{pi^2 * (Lf1/d1^5 + Lf2/d2^5)}$$



In_1 refers to Pa, atmospheric temperature.
In_2 refers to T1, temperature in the first effect.

In_3 refers to N0, speed of the feed pump.
In_4 refers to LT, level in the feed tank.

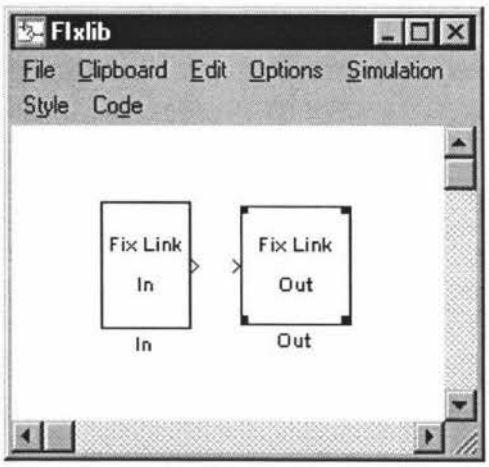
All the inputs to the Mux block are grouped to create a vector of inputs to the function block, Fcn (The width of the vector line from Mux block is same as the number of inputs specified). In the block Fcn u[1] corresponds to the first input, u[2] corresponds to the second input and so on. The Saturation block limits the range of Q0 to be between 0 and 500l/hr. This is done to make sure a negative number is not passed to the square root block following. The function block, Fcn1, calculates the square root of its input and passes the output to the Switch block.

The Switch block will output the calculated Q0 or 0 if the tank is empty (LT<=0).

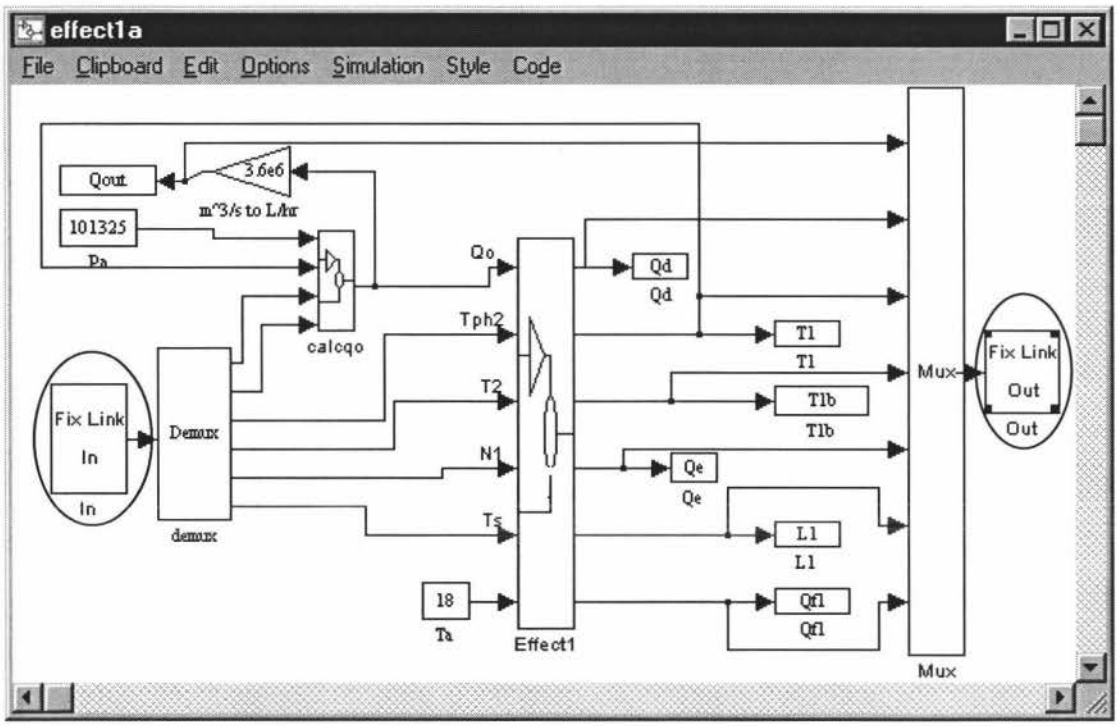
For the detailed discussion of the subsystems refer to Appendix 1, Chapter 5.

5.1 Adding Device Drivers to the Model

The simulation model and the FIX database communicate through device driver blocks. The device driver blocks designed for this purpose are stored in the *Fixlib* block library and labeled Fixlink In and Fixlink Out.

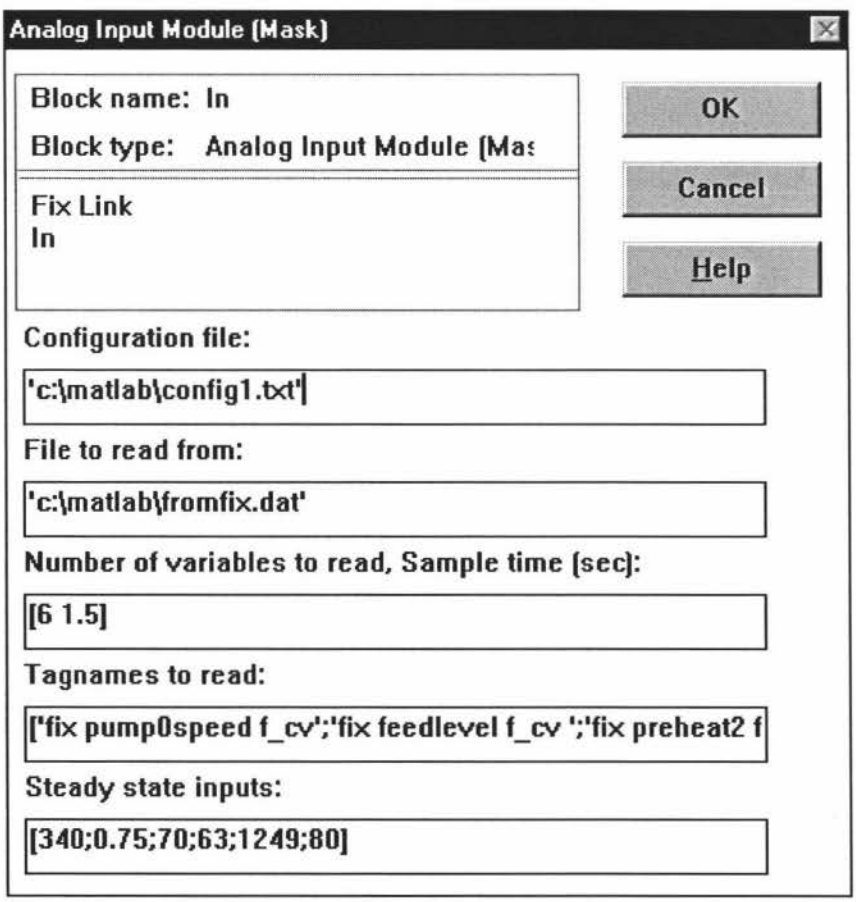


Fixlink In and Fixlink Out are connected to the model inputs and model outputs respectively. An example of the use of the FixLink Out block is shown in the following diagram of the evaporator model.

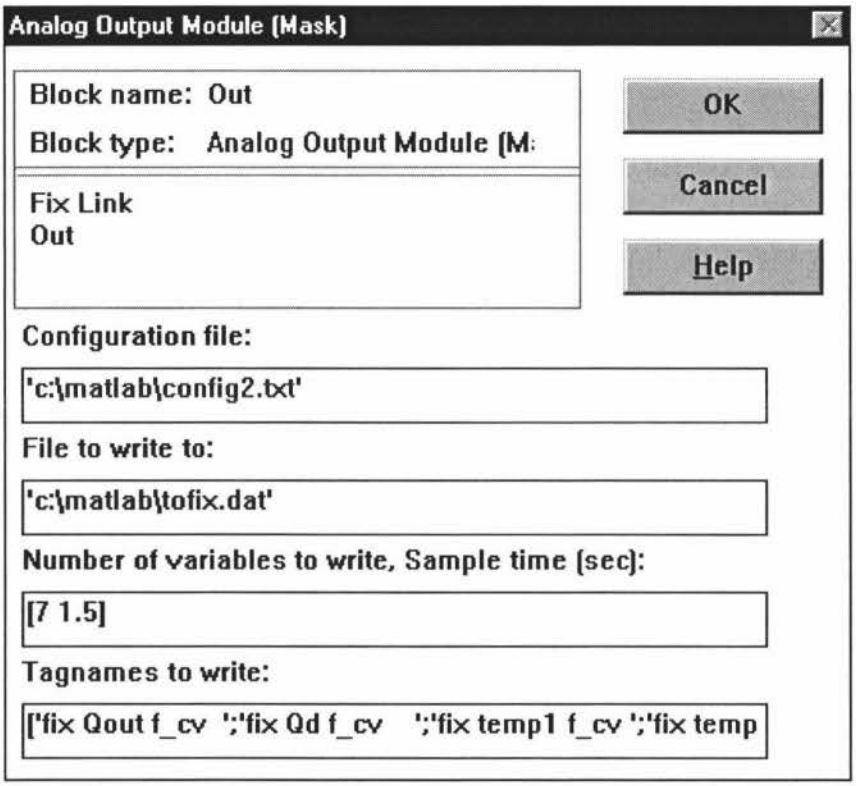


5.2 Configuring Device Drivers

Each device driver block has a dialogue box where the parameters are set. The dialogue boxes below are for the Fixlink In and Fixlink Out device drivers. They show the parameters used in the evaporator example.



Fixlink Out



The various parameters to be entered into the dialogue boxes are:

Configuration file by default is set to c:\matlab\config1.txt and c:\matlab\config2.txt.

File to read from/File to write to is set by default to c:\matlab\fromfix.dat and c:\matlab\tofix.dat.

Number of variables to read/Number of variables to write is the number of model inputs to read/write and it must match with the number of tagnames to read/write specified in the next entry of the dialogue box. The sample time entered is in seconds and must be the same or multiple of the model step size set in the nonreal-time options dialogue box.

Tagnames to read/write are in node:tag.field format with spaces between them and are placed between quotes. Each tagname to read is separated by semicolon.

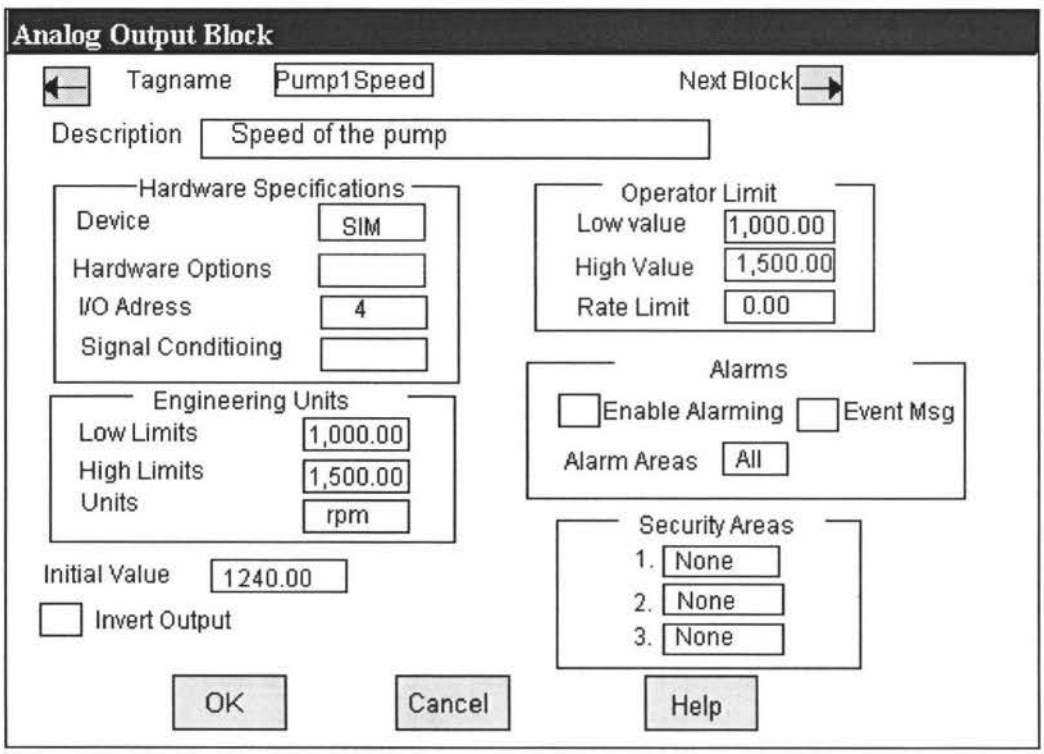
Steady state inputs for Fixlink In is a vector that is of the same length as number of tagnames.

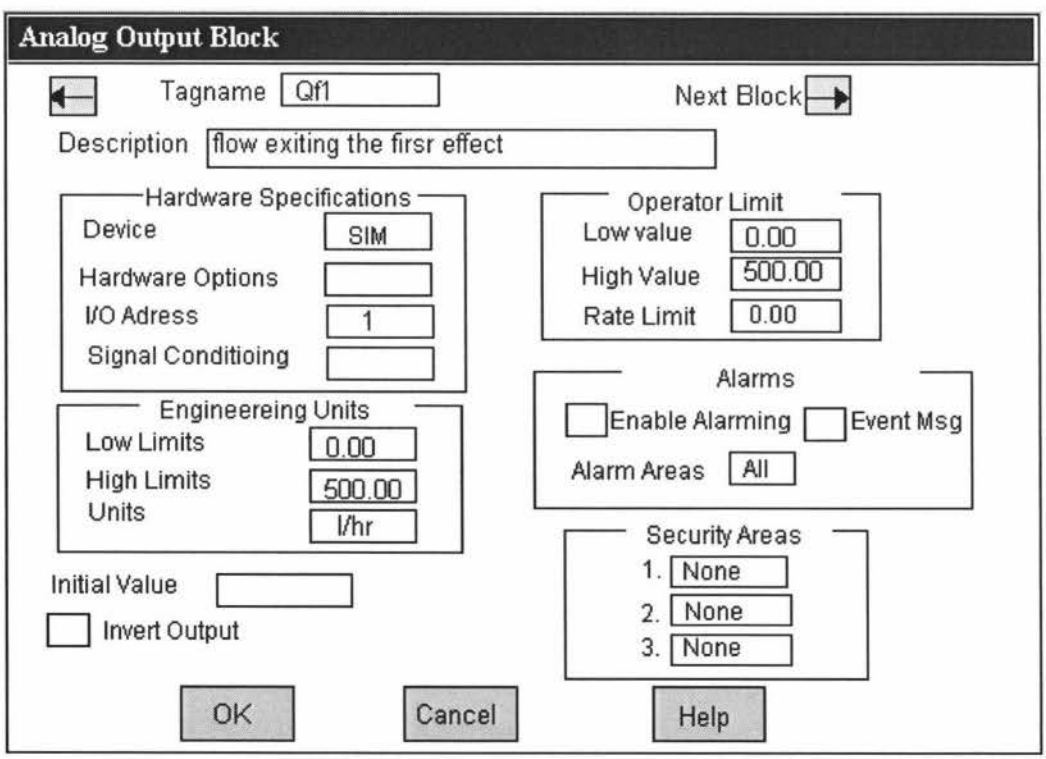
5.3 Setting up the Database in FIX:

FIX applications read and write data to a process database. A FIX database is

made up of blocks. Each block requires several parameters to be provided through dialog boxes in the database builder. The author added all the evaporator model inputs and outputs to the database as analog output blocks.

Analog Output blocks for model input, speed of the pump, and the model output, Qf1, volumetric flow exiting from the first effect, are shown below:





The parameters entered are :

Tagname: The Tagname field displays the name of the block. Each tagname must be unique in the database. The tagname entered should be same as the corresponding tagname in the 'tagnames to read'/'tagnames to write' field of the input/output device driver block.

Description: A brief description of the block.

Hardware Specifications: The hardware specifications of the I/O driver are entered here. FIX comes with a simulation driver called SIM. The simulations driver is useful in testing systems or, in this case, in storing values to access by external tasks. The I/O address is entered in the I/O address field. For analog values the I/O address ranges from 0 to 2000.

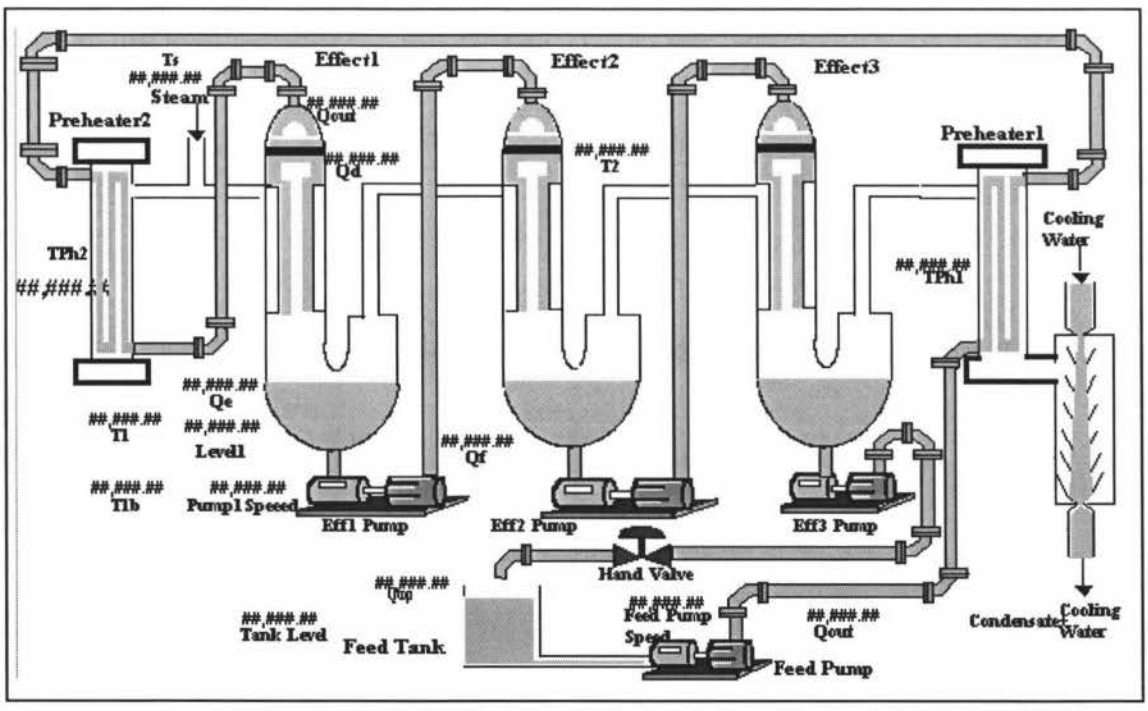
Engineering Units: The Low Limit and the High Limit for the data are entered. The Units field specifies a label for the Low and High Limit fields.

Initial Value: A value between the Low and High Limit fields for the Analog Output block (Applicable for model inputs only).

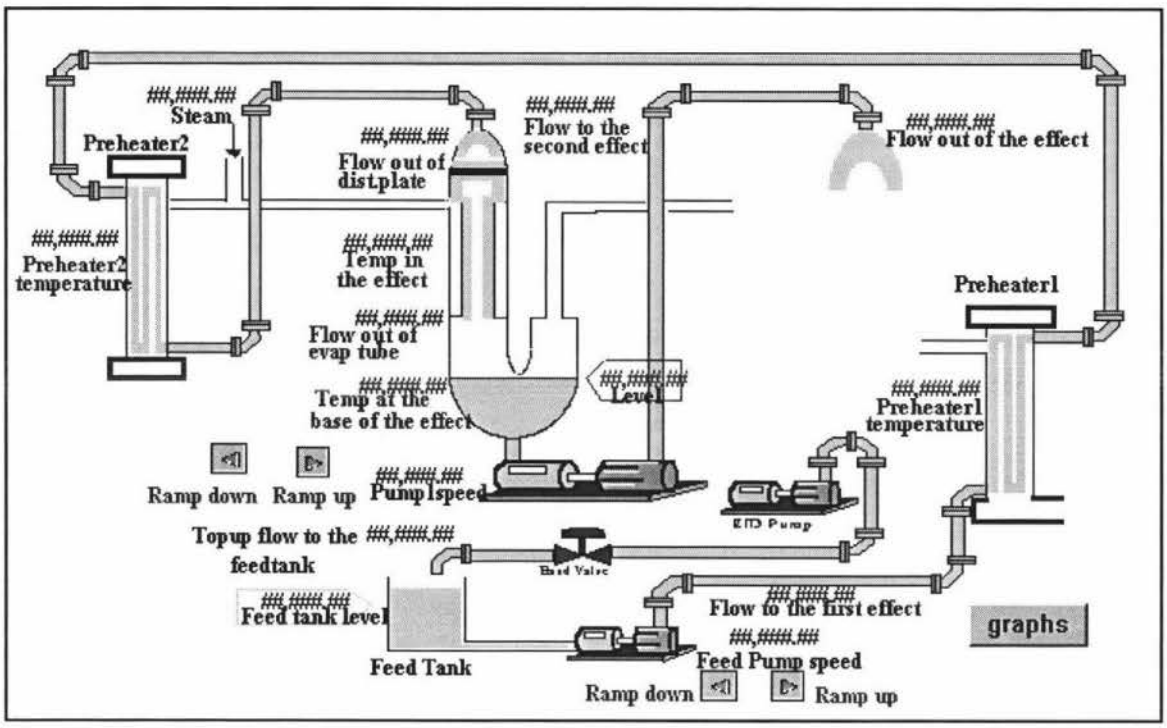
Operator Limits: A value between the Low and High Limit (EGU) fields is entered. If appropriate, the Low Limit /High Limit (EGU) can be used for low value field/high value fields.

5.4 The Evaporator set up in FIX Draw:

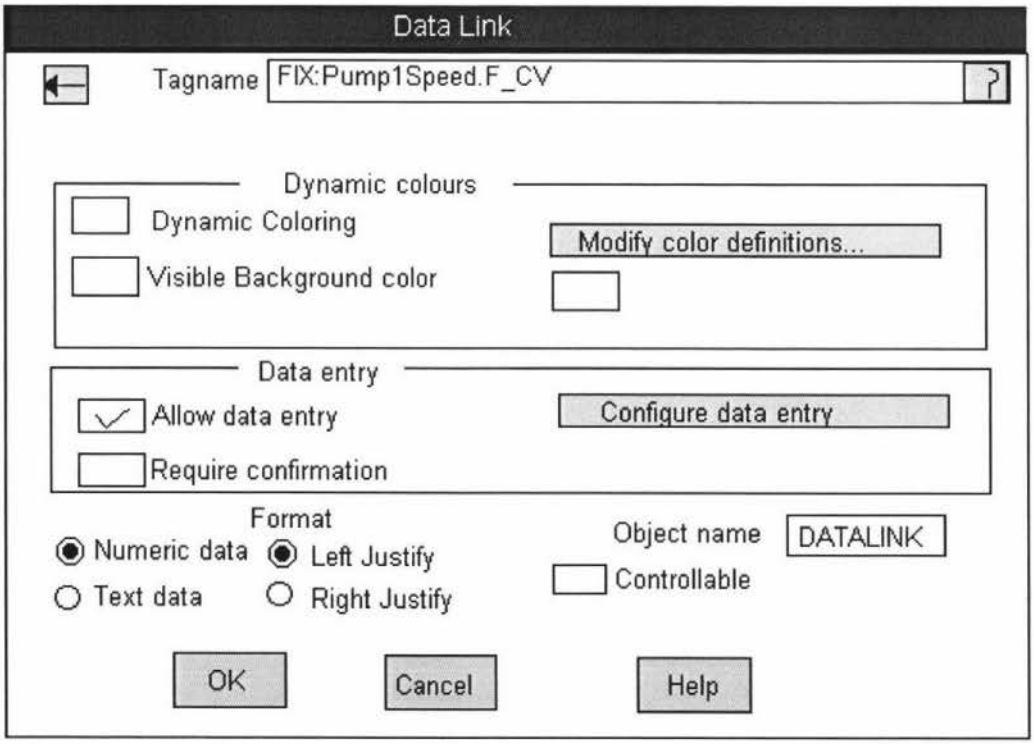
Control screens for the three stage falling film evaporator were set up in FIX Draw by the author(see below). Values from the database would be written where ##,###,## placeholders appear.



The FixSim Software developed is tested by converting the Simulink model of the first effect of the falling film evaporator to run under FIX. The control screen for the first effect of the model is as shown below:



Connections between the control screens above and the database blocks are made by Data Links. The Data Link is the primary tool that allows read/write information from/to the database.



The tagname for the model input/output is entered in tagname field. It can also

be selected by ? button which uses the Field Select dialog box. FIX draw fills in the SCADA node and field. If numeric data is specified then Draw enters F_CV (floating point current value) as the field.

The value displayed by the Data Link can change color according to the value of the database field. To allow data entry into the link by the operator Allow Data Entry check box is checked.

The Fix View program provides the real-time display of the graphics (Draw) application.

6 RESULTS

A simulation of the first effect of the evaporator model was implemented in Matlab/Simulink environment.

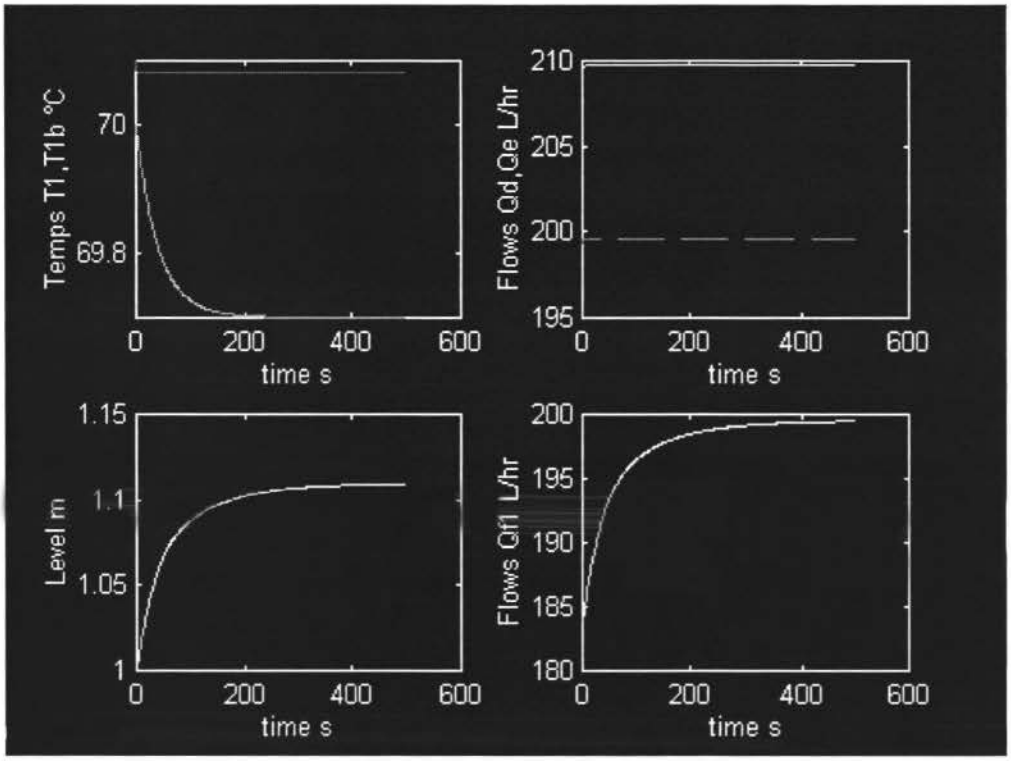
The simulation was an implementation of all the state equations and related algebraic equations developed for a single effect. The model supplied with the operating point inputs was tested to check for the outputs from the system and to see if the simulation of the model was correct.

The input values for the system are summarised as below:

Input	Value	Unit
Feed flow,Q0	209.73	L/hr
Preheater two temperature, (feed temperature) Tph2	70	°C
Effect two temperature, T2	63	°C
Pump speed, N1	1240	rpm
Steam temperature, Ts	80	°C
Ambient temperature, Ta	18	°C
Atmospheric Pressure, Pa	101325	Pascals
Feed pump speed, N0	340	rpm
Feed level, LT	0.75	m

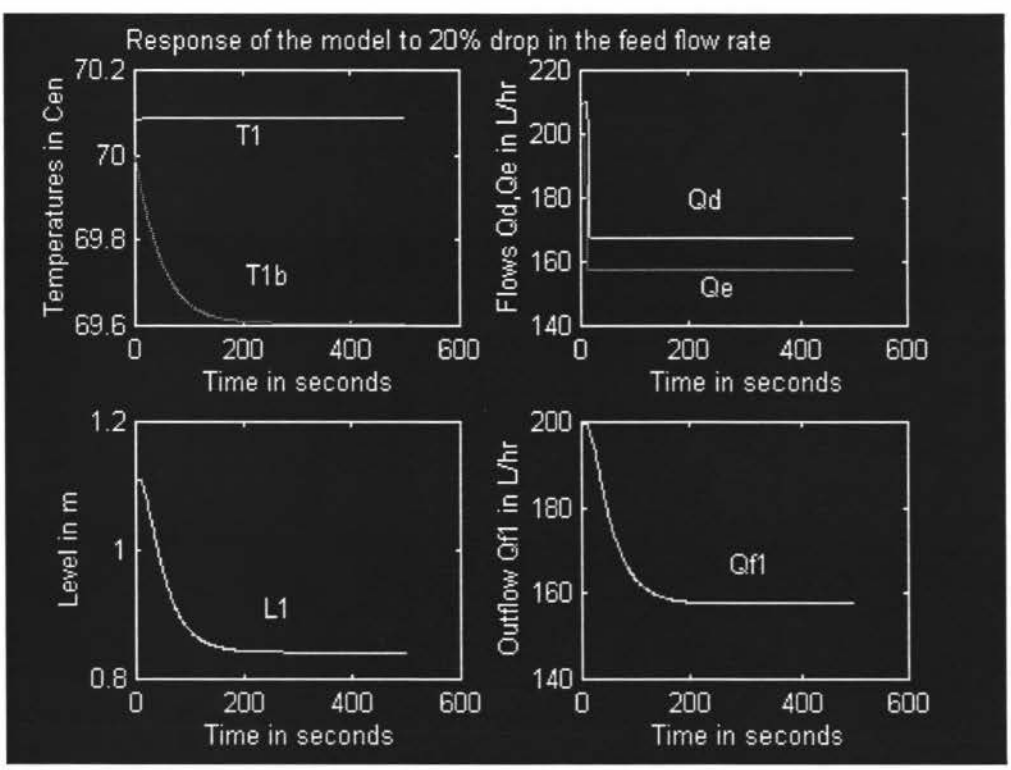
The Integration Algorithm chosen was Runge-Kutta 5 (fifth order) method. Fixed step size of 1.5 seconds was used and the Simulation was run for 500 seconds.

The following plot shows four plots containing various output results of interest. The top left shows that the temperatures T1 and T1b settle very close to the operating point values of 70.1°C and 70°C. The bottom left plot shows the level settling at around 1.1m. The figure on the top right show flows Qd, distributed flow, at around 209L/hr and Qe, flow out of evaporation tube, at around 199L/hr with evaporation rate of 10L/hr. The bottom right shows output flow, Qf1, at around 199L/hr. All values obtained from the simulator agreed well with the results obtained when all the state derivatives were set to zero, and an equilibrium point calculated for the same input values. It was therefore concluded that the simulation of the model had been successful.



6.1 Response of the Model to 20% drop in the feed flow rate

The feed flow rate was reduced to 20% from operating point value of about 209 L/hr to about 167 L/hr and the response of the outputs of the model were observed.



The top left plot shows that temperatures remain relatively unaffected by the

drop in the feed flow rate.

The top right plot shows the new steady state values for Qd, the flow out of the distribution plate, and Qe, the flow out of the evaporator tube. A 20% drop in the feed flow rate results in a 20% drop in Qd. Since at steady state Qd equals the feed flow this drop was as expected. Flow out of the evaporator tube, Qf1, dropped to around 157L/hr from its initial value of 199 L/hr. Again this is roughly the 20% which would be expected.

The bottom left plot shows that the liquid level at the bottom of the separator tank, L1, drops to a new steady state value of 0.85m. The liquid level drops because of the decrease in the feed flow rate. (There is no level control in this simulation as there would normally be on the pilot plant.) Due to the drop in the liquid level, the discharge flow to the next effect, Qf1, is reduced. Because of this reduction in the discharge flow, the liquid level reaches a new steady state.

The bottom right plot shows the discharge flow,Qf1, to the next effect dropping to around 157L/hr. This drop was same as Qe, flow out of the evaporator tube. Since at steady state, flow out of the evaporator tube, Qe, must be equal to the discharge flow,Qf1, this was to be expected.

Overall then it was concluded that the simulation of the model was correct.

6.2 Building the Real-Time Model

Once the evaporator model was successfully tested in Simulink it was ready to be implemented as real-time application running under Fix 32. The device driver blocks were connected and the real-time model was built.

The completed real-time application was run on a 100MHz Pentium computer with 32MB of memory under Windows NT. It was connected to a small database containing 17 blocks which were updated at 1 second intervals. The simulation sampling rate was initially set to 1 second but was eventually slowed to 1.5 seconds before it could be run without significant computation overruns. The simulation used a 4th/5th order Runge-Kutta integration algorithm which could have been replaced by a simpler one to reduce computation time further at the cost of some accuracy.

The simulation model did not show any apparent signs of degraded performance during testing which indicated that the conversion from a Simulink model to a stand-alone real-time application had been accomplished without significant change.

7 CONCLUSIONS AND RECOMMENDATIONS

By incorporating the simulation models into SCADA package the process industry could benefit in number of areas.

1. The plant operator/technician can have an understanding of the process in advance and handle the emergency situations better.

2. The industrial automation system configuration and control strategy can be checked before implementation.

3. Various operating strategies and 'what if' scenarios can be tried without any risk to the plant.

4. Production levels and grades can be changed to determine their impact.

5. Malfunctions and process disturbances can be simulated before they occur and different strategies can be reviewed to determine the best course of action.

The fact that the simulation models are not used in the industry may be due to lack of awareness and due to the lack of suitable development and implementation environments or a ready means of converting from development environment to an implementation environment.

In this project Matlab/Simulink is used as a development environment to develop the simulation model for the first effect of a falling film evaporator. This model is automatically compiled and linked by the Real-Time Workshop into a Windows NT character mode application which is then connected to SCADA database. The SCADA package used was FIX from Intellution.

The simulation model of a single effect falling-film evaporator converted to real-time application has been shown to work successfully and reliably. The simulation did not show any apparent signs of degraded performance during testing which indicated that the conversion of the simulation model to a stand alone real-time application had been accomplished without any significance change.

The current solution converts the single effect of a falling film evaporator into a stand alone real-time application. The complete pilot model of the three effect falling film evaporator could be converted into a real-time application. This requires all the modeling work to be done including the feed system, preheaters and venturi condensers. Once the modeling is completed the simulation model could be developed in the Simulink and a real-time application could be generated using the 'FixSim' software developed.

8 LIMITATIONS AND FUTURE DEVELOPMENT

The real-time application and SCADA database now communicate through disk files. The RTW does not convert Matlab function blocks and S-function blocks to C code which restricts the developer. With the next release of RTW software it should be possible to remove these restrictions.

If RTW can convert Matlab function blocks and S-function blocks to C code then it would be much simpler to implement the whole three effect evaporator.

Possible Scenarios:

1. The Real-Time Workshop supports Microsoft Visual C compiler allowing everything to be done this way.

2. EDA is encapsulated, or released by Intellution, as a DLL file which can be linked to by the Watcom compiler.

APPENDIX 1

DOCUMENTATION MANUAL PRODUCED FOR THE COMPANY, INDUSTRIAL INTERFACE COMPANY

Title Incorporating Simulink Models Into FIX

Table of Contents Page No

SUMMARY

The Matlab/Simulink Real-Time Workshop (from The Mathworks, USA) can build real-time simulation tasks from Simulink graphical models. Data is passed between the stand alone simulation model and the real world by the use of specific Real-Time Workshop (RTW) device drivers which are written in C. These device drivers can be used to communicate with the FIX SCADA database using a library of EDA (Easy Database Access) functions. The device drivers currently exchange data between the simulation model and the SCADA database via disk files.

This document describes:

- A method for automatically generating background simulation tasks from Simulink graphical models that can access a SCADA database in real-time.

- The application of this method to a simulation model of a falling-film evaporator.

1 INTRODUCTION

The Matlab/Simulink environment consists of a numerical (Matlab) and simulation environment (Simulink) which are closely linked. Models are developed using Simulink, which is a graphical simulation environment, by inserting and linking blocks. There are a large number of blocks in the Simulink block library which provide for flexible simulation options.

The Matlab/Simulink Real-Time Workshop converts the graphical models into C code and builds programs that can be run in real-time.

FIX is a SCADA package from Intellution. FIX comes with the Easy Database Access (EDA) library which allows user applications to access the information in the database and to write back results by including calls to this library in their code and linking with the EDA library.

The aims of this project were:

1. To produce software that would allow Simulink models to be automatically compiled into a background task using the Real-Time Workshop.

2. To produce an example simulation model that would demonstrate the practicality of the software.

3. To document the software, methods and procedures developed.

This report includes the following:

1. A description of the example simulation model.

2. An introduction to Matlab, Simulink and Real-Time Workshop.

3. A description of the software, developed.

4. A guide to converting a model into a real-time simulation running under FIX.

2 COMPLETE THREE EFFECT EVAPORATOR DESCRIPTION

A three stage falling-film evaporator has been installed at the Department of Production Technology, Massey University. The evaporator has been designed to evaporate a range of solutions, with normal evaporation of 10kg/hr in each calendria (evaporation tubes). At present it is operating only with water. The evaporator itself is the smallest ever falling-film evaporator to be built in New Zealand by orders of magnitude. Hence it contains dynamics significantly faster than industrial evaporators.

The complete diagram of the evaporator in straight-through flow configuration, is shown in the figure 2.1.

The product (cold water) is fed from the tank to the first effect through two preheaters which bring the temperature of the product to that of the first effect. The first preheater gets the heat energy from the vapour produced by the last effect. The second preheater uses steam as energy source. The product is heated to about 70^{o} C. After the three effects the product is re-entered into the feed tank through a hand valve.

The vapour from each effect is condensed in the shell of the next effect. The venturi condenser draws the condensate from each effect except the last effect where the vapour is partially condensed in the pre-heater.

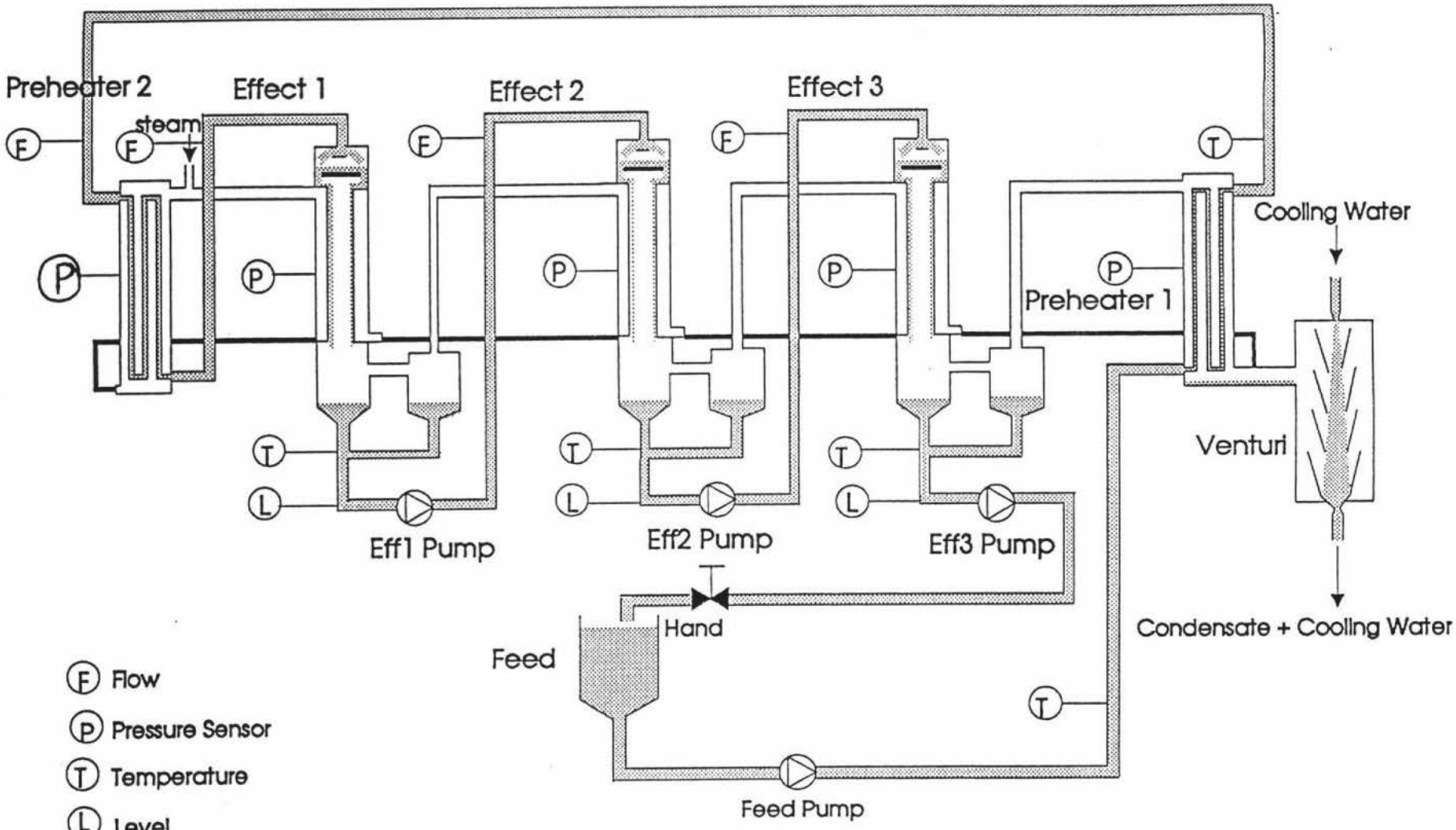
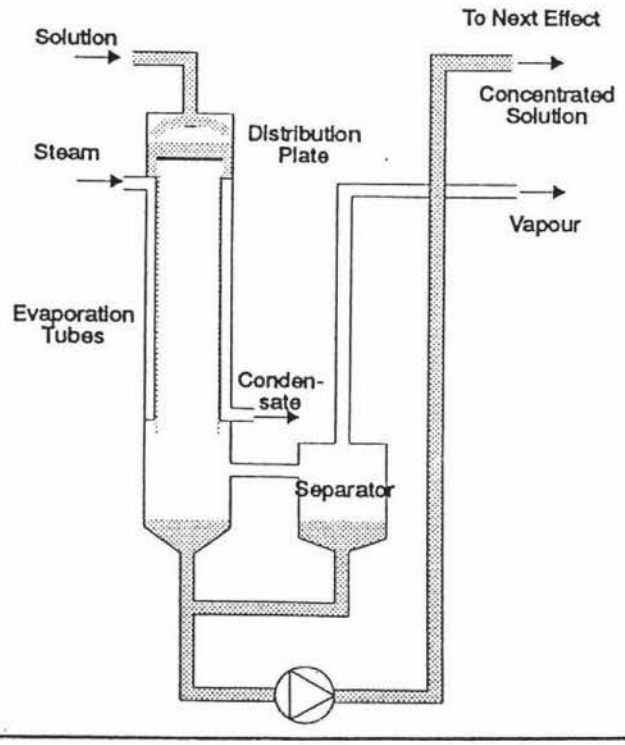


Figure 2.1 Three effect evaporator of straight-through configuration

3. SYSTEM DESCRIPTIONS OF A SINGLE EFFECT OF A FALLING-FILM EVAPORATOR

A single effect of a falling-film evaporator is as shown in the figure 3.1. The liquor to be evaporated, which is usually preheated, is pumped to the top of the effect where it reaches the distribution plate. The distribution plate evenly distributes the liquid amongst the externally heated evaporation tubes. It flows downwards in the form of a film along the inner walls of the tubes and is partially evaporated in the process due to heating of the tubes by condensing water vapour in the jacket. The evaporation rate is a function of the number of tubes, the length of the tubes, the product flow, and the temperature difference between the condensing vapour and the product. Vapour-liquid separation usually takes place at the bottom in the separator tank located at the foot of the evaporator.

Figure 3.1 A single effect of a standard falling film evaporator



3.1 Dynamic Sub-System Descriptions

For modeling purposes, each effect of the falling film evaporator is divided into four sub-systems,

- distribution plate sub-system
- evaporation tube sub-system
- product transport sub-system
- energy balance sub-system

Each of these sub-systems is described in the following sections. A table of symbols is available on page 48.

3.1.1 Distribution Plate Sub-System

The distribution plate is considered as an ideally mixed tank with perforations which allow even distribution of the liquid into the externally heated evaporation tubes. Using Bernoulli's equation and performing an energy balance between points 1 and 2 in the figure 3.2 the volumetric flow out of the distribution plate, Qd can de obtained as:

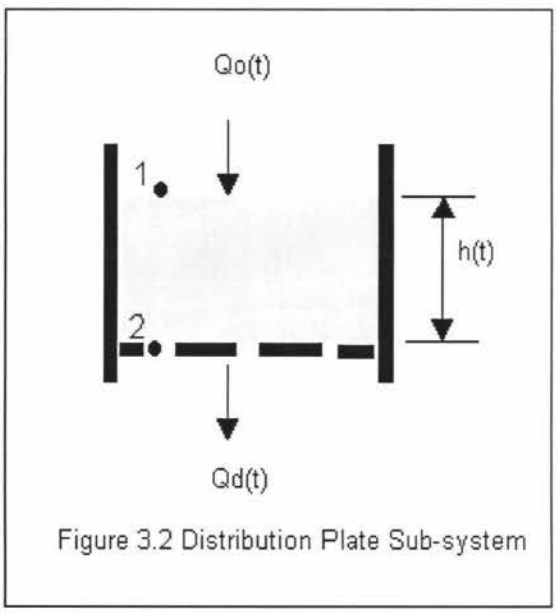


Figure 3.2 Distribution Plate Sub-system

$$p1 - p2 + \rho g(h1 - h2) + \frac{\rho}{2}(v_1^2 - v_2^2) = 0$$

$$0 + \rho g h(t) + \frac{\rho}{2}\left(0 - \mu\left(\frac{Q_d(t)^2}{A_{hl}}\right)\right) = 0$$

$$Q_d(t) = A_{hl}\sqrt{2gh(t)/xi} \qquad (3.1)$$

The volumetric flow depends on the height of the liquid above the distribution plate, h which is a state variable, area of the holes, A_{hl} and friction factor x_i which is dependent on the size of holes in the distribution plate.
By performing volumetric flow balance between points 1 and 2 the height of the liquid above the distribution plate is obtained in the form of differential equation as:

$$\frac{dh(t)}{dt} = \frac{Q_0(t) - Q_d(t)}{A_d} \qquad (3.2)$$

3.1.2 Evaporation Tube Sub-System

Figure 3.3 shows a generalised diagram of the evaporator tube section.

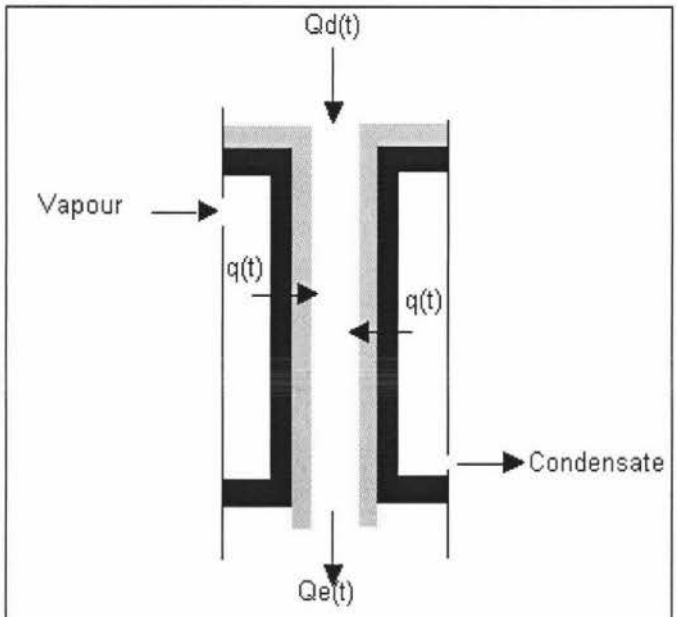


Figure 3.3 Evaporator tube Sub-system

To describe this sub-system completely two assumptions are made:

- The heat supply across the tube walls is considered to be uniformly distributed.
- The fall velocity of the product within the tubes is assumed to be constant. This implies that the residence time, te, is not variable.

By performing a mass balance over the evaporator tube subsystem the mass flow of the product out of the subsystem can be obtained. It is equal to the mass flow in at time t-te minus the amount of liquor evaporated in the evaporation tubes during the time te and given by equation 3.3.

$$\rho_e(t)Q_e(t) = \rho_d(t-t_e)Q_d(t-t_e) - \frac{1}{te}\int_{t-te}^{t}\frac{q_1(\lambda)}{r_1(\lambda)}d\lambda \qquad (3.3)$$

3.1.3 Product Transport Sub-System

Figure 3.4 shows the flow diagram for product transport sub-system.

State Equation For The Product Level

The height of the product in the separator tank i.e., on the suction side of the pump, L(t) is a state variable and is given by the differential equation:

$$\frac{dL(t)}{dt} = \frac{Q_e - Q_f}{A_b} \qquad (3.4)$$

Where Q_e is the flow coming out of the evaporation tube, Q_f is the volumetric outflow going to the next effect and A_b is the surface area of the level.

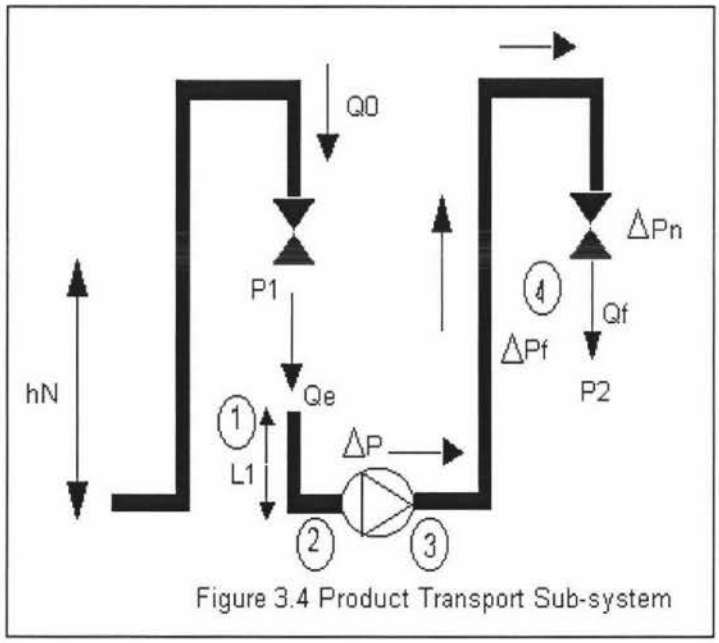


Figure 3.4 Product Transport Sub-system

Sub-System Energy Balance Using Bernoulli's Equation

The algebraic equation relating the outflow, Q_f , and pump speed, N, can be obtained by performing an energy balance (using Bernoulli's equation) between the input point which is level on the suction side and the discharge point (the next effect).

Referring to Figure the balances are as follows:

1 to 2: $$P1(t) + \frac{\rho_1 v_1^2}{2} + \rho_1 g L(t) = p_2 + \frac{\rho_2 v_2^2}{2}$$

2 to 3: $$p_2 + \frac{\rho_2 v_2^2}{2} + \Delta P = p_3 + \frac{\rho_3 v_3^2}{2}$$

3 to 4: $$p_3 + \frac{\rho_3 v_3^2}{2} = \Delta P_f(t) + \Delta P_n(t) + P2(t) + \rho_4 g h_n + \frac{\rho_4 v_4^2}{2}$$

Overall balance 1 to 4:

$$P1(t) + \frac{\rho_1 v_1^2}{2} + \rho_1 g L(t) + \Delta P = \Delta P_f(t) + \Delta P_n(t) + P2(t) + \rho_4 g h_n + \frac{\rho_4 v_4^2}{2} \qquad (3.5)$$

P1 is the equipment pressure of the current effect and P2 is the discharge pressure for the flow Q_f. p_i, v_i, ρ_i and are the local pressures, velocities and densities which are functions of time. Velocities are calculated from equation:

$$Q_x = A_x v$$

Frictional Losses

ΔP_f is the friction loss in the discharge pipe and is given by the formula:

$$\Delta P_f = 4f \frac{L_e}{d} \frac{v^2}{2} \rho 1$$

Transforming the liquid velocity into volumetric flow, the above equation is given by:

$$\Delta P_f = \frac{32 \rho 1 f L_e Q_f^2}{\pi^2 d^5} \qquad (3.6)$$

The friction loss through discharge nozzle is modelled using the standard flow equation $Q = k\sqrt{\Delta P}$ and is given by:

$$\Delta P_n = \frac{Q_f{}^2}{c_n^2} \qquad (3.7)$$

Where c_n is the nozzle characteristic.

3.1.4 Energy Sub-System

The Energy sub-system is concerned with the heat gains and losses by the effect, including vapour transmission to the next effect. Figure 3.5 describes the boundaries used to establish heat gained and heat lost in the first effect of an evaporator system.

Heat transfer coefficients are the overall coefficients from one stream to other, whether the streams are liquid, vapour, evaporating, condensing or ambient air.

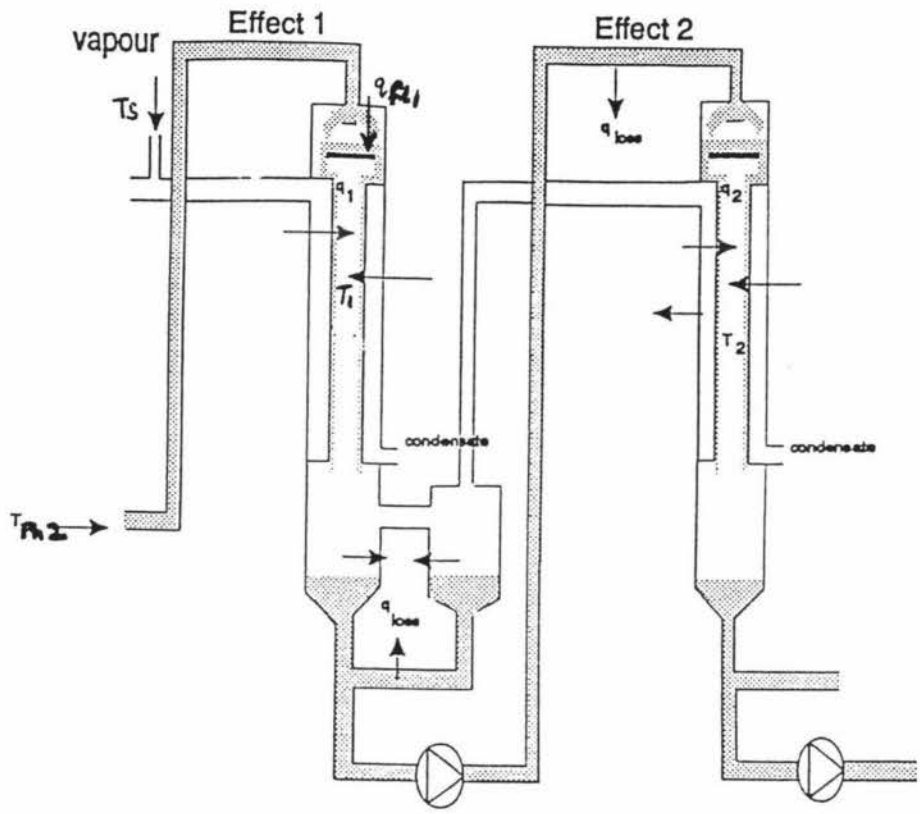
For the purposes of a mass balance, the mass of the product within the effect is assumed to be constant. In reality mass will vary in accordance with level, as the bottom of the separator tank is the place where a significant amount of liquid can be stored. One more assumption is that the heat capacity of the product is also assumed to be constant.
The temperature of the effect, T_1, is a state variable that is assumed to be constant throughout the effect at any given time.

The four energy streams that affect the product temperature, T_1, within the effect are as shown in the figure 3.5. These are combined together to give the energy balance for the first effect of an evaporator. A differential equation is obtained as :

$$\frac{dT_1}{dt} = \frac{q_1(t) + q_{fl1}(t) - q_2(t) - q_{loss}(t)}{Mc_p}$$

Where q_1 is the energy supply for evaporation, and q_2 is the energy supplied to the next effect in watts. They are given by:

Figure 3.5 Energy Sub-System



$$q_1(t) = U1 * A_{t1} * (T_S(t) - T_1(t)) \quad (3.8)$$

$$q_2(t) = U2 * A_{t2} * (T_1(t) - T_2(t)) \quad (3.9)$$

Where U1 and U2 are the heat transfer coefficients, A_{t1} and A_{t2} are heat transfer surface areas of the tubes in the first and second effects.
q_{fl1} is the heat flow into effect due to product feed. The product feed is preheated by preheater two to a temperature of 70°C. qfl1 is given by equation:

$$q_{fl1} = Qdd1 * rhoph2d * Cp * (T_{ph2}(t) - T1(t)) \quad (3.10)$$

Where Qdd1 is the flow out of distribution plate delayed by one residence time and rhoph2d is density of liquor in preheater two delayed by one residence time.
The heat lost to the surroundings is given by:

$$q_{loss}(t) = Uloss * A_S * (T_1 - T_o) \quad (3.11)$$

Where Uloss is the heat transfer coefficient to the surroundings and is governed by natural convection process. A_s is the total surface area of the effect and T_o is the

M is the total mass of the product in the effect and Cp is the specific heat capacity of product in the effects and preheaters.

For further analysis temperature at the base of the effect T_{1b} is considered as a state variable given by the differential equation:

$$\frac{dT1b(t)}{dt} = \frac{(q_{in} - q_{lossb})}{Mb/Cp} \qquad (3.12)$$

Where q_{in} is the energy at the base of the effect and q_{lossb} is the heat lost to the surroundings at the base of the effect given by the following equations:

$$q_{in} = Qe * \rho 1 * Cp * (T_1 - T_{1b}) \qquad (3.13)$$

$$q_{lossb} = Ulossb * Abs * (T_{1b} - Ta) \qquad (3.14)$$

Mb, the mass at the base of the effect, is given by:

$$M_b = V_b * \rho 1b \qquad (3.15)$$

3.2 Non-linear State Space Description

3.2.1 Introduction

The overall energy balance equation (3.5) in the product transport sub-system requires the calculation of velocity head terms. In the equation a velocity term appeared on both sides and hence only the difference between them needed to be calculated. The velocity head terms were found to be an order of magnitude smaller than the other terms, and the differences between them even smaller, given the standard operating flows and the geometry of the evaporator. Hence velocity head terms were neglected in the product transport sub-system.

The density of the fluid in the effect is only a function of temperature and hence time and assumed to be constant throughout the effect.

Therefore the equation 3.5 can be re-written as :

$$P1(t) + \rho 1gL(t) + \Delta P = \Delta P_f(t) + \Delta P_n(t) + P2(t) + \rho 1gh_n \qquad (3.16)$$

3.2.2 Variable Classification

The following table gives the classification of all variables:

State variables	Disturbance variables	Controllable input variables	Output variables

h L T_1 T_{1b} Mv1	Ta T_2	Q_0 Ts,T_2 T_{ph2} N	Qd, Qe, Qf, q1,q2,qfl1,qloss h, L, T_1, T_{1b},Mv1

3.2.3 Development of The State Equations

As there are four state variables, the state space description contains four differential equations. These were developed from equations (3.2), (3.4), (3.7) and (3.12) which can be expressed as follows:

$$\dot{h} = f(Q_o, Q_d)$$

$$\dot{L} = f(Q_e, Q_f, A_b)$$

$$\dot{T}_1 = f(q_1, q_{fl1}, q_2, q_{loss}, M)$$

$$\dot{T}_{1b} = f(q_{in}, q_{lossb}, M_b)$$

The terms A_b, M and M_b have been included as they are functions of evaporator geometry and level L.
To obtain a state space description, all the bracketed variables in the above equations were found to be explicitly in terms of state and input variables. The following expressions were obtained:

$$\dot{h} = f(Q_o, h) \qquad (3.17)$$

$$\dot{L} = f(Q_e, T_1, T_2, N, L) \qquad (3.18)$$

$$\dot{T}_1 = f(T_1, T_a, T_2, T_s) \qquad (3.19)$$

$$\dot{T}_{1b} = f(T_1, T_{1b}, Ta) \qquad (3.20)$$

It can be seen that Q_e need to be eliminated to complete state space description.

3.2.4 Solution Strategy

1. Find the general form of pump characteristic curve.
2. Eliminate all velocity head terms from equation (3.5)
3. Solve equation (3.5) for the output flow rate Qf.

3.2.5 Evaluation Of The Output flow

After removing velocity head terms from equation (3.5) substituting various relevant terms the output flow obtained was as follows:

$$Q_f = \sqrt{\frac{P1(T_1) + \rho 1 g L + \Delta P - \rho 1 g h_n - P2(T_2)}{\left(\frac{32\rho 1 f L_e}{\pi^2 d^5} + \frac{1}{c_n^2}\right)}} \tag{3.21}$$

The pump characteristic equation used in the model was:

$$\Delta P = a_p N^2 \tag{3.22}$$

3.2.6 Implementation of Liquid Evaporation Equation

The mass flow of the product out of the evaporation tube subsystem Qe is given by equation (3.4) as:

$$\rho_e(t) Q_e(t) = \rho_d(t - t_e) Q_d(t - t_e) - \frac{1}{te} \int_{t-te}^{t} \frac{q_1(\lambda)}{r_1(\lambda)} d\lambda$$

The above equation can be split into two parts as:

$$\rho_e(t) Q_e(t) = \rho_d(t - t_e) Q_d(t - t_e) - \frac{1}{t_e}\left(\int_0^t \frac{q_1(\lambda)}{r_1(\lambda)} d\lambda - \int_0^{t-t_e} \frac{q_1(\lambda)}{r_1(\lambda)} d\lambda\right)$$

The two integrals each represent the total mass of vapour generated by evaporation over the integral limits. If the value of q(t) is assumed to be zero for t<0, the two integrals can be recombined as:

$$\rho_e(t) Q_e(t) = \rho_d(t - t_e) Q_d(t - t_e) - \frac{1}{t_e}\left(\int_0^t \frac{q_1(\lambda)}{r_1(\lambda)} d\lambda - \frac{q_1(\lambda - t_e)}{r_1(\lambda - t_e)} d\lambda\right)$$

The combined integral is the total mass of vapour generated in the tube over the previous te seconds. This is the fourth state variable in the model Mv1:

$$Mv1 = \frac{1}{t_e}\left(\int_0^t \frac{q_1(\lambda)}{r_1(\lambda)} d\lambda - \frac{q_1(\lambda - t_e)}{r_1(\lambda - t_e)} d\lambda\right)$$

$$\dot{Mv1} = \frac{q_1(t)}{r_1(t)} - \frac{q_1(t - t_e)}{r_1(t - t_e)}$$

$$\dot{M}v1 = \frac{q_1(t)}{r_1(t)} - \frac{q_1 d}{r_1 d} \qquad (3.23)$$

q1d and r1d denote heat flow across tube walls and latent heat of vaporisation delayed by one tube residence time respectively.

Hence the expression defining the flow leaving the evaporation tubes is:

$$Q_e(t) = \frac{\rho_d(t - t_e)}{\rho_e(t)} Q_d(t - t_e) - \frac{1}{t_e \rho_e(t)} Mv1$$

Since ρd corresponds to the density of liquor before entering distribution plate it can be replaced by ρph2d which is the density of the product feed. Also ρe is the density in the effect and can be replaced by ρ1. Hence the above equation can be written as:

$$Q_e(t) = (\rho ph2d * Qdd1 - 1/te * Mv1) / \rho 1 \qquad (3.24)$$

r1 and density are functions of temperature and found from thermodynamic water tables prepared by Cooper and Le Fevre (1969).

From Equation 3.24 and 3.23 it follows that:

$$Q_e = f(h, Mv1, T_1) \qquad (3.25)$$

$$\dot{M}v1 = f(T_1, T_s) \qquad (3.26)$$

By substituting equations 3.25 and 3.26 in the original state equations (3.17 to 3.20), a complete five-state description of the evaporator is obtained.

$$\dot{h} = f(Q_o, h) \qquad (3.27)$$

$$\dot{L} = f(h, Mv1, T_1, T_2, N, L) \qquad (3.28)$$

$$\dot{T}_1 = f(T_{ph2}, T_1, Ta, T_2, T_s) \qquad (3.29)$$

$$\dot{T}_{1b} = f(T_1, T_{1b}, Ta) \qquad (3.30)$$

$$\dot{M}v1 = f(T_1, T_s) \qquad (3.31)$$

3.2.7 Feed Flow Rate Calculation

The feed flow rate Q_0 is function of feed tank level, feed pump speed, atmospheric pressure and pressure in the first effect. It is calculated using the following equation:

$$Q_0 = \sqrt{\frac{(Pa - P1 - Ppump + P\tan k - Ptop)}{1/Cvn^2 + Pfric}} \qquad (3.32)$$

$P1=989.58-18.605T_1+4.7672*T_1^2-4.852e-2*T_1^3+1.0301e-3*T_1^4$
$Ppump=ap*N0^2$
$Ptank=g*rho1*LT$
$Ptop=g*rho2*hN$

$$Pfric = \frac{32 * rho3 * f}{pi^2 * (Lf1/d1^5 + Lf2/d2^5)} \qquad (3.33)$$

Table 3.1 Nomenclature of Symbols Used

A_b	Area of the liquid surface in the bottom of the effect	m^2
A_d	Area of the liquid surface on the distribution plate	m^2
A_{hl}	Area of the holes in the distribution plate	m^2
A_s	Surface area of the effect	m^2
A_t	Heat transfer area of the evaporation tubes	m^2
ap	Pump flow characteristic constant	Pa/rp m^2
c_n	Flow characteristic constant for the discharge nozzle	$m^3/(Pa^{1/2}s)$
c_p	Heat capacity of the effect	$J/kg/K$
C_v	Valve flow characteristic	$m^3/Pa^{1/2}s$
d	Diameter of the pipe for section 3 to 4 of the prod trans Subsystem	m
f	Dimension less friction factor for fluid flow	-
g	Acceleration due to gravity	m/s^2
h	Height of the liquid on the distribution plate	m
hN	Height of the discharge nozzle	m
k	Geometric constant used in calculations of Ab	-
L	Height of the liquid level in the separator tank	m
L_e	Equivalent length of a pipe-line	m
Lf1,2	Equivalent pipe length for feed system	m
LT	Liquid level in the feed tank	m
M	Total mass of product in the effect	kg
N1	Pump speed	rpm
N0	Feed pump speed	rpm
P1	Pressure in effect 1 from steam tables	Pa
P2	Pressure in effect2 from steam tables	Pa
Pa	Ambient pressure	Pa
Q_d	Liquid flow through distribution plate	m^3/s
Q	Liquid flow out of evaporation tube	m^3/s
Q_f	Volumetric flow leaving the effect	m^3/s

Qdd1	Flow through distribution plate delayed by one residence time	m^3/s
Q_0	Volumetric flow fed to the effect	m^3/s
q1,2,3	Heat flow for evaporation across tube walls in the effect	W
q_{loss} 1,2, 3	Heat loss from the effect to the surroundings through outer walls	W
qfl1,2,3	Heat flow into effect due to product feed	W
r1	Latent heat of vaporisation for water	J/kg
r1d	Latent heat of vaporisation for water delayed by one residence time	J/kg
rho1,2,3, (ρ 1,2,3)	Density of liquid water calculated from steam tables	kg/m^3
rhoph1,2 (ρ 1,2)	Density of liquid water calculated from steam tables	kg/m^3
rho1,2,3 ph1,2d (ρph1,2d)	Density delayed by one residence time	kg/m^3
te	Residence time	second s
$T_{1,2,3}$	Temperature of the vapour/liquid system in the effects	$^\circ C$
T1d	Temperature in the effect 1 delayed by one residence time	$^\circ C$
T1b	Temperature at the bottom of the effect	$^\circ C$
T1bd	Temperature at the bottom of the effect delayed by one residence time	$^\circ C$
T_{ph1}, T_{ph2}	Temperature of the product feed leaving preheaters	$^\circ C$
T_{ph1d}, T_{ph2d}	Temperature of the product feed leaving preheaters delayed by one residence time	$^\circ C$
Ta	Ambient temperature	$^\circ C$
Ts	Temperature of steam supply to the shell of Eff1 and ph2	$^\circ C$
v_1, v_2	Velocity of fluid at sections 1 and 2 in the prod trans Subsystem	m/s
v_3, v_4	Velocity of fluid at sections 3 and 4 in the prod trans Subsystem	m/s
U	Overall heat transfer coefficient for evaporation tubes	$W/(m^2K)$
Uloss	Overall heat transfer coefficient for heat loss to the surroundings	$W/(m^2K)$
Vb	Volume of the product collected at the base of the effect	m^3
ΔP	Pressure increase through the pump	Pa

(dPf) ΔP_f	Friction loss through section 3 to 4 of discharge pipe	Pa
ΔP_n	Friction loss through discharge nozzle	Pa
xi	Dimensionless friction factor for flow through the distribution plate	-

3.3 Parameter Identification (Mark Illingworth 93)

Following table lists all the constants that are required in the application of the model, their values for each of the three effects, and the means of identification or calculation.

Symbol	Unit	Effect1	Effect2	Effect3	ID method
A_{hl}	m^2	2.36×10^{-4}			Evaporator Geometry
A_d	m^2	$3.88 \times 10^{-3} : h \le 0.0026$ $3.88 \times 10^{-3} + 0.06\left(\sqrt{0.0235^2} - \sqrt{(0.0495 - h)^2}\right)$ $: 0.026 \le h \le 0.0073$			Evaporator Geometry
A_b	m^2	$1.795 \times 10^{-3} : 0 \le L \le 1.02$ $2.168 \times 10^{-3} : 1.02 \le L \le 1.06$ $\frac{\pi}{4}(0.0218^2 + (0.5(L - 1.06) + 0.0478)^2)$ $: 1.06 \le L \le 1.11$ $4.536 \times 10^{-3} : 1.11 \le L \le 1.39$ $\frac{\pi}{4}(0.0728^2 + (3.04(L - 1.06) + 0.0218)^2)$ $: 1.39 \le L \le 1.415$			Evaporator Geometry
A_t	m^2	0.3313	0.3313	0.2137	Evaporator Geometry
A_s	m^2	1.917	1.917	1.610	Evaporator Geometry
U	$(W / m^2 K)$	3850	3750	3500	Heat Transfer Formulae
Ulos s	$(W / m^2 K)$	4.6	4.2	3.6	Heat Transfer Formulae
f	-	0.006			Estimated typical steel properties
g	m / s^2	9.81			Gravitational Acceleration
hN	m	4.82			Evaporator Geometry
M	kg	$\rho\left(\int_0^t A_b dl + 0.002395\right)$			Evaporator Geometry

Cp	J/kg	4230	4210	4200	Est.Thermodynamic properties of water
Le	m	11			Evaporator Geometry
Lf1	m	21.5			Evaporator Geometry
Lf2	m	20.5			Evaporator Geometry
d (d1)	m	0.0218			Evaporator Geometry
d2	m	0.0095			Evaporator Geometry
xi	-	1			zero friction assumed
te	seconds	4			Experiment
a_p	Pa/rpm^2	0.02267	0.02267	0.02266	Experiment
Cvn 1	$m^3/Pa^{1/2}s$	$6.65*10^{-7}$	$6.65*10^{-7}$	$6.65*10^{-7}$	Experiment

4 DEVELOPMENT ENVIRONMENT: MATLAB/SIMULINK AND REAL-TIME WORKSHOP

4.1 Matlab

Matlab uses a command line interface and a command interpreter. All the variables created from a numerical calculation or from a simulation are stored in the Matlab workspace. One can view them by typing 'who' at the Matlab prompt. The command 'help' with no arguments displays a list of directories that contain Matlab related files. Terminating a Matlab session deletes all the existing variables from the workspace. Before quitting, the workspace can be saved for later use. The next time Matlab is invoked the 'load' command restores the workspace from Matlab.mat. One can use 'save' and 'load' with other filenames or to save only selected variables.

Matlab applications are comprised of: M files, Mat files and MEX files.

M-files: Text files that contain Matlab statements are called M-files because they have a file extension of .m. Two types of M- files can be used: script files and function files. Script files automate long sequences of commands. When a script file is invoked Matlab simply executes the commands in the file via its command interpreter. The statements in a script file operate globally on the data in the workspace. Function files allow the user to add new functions to the existing library. These files contain the word 'function' at the beginning of the first line. A function differs from a script in that arguments may be passed, values may be returned and variables defined and manipulated inside the file are local to the function and do not operate globally on the workspace. Both scripts and functions are ordinary ASCII text files. M files are machine independent.

Mat files: These files contain binary data. This is the file format that Matlab uses for saving data to disk. Mat files offer a simple and convenient mechanism for transporting data between platforms.

MEX Files: These files are compiled code resources used to speed up functions eliminating the requirement that Matlab interpret them.

4.2 Simulink

Simulink adds a dynamic simulation layer to Matlab that features a graphical programming language while retaining all of Matlab's general functionality. Simulink is a graphical mouse-driven environment that allows the user to model systems via block diagrams. It can handle linear, non-linear, continuous time, discrete time and multivariable systems. Simulink models can incorporate blocks containing Matlab commands or functions and Matlab's complete set of numerical and visualisation tools that can be used to analyse and display the results of Simulink simulation.

Simulink has two phases of use; model definition and model analysis. To facilitate model definition, Simulink adds a new class of windows called block diagram windows. After a model is defined it can be analysed by choosing options from the Simulink menus or by entering commands in Matlab's command window. The progress of simulation can be viewed while the simulation is running and the final results can be made available in the Matlab workspace when a simulation is complete.

Simulink can be opened from Matlab command window by typing 'Simulink'. This command displays a new window containing icons for the Subsystem blocks that make up the standard library. These Subsystems blocks can be opened by double clicking to produce the windows containing the blocks. These blocks can be copied into the models. Blocks can also be copied from one window to another by dragging them from their original location to the new window. These copied blocks contain the same internal parameters as the original block. To change the parameters of a block double click on its icon that displays a dialogue box that contains respective parameter list. The angle bracket (>) pointing out of block's icon represents its output port(s) and the angle bracket pointing into the icon represents its input port(s). To connect these two blocks, use the left mouse button to click either the output or input port of one block, drag to the other block's input or output port to draw a connection line, and then release the left mouse button.

Once the model is built simulation can be started by choosing the simulation parameters from the simulation menu. Selecting the parameters command from 'Simulation' menu displays a dialogue box in which the integration algorithm and other simulation parameters (such as start and stop times and minimum and maximum step size) to be used in the current simulation can be specified.

The simulation algorithms available are

- linsim Liner Simulation method
- rk23 Runge-Kutta third order method
- rk45 Runge-Kutta fifth order method
- adams Adam's method
- gear Gear's method
- euler Euler's method

linsim

linsim is used for linear models. Linear models are composed of Transfer Function, State-Space, Zero-Pole-Gain, Sum and Gain blocks.

If the system is primarily linear but contains a few non-linear blocks *linsim* also works well. *linsim* is particularly good compared to other methods when the linear blocks have both fast and slow dynamics.

rk45, rk23

Runge-Kutta methods are used when the system is highly non-linear and/or discontinuous. These methods do not work well when the system has both fast

and slow dynamics. The methods perform well for mixed continuous and discrete time systems.

gear
The *gear* algorithm is used when the system is smooth and non-linear.

adams
This method is used for systems which are smooth and nonlinear but do not have widely varying time constants.

Minimum Step Size
The minimum step size is the step size that is used at the start of the simulation. The integrators do not step below the minimum step size when generating the output points, unless the system contains discrete blocks with sampling periods smaller than the minimum step size.

Maximum Step Size
By limiting the maximum step size a smooth plot results.

Fixed Step Size
Variable step methods like *rk45* and *rk23* can be converted into fixed step methods by setting minimum step size equal to the maximum step size. The *adams* and *gear* methods take an undetermined number of points between outputs and therefore, cannot be converted to fixed step methods.

System Functions or S-functions:
S-functions are Matlab functions with special calling syntax that allow access to a model's dynamic equations. In ordinary graphical interaction with Simulink S-functions remain transparent to the user. S-functions are created by Simulink using the information from Simulink block diagram. Every block diagram has a system function with the same name as the model name. Simulink interacts with this S-function for simulation and analysis. They are the internal representation of the model and used by the integration routines. One can side step this process by defining an S-function in standard M-file or MEX file written in C. Such files must have a standard format as the basic idea is to provide Simulink with a clear definition of the variables that are simulated, how the time rates of change of those variables are determined by the current values of the variables and system inputs, and how the output of the System, if any, is determined.

S functions can be represented in three types:

Graphical ---- Block diagram
M-file ---- Matlab language
MEX file ---- C language subroutines

4.3 Real-Time Workshop And Device Drivers

The Simulink Real-Time Workshop (RTW) is an automatic C language code generation environment for Simulink. It produces C code directly from Simulink graphical models then compiles and links this code with other application modules and user written modules to produce a stand alone program using one of the available compilers. The compiler is called from Simulink so that the entire sequence can be run from within Simulink.

The RTW application modules are designed to perform the following:

- Read data from the external hardware to obtain inputs to the model.
- Calculate the outputs of the model using input data.
- Write the model outputs to the external hardware.

The RTW can produce real-time or nonreal-time applications.

Device Drivers

The RTW device driver software handles communication between the model and the real world. Device drivers contain necessary code for interfacing to specific applications. They are implemented as Simulink S-functions written in C. The C code for a device driver block is compiled as a MEX file so that it can be called by Simulink. These drivers can be added to the Simulink model like any other block. The code generator compiles and links the block's source along with the code generated for the model and the source files that are used to build a real-time application. Each device driver block has a dialogue box where configuration parameters are set. One can write his/her own C code S-function that implements device drivers to suit a particular application. The device driver MEX-files must be in the directory on the Matlab path so that the code generator can perform argument checking before generating the code for the model.

The drivers are converted into C source code using the RTW. The C source code is built into a MEX file using the CMEX batch file (used to build C language MEX-files), while the same code is compiled and linked with the generated code in the stand-alone program.

Before building MEX-files the CMEX batch file is edited to match the system configuration and to specify the locations of the compiler and linker. It can be opened in a text editor. It is often convenient to create MEX-files from within Matlab by using the shell escape feature to execute CMEX from the Matlab prompt. For example, one can simply type

```
!cmex tofile.c
```

to create the MEX-file version of *tofile* device driver from within the Matlab environment.

Figure 4.1 illustrates the functionality of device drivers.

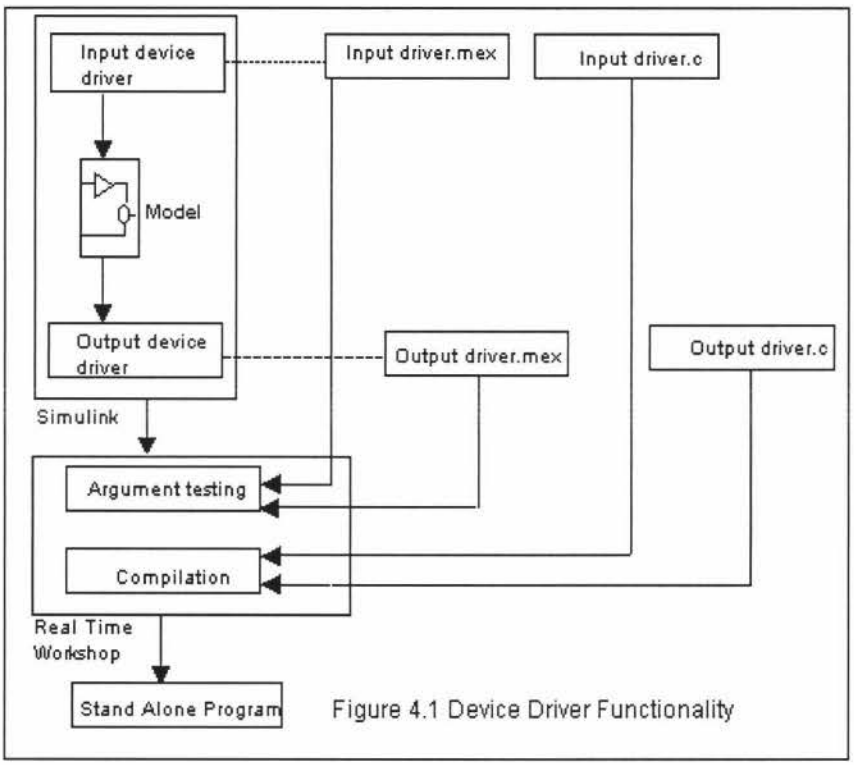


Figure 4.1 Device Driver Functionality

4.4 Limitations of Real-Time Workshop

- All Simulink blocks are automatically converted into C-code, with the exception of Matlab function blocks and S-function blocks that invoke M-files. S-function blocks written in C can be converted as long as they stick to Simulink 1.3 S-function format.

- The Real-Time Workshop does not produce code that solves algebraic loops.

- The Real-Time Workshop does not convert blocks that depend on absolute time.

- All blocks must run at a multiple of the fixed step size including the device drivers.

4.5 Description of The Major RTW C Files

Application Modules

Building a real-time application requires a number of source files in addition to the generated code, model.c. These source files are:

A main program	nrt/rt	main.c
Code to drive execution of the model code	nrt/rt	sim.c
Code to implement an integration algorithm	nrt/rt	

(any one of the algorithms)		euler.c, gear.c
rk45.c, rk23.c		
Code to carry out data logging	nrt/rt	log.c
Code to create Simstruct data structure	simstruc.c,	simstruc.h

Template Files

The template files contain the macros that define information such as; the path to include files, compiler options, source files, etc. that are needed to build a program, real-time or nonreal-time. The RTW automatic program builder uses a Matlab M-file, (make-rt.m or make-nrt.m) to process a template file into a Makefile. This processing incorporates the parameter settings from the real-time/nonreal-time options dialogue box into a Makefile. The make utility uses rules and dependencies defined in a Makefile to determine which code modules to compile and link.

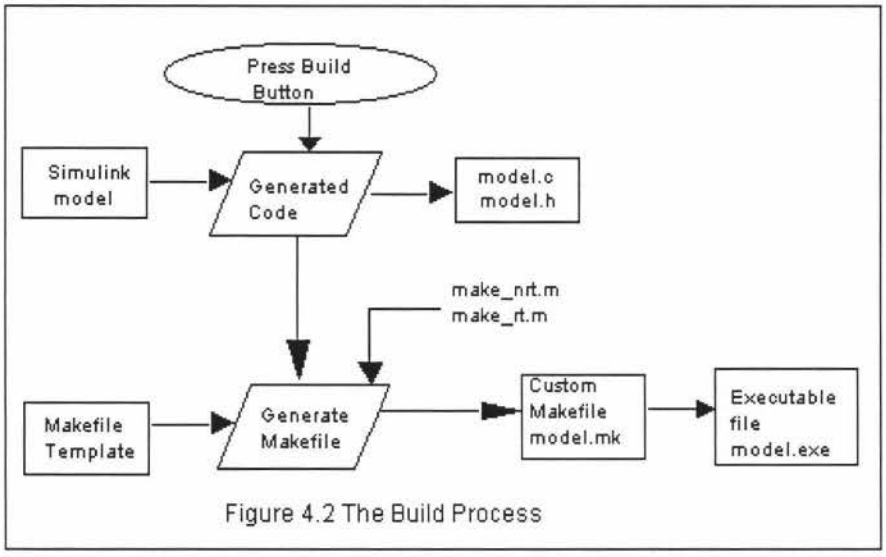


Figure 4.2 The Build Process

In general templates contain the following categories:

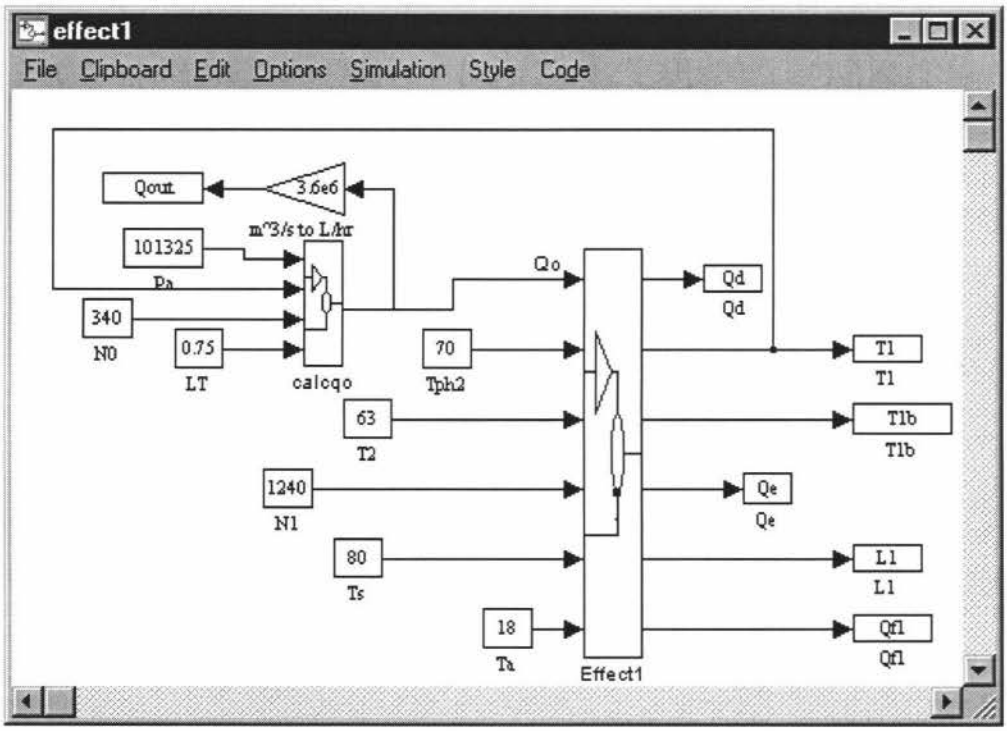
- Macros read by make-rt /make-nrt --- Basic build information.
- Customisation macros --- Macros specified by make-rt /make-nrt according to values obtained from the Simulink model and RTW dialogue boxes.
- Tool locations --- The path to compiler and linker.
- Tool definitions and configurations--- Specific names of build tools.
- Include path --- The path to include files.
- C flags --- Compiler options.
- Source files --- The source files used to build the program.
- Rules --- Statements used by Make to determine when the program needs to rebuilt.
- Dependencies --- Statements used by Make to determine which object files depend on values in the generated code and Makefile.

5 BUILDING AN EVAPORATOR MODEL

The 'FixSim' software was tested by the first effect of Production Technology evaporator model to run under the FIX.

The model named *effect1* resides in c:\matlab. It is run by typing *effect1* at the Matlab prompt or can be opened from Simulink. (To open from Simulink first invoke Simulink by entering *simulink* at the Matlab prompt. Then choose 'open' from *File* on the menu bar and select *effect1* from c:\matlab directory.)

The model implemented was the first effect of a falling film evaporator run in straight flow configuration as shown below:



The Real-Time Workshop does not convert Matlab function blocks, and S-function blocks that invoke M-files, to C code. Hence the S-function blocks and Matlab function blocks were avoided totally and instead ordinary function blocks were used to implement functions and Switch blocks were used for conditional if else statements. Transport delay blocks were used for the tube residence time.

The simulation is an implementation of all the state equations and related algebraic equations developed for the first effect including the feed system.

The steady state inputs to the model are:

Tph2 Temperature of the preheater two at 70°C.
T2 Temperature in the second effect which is disturbance input at 63°C.
N1 Speed of the pump at 1240 rpm.
Ta Atmospheric temperature at 18°C.

Ts Steam temperature.
Q0 Feed flow rate is a function of atmospheric pressure Pa, feed pump speed N0, feed level LT and pressure in the first effect P1 with following values.

Pa 101325 Pascals.
N0 340 rpm.
LT 0.75m.

The outputs from the model are:
Qd Volumetric flow out of the distribution plate.
T1 Temperature in the first effect.
T1b Temperature at the base of the effect.
Qe Volumetric flow out of evaporation tube.
L1 Liquid level in the separator tank.
Qf1 Outflow to the next effect.

The feed flow rate Qout is also set as an output to observe the amount of feed that is being fed to the model.

Description of The Model Subsystems

The top level model has two subsystems, *calcqo* which calculates the amount of feed flow to the first effect and *Effect1* which calculates all flows and temperatures. The following sections explain these subsystems in detail by exploding them. In order they are:

- Explosion of subsystem *calcqo*
- Explosion of *Effect1*

Effect1 is divided mainly into four subsystems:

- *Flows and levels*
- *Temperature and heatflows*
- *T1b and heatflows*
- *Mv1*.

Flows and levels calculates all flows, the liquid level on the distribution plate and the liquid level in the separator tank and the required densities.

Temperature and heatflows calculates T1, the temperature in the first effect, T1d, the temperature delayed by one residence time and all heat flows.

T1b and heatflows calculates T1b, the temperature at the base of the effect and the corresponding heat flows.

*Mv1*calculates the evaporation rate.

Explosion of Subsystem *calcqo*:

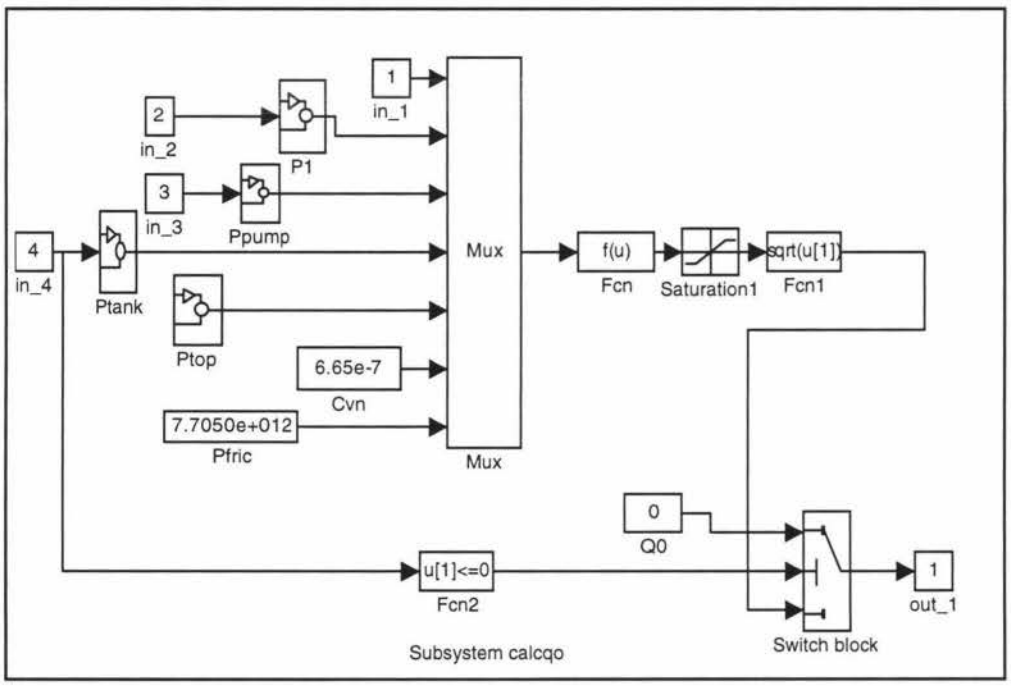
This Subsystem calculates feed flow rate,Q0, to the model.

Q_0 is calculated using the equation:

$$Q_0 = \sqrt{\frac{(Pa - P1 - Ppump + P\tan k - Ptop)}{1/Cvn^2 + Pfric}}$$

Where Pfric is given by

$$Pfric = \frac{32 * \rho 3 * f}{pi^2 * (Lf1/d1^5 + Lf2/d2^5)}$$



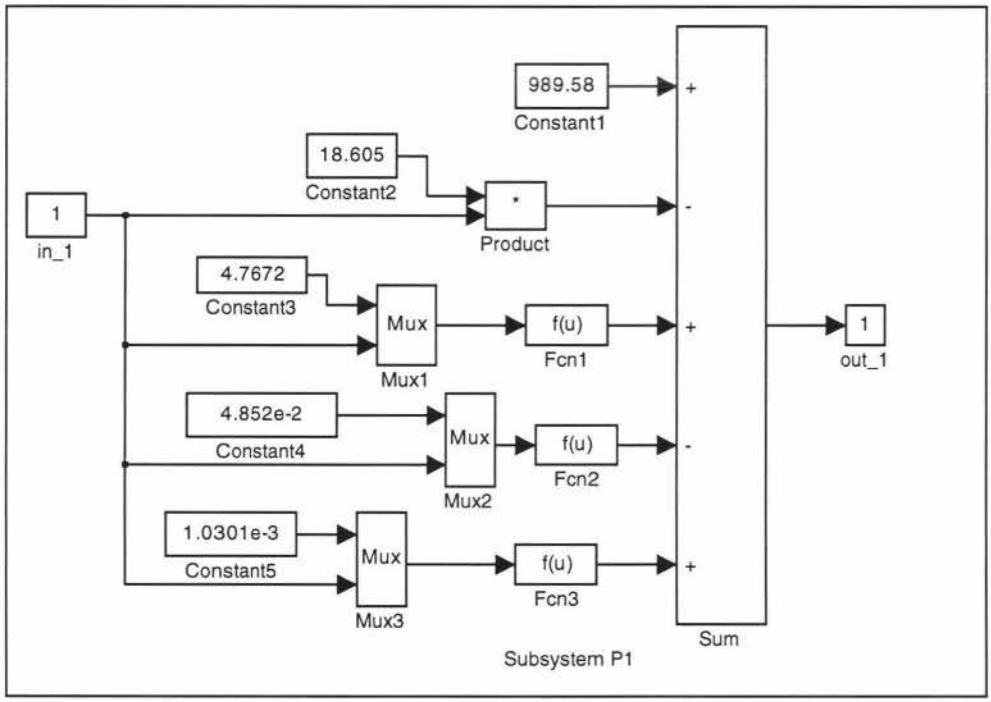
in_1, in_2, in_3 in_4 are input ports. Out_1 is output port.
in_1 refers to Pa, atmospheric temperature.
in_2 refers to T1,temperature in the first effect.
in_3 refers to N0, speed of the feed pump.
in_4 refers to LT, level in the feed tank.

All the inputs to the Mux block are grouped to create a vector of inputs to the function block, Fcn (The width of the vector line from Mux block is same as the number of inputs specified). In the block Fcn u[1] corresponds to the first input, u[2] corresponds to the second input and so on. The Saturation block limits the range of Q0 to be between 0 and 500l/hr. This is done to make sure a negative number is not passed to the square root block following. The function block, Fcn1, calculates the square root of its input and passes the output to the Switch block.

The Switch block will output the calculated Q0 or 0 if the tank is empty (LT<=0).
All the blocks are selected and grouped together to form 'calcqo'.

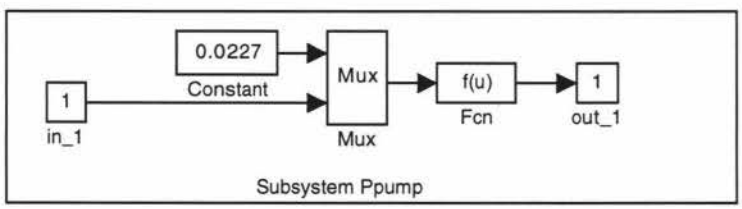
P1 Subsystem P1 calculates pressure in the first effect using the equation: $P1=989.58-18.605T_1+4.7672*T_1^2-4.852e-2*T_1^3+1.0301e-3*T_1^4$.

Referring to subsystem P1,Input port in_1 refers to T1 and the output port is P1.
Constant 18.605 and T1 are inputs to the Product block. The output from the Product block is 18.605*T1. 'Mux1' groups constant 4.7672 and T1 to create a vector of input to function block, Fcn1.
In the block Fcn1 u[1] corresponds to the first input which is 4.7672 and u[2] corresponds to second input T1.



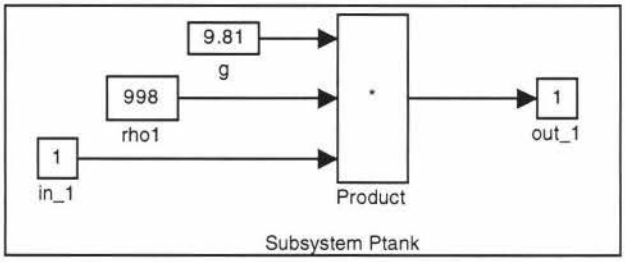
The output from the Fcn1 block is $4.7672*T_1^2$. Similarly the output from theFcn2 block is $4.852e-2*T_1^3$ and output from the Fcn3 block is $1.0301e-3*T_1^4$. The Sum block adds all the inputs according to the signs to produce the required output (no spaces in the Sum block between plus and minus signs). The output from the Sum block is $989.58-18.605T_1+4.7672*T_1^2-4.852e-2*T_1^3+1.0301e-3*T_1^4$.

Ppump



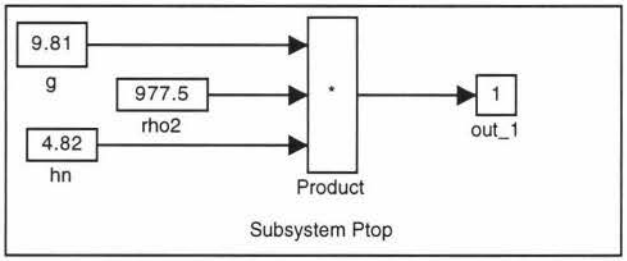
The input port refers to N0 and the output port is Ppump.
Subsystem 'Ppump' calculates pressure in the pump using the formula: $Ppump=ap*N0^2$. The output from the Fcn block is $ap*N0^2$.

Ptank



Subsystem 'Ptank' calculates the pressure in the feed tank using the formula: Ptank=g*rho1*LT. In_1 refers to LT, feed tank level. The output from the Product block is g*rho1*LT.

Ptop



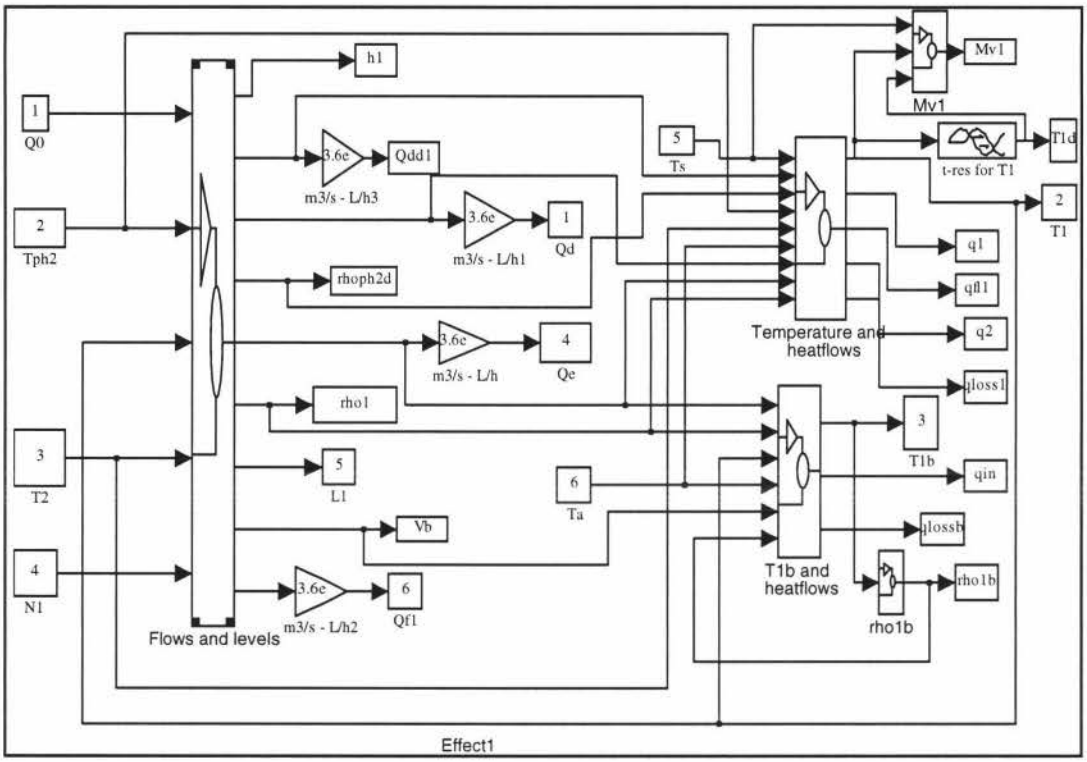
Subsystem 'Ptop' calculates the topup pressure in the feed tank (corresponding to rate at which fresh water is added to the tank) using the formula: Ptop=g*rho2*hN. The output from the Product block is g*rho2*hN.

Cvn is constant block with value of 6.65e-7.
Pfric is frictional losses and calculated using the formula:

$$Pfric = \frac{32 * \rho 3 * f}{pi^2 * (Lf1 / d1^5 + Lf2 / d2^5)}$$

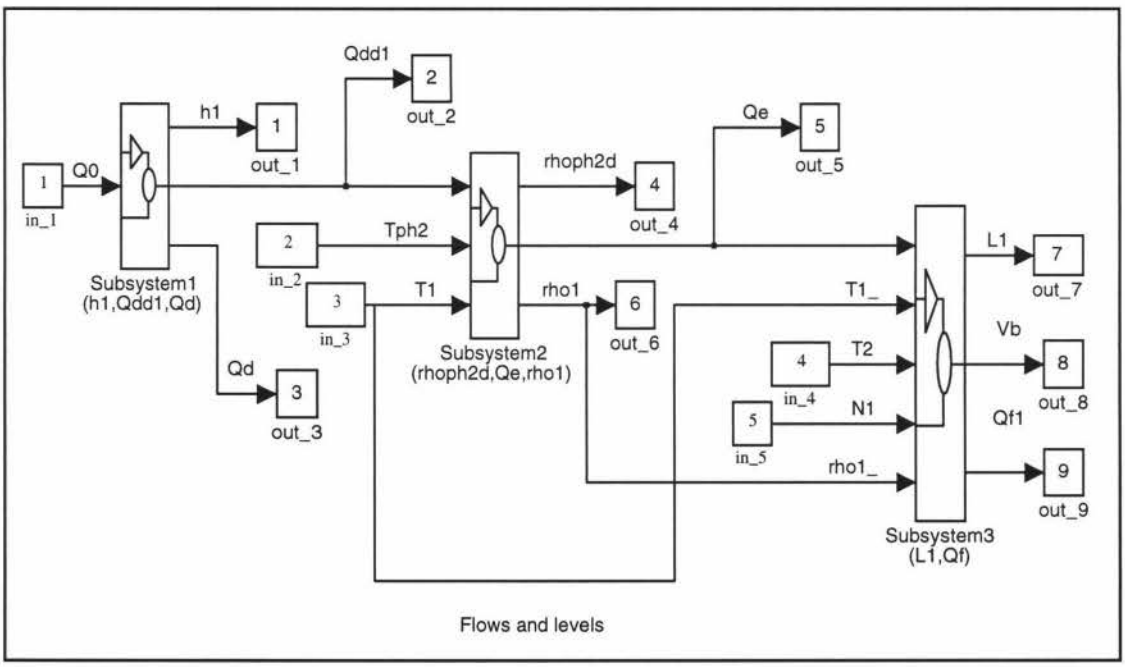
Explosion of 'Effect1'

The Subsystem 'Effect1' is as shown below:



All the Subsystems in Effect1 are explained in detail :in the following sections.

'Flows and levels'



The inputs to this Subsystem in order are:

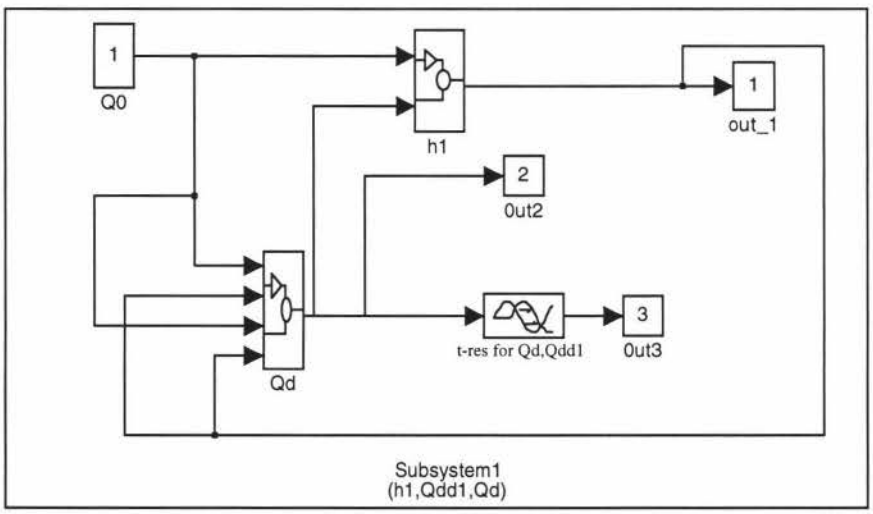
Q0 Feed flow rate.
Tph2 Temperature in the preheater2.
T1 Temperature in the first effect.
T2 Temperature in the second effect.
N1 Speed of the pump.

Outputs from this Subsystem in order are:

h1 Height of the liquid level above the distribution plate.
Qdd1 Volumetric flow out of the distribution plate delayed by one residence time.
Qd Volumetric flow out of the distribution plate.
rhoph2d Density of liquid water in the preheater two delayed by one residence time.
Qe Volumetric flow out of the evaporation tube.
rho1 Density of liquid water in effect one.
L1 Liquid level in the separator tank.
Vb Volume of the product collected at the base of the effect.
Qf1 Volumetric flow into the second effect.

'Flows and levels' is again subdivided into three Subsystems:
Subsystem1, calculates h1,Qdd1 and Qd.
Subsystem2, calculates rhoph2d, Qe and rho1
Subsystem3,. calculates L1, Vb and Qf1
All these Subsystems are grouped to form 'Flows and levels'.

Subsystem1 (h1,Qdd1,Qd)

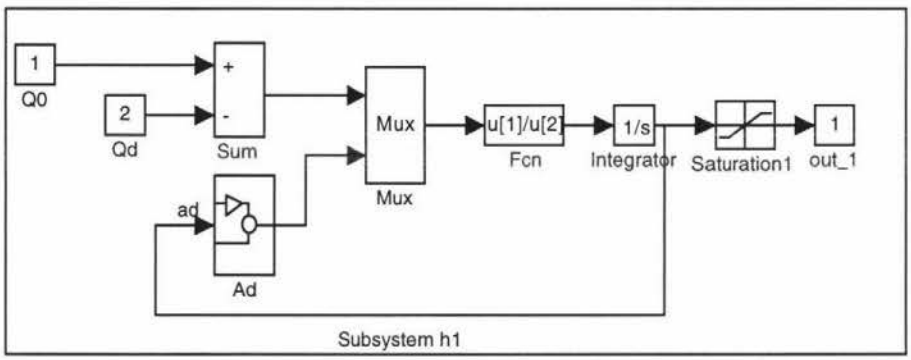


The input to this Subsystem block is Q0 and outputs are h1, Qd and Qdd1.
Subsystem h1 calculates height of the liquid above distribution plate. Subsystem Qd calculates volumetric flow out of the distribution plate which is then passed through the Transport Delay block to calculate Qdd1 (distributed flow delayed by 4 seconds one residence time). The initial input in the Transport Delay block is equal to, 209.6496l/hr (Qd).

The height of the liquid above distribution plate is given by the differential equation:

$$\frac{dh(t)}{dt} = \frac{Q_0(t) - Q_d(t)}{A_d}$$

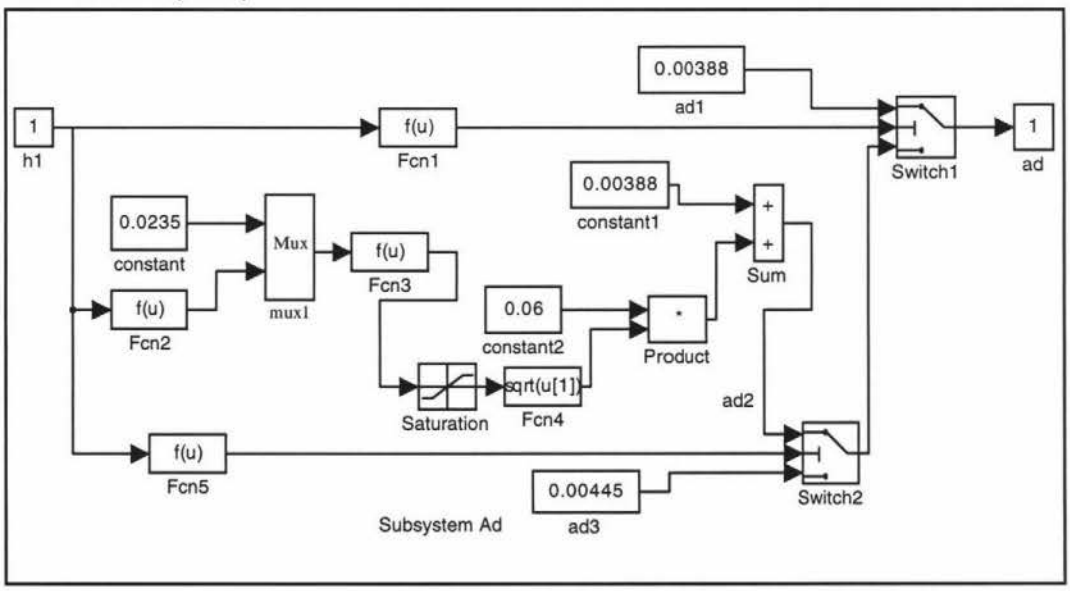
Referring to the *Subsystem 'h1'* the output from the Sum block is the difference between Q0 and Qd. The Saturation block imposes limits between 0 and 500l/hr. This is done to make sure no negative or complex number values pass through. The Fcn block which has inputs u[1], (Q0-Qd), and u[2] (Ad) determines the rate of change of height dh/dt.



This is then passed through the Integrator block with initial condition of 0.0031m to determine height of the liquid above distribution plate. The Saturation1 block limits the height to be minimum of 0m and maximum of 0.071m. All the blocks are selected and grouped together to form the Subsystem 'h1'

Subsystem *Ad* calculates the area of liquid surface on the distribution plate. It represents the following conditional statement:

```
if (h <= 0.026),
        Ad=0.00388;
elseif (h < 0.071) & (h > 0.026),
        Ad=0.00388+0.06*sqrt(0.0235^2-(0.0495-h)^2);
else Ad=0.00445; end;
```



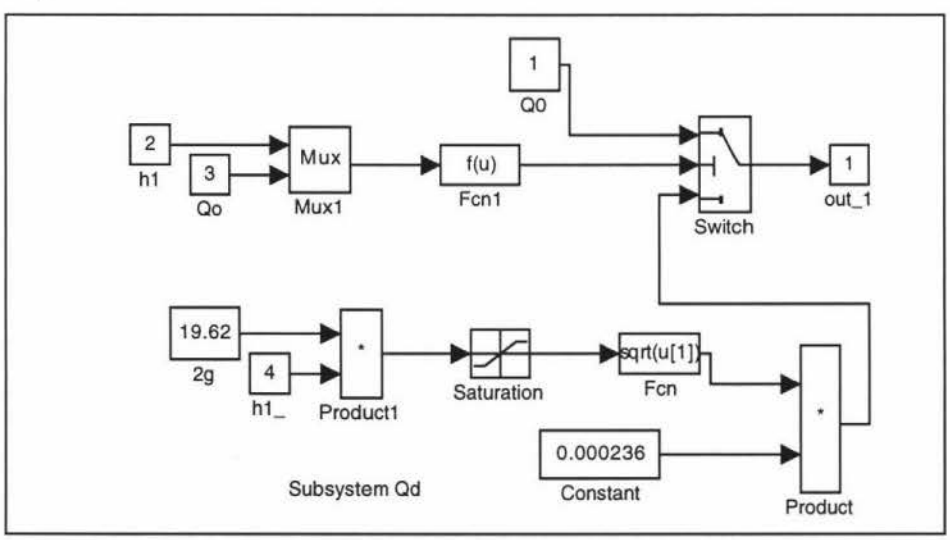
The Switch1 block checks for the condition h<=0.026which is contained in the Fcn1 block for true or false. If the condition is true it passes the first input, ad1, as the output from the Switch1 block. If the condition is false it passes the second input as the output.

The Switch2 block checks for the condition (h < 0.071) & (h > 0.026) for true or false. If the condition is true it passes the first input, ad2, as the output from the Switch block. If the condition is false it passes the second input, ad3, as the output.

Ad2 is calculated using the equation Ad=0.00388+0.06*sqrt(0.0235^2-(0.0495-h)^2).
The block Mux1 groups constant 4.7672 and function (0.0495-h) represented in the block Fcn2 as (0.0495-u[1]) to create a vector of input to the function block Fcn3. In the block Fcn3 u[1] corresponds to the first input which is 4.7672 and u[2]corresponds to second input (0.0495-h). The output from the Fcn3 block which is (0.0235^2-(0.0495-h)^2) sent through the Saturation block, where the limits for Ad are set in the range of 0 and 0.05, to make sure no negative or complex number values pass through. The block Fcn4 calculates the square root of its input. The output from the Sum block is 0.00388+0.06*sqrt(0.0235^2-(0.0495-h)^2).

Subsystem *Qd* represents the following conditional statement:

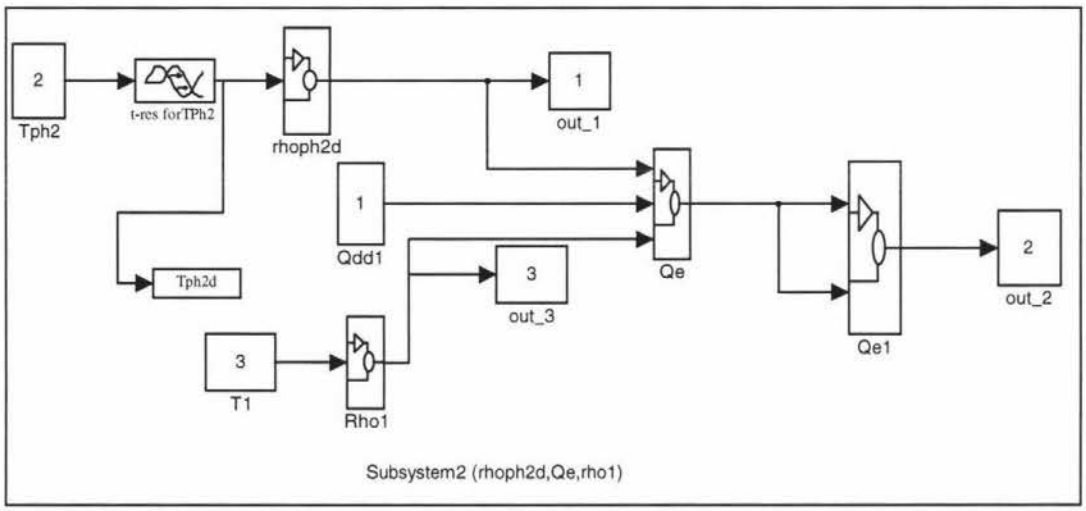
```
if (h1 >= 0.071) & (Q0 > (Ahl*sqrt(2*g*0.071/xi))),
                Qd1=Q0;
        else
                Qd1=Ahl*sqrt(2*g*h1/1);
end;
```



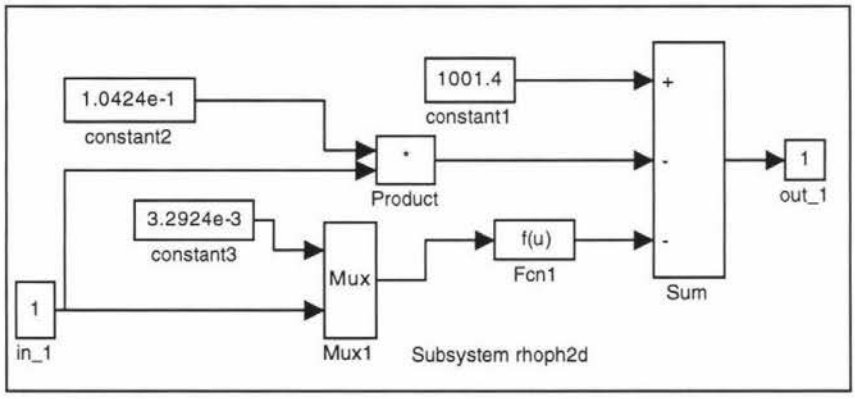
The Switch block checks for the condition (h1 >= 0.071) & (Q0 > (Ahl*sqrt(2*g*0.071/xi))) contained in the Fcn1 block for true or false. If the condition is true the first input is propagated and if the condition is false third input drives the output.

Subsystem2 (rhoph2d, Qe,rho1)

The inputs to this Subsystem are Qdd1, Tph2 and T1 and the outputs are rhoph2d, density of the product in the preheater two delayed by one residence time, Qe, volumetric flow out of the evaporation tube and rho1 density in the first effect. Tph2, temperature in the preheater two is passed through the Transport Delay block with delay of 4 seconds and initial input of 70 °C to calculate Tph2d.

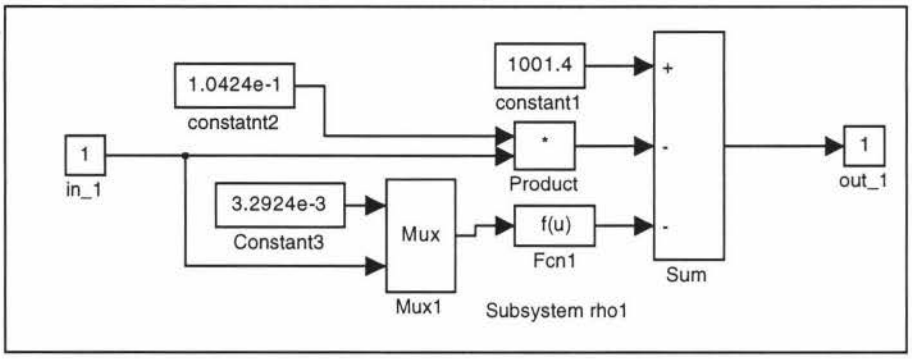


Subsystem *rhop2d* implements the following expression:
rhoPh2d=1001.4-1.0424e-1*TPh2d-3.2924e-3*TPh2d^2.

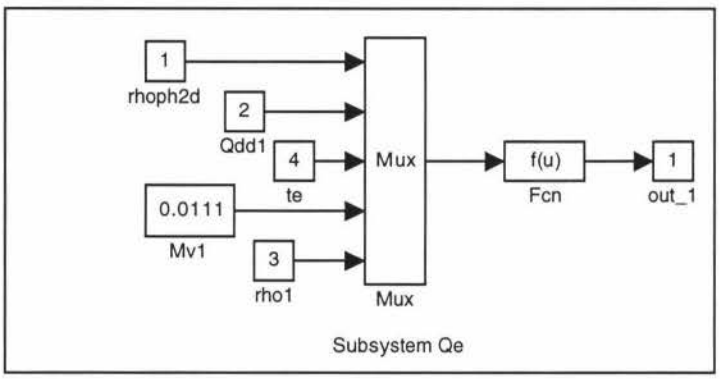


Input port in_1 refers to Tph2d and outport refers to rhoph2d. The output from the Product block is 1.0424e-1*TPh2d and output from the Fcn1 block is 3.2924e-3*TPh2d^2. The output from the Sum block is 1001.4-1.0424e-1*TPh2d-3.2924e-3*TPh2d^2.

Subsystem *rho1* implements the following expression:
rho1=1001.4-1.0424e-1*T1-3.2924e-3*T1^2.



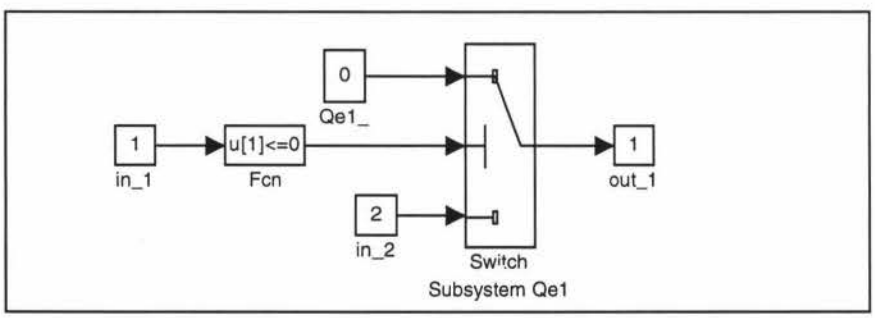
Subsystem *Qe* calculates the flow out of evaporation tube is calculated using the equation Qe = (rhoPh2d*Qdd1-1/te*Mv1)/rho1. All the inputs group through the Mux block to form a single vector output. The output from the block Fcn is the required flow.



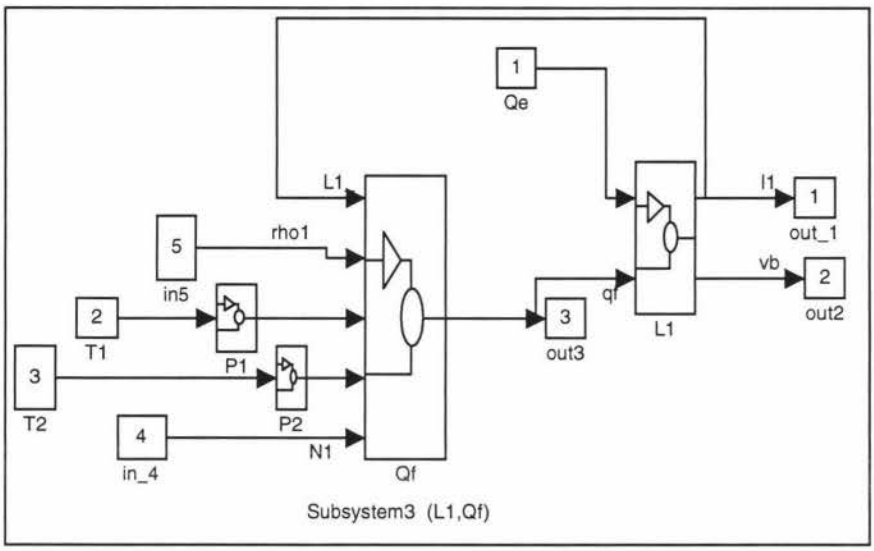
Subsystem *Qe1* implements the following code:

```
if Qe1<=0,
        Qe1=0;
end;
```

The Switch block checks for the conditional statement contained in the function block Fcn for true or false. If the condition is true first input is the output and if false third input drives the output. Inports in_1 and in_2 both refer to Qe.



Subsytem3 (L1,Qf)



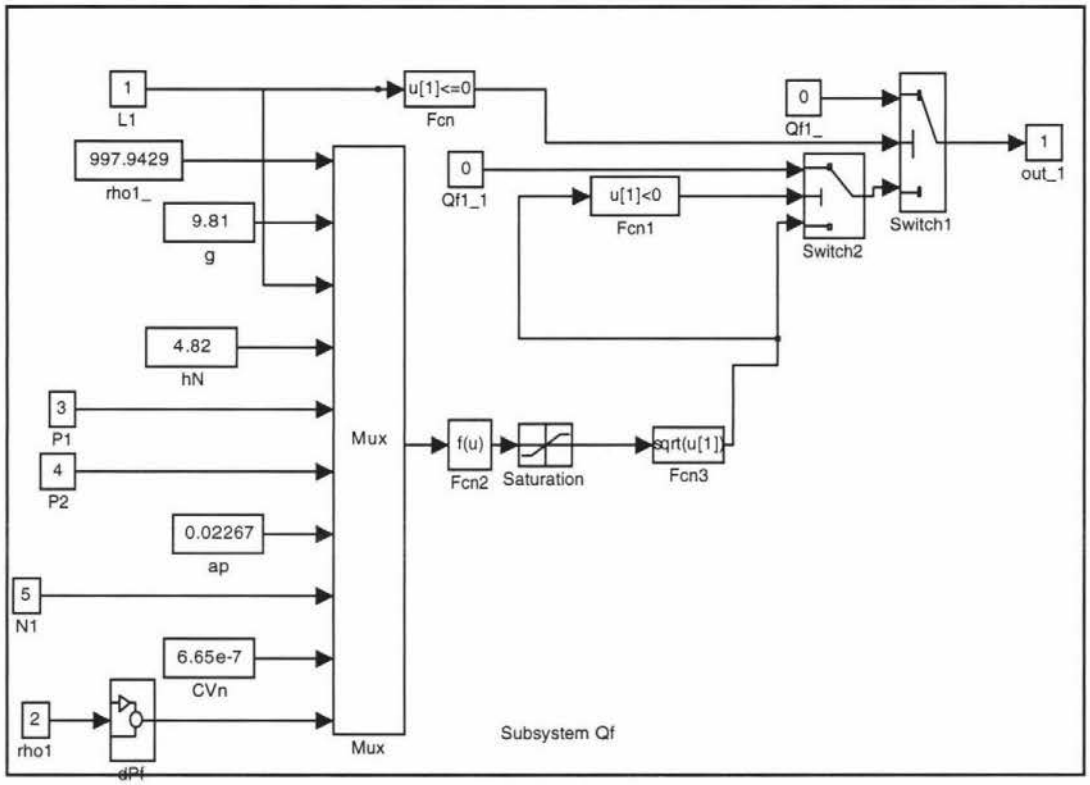
Subsystem3 calculates, L1, liquid level in the separator tank and Qf, volumetric flow to the next effect.

The outflow to the next effect, Qf, is calculated using the expression:
Qf1 = sqrt((rho1*g*(L1-hN)+P1-P2+ap*N1^2)/(1/(Cvn2^2)+dPf)).
The Subsystem 'Qf' implements this conditional statement:

```
if (L1 <= 0),
        Qf1=0;
    else
        dPf=32*rho1*f*Le/(pi^2*d^5);
        Qf1=sqrt((rho1*g*(L1-hN)+P1-
P2+ap*N1^2)/(1/(Cvn2^2)+dPf));
        if (Qf1 < 0)
            Qf1=0;
        end;
    end;
```

The block Switch1 will output the calculated Qf or 0 if separator tank is empty (L1<=0).

All the inputs are grouped through the Mux block to form a single vector output whose width is same as the number of inputs specified. The Fcn2 block contains the expression (rho1*g*(L1-hN)+P1-P2+ap*N1^2)/(1/(Cvn2^2)+dPf). u[1] denotes to first input u[2] denotes the second input and so on. The output from the Fcn2 block is passed through the Saturation block which imposes lower and higher limits on Qf from 0 to 500l/hr. The Fcn3 block calculates the square root of its input.



The block Switch1 will output the calculated Qf or 0 if separator tank is empty (L1<=0).

All the inputs are grouped through the Mux block to form a single vector output whose width is same as the number of inputs specified. The Fcn2 block contains the expression (rho1*g*(L1-hN)+P1-P2+ap*N1^2)/(1/(Cvn2^2)+dPf). u[1] denotes to first input u[2] denotes the second input and so on. The output from the Fcn2 block is passed through the Saturation block which imposes lower and higher limits on Qf from 0 to 500l/hr. The Fcn3 block calculates the square root of its input.

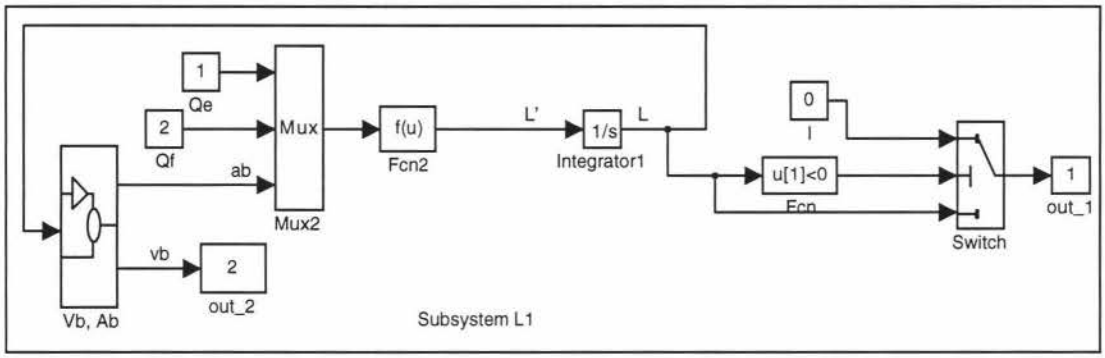
The height of the product in the separator tank L(t) is a state variable and is given by the differential equation:

$$\frac{dL(t)}{dt} = \frac{Q_e - Q_f}{A_b}$$

Qe is the volumetric flow out of the evaporation tube
Qf is the flow into next effect
Ab is the area of liquid surface at the base of the effect

Subsystem L1 is as shown below:

The inputs to the Mux block are grouped to create a vector of input to the function block Fcn2. In the block Fcn2 u[1] corresponds to the first input, u[2] corresponds to the second input and u[3] corresponds to third input. The output from the Fcn2 block dL/dt is passed through the Saturation block to the Integrator block to produce the required output L .



The Switch block checks for the condition L<0 contained in the Fcn block. If it is true sends the first input (L=0) to the Switch block and if the condition is false sends the third input.

Subsystem *Vb, Ab* calculates Vb ,volume of product collected at the base of the effect and Ab, area of liquid surface at the bottom of the effect. It implements the following conditional statement:

```
if (L < 1.02),
        Ab=0.001792;
        Vb(L+0.2)*Ab;
elseif (L < 1.06),
        Ab=0.002168;
        Vb.(L+0.2)*0.001792+(L-1.02)*0.000376;
elseif (L < 1.11),
        Ab=pi/4*(0.0218^2+(0.5*(L-1.06)+0.0478)^2);
        Vb=0.002273+pi/6*((0.5*(L-1.06)+0.0478)^3-
0.0478^3)+pi/4*0.0218^2*(L-1.06);
elseif (L < 1.39),
        Ab=0.004536;
        Vb=0.0024365+(L-1.11)*Ab;
elseif (L < 1.415),
        Ab=pi/4*(0.0728^2+(k*(L-1.39)+0.0218)^2);
        Vb=0.0037066+pi/12/k*((k*(L-1.39)+0.0218)^3-
0.0218^3)+pi/4*0.0728^2*(L-1.39);
else
        Ab=pi/4*((k*0.025+0.0218)^2+0.0728^2);
        Vb=0.00389+(L-1.415)*Ab;
end;
```

Temperature and Heatflows

The differential equation involving T1 and heat flows is given by

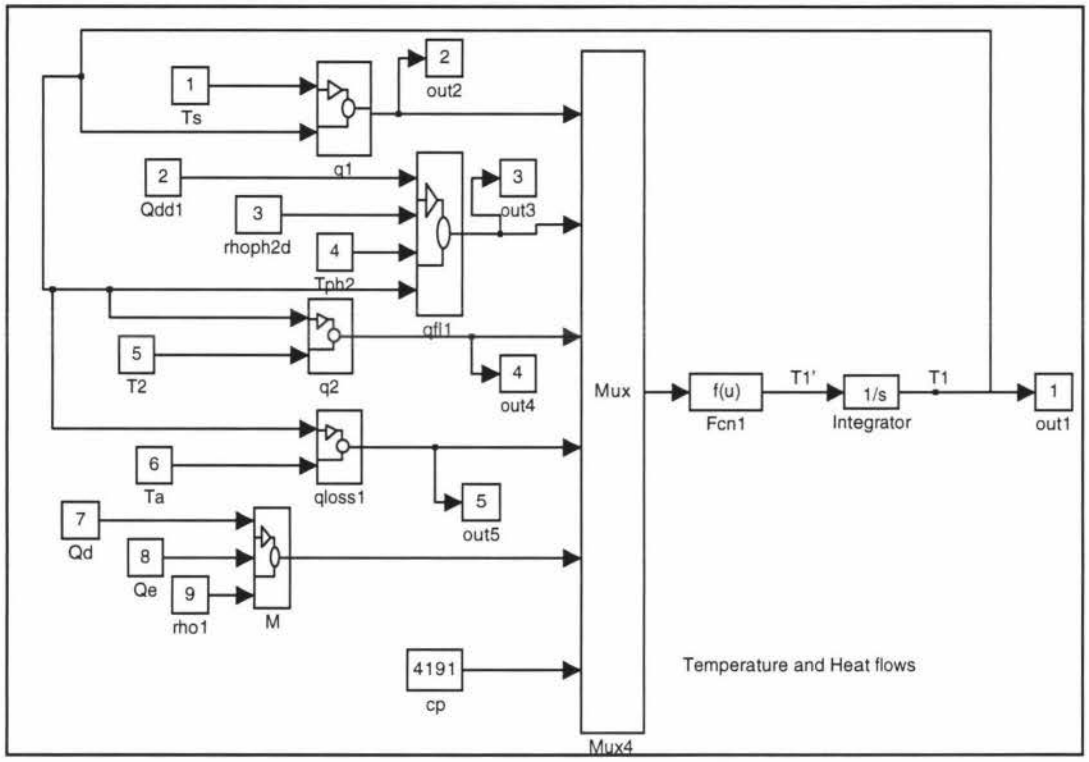
$$\frac{dT_1}{dt} = \frac{q_1(t) + q_{fl1}(t) - q_2(t) - q_{loss}(t)}{Mc_p}$$

q_1 is the energy supply for evaporation and q_2 is the energy supplied to the next

effect and are given by:

$$q_1(t) = U1 * A_{t1} * (T_S(t) - T_1(t))$$

$$q_2(t) = U2 * A_{t2} * (T_1(t) - T_2(t))$$



U1 and U2 are the heat transfer coefficients, $A_{t1}$and A_{t2} are heat transfer surface areas of the tubes in the first and second effects.

q_{fl1} is the heat flow into effect due to product feed and is given by equation:

$$q_{fl1} = Qdd1 * rhoph2d * Cp * (T_{ph2}(t) - T1(t))$$

Qdd1 is the flow out of distribution plate delayed by one residence time and rhoph2d is density of liquor in preheater two delayed by one residence time. Cp is the specific heat capacity of product in the effects and preheaters.

The heat lost to the surroundings is given by:

$$q_{loss}(t) = Uloss * A_S * (T_1 - T_o)$$

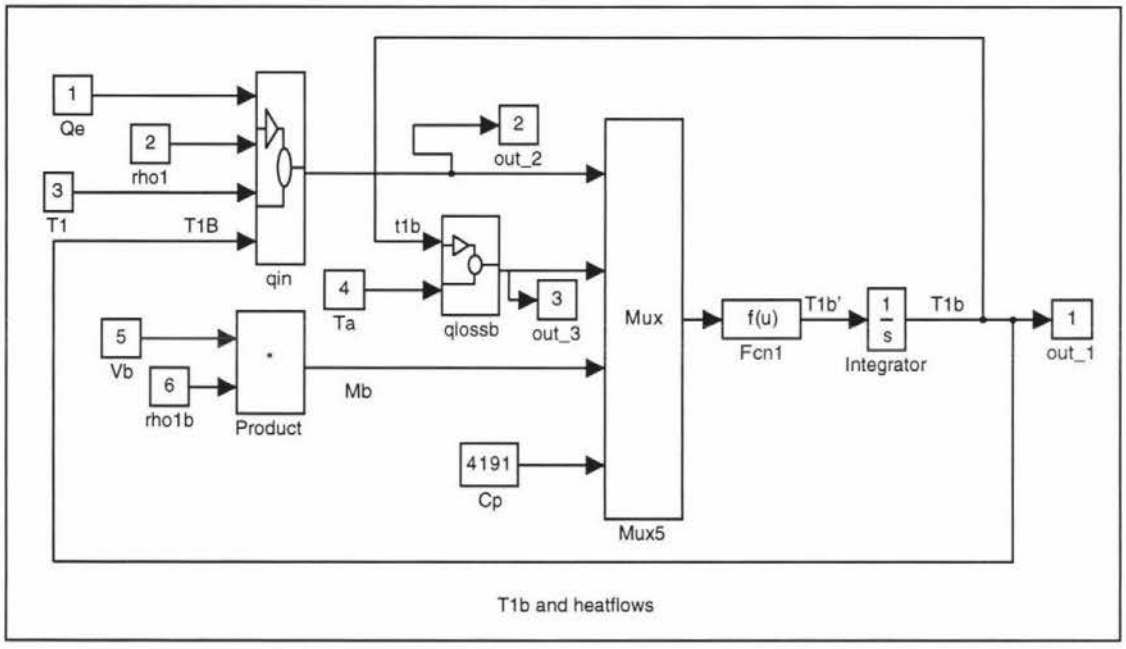
Uloss is the heat transfer coefficient to the surroundings, A_s is the total surface area of the effect and T_o is the ambient temperature.

M is the total mass of the product calculated using the equation:
M=(Qd1+Qe1)/2*te*rho1

In the Subsystem *temperature and heat flows* the inputs, q_1, q_2, q_{fl1}, q_{loss}, M, Cp
to the Mux block are grouped to create a vector of input to the function block Fcn1. In the block Fcn1 u[1] corresponds to the first input, u[2] corresponds to the second input and so on. The output from the Fcn2 block dT1/dt is passed through the Integrator block with initial condition of 70.1°C to produce T1. All

the heat flows are also available as outputs.

T1b and heatflows



T1b is the temperature at the base of the effect given by the differential equation:

$$\frac{dT1b(t)}{dt} = \frac{(q_{in} - q_{lossb})}{Mb/Cp}$$

q_{in} is the energy at the base of the effect and q_{lossb} is the heat lost to the surroundings at the base of the effect given by the following equations:

$$q_{in} = Qe * rho1 * Cp * (T_1 - T_{1b})$$
$$q_{lossb} = Ulossb * Abs * (T_{1b} - Ta)$$

Mb, the mass at the base of the effect, is given by:
$M_b = V_b * rho1b$

The inputs to the Mux block q_{in} q_{lossb}, Mb, rho1b are grouped to form a single vector input to the Fcn1 block. The expression in the Fcn1' block is (u[1]-u[2])/u[3]/u[4]), u[1] represents first input, u[2] second ,u[3] third and u[4] fourth. The output from the Fcn1 block dT1b/dt passes through Integrator block with initial condition of 70°C to give output T1b. Heat flows q_{in} and q_{lossb} are also available as outputs.

Mv1

Evaporation rate Mv1 is a state variable given by

$$\dot{Mv1} = \frac{q_1(t)}{r_1(t)} - \frac{q_1 d}{r_1 d}$$

q1d and r1d denote heat flow across tube walls and latent heat of vaporisation delayed by one tube residence time respectively.

q1 is given by

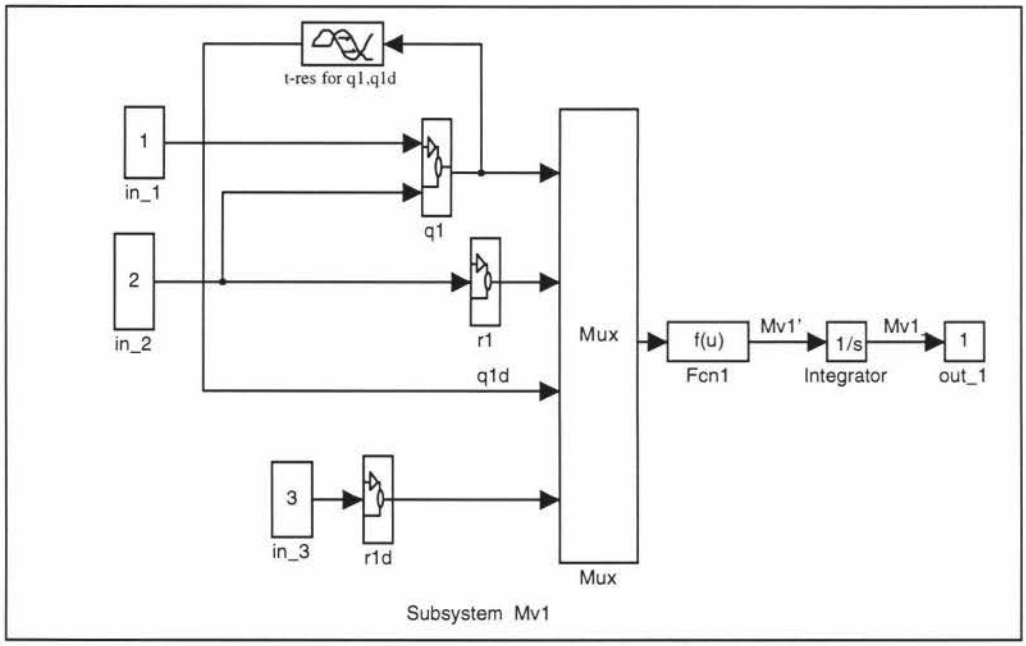
$q_1(t) = U1 * A_{t1} * (T_S(t) - T_1(t))$

q1 is passed through the Transport Delay block with delay of 4 seconds to give q1d.

The equations for latent heats are

r1=(2497.9-2.2063*T1-2.0131e-3*T1^2)*1e3;

r1d=(2497.9-2.2063*T1d-2.0131e-3*T1d^2)*1e3



The inputs to the Mux block q1, r1, q1d, r1d are grouped to form a single vector input to Fcn1 block. The expression in the Fcn1 block is u[1]/u[2]-u[3]/u[4], u[1] represents first input, u[2] second ,u[3] third and u[4] fourth. The output from the Fcn1 block dMv1/dt passes through Integrator block with initial condition of 0.0111 to give output Mv1. Heat flows q_{in} and q_{lossb} are also available as outputs.

6 TESTING THE EVAPORATOR MODEL

The model supplied with the operating point inputs was tested to check for the outputs from the system and to see if the simulation of the model was correct. The input values for the system are summarised as below:

Input	Value	Unit
Feed flow,Q0	209.73	L/hr
Preheater two temperature, (feed temperature) Tph2	70	°C
Effect two temperature, T2	63	°C
Pump speed, N1	1240	rpm
Steam temperature, Ts	80	°C
Ambient temperature, Ta	18	°C
Atmospheric Pressure, Pa	101325	Pascals
Feed pump speed, N0	340	rpm
Feed level, LT	0.75	m

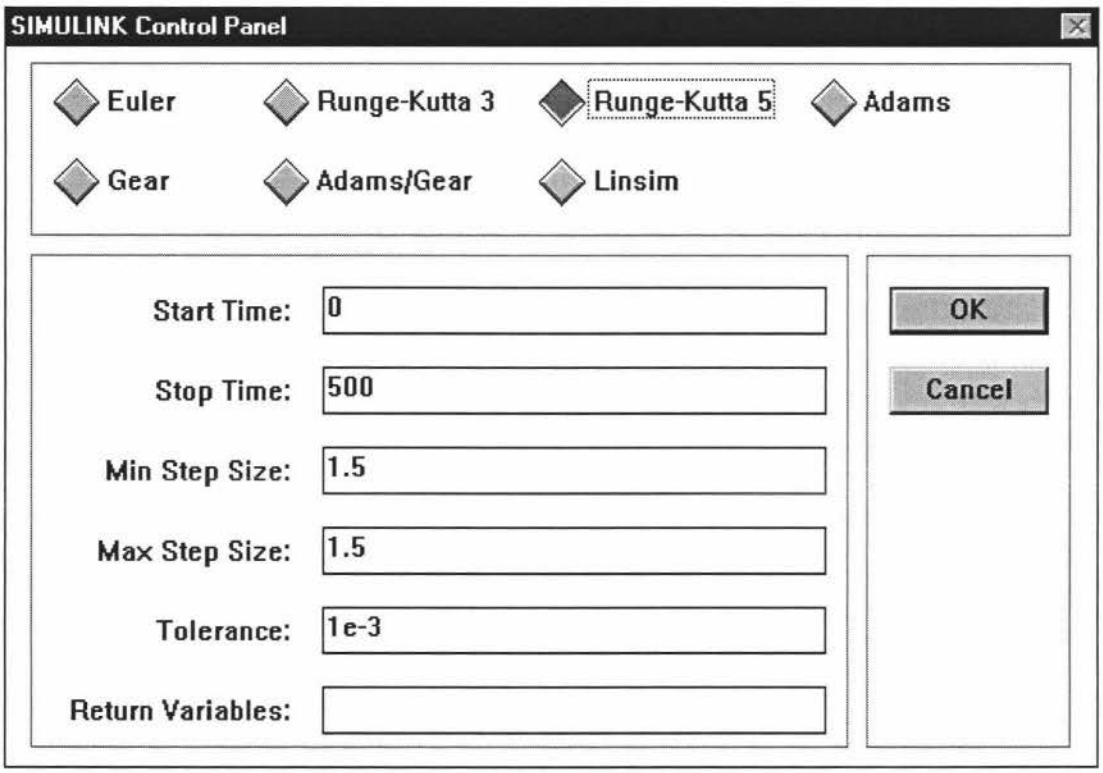
The model simulation can be started by selecting the integration technique and parameters used during simulation. This can be done by pulling down the *Simulation* menu and choosing *Parameters.* (A simulation can also be started from the Matlab command line). A dialogue box opens showing all the default parameters that can be modified.(shown in the next page)

The Integration Algorithm chosen was *Runge-Kutta 5* (fifth order) method but can be replaced by others.

Fixed step size (Min Step size = Max Step size) of 1.5 seconds was used.

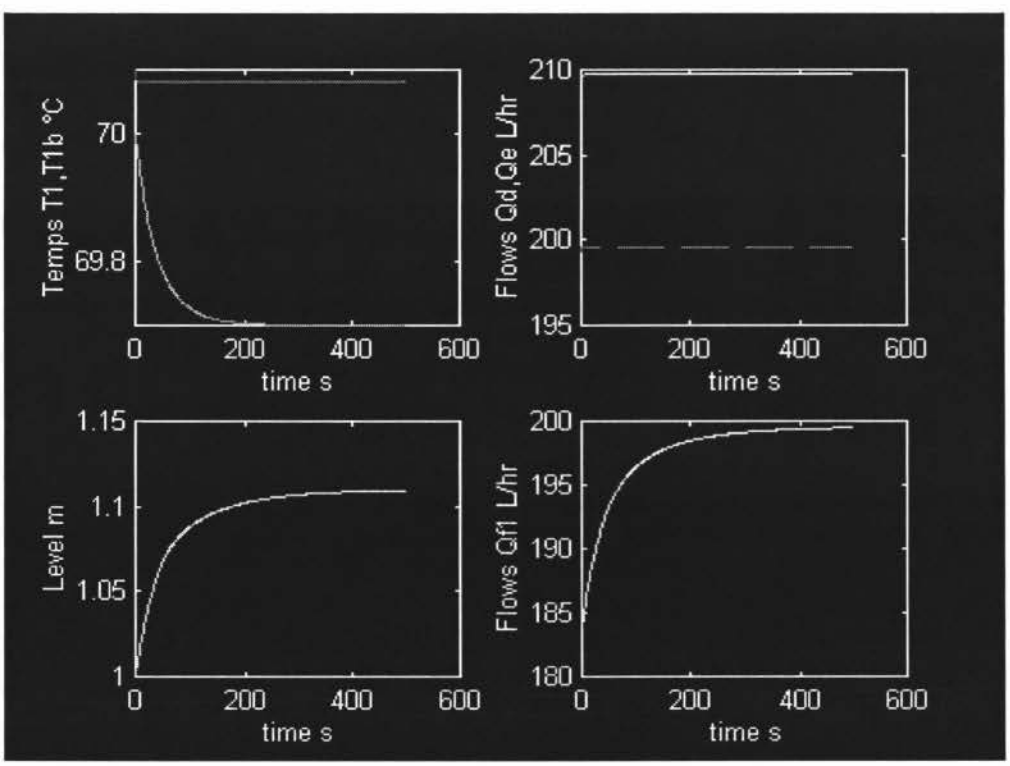
The *Simulation time* was for 500 seconds.

Parameters, Tolerance and Return Variables, were ignored.
Once the parameters are set, click the *OK* button to close the simulation dialogue box and start the simulation by choosing *Start* from the *Simulation* menu.



Note: The initial values displayed in the dialogue box are just defaults and do not necessarily match the simulation parameters set for this particular model

The figure below shows four plots containing various output results of interest for a simulation of 500 seconds.



The top left shows that the temperatures T1 and T1b settle very close to operating point values of 70.1°C and 70°C. The bottom left plot shows the level settling at around 1.1m. The figure on the top right show flows Qd, distributed flow at around 209L/hr and Qe, flow out of evaporation tube at around 199L/hr with evaporation rate of 10L/hr. The bottom right shows output flow Qf1 at around 199L/hr. All values obtained from the simulator agreed well with the results obtained when all the state derivatives were set to zero, and an equilibrium point calculated for the same input values. It was therefore concluded that the simulation of the model had been successful.

7 REASONS FOR FINAL SOFTWARE DESIGN

7.1 Platform

This project initially was aimed at producing a real-time application that runs under Windows 3.1 which is a 16 bit operating system. The RTW is designed to produce real-time applications that run under 32 bit operating systems. A Windows 3.1 solution was not, therefore, possible.
Applications could be produced to run under extended DOS via one of a number of 32 bit DOS extenders but this imposed some hefty performance penalties. The RTW can also produce NT character mode executables that are essentially DOS applications that use the 32 bit DOS emulation native to Windows NT. Since FIX also runs as a 32 bit application this was the selected platform.

7.2 Compiler

EDA (Easy Database Access) is a library of subroutines that provide easy access to FIX data. The EDA library for FIX is built using the Microsoft Visual C compiler/linker. The RTW, however, does not support Microsoft C and must use the Watcom or Metaware compilers to produce real-time applications. The Watcom compiler cannot read the Microsoft library format. In order to use the EDA library the compiler needs to be able to read the Microsoft library format; the Microsoft Visual C compiler/linker is therefore the best option. Because of this incompatibility between compilers the device drivers could not communicate directly with the SCADA database. Instead the real-time model (through its device drivers) and the EDA library must exchange data through disk files. The Watcom compiler is used to produce the model application with its device drivers and the Visual C compiler is used to produce the SCADA database communication program.

7.3 Real-Time Vs Nonreal-Time

Using the real-time model imposes a lower limit on the model sample rate of 18Hz or 0.056 seconds due to the hardware timer used. This is much faster than required to run a real-time application for the process industries. The real-time model code also has the unwanted behavior of stopping the simulation if any overrun occurs. For these reasons the nonreal-time code (nrt-main.c) was used as a basis and rewritten to provide more suitable sampling rates and overrun handling (prt-main.c). Other changes made to the nonreal-time code will be documented later.

8 DESCRIPTION OF THE SOFTWARE DEVELOPED

8.1 Overall Design

The over all design of the software falls into the following categories:

1. The RTW's nonreal-time code rewritten to provide a 'pseudo real-time' application.

2. The RTW device drivers written to read inputs from a file and write outputs to a file.

3. The template file modified.

4. The Fixend software that communicates with the real-time model via disk files.

8.2 Placement of All Software:

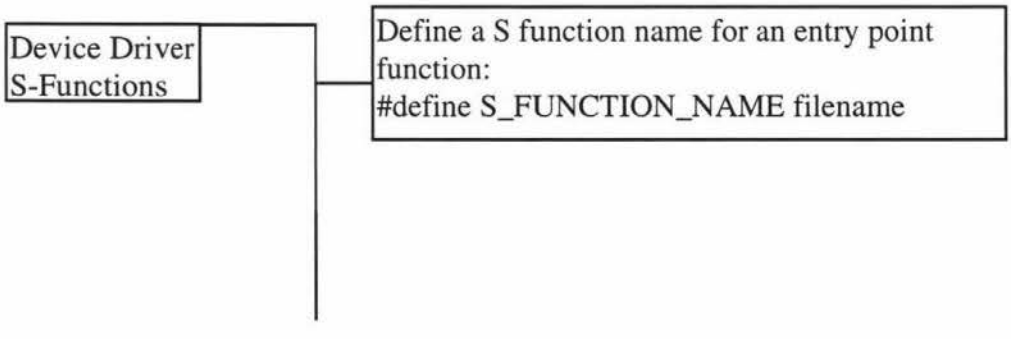
The developed software resides in the following directories:
NB: These are additional to the RTW files.

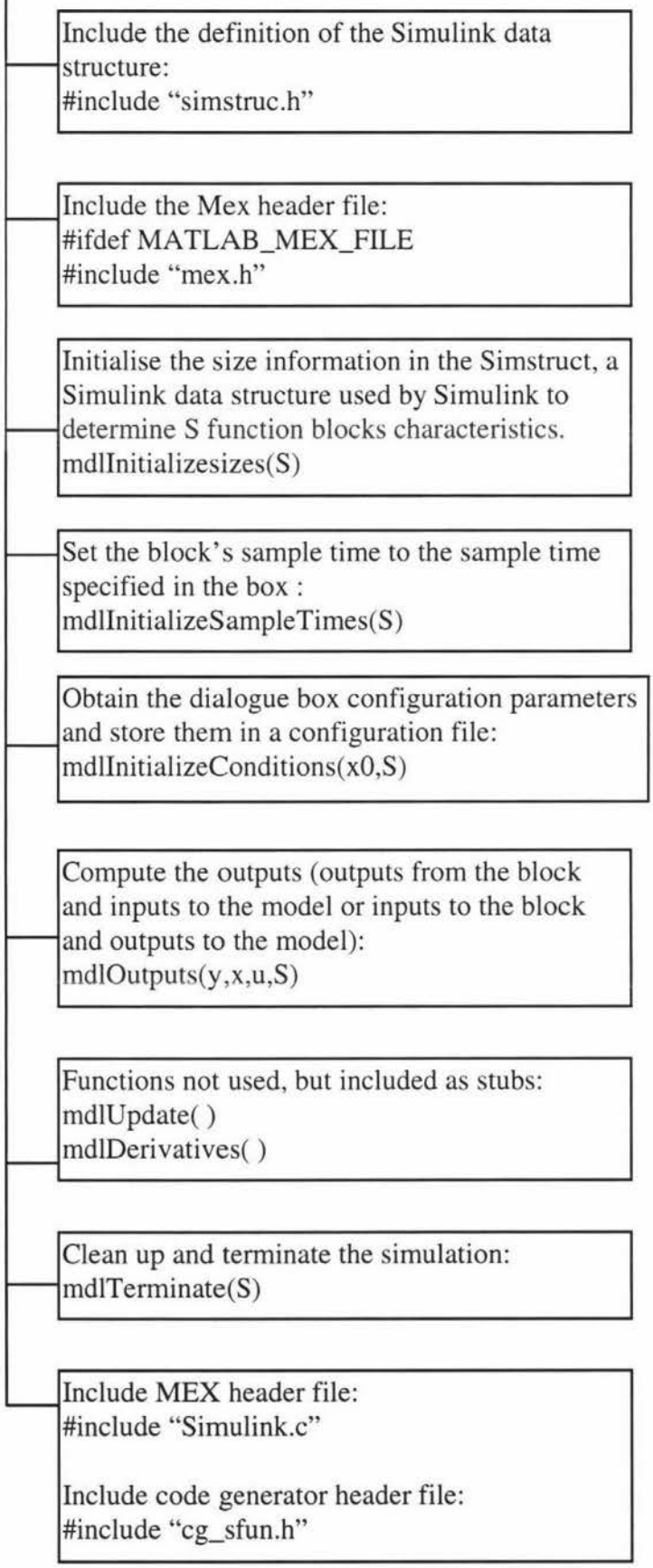
Prt_main.c	C:\Matlab\codegen\nrt\
Fix_watg.tmf	C:\Matlab\codegen\nrt\tmf\
Fromfile.c	C:\Matlab\codegen\rt\dos\devices\
Tofile.c	C:\Matlab\codegen\rt\dos\devices\
Fromfile.MEX	C:\Matlab\codegen\rt\dos\devices\
Tofile.MEX	C:\Matlab\codegen\rt\dos\devices\
Fixend.c	C:\Fix32\Fixend
Fixend.exe	C:\Fix32\
Fixlib.m	C:\Matlab\codegen

8.3 Structure And Function of RTW Device Drivers; Fromfile.C And Tofile.C

The directory Matlab/codegen/rt/dos/devices contains the MEX-files and the source code files (Fromfile.c and Tofile.c) for the device driver blocks Fixlink In and Fixlink Out. This directory is automatically added to the Matlab path when the device driver blocks from the 'Fixlib' are included in the model.

It is necessary to define specific functions, certain defined values and include files in order for the device drivers to perform required operations. The structure of the device driver S-functions is as shown below:





The main functions of device driver *Fromfile*.c are:

On initialisation

- To check that all the input arguments in the dialogue box are specified (five in total).

- To check whether the number of variables read are less than the maximum number of inputs.

- To check that the number of variables to read is equal to the number of tagnames specified.

- To set the block's sample time to the sample time specified in the dialogue box.

- To store the steady state inputs in the simstruct data structure where they can be later retrieved.

- To read the configuration parameters (readfile name, the sample time in milliseconds and the tagnames to read) from the block dialogue box and write them to a configuration file where they can be read by the Fixend program.

During Execution

- To read the variables from the read file (fromfix.dat) and store them as a vector of outputs from the block .
- To output the steady state inputs if the read file is not present.

On termination

- Remove the read file and the configuration file to signal the Fixend application which then restarts and waits.

The main functions of device driver *Tofile.c* are:

On initialisation

- To check that all the input arguments in the dialogue box are specified (four in total).

- To check that the number of write variables are less than the maximum number of outputs.

- To check whether the number of variables to write is equal to the number of tagnames specified.

- To set the blocks sample time to the sample time specified in the dialogue box.

- To read the configuration parameters (write file name, the sample time in milliseconds and the tagnames to write) from the block dialogue box and write them to a configuration file where they can be read by the Fixend program.

During Execution

- To read the inputs to the block and write them to the write file (tofix.dat).

On termination

- To remove the write file and the configuration file to signal the Fixend application which then restarts and waits.

8.4 Functions of Prt_Main.C And Changes From Nrt_Main.C

The RTW real-time code is written for real-time simulation and control of fast mechanical systems. Due to hardware considerations this imposed a limit on slow and fast sample rates. The slowest sample rate one can define is 18Hz or 0.056 seconds and the fastest sample rate is 4×10^{5} Hz or 2.5×10^{-6} seconds. The sample rate required for the process industries are well below this range; generally seconds to minutes. In addition the real-time code handles overruns by stopping the simulation. A more useful response would be to signal an overrun in some way and either leave the uncompleted simulation stop or complete it and start the next one late (while maintaining the given sample rate). Of these two the latter is more acceptable since the previous simulation step must be completed so the next one can occur. The result of overruns then is an apparent increase in the simulation model time constants i.e., the model runs more slowly than it should. To provide more suitable sample rates and to include the new method of handling overruns the nonreal-time code was rewritten to produce a 'pseudo real-time' application, the new file being named 'prt_main.c'. This still uses the nonreal-time dialogue box to accept parameters and some of these are ignored or must be constrained.

The prt_main.c produces real-time simulations that can be run at sample rates ranging from 5965 hours (slowest sample time) to 0.01seconds (fastest sample time). If overruns occur the code would simply run as fast as possible and write time stamped warnings to the application's window.

Most real-time programs are intended to run for indefinite lengths of time. Hence the stop time entered into the dialogue box is ignored and the program is stopped externally by pressing Ctrl-X.

8.5 Changes To Template File Dos_Watg.Tmf

The only change required of the template file is to include the new main program file. The source files option in the template file dos_watg.tmf is changed from nrt_main.c to prt_main.c. The new template file is called Fix_watg.tmf.

8.6 Functions And Design of Fixend Software

The functions of Fixend are:

- To read configuration information from a configuration file.

- To use this regularly to read information from a file and put it into the database.

- To use this to regularly write information from a database into a file.

The Fixend.exe file can be placed anywhere but would best go in the FIX 32 directory. The FIX system Configuration program can be used to make the Fixend.exe task start when the FIX starts up. The configuration files (c:\Matlab\config1.txt, c:\Matlab\config2.txt by default) are included as command line arguments in the FIX system configuration task although the 'DOS' window will need to be closed manually on shut down.

The start up mode for this task can be 'minimised' but should not be background or it cannot be shut down when finished.

Every time FIX is started it performs the following tasks:

On initialisation:

1. Checks that the FIX node is there by initialising access.

2. Remove the FIX read data file so that the model does not read incorrect data.

3. Attempts to find the configuration files. If not found it signals an error and tries again a second later.

4. The configuration information for the reads (FIX $\mapsto$ model) is read from the configuration file given in the first command line argument.

$\Rightarrow$ get the name of the read file from the configuration file.
$\Rightarrow$ get the sample time.
$\Rightarrow$ get the read tagname list.

5. The configuration information for writes (model $\mapsto$ FIX) is read from the configuration file given in the second command line argument.

$\Rightarrow$ get the name of the write file from the configuration file.
$\Rightarrow$ get sample time.
$\Rightarrow$ read the write tagname list.

6. Make the sample time the minimum of the read and write sample times.

$\Rightarrow$ Open the read file, get data from the SCADA database and write to the read file.
$\Rightarrow$ Open the write file, read the data from the write file and send the values to the SCADA database.

9 SOFTWARE USER'S GUIDE

The implementation environment used was FIX6.0 from Intellution. FIX6.0 comes with Easy Database Access (EDA) library that allows user applications to access the information in the SCADA database and to write back the results by including calls to this library in their code and linking with the EDA library.

9.1 System Configuration

Hardware Requirements

- A 100 MHz Pentium computer with minimum of 32 Mb of memory, running Windows NT is required. The RTW does not work under Windows 95 although Matlab and the compiled model will.

Software Requirements

- Simulink 3.1c is required to develop the model and the Real-Time Workshop1.1c and all its support files are required to create the code for the model.

- The Watcom 10.1 Compiler or later is required to compile and link the model.

- The FixSim software (device drivers, Fixend and all support files)

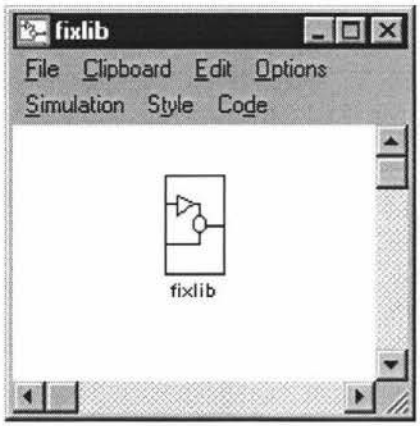
- FIX 6.0 is required to run the model.

9.2 Adding Device Drivers To The Model

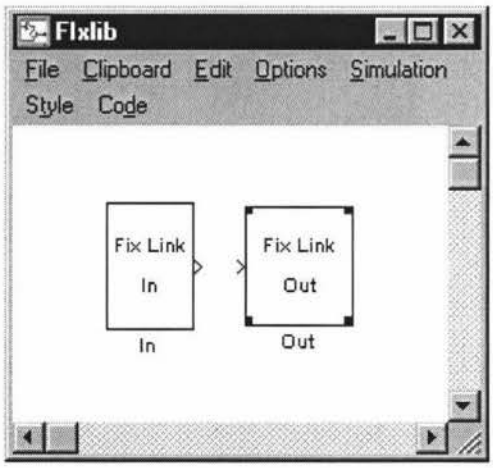
The simulation model and the FIX database communicate through the device driver blocks. The device driver blocks designed for this purpose are stored in the *Fixlib* block library and reside in the directory c:\matlab\codegen. To display this library make c:\matlab\codegen present working directory (should not be necessary if the codegen directory was put into the Matlab path properly.) and type *Fixlib* at the Matlab prompt.

```
>> cd c:\matlab\codegen
>>Fixlib
```

This displays the window shown in the next page

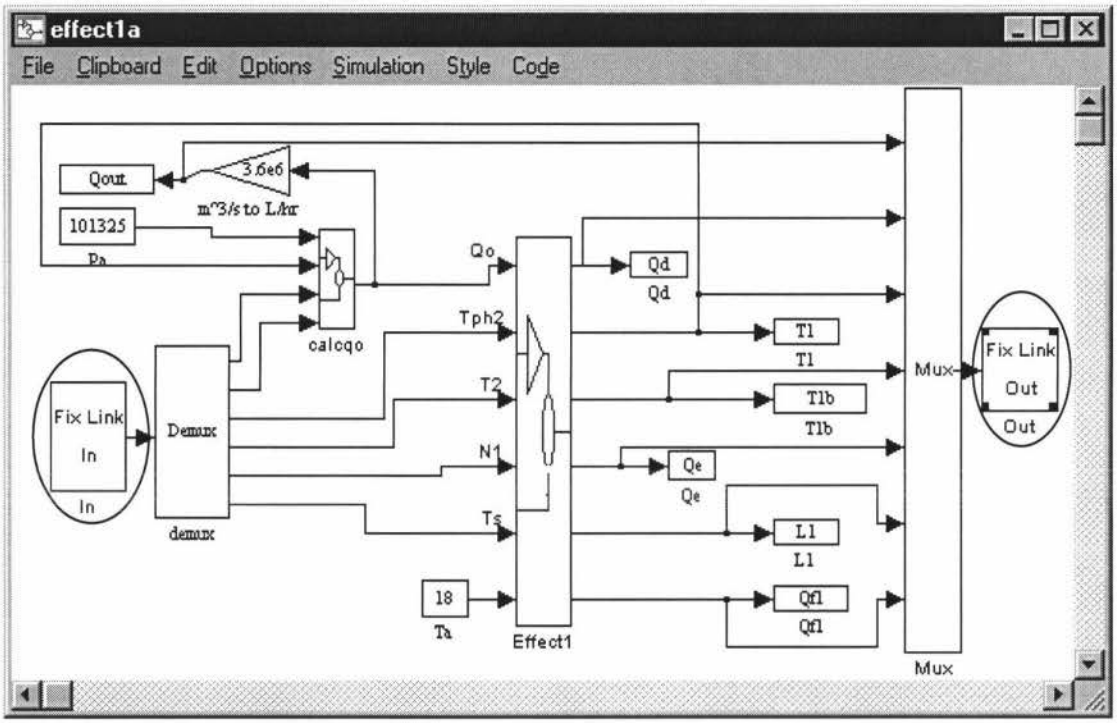


To access the device drivers open *Fixlib* by double clicking on the icon. The blocks are labeled Fixlink In and Fixlink Out.



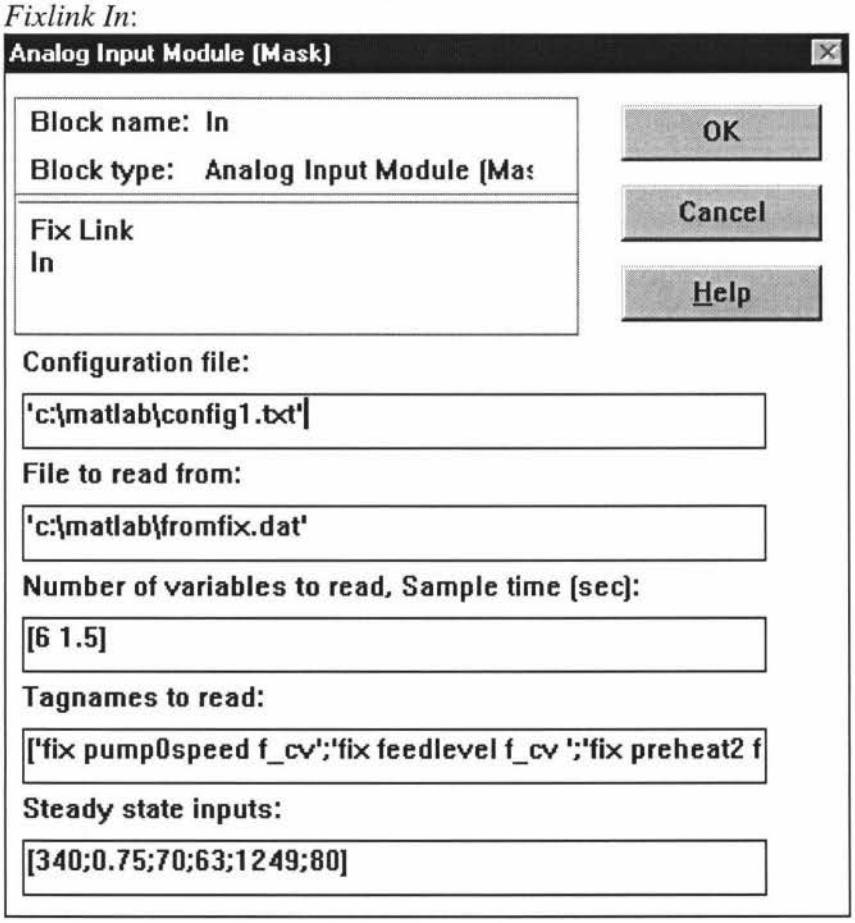
Drag the blocks into the *effect1* model and connect Fixlink In to the model inputs and Fixlink Out to the model outputs. Fixlink In has one output port and if there is more than one output (that is inputs to the model and outputs from the device driver block) use the block Demux to separate its outputs. Similarly Fixlink Out has one input port and if there is more than one input use the block Mux to vectorize its input (i.e. outputs from the model and inputs to the device driver block).

Note: You can use Fixlink In and Fixlink Out more than once in the same model. For this to work you have to have the second copy of the Fixend programme talking to different configuration files and different read write files.
When the model code is generated the C code used to implement the device drivers is automatically linked with the generated code.



9.3 Configuring Device Drivers

Each device driver block has a dialogue box where the parameters are set. Double click on each block to display its dialogue box. The Analog Input module below displays setting of parameters for the device driver block *Fixlink In*:



Configuration file by default is set to c:\matlab\config1.txt. The location for the configuration file can be any other directory or it can be a relative address. Since the real-time executable file produced is written to the Matlab directory, this was the chosen location. This configuration file must be the file specified when the Fixend software is run. If the Fixend software can read the configuration file then it will be told what values to read.

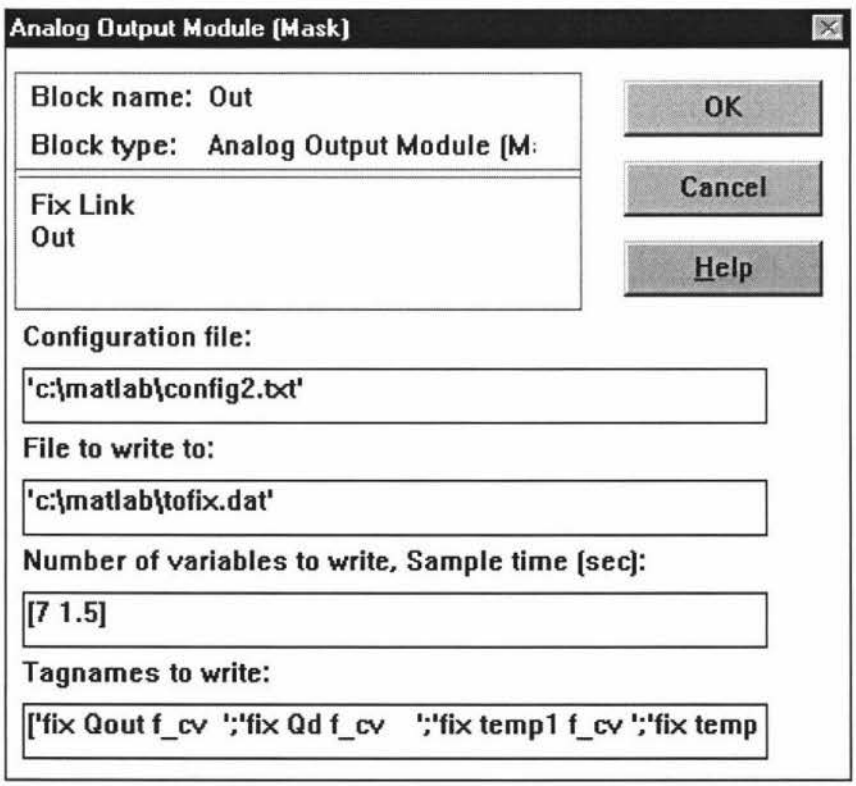
File to read from is set by default to c:\matlab\fromfix.dat but the location can be any other directory.

Number of variables to read is the number of model inputs to read and it must match with the number of tagnames to read specified in the next entry of the dialogue box. The sample time in seconds must be the same or multiple of the model step size set in the nonreal-time options dialogue box.

Tagnames to read are in node:tag.field format with spaces between them and are placed between quotes. Each tagname to read is separated by semicolon. The length of the tagnames must be the same as the longest one and trailing spaces should be put to pad for the shorter ones. The tagnames can also be defined in the Matlab environment by assigning the tagnames to a variable and placing that variable name in the dialogue box. This would be a more useful option when there are lot of variables to read.

Steady state inputs is a vector that is of the same length as number of tagnames.

Fixlink Out



Configuration file by default is set to c:\matlab\config2.txt.

File to write to is set by default to c:\matlab\tofix.dat.

Number of variables to write is the number of model outputs to write and it must match with the number of tagnames to write specified in the next entry of the dialogue box. The sample time in seconds must be the same or multiple of the base sample time (model step size set in the nonreal-time options dialogue box) for the simulation.

Tagnames to write are in node:tag.field format with spaces between them and are placed between quotes. Each tagname is separated by semicolon. The length of the tagnames must be the same as the longest one and trailing spaces should be put to pad for the shorter ones. The tag names can also be defined in the Matlab environment by assigning the tagnames to a variable and placing that variable name in the dialogue box. This would be a more useful option when there are lot of variables to write.

Note: The initial values displayed in the dialogue box are just defaults and do not necessarily match the configuration parameters set for this particular model.

9.4 Building a Real-Time Application

Select the *Nonreal_Time* options from the code menu. This displays the dialogue box shown in the next page.

The following options are set:

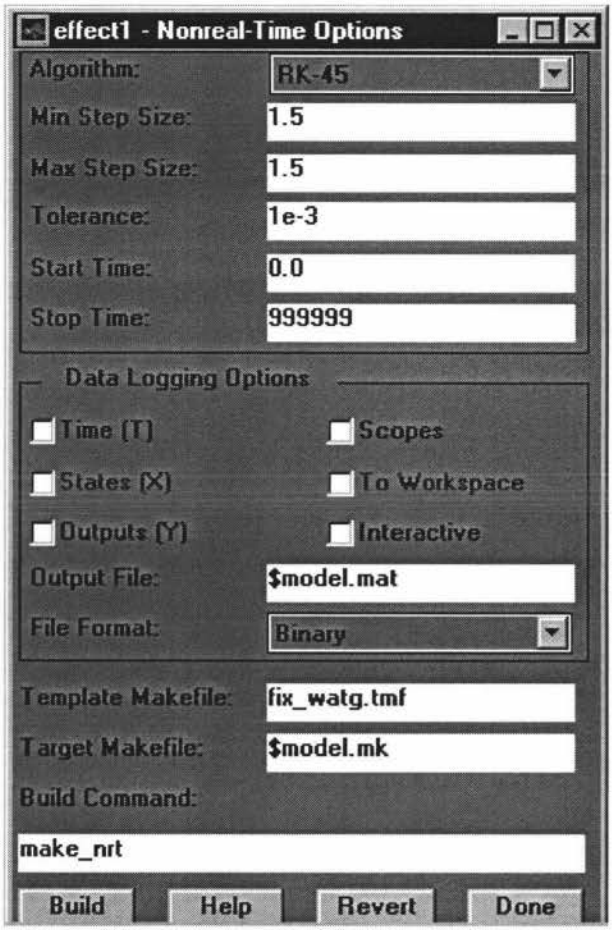
The *Algorithm* is set to RK-45 but any algorithm can be used.

Maximum step size and *Minimum step size* must be the same.
Before producing the real-time application the simulation in Simulink should be run with the same algorithm and same fixed step size as mentioned in the Nonreal-Time options dialogue box to make sure it works.

Tolerance can be ignored since it deals with step size adjustments.

Start time is usually zero but can be given any valid value to start the simulation after the specified time has elapsed.

Stop time is ignored by the real-time application which is stopped externally by pressing a key.



Data logging is optional and might be useful for debugging (data logging is used only to allow analysis and presentation of the data in Matlab).
Template make file is changed to fix_watg.tmf as this contain the new main program file prt_main.c.

Note: The initial values displayed in the dialogue box are just defaults and do not necessarily match the simulation parameters set for this particular model.

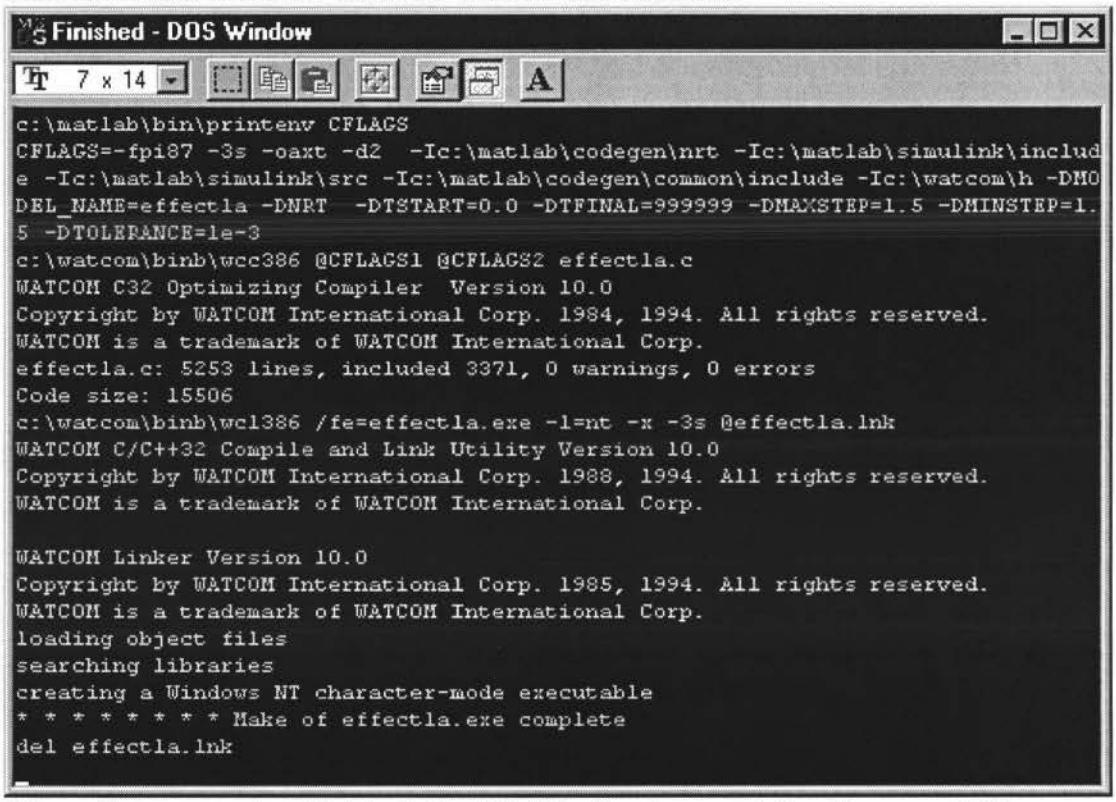
After setting the proper entries in the dialogue box, the real-time program is built by clicking on the **Build** button. This action causes the Matlab to display the following messages:

Writing 7 channels.
Write file is c:\matlab\tofix.dat.
Write sample time =1500ms.
Reading 6 channels.
Read file is c:\matlab\fromfix.dat.
Read sample time = 1500ms.

(As explained already the device driver blocks are compiled as MEX files which means they can be run from within Simulink. Matlab calls these MEX files before they are compiled into a real-time program to make sure there are no errors and they work correctly. The messages in the Matlab are produced

from the real-time program when it runs).

Then the DOS compilation window appears . If the object files for the device drivers are not already there they will be compiled again. Once the compilation is finished it gives a message saying it is creating Windows NT character mode executable and make of the target, effect1.exe (model name) is complete and deletes the link file. Now the DOS window can be closed.



(Once the process has finished, the Matlab directory contains the executable images of effect1.exe and the object files created in the process of building the source modules.)

Note: If the model is already running from this location the new exe file cannot be written.

9.5 Running the Real-Time Application

The real-time test programme can be run from the Matlab window by the following command :
>>!effect1
Or it can be run from the file manager by double clicking on 'effect1.exe' icon.
It comes up with the following message window:
Reading 6 channels.
Read file is c:\matlab\fromfix.dat.
Read sample time = 1500ms.
Writing 7 channels.
Write file is c:\matlab\tofix.dat.
Write sample time =1500ms.

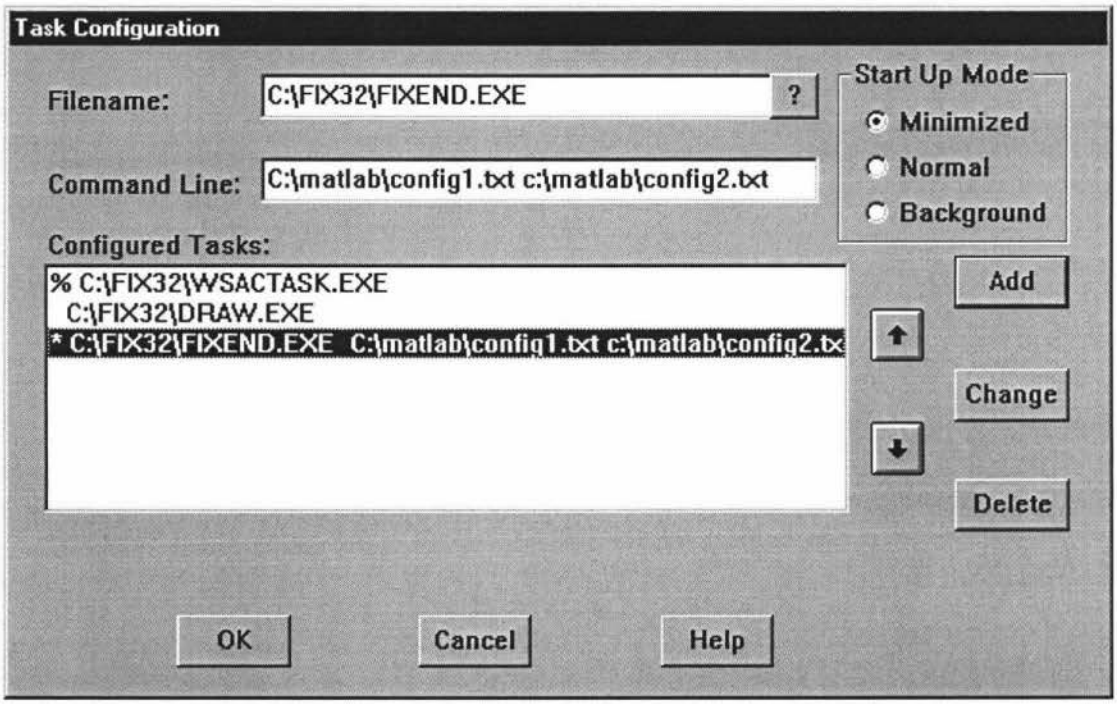
9.6 Configuring and Starting Fixend

Configuring the Fixend includes:

1. Setting up the SCADA configuration to call the Fixend programme.

2. Including the two configuration files that are specified in the dialogue boxes for the two device driver blocks as command line arguments in the FIX task configuration.

3. The start up mode for this task can be 'minimised' but it cannot be 'backgrounded' since it cannot be shut down manually when finished. The rest of the information comes from the configuration files themselves. This will run the Fixend programme.



If the Fixend is started before the test programme is run it simply says 'can't find the read configuration file'. Fixend cannot read them until the test programme actually produces it.

When both programs are running the Fixend programme comes up with the following message:

Read file is c:\matlab\fromfix.dat
Stepsize = 1500ms
Read tag [0] = FIX! pump0speed: f_cv
Read tag [1] = FIX! feedlevel: f_cv
Read tag [2] = FIX!preheater2: f_cv
Read tag [3] = FIX!temp2: f_cv
Read tag [4] = FIX!pump1sped: f_cv

Read tag [5] = FIX!steamtemp: f-cv

c:\matlab\tofix.dat
Step size = 1500
Write tag[0] = FIX!Qout: f_cv
Write tag[1] = FIX!Qd: f_cv
Write tag[2] = FIX!temp1: f_cv
Write tag[3] = FIX!temp1b: f_cv
Write tag[4] = FIX!Qe: f_cv
Write tag[5] = FIX!Level: f-cv
Write tag[6] = FIX!Qf1: f-cv

9.7 Configuring FIX Blocks

All the read and write tags are added to the FIX database as blocks with the same tagnames as in the device driver block dialogue boxes. The block type should be Analog output regardless of direction since this provides the greatest flexibility. Blocks should also be set to 'manual' mode.
Once all the blocks are added, the View programme in the FIX can be opened up to communicate with the real-time application.

9.8 Stopping the Fixend

The Fixend is stopped by pressing Ctrl-X.

9.9 Stopping the Real-Time Application

The real-time application is also stopped by pressing Ctrl-X.

APPENDIX 2

Block Libraries

The following are some of the blocks that will be used in the simulation model:

Sources

Constant Inject a constant value. The output of the Constant block is the value of the specified parameter.

Step function Generate a step function

Sinks (For testing only. Must be removed before RTW is used.)

To Workspace Write data to a matrix. The To Workspace block creates a matrix in the Matlab Workspace with the specified name containing the data from the vector input line. The matrix is not available until the simulation is terminated. Every time a simulation is finished, any existing matrix of the same name is overwritten.

Scope Display signals during simulation.

Linear

Gain Multiply input by a constant

Integrator The integrator block integrates its input. Its single parameter is its initial condition. The output of the integrator is its state, the integral. The initial value can be entered as a constant or a variable.

Sum The sum block accepts scalar inputs only and adds the value of each input to produce a scalar output. The list of signs parameter can be represented with a constant or a combination of the symbols (+-).

Non-linear

Fcn The function block applies the specified function to its input. The first element of the input vector is denoted as u[1]; u alone is taken as u[1]. A vector of inputs is created using the Mux block.

Product Multiply inputs together. The product multiplies the value of each input to produce the output. The number of input parameters can be represented with a constant or a variable.

Switch To Switch between two inputs. The Switch block uses its second input to control which of its other two inputs are propagated to its output. If the

signal on the second input is true, the first input is propagated; otherwise, the third input drives the output.

Saturation The Saturation block imposes upper and lower bounds on a signal. Within the range of the specified limits, the input signal passes through unchanged. Outside these bounds, the signal is clipped to the maximum or minimum bound.

Transport delay To delay the input by a given amount of time. The Transport Delay block can be used to simulate a time delay. It delays the input by a given amount of time. At start-up time, the block outputs the initial condition parameter until the simulation time exceeds the time delay.

Connections

DeMux Separate vector signal into scalar signals. The DeMux block separates a vector line into several scalar lines.. The input to a demultiplexer can be a vector of any width, and each output is the value of the first element of the input vector. The scalar value at each successive output is the value of the next element of the vector input. The demultiplexer block redraws its ports automatically to comply with the specified number of outputs.

Inport Provide link to an external input. Inports are the links from the outside world into a system. Inside a Subsystem block there is an inport corresponding to each input port on the block. A signal that arrives an input port on a Subsystem block flows out of the corresponding inport within that block. The inports within a Subsystem block must be numbered consecutively, starting with 1. The inport numbered 1 corresponds to the highest input port on the Subsystem; number 2 is the next highest; and so on. If two or more Inport blocks are included when some blocks are grouped together to create a new Subsystem block, the ports on the Inport blocks are renumbered automatically. When the Subsystem block is opened the port numbers may be different than they were in the original block diagram.

Mux Group scalar lines into a vector line. The Mux block groups several scalar lines into a single vector line. The width of the vector line is the same number of inputs specified. The Multiplexer redraws its ports to comply with the number of inputs set in the dialogue box.

Outport Provide link to an external output. The Outport block provides a mechanism for labelling a system's outputs. In a Subsystem, output ports correspond to outputs on the Subsystem block. If two or more outport blocks are included when some blocks are grouped together to create a new Subsystem block, the ports on the outport blocks are renumbered automatically. When the Subsystem block is opened the port numbers may be different than they were in the original block diagram.

Subsystem Groups blocks into a Subsystem. Subsystem blocks represent one system with another system. Any set of blocks and lines can be converted into

a Subsystem with the 'group' command on the options menu. The 'group' command removes all selected objects from the active window and replaces them within a Subsystem block. This new block when opened redisplays all of the grouped objects. The number of input ports drawn on the Subsystem block corresponds to the number of inports contained in the Subsystem. The top input connects to inport1, the next input connects to inport2, and so on. The output ports drawn on the block correspond in the same way to the number of outports contained within the Subsystem.

When a Subsystem is created unconnected input and output lines are replaced within the Subsystem with inports and outports respectively and are given corresponding ports on the outside of the Subsystem block. Any changes to the number of inports and outports within the Subsystem cause the block to be immediately redrawn with the correct number of input and output ports.

Note: Since some of the non-linear blocks (Saturation block, Switch block and Transport Delay block) are used in the simulation the integration algorithm 'linsim' cannot be used.

REFERENCES

1. Billet R. (1989), *Evaporation Technology,* VCH Publishers, New York.

2. Coulson J.M and Richardson.J.F. (1968), *Chemical Engineering Volume Two*, Pergamon New York.

3. Citect User's Manuals from Ci Technologies, Australia.

4. FIX User's Manuals and from Intellution Inc. USA .

5. Illingworth M. (1993), *Dynamic Modeling For A Three-Effect Falling-Film Evaporator* , Final Year Research Project, Massey University.

6. Intouch User's Manuals from Wonderware, USA.

7. Kinsky R. (1982), *Applied Fluid Mechanics* , McGraw Hill, New York.

8. McAdams W.H.(1954), *Heat Transmission* , Mcgraw Hill, New York.

9. Malab / Simulink / Real-Time Workshop User's Guide from The Math Works Inc, Natick, Massachusetts, USA.

10. Ogata K. (1990), *Modern Control Engineering* , Prentice Hall, New York.

11. Parr E.A. (1989), *Industrial Control Handbook Volume Three* , BSP Professional Books, London.

12. Perry J.H. (1963), *Chemical Engineers' Handbook*, McGraw Hill, New York.

13. Quaak P. &Gerritsen J.B.M.(1990), *Modeling Dynamic Behavior of Multiple-Effect Falling Film Evaporators* , Computer Applications in Chemical Engineering pp59-64, Elsevier Publishers, Amsterdam.

14. *Reducing Development Time For Real-time Simulations in SCADA Environment*, Control 96 Conference Paper presented by *Huub Bakker, Senior Lecturer* Department of Production Technology, Massey University, Palmerston North.