

PSIG 05B3

## Simulating pipeline control systems for operator training

David Basnett, Atmos International Ltd, John Lewis, Atmos International Inc.

Copyright 2003, Pipeline Simulation Interest Group

This paper was prepared for presentation at the PSIG Annual Meeting held in New Orleans, Louisiana, 26 October – 28 October 2005.

This paper was selected for presentation by the PSIG Board of Directors following review of information contained in an abstract submitted by the author(s). The material, as presented, does not necessarily reflect any position of the Pipeline Simulation Interest Group, its officers, or members. Papers presented at PSIG meetings are subject to publication review by Editorial Committees of the Pipeline Simulation Interest Group. Electronic reproduction, distribution, or storage of any part of this paper for commercial purposes without the written consent of PSIG is prohibited. Permission to reproduce in print is restricted to an abstract of not more than 300 words; illustrations may not be copied. The abstract must contain conspicuous acknowledgment of where and by whom the paper was presented. Write Librarian, Pipeline Simulation Interest Group, P.O. Box 22625, Houston, TX 77227, U.S.A., fax 01-713-586-5955.

### ABSTRACT

Most pipeline simulations are concerned primarily with the hydraulic performance of the pipeline. As a result, an idealised model of the control systems is assumed. Control devices are often assumed to operate instantaneously and perfectly. This is sufficient for a wide range of simulation tasks. However, for simulations that are intended to be used in training pipeline operators this is no longer suitable. What is required is to provide a 'virtual pipeline' that responds in the same way as the real pipeline would. This means that the delays, limitations and interactions of the control system need to be modelled in addition to the hydraulics of the pipeline.

This paper describes how a generic control capability was added to an existing liquid simulation package. The practical lessons learned from implementing a simulation of the full control system for a pipeline training system will be discussed. The potential drawbacks of the method chosen will also be highlighted.

### BACKGROUND

In a simple model of a pipeline, if you ask the simulation to close a valve or start a pump, then it does. In the real world, if an operator asks a Supervisory Control And Data Acquisition (SCADA) system to close a valve, any number of things may happen:

- The valve might close as requested
- The valve might not close because the pressure downstream is too low for the system to allow this.

- The valve might close, and a valve on an alternative route may open
- The valve might close, and several other related valves may also close.
- The valve might close, and after some time various other changes occur automatically.

The overall effect on the operation of the pipeline will obviously be different in each case. Similarly, the behaviour of the real control system to a change in set point may be completely different than that exhibited by an ideal control. As a result when training an operator in the running of the pipeline, it is important that the training system can mimic this behaviour.

In a real pipeline, this sort of control can be implemented in several places. It can be built directly into the SCADA control screens that the operator uses, implemented in the Programmable logic controllers (PLCs) used to directly control the pipeline objects, or some combination of the two. Training systems are required to simulate various levels of this control. In addition training systems often need the ability for a trainer to interfere with the normal operation of the pipeline or control logic in order to instruct a trainee in dealing with various faults.

In a pipeline, there are also usually two main forms of control, which interact with each other. The first is the 'process' control, which is responsible for controlling the pressure and flows in the pipeline by changing the positions of valves and the speeds of pumps or compressors in order to meet operator supplied set points. The second form of control is the 'logical' control, which is responsible for enforcing a set of rules for safe operation of the pipeline. The logical control is used for emergency shut down and various pump and valve interlocks, and is mostly independent of the operator.

In order to meet these requirements, a training simulation system needs to simulate some or all of both aspects of the control system, in addition to the hydraulic behaviour of the pipeline. The rest of this paper will give details of how both process control and logic control simulation were added to the Atmos LSIM liquid simulation model produced by Atmos

International.

## SYSTEM DESIGN

### Overview

It was expected that the majority of the people who would configure and use the new control simulation would not be simulation or control experts. As a result it was decided that the method for configuring item control should be clear and as simple as possible. For this reason a graphical configuration system was chosen.

### Architecture

The two main objects that are controlled in pipelines are valves and pumps (compressors for gas pipelines). It was decided to model the valves as being only controlled by the fraction open, and the pumps by the speed. This meant that for each item that was being controlled, a single graphical control diagram with a single output value could be used. Within each of these diagrams graphical control blocks represent the data sources and control logic elements. Drawing lines between these blocks shows the flow of information through the diagram.

In a real pipeline, many of the controllers originally operated in a continuous analog fashion. With the increasing use of computer based control systems, the equivalent control is provided by discrete controls that are updated at a fixed rate. Since the physical pipeline model that the control simulation interacts with runs using a fixed time step, it was decided to run the control simulation at the same time step. The control simulation is updated after each step of the engine, using the current simulation values as inputs. The updated control outputs then provide inputs for the next step of the pipeline simulation.

The control simulation has been designed and implemented using object oriented programming techniques. Each control block has been implemented as a separate object, which takes a set of inputs, performs some action on them and produces an output value. Each control diagram has a list of the control blocks that it contains, and the information flow links between them.

In a real pipeline control system, there is a mixture of two types of value:

- Analog values, such as pressures and pump speeds, which are used in the 'control' aspects of the system.
- Boolean values, which are used in the 'logic' part of the system, such as Emergency Shut Down (ESD) conditions.

Since both types of value can affect a single pipeline item, the control simulation needs to combine them in a single control diagram. To do this, all the values in the diagram are stored as analog values, and the convention that 0 is false and any other value is true is used when a Boolean value is expected.

### Communications

The control simulation needs to be able to communicate with the SCADA systems used for the real pipeline if they are available. For training purposes, it is useful to allow more than one user to interact with the simulation at the same time. In order to do this OPC, an industry standard for data exchange, was chosen for the communications. Since OPC was already used for the run-time inputs to and outputs from the physical pipeline model, this also simplified the integration of the control and pipeline simulation.

It is useful to be able to examine the inputs and outputs of individual blocks within a control diagram. For both training situations, this allows particular parts of the control of the pipeline to be overridden. In addition, during the configuration of a training system, it simplifies the process of 'debugging' the control systems. Generally both the inputs and outputs of the blocks are of interest, and so a separate OPC value is created for each one.

## CONFIGURING CONTROLLERS

Each of the control blocks available for creating a control diagram has a fixed number of inputs and a single output connection. There are two exceptions to this. The source blocks are used to bring values into the control diagram from external sources and so only have an output. Each control diagram also has a single output block, which represents the pipeline item to be controlled, and only has an input connection.

To create a control diagram, the control blocks necessary to create the required behaviour are placed onto the control diagram using a simple drag and drop scheme. Each block has a name, which is set to a default value, but can be replaced with a more meaningful name. Some of the blocks have further fixed parameters that can be set from the editing screen. To send the output value from one control block to the input value of another block, a link is drawn between them.

There are currently 15 types of control block that can be used to create control diagrams. Details of each type of control block are given in Table 1.

## RUNNING CONTROL SIMULATION

The running of the control simulation is split into two parts, a set of initial calculations that are performed once at the start of the simulation, and the actions that are taken for every simulation step.

### Initial Calculations

Since each control block is evaluated once per simulation step, the main issue we have is determining the order in which the control blocks should be considered. It is a sensible requirement that all the blocks that provide input to a given block are evaluated before that block is.

The only problem that we run into is when there is a feedback loop within the control system. Since it is required to only evaluate each control block once for each simulation step, these feedback loops need to be 'broken' in some way. The effect of doing this is that a delay of one timestep is introduced into the feedback loops. Since the PLCs used in a real pipeline usually execute their programs in a similar way, this is acceptable if the time steps used are short enough.

The method used for determining in which order to evaluate the control blocks is based on techniques from graph theory [1]. The idea is to make a directed graph where the nodes of the graph represent the control blocks and the arcs of the graph represent the links between them. The arcs in the graph are directed, and so represent the flow of information between the blocks. The problem now is finding a scheme for assigning numbers to the nodes such that the number at the 'from' end of an arc is always lower than the number at the 'to' end of an arc. Under such a scheme, all the nodes that have arcs into a given node will have lower numbers. This means that if we evaluate the control blocks in an order based on the numbering of the matching graph nodes, all the inputs of a control block will be evaluated before that block is.

Since the source blocks can be set to take a value that is the output of another control block, we cannot consider each control diagram separately. For these source blocks we need to add an extra arc to the graph to represent the dependence of the block on the output of another control block.

The procedure used to number the nodes in the graph is as follows:

- Loop through each node in the graph
  - If the node has not already been assigned a number, and has no inward arcs, start a recursive numbering procedure from that node.

The nodes that have no inward arcs will be those that represent source nodes that take values from outside the control simulation, and so can always be evaluated before the other control blocks.

Each new recursive assignment starts with the node being assigned a number of one. For each node that we consider, we first look at any inward arcs, and call the recursive procedure for the node at the 'from' end of the arc with a number one lower than the current node. Then we look at any outward arcs, and call the recursive procedure for the node at the 'to' end of that arc with a number one higher than the current node. As each arc is considered, it is labelled so that it is not considered again. This has the effect of arbitrarily 'breaking' any loops by assigning a set of numbers that is compatible with the direction of all but one of the arcs in the loop. Other than one arc per loop, all the other arcs now have the number for the 'from' node one lower than the number at the 'to' node

as required.

Note that with this procedure it is possible to assign numbers lower than one to the nodes. To simplify later calculations, we would prefer all the nodes to have a number of 1 or greater. To allow this, in addition to the number, each node considered by the recursive procedure is marked as belonging to a sub set. The nodes in this subset are connected to each other, and nothing else. This means that a fixed value can be added to the number assigned to each node whilst keeping a numbering that is compatible with the arc directions. This allows us to increase the numbers for all nodes in the sub set to make the lowest node number 1 if necessary.

The control blocks are then sorted into arrays for each 'layer' of the control. All the blocks with corresponding nodes that are numbered 1 are placed in the array for the first layer. The second layer contains all the blocks with nodes numbered 2 and so on.

### Each Simulation Step

Since all the blocks in a layer depend only on the blocks in lower layers, we can simply evaluate all the control blocks in layer one, then all the blocks in layer two and continue until all the identified layers have been evaluated.

The procedure for evaluating each block is the same. First, the input OPC values are obtained from the OPC communication system. Then the calculations particular to that type of block are carried out. Finally the output value is written back to the OPC system.

The simpler blocks, such as a minimum or a logic AND only require the current input values to determine the output. For the more complicated blocks, such as the PID block, extra internal storage is required between simulation steps. The object oriented approach means that these internal storage values can be made properties of the implementation of the control block, and the main control simulation does not have to do anything special to deal with them.

### Simulating Abnormal Conditions

The basic operation of the control simulation as described above is sufficient to train an operator in the normal running of the pipeline. However, many companies also wish to train their operators in how to deal with a variety of abnormal operating conditions. Several extensions have been made to the basic system to allow this kind of training.

Most pipelines are controlled via a SCADA system from a central control room. However, at each pumping station or similar location there is usually a control panel that allows an engineer at that location to switch the control from remote to manual. When this happens, the central control room has reduced or no control of that station. In the control simulation

system that has been created, each Control diagram and PID control block can be set into manual override mode. For a control diagram, the manual override mode allows a new value for the output of the whole diagram to be specified. All the control blocks within the control diagram are still evaluated as normal, but the output value is replaced with a value provided by the trainer. The manual override for the PID control block is similar, but only affects the output of the PID block. For PID controls, it is often required that whilst they are in manual override the set point will track the process value. This prevents a sudden change in output when they are switched back into automatic operation. The PID control blocks can be configured to track the set point or continue normal calculations when in manual override mode.

If a trainer requires a lower level method of interfering with the operation of the control system, then it is possible to overwrite the values of the individual inputs or outputs of the control blocks. When a value in the OPC communications system is written to in this way, a flag is set for this value. The flag instructs the control logic simulator not to overwrite the value of that tag when carrying out the normal simulation step updates.

There are also other abnormal conditions that should be simulated in a training system that are not related directly to the control system. These include leaks, which will be covered by the hydraulic pipeline simulation, but are likely to cause subsequent effects in the ESD logic of the pipeline. Most of the other abnormal conditions that are required can be obtained by making changes in the communications layers rather than changing the simulation itself. To simulate a valve that has failed closed; the position for the valve that is sent to the pipeline simulation can be set to closed regardless of the values sent by the trainee. Valves that have failed open or become stuck can be treated in a similar way.

## DRAWBACKS

Although the control simulation system works well for the training simulators that it has been used for, there are a few theoretical drawbacks to the approach that has been described above. This section of the paper will give an overview of these drawbacks and their implications.

- The system can only be used to simulate systems when the underlying pipeline model is running with reasonably short fixed time steps. Depending on the complexity of the simulation and the speed of the underlying engine, this may limit the ability to 'fast forward' a simulation to reach a time of interest.
- Each control diagram can only affect a single item. This makes modeling a sequence of actions that affects several items, such as pump start up) more difficult, since the control has to be spread over several diagrams.
- Each control block is evaluated once in each stimulation step. This introduces a delay of one simulation step into any feedback loops. Practical experience has not noted

any problems with this, however it is worth noting.

- Replacing ideal control with a real control model makes it much easier to create simulation instabilities. For example, poor tuning of a PID control can cause the output of the control to oscillate, which will then affect the pipeline model. Although this is an accurate simulation of the model provided, it is generally undesirable. The tuning of the controllers to avoid this will increase the time and effort needed to create a simulation model.
- Determining the behavior of the control system during a steady state solution can be complicated. The system described above forms a dynamic model of the controllers, but there is no method for determining the steady state solution for the combined pipeline and control simulation. Since the control simulation can be highly non-linear an iterative approach alone is not always able to find a valid solution. The approach that has been taken to overcome this is to disable parts of the control during the steady state by fixing the output values. This allows a steady state for the pipeline model with parts of the control to be found. A dynamic simulation with fixed inputs is run from this reasonable starting point to find a steady state solution for the pipeline and control together.

## EXAMPLES OF CONTROL IN TRAINING SYSTEMS

This section will show examples of the control logic that can be built using the described system. It will begin with a few simple examples and then continue to cover some more complicated examples from the system created for Enterprise Products to train operators for their Cameron Highway Oil Pipeline System (CHOPS) pipeline.

Figure 1 shows the control diagram for a simple PID control of the discharge pressure of a valve. Figure 2 gives an example of how the actual discharge pressure of the valve varies under this sort of control with changes in the set point. Note that the control can overshoot the new set point, depending on the tuning of the PID. This could cause under or over pressures that would not be displayed with ideal control.

The next example, in figure 3, shows a pressure control valve that has upstream pressure control with a setpoint of 250 psi, and downstream pressure control with a setpoint of 130 psi. A select block is used to allow the control to be switched to either upstream, downstream, or the minimum of the two PID outputs. When the minimum is selected then the control will act as upstream pressure control with a downstream pressure control override.

Figure 4 shows a more involved pump control scheme from the CHOPS trainer. In this case, from the client's Cause and Effect diagrams, we have numerous ESD conditions that can

trigger the shutdown of a pump. We also have pump start and stop commands that are sent from the training SCADA to the simulation. Note that a 'valve command' block is used to handle the pump start and stop commands, since the same functionality is required here as for a valve position. This control diagram actually controls the position of the suction valve for the pump. The control logic for the pump includes the part of the start up sequence that introduces a time delay between opening this valve and the pump starting, as shown in figure 54.

The CHOPS pipeline is expected to operate over a wide range of flows. In order to accurately meter the flows, a set of several flow meters in parallel are used. The documentation for the control system gives a series of high flow thresholds at which the valves allowing flow through the next meter run should be opened. There are also a series of low flow thresholds at which the valves for the last open run are closed. Each of these thresholds must be exceeded for a fixed time interval for the valves to be opened or closed. Figure 6 shows how the required control logic can be implemented in the system described above.

## CONCLUSIONS

The method of implementing control simulation described in this paper provides training systems with a more realistic model of the real pipeline response. The method allows the simulation of most pipeline controls and Cause and Effect logic, but sequences of events affecting multiple items are cumbersome to configure.

## REFERENCES

1. "Graph Theory", Gould, Ronald, 1988, Benjamin/Cummings Publishing Company, California
2. "Comparison of PID Control Algorithms" - <http://www.expertune.com/artCE87.html>

## ACKNOWLEDGEMENTS

The authors wish to thank ATMOS International for permitting us to present this paper. The authors also wish to thank Enterprise Products for giving permission to take examples from the training system produced for the CHOPS pipeline.

## TABLES

Block Type	Description
Source	The means of obtaining values from outside the control diagram. This is the only block type that has no input connection. Several potential data sources are available: <ul style="list-style-type: none"> <li>• A fixed value</li> <li>• A value that varies with a given time profile</li> <li>• An output from the pipeline simulation, such as a pressure or flow.</li> <li>• Any output from another control block in the control simulation.</li> <li>• A value read from an external OPC device, such as a training SCADA system</li> </ul>
Output	This block represents the output of the control diagram back to the pipeline item being controlled. This is the only block with no output connection.
Convert	Used to carry out unit conversion or simple scaling and offset of the input value.
Min	The output is the minimum of the two inputs
Max	The output if the maximum of the two inputs
Select	A block with three inputs, two signals and a select. The block operates in three modes: When the select is 1, the output is the first input When the select is 2, the output is the second output When the select is any other value, the output is the minimum of the two input signals.
Compare	Allows comparison of two input values. Is configured with the output values for the cases when the first input is less than, equal to or greater than the second output.
PID	The main type of control used when a setpoint is to be achieved. Allows the P,I and D values to be modified during the simulation and so has 5 inputs. See [2] for more details of PID controllers.
Limit	Is configured with a trigger value and set to be a low or high limit. As a low limit, it outputs false (0) when the input is above the trigger value, and true (1) if the input is less than then trigger value. Similarly for the high limit
AND	Logical AND control. Since control block have a fixed number of inputs there are versions of this with 2 and 3 input connections.
OR	Logical OR control. Since control block have a fixed number of inputs there are versions of this with 2 and 3 input connections.
NOT	Logical NOT control.
Timer	A time delay for Boolean signals. Is configured with a time delay in seconds. Originally the output is false, when the input becomes true, a counter is started, if the input stays true for the configured delay, the output becomes true. Can be configured so that the output is true for a single time step, or stays true until the input becomes false.
Sequence	A simple counter control. Is configured with a maximum and minimum output value. There are two inputs to increment and decrement the output value. If the increment input is true the output value will be one higher than it was in the last simulation step. If the decrement input is true the output will be one lower than in the last simulation step.
Valve control	This is a specialised control created to simulate a repeated piece of logic required in a training system. The two inputs are a valve open and close signal, the output is the required position of the valve. (The training SCADA required that the open signal be reset when the close signal is sent and vice versa, which is also done by this control)

**Table 1 – Types of Control Block**

# FIGURES

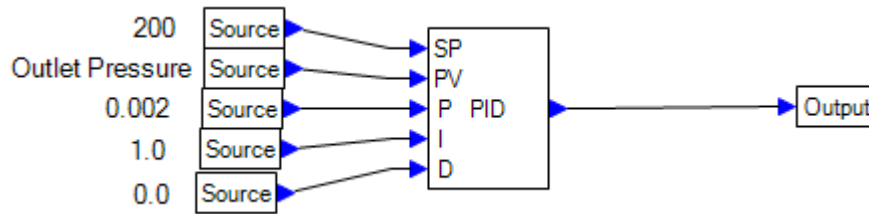


Figure 1 –Simple pressure control

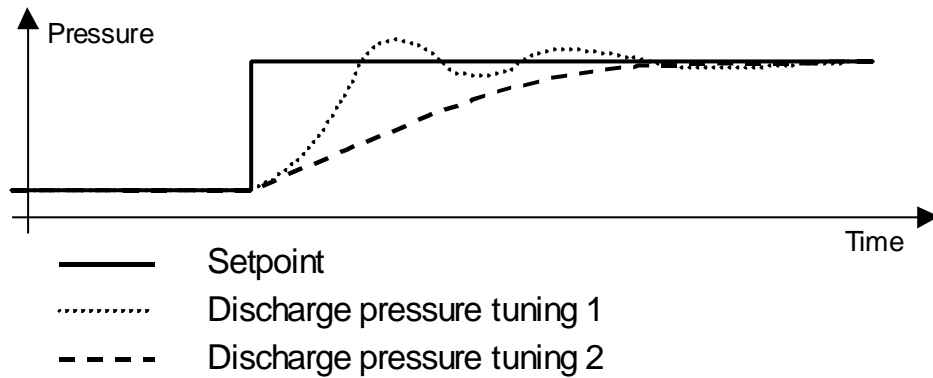


Figure 2 –Response of PID control to setpoint change

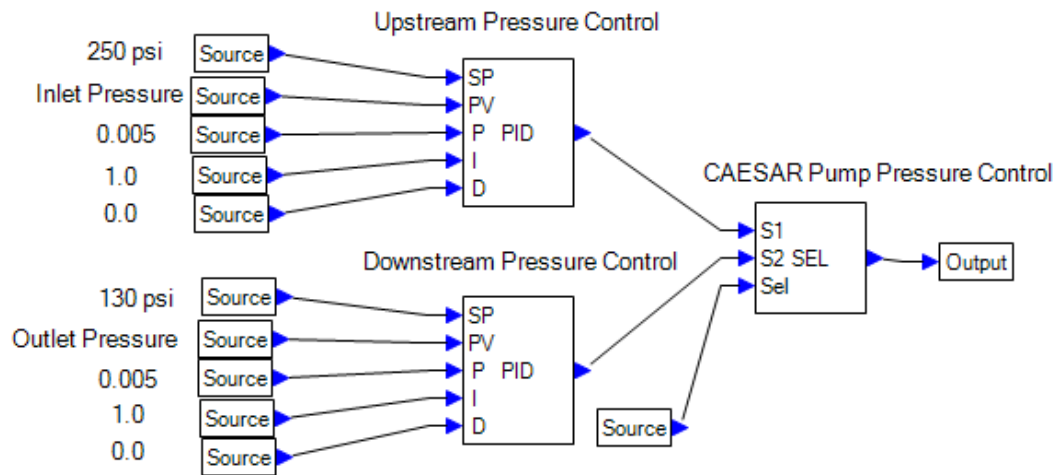


Figure 3 – Pressure control valve diagram

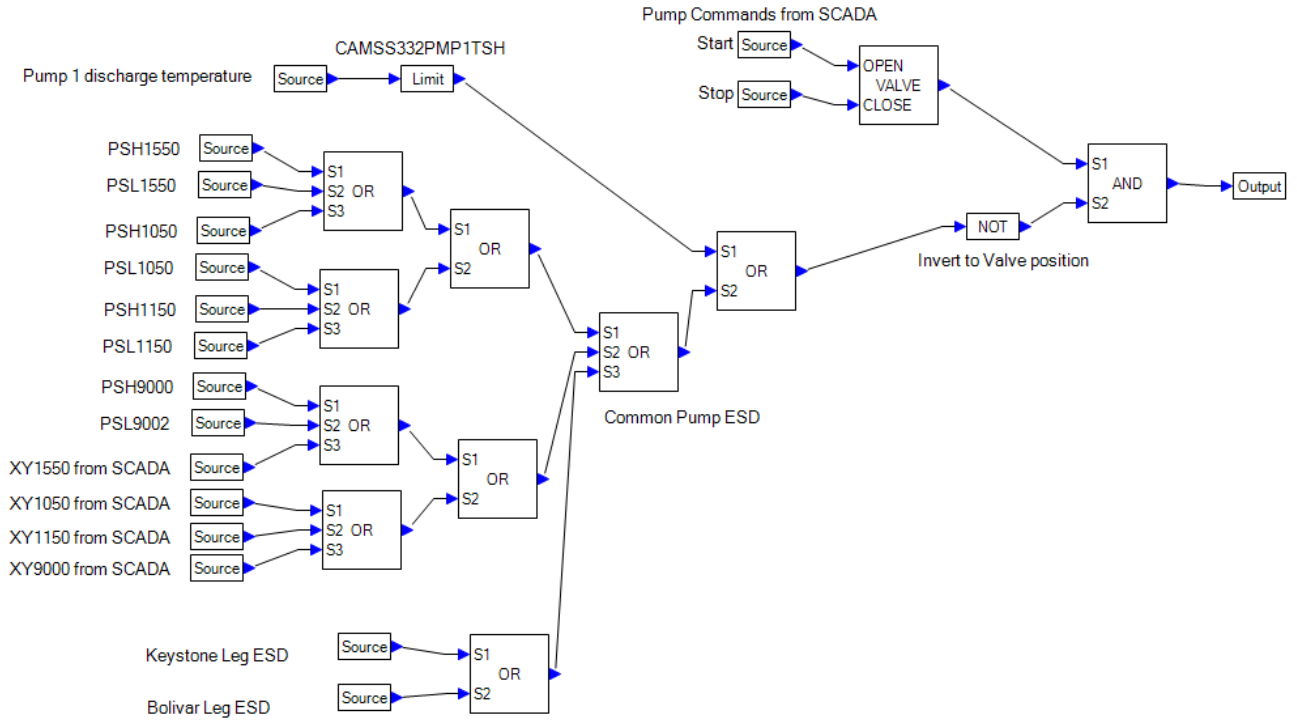


Figure 4 – Pump startup control with ESD logic

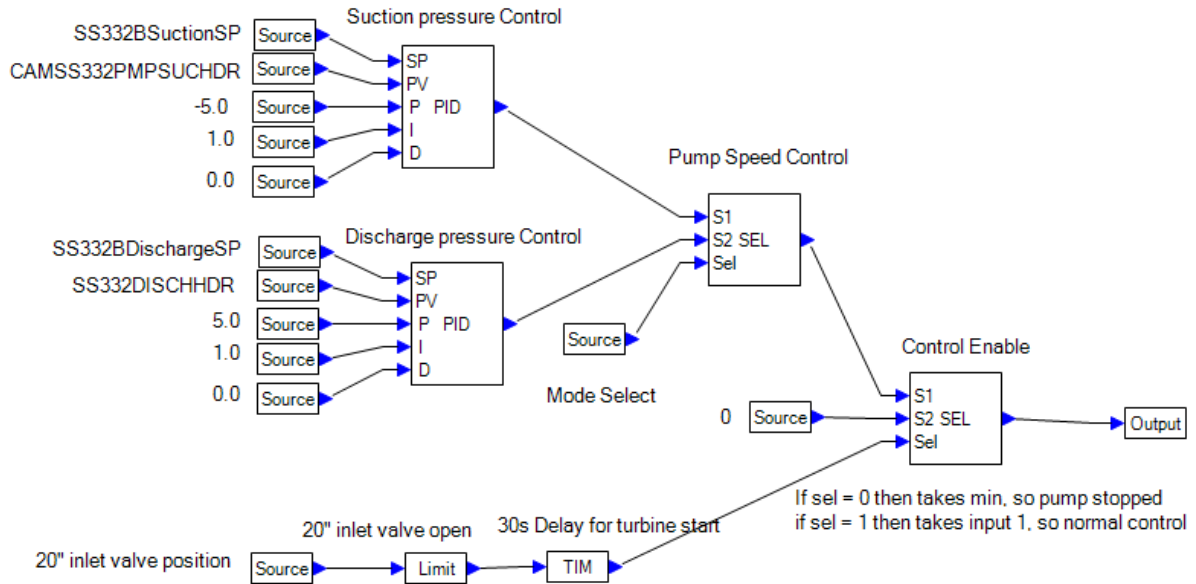


Figure 5 – Pump speed control with sequence delay

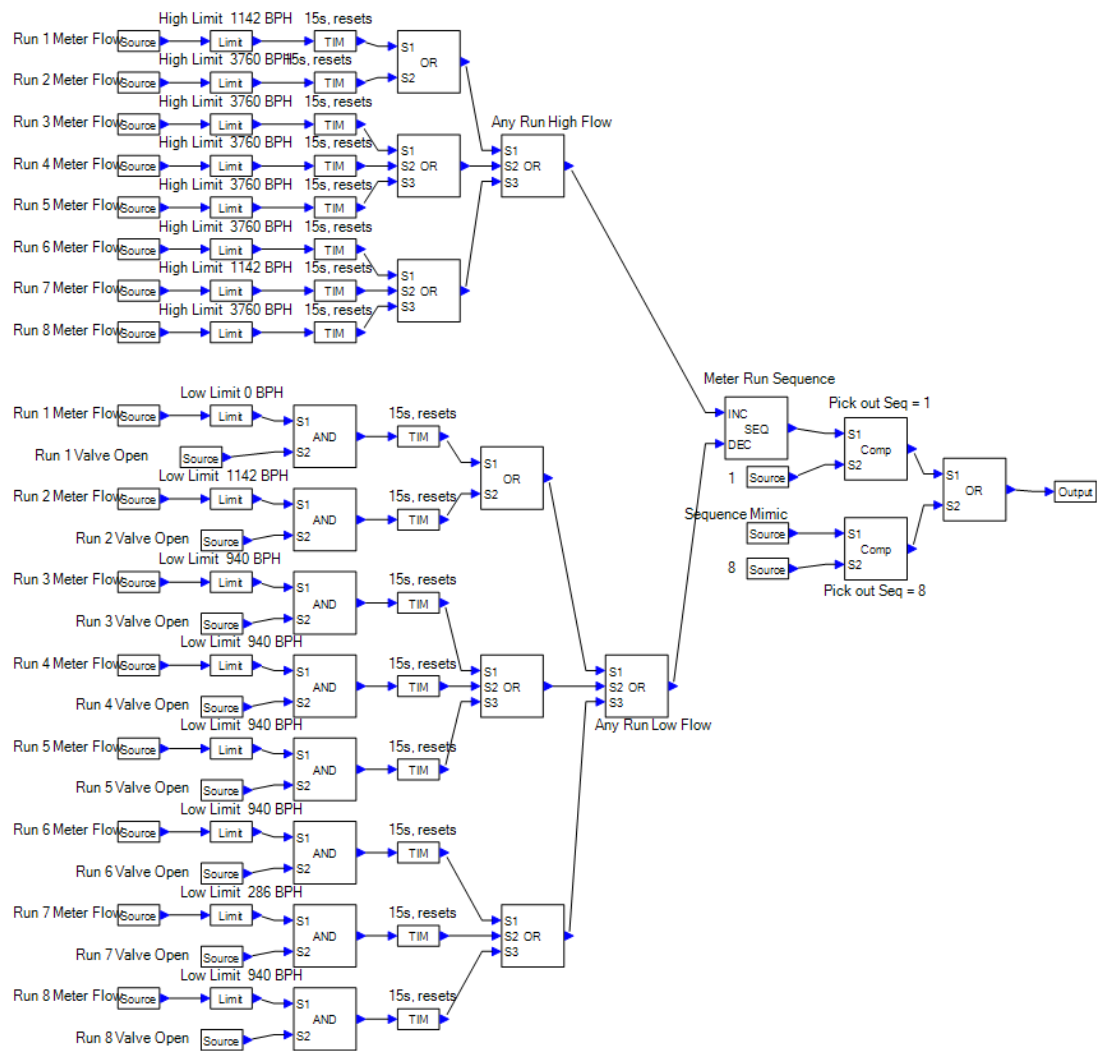


Figure 6 – Meter run switching logic diagram