Design and construction of a multi-level control system using the Compact RIO controller and LabVIEW

Kane Blay, 19808328

2010

A report submitted to the School of Engineering and Energy, Murdoch University in partial fulfilment of the requirements for the degree of Bachelor of Engineering.

Supervisor: Associate Professor Graeme Cole
1 ABSTRACT

This report outlines the design and construction of a multi-level control system for the Murdoch University Universal Water System (UWS). The UWS can be re-configured (using a series of manual valves) to behave as a single integrated system, three individual systems, or any 2-1 combination of the systems. Due to the high interconnectivity between the tanks the system displays a high level of control loop interaction. This is desirable as it allows students the opportunity to test advanced control models (both SISO and MIMO).

The project is now complete, with the UWS control system in place and successfully commissioned and tested. The control system is designed to be used by operators with varying levels of abilities. The client program template can be used as a standard operator interface, with default control loop selection and tuning. The client template can also be used to select alternate control loops and tuning parameters. Advanced users can even add custom controller models to the system by use of the provided code modules. Alternatively, users can connect to the system via OPC and construct controllers using Excel or MATLAB interfaces.

This report also covers the extensive documentation and user guides that have been generated for the UWS project.
2 ACKNOWLEDGEMENTS

For their support and assistance throughout this project, the author would like to acknowledge and thank the following Murdoch staff:

Associate Professor Graeme R Cole, Lecturer
Dr Gareth Lee, Lecturer
Professor Parisa A Bahri, Head of School
John Boulton, Pilot Plant Maintenance Manager
Will Stirling, Murdoch Engineering IT Officer
Dr Linh Vu, Lecturer

And on a personal note, the author would like to thank his family and friends for their support and distractions, his fellow students for keeping the degree interesting and for the indoor football, and the process control team at Newmont Boddington Gold, for giving him such flexible hours in order to complete this project.
3 TERMS AND ABBREVIATIONS

- **Pilot Plant**: Murdoch University Pilot Plant at the South Street Campus
- **Solenoid Valve**: Valve controlled by an electrical solenoid, can only be OPEN or CLOSED
- **Proportional Valve (Flow Control Valve)**: Valve able to be opened to a range of positions
- **Interlocks**: One or more conditions that must be satisfied in order for an event to occur
- **Controller**: Compact RIO Controller used for the Universal Water System
- **Server**: Windows machine running LabVIEW 2009 and associated Shared Variable Engine
- **Operator Machine**: Remote computer system that can be used to control the Universal Water System via the Server
- **UWS**: Universal Water System
- **NI**: National Instruments, the company which produces the hardware and the software to be used in the UWS.
- **LabVIEW**: Laboratory Virtual Instrumentation Engineering Workbench, the software package provided by National Instruments, for this project LabVIEW 2009 is used
- **cRIO**: National Instruments Compact Reconfigurable Input and Output
- **VI**: Virtual Instrument, a program coded in LabVIEW
- **PD**: Positive Displacement
- **IO**: Input / Output
- **OPC**: Object linking and embedding for Process Control, a commonly used protocol for the hosting and sharing of process variables
- **PID**: Proportional Integral Derivative Control
- **P&ID**: Piping and Instrumentation Diagrams
- **VSD/VFD**: Variable Speed Drive (Variable Frequency Drive)
- **DOL**: Direct On-Line (Motors that are only either ON or OFF)
- **MAX**: National Instrument’s Measurement and Automation Explorer (program)
- **SISO**: Single Input Single Output, typically with reference to control loops.
- **MIMO**: Multiple Input Multiple Output, typically with reference to control loops.
4 CONTENTS

1 ABSTRACT.................................................................................................................. 1

2 ACKNOWLEDGEMENTS ............................................................................................ 2

3 TERMS AND ABBREVIATIONS ................................................................................. 3

4 CONTENTS .................................................................................................................. 4

5 TABLE OF FIGURES .................................................................................................. 6

6 BACKGROUND ........................................................................................................... 7

7 EARLIER WORK ON THE UWS ............................................................................. 11

8 PROJECT OBJECTIVES ............................................................................................ 14

  8.1 MULTIPLE LEVELS OF INTERACTION ................................................................. 14

  8.2 CONTROL LOOP FLEXIBILITY ........................................................................... 17

  8.3 DOCUMENTATION AND USER GUIDES ......................................................... 17

  8.4 PROJECT COMPONENTS .................................................................................... 18

9 SYSTEM DESIGN ....................................................................................................... 19

  9.1 EQUIPMENT BEHAVIOUR .................................................................................. 19

  9.2 PLATFORM & VARIABLE PASSING ................................................................... 21

  9.3 DATA STRUCTURE ............................................................................................. 22

  9.4 SYSTEM LIBRARY .............................................................................................. 25

  9.5 cRIO AND SERVER INTERACTION .................................................................... 25

  9.6 INTERLOCKS ........................................................................................................ 27

  9.7 SECURITY ............................................................................................................. 29

10 THE COMPACTRI O CONTROLLER ....................................................................... 30

  10.1 CONFIGURATION AND HARDWARE ................................................................. 30

  10.2 PROGRAMMING THE COMPACTRI O ............................................................... 31

  10.3 TASKS OF THE COMPACTRI O PROGRAM .................................................... 33

  10.4 SHARED VARIABLE LIBRARY .......................................................................... 34

  10.5 SETTING THE cRIO TIME .............................................................................. 36

  10.6 PROGRAM EXECUTION ..................................................................................... 37

  10.7 cRIO CODE ......................................................................................................... 38

  10.8 COMPILING THE cRIO CODE ......................................................................... 43

  10.9 TESTING THE cRIO CODE ............................................................................... 43

11 THE UWS SERVER .................................................................................................. 48
11.1 BASIC CONFIGURATION ................................................................. 48
11.2 UWS SERVER Tasks .................................................................... 49
11.3 THE NEED FOR THE UWS SERVER ........................................... 50
11.4 LIBRARY BEHAVIOUR ................................................................. 51
11.5 LOGGING AND EXTRACTION ..................................................... 53
11.6 TESTING PAGE ............................................................................ 54
11.7 UWS SECURITY ............................................................................ 54
11.8 SYSTEM ADMINISTRATION .......................................................... 55
11.9 OPC CONNECTIVITY ................................................................. 56

12 CLIENT SOFTWARE ........................................................................... 57
12.1 UWS CLIENT BASICS ................................................................. 57
12.2 LOOK AND FEEL (ASM COMPLIANCE) ....................................... 58
12.3 SIMPLE ALARMING .................................................................... 60
12.4 UWS CLIENT CODE ................................................................... 61
12.5 USER INTERFACE ....................................................................... 66

13 TESTING AND OPERATING THE UWS .......................................... 72
13.1 TESTS PERFORMED ................................................................. 72
13.2 ALTERNATE CONFIGURATIONS ............................................... 74

14 DOCUMENTATION ......................................................................... 76
14.1 DRAWINGS AND DIAGRAMS (0xxx) ............................................ 77
14.2 DATASHEETS AND REFERENCE (1xxx) ...................................... 77
14.3 UWS SOFTWARE (2xxx) .............................................................. 77
14.4 STANDARD OPERATING PROCEDURES (3xxx) ......................... 78
14.5 FUNCTIONAL DESCRIPTIONS (4xxx) .......................................... 78
14.6 CHAD WEBSITE ...................................................................... 79

15 PROJECT CLOSE-OUT AND FUTURE DIRECTIONS ....................... 80

16 APPENDICES .................................................................................. 81

17 REFERENCES .................................................................................. 82
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6-1</td>
<td>Universal Water System (1)</td>
<td>9</td>
</tr>
<tr>
<td>6-2</td>
<td>Murdoch UWS P&amp;ID</td>
<td>10</td>
</tr>
<tr>
<td>7-1</td>
<td>Previous Design</td>
<td>11</td>
</tr>
<tr>
<td>8-1</td>
<td>Proposed ‘Levels’ of Interaction</td>
<td>14</td>
</tr>
<tr>
<td>8-2</td>
<td>One Pump, Multiple Variables</td>
<td>17</td>
</tr>
<tr>
<td>8-3</td>
<td>Project Scope</td>
<td>18</td>
</tr>
<tr>
<td>9-1</td>
<td>Data Movement Between cRIO and Server</td>
<td>26</td>
</tr>
<tr>
<td>9-2</td>
<td>Pump 01 System</td>
<td>27</td>
</tr>
<tr>
<td>10-1</td>
<td>NI cRIO-9014 Real-Time Controller (3)</td>
<td>30</td>
</tr>
<tr>
<td>10-2</td>
<td>Reconfigurable FPGA Chassis(3)</td>
<td>30</td>
</tr>
<tr>
<td>10-3</td>
<td>Tasks carried out by the cRIO Controller</td>
<td>33</td>
</tr>
<tr>
<td>10-4</td>
<td>Main cRIO Code</td>
<td>37</td>
</tr>
<tr>
<td>10-5</td>
<td>Input Processing</td>
<td>38</td>
</tr>
<tr>
<td>10-6</td>
<td>Valve Interlock</td>
<td>39</td>
</tr>
<tr>
<td>10-7</td>
<td>Pump Logic Code (Showing Sub-VIs)</td>
<td>41</td>
</tr>
<tr>
<td>10-8</td>
<td>VSD Pump VI</td>
<td>42</td>
</tr>
<tr>
<td>10-9</td>
<td>Monitor.VI Testing Interface</td>
<td>44</td>
</tr>
<tr>
<td>10-10</td>
<td>cRIO Performance</td>
<td>46</td>
</tr>
<tr>
<td>11-1</td>
<td>Tasks carried out by the UWS Server</td>
<td>49</td>
</tr>
<tr>
<td>11-2</td>
<td>System without server, multiple cRIO connections</td>
<td>50</td>
</tr>
<tr>
<td>11-3</td>
<td>System With Server, Single cRIO Connection</td>
<td>51</td>
</tr>
<tr>
<td>11-4</td>
<td>PU01 Variable Passing</td>
<td>52</td>
</tr>
<tr>
<td>11-5</td>
<td>UWS Database Administration</td>
<td>53</td>
</tr>
<tr>
<td>12-1</td>
<td>System Overview Page</td>
<td>58</td>
</tr>
<tr>
<td>12-2</td>
<td>Pump Icon in Visio</td>
<td>59</td>
</tr>
<tr>
<td>12-3</td>
<td>Examples of Alarms in the UWS</td>
<td>61</td>
</tr>
<tr>
<td>12-4</td>
<td>Pump Display</td>
<td>63</td>
</tr>
<tr>
<td>12-5</td>
<td>Interlocked Pump, Showing Tip Strip</td>
<td>64</td>
</tr>
<tr>
<td>12-6</td>
<td>Pages of the UWS Client Interface</td>
<td>66</td>
</tr>
<tr>
<td>12-7</td>
<td>PID Control</td>
<td>67</td>
</tr>
<tr>
<td>12-8</td>
<td>PV Selection</td>
<td>68</td>
</tr>
<tr>
<td>12-9</td>
<td>Trend Page</td>
<td>69</td>
</tr>
<tr>
<td>12-10</td>
<td>Change Current Group</td>
<td>69</td>
</tr>
<tr>
<td>12-11</td>
<td>Data Extraction Interface</td>
<td>71</td>
</tr>
<tr>
<td>13-1</td>
<td>Trend showing Tank Levels</td>
<td>74</td>
</tr>
<tr>
<td>13-2</td>
<td>UWS as a Continuous System</td>
<td>75</td>
</tr>
<tr>
<td>13-3</td>
<td>UWS Configured as three Discrete Systems</td>
<td>75</td>
</tr>
<tr>
<td>14-1</td>
<td>Project Documentation</td>
<td>76</td>
</tr>
</tbody>
</table>
6 BACKGROUND

Murdoch University has a strong reputation for giving its students a high level of hands-on experience as part of both the Instrumentation and Control, and Industrial Computer Systems Engineering Degrees. Students from Second Year onwards have the opportunity to apply their knowledge in a series of laboratories and ultimately in the Murdoch Pilot Plant, using industrial grade equipment and instruments. This exposure to real-world equipment and processes allows students to apply their theoretical knowledge. Currently the Murdoch University operates a single Pilot Plant modeled on the grinding-digestion-precipitation phases of the Bayer Process for Alumina Refining.

The Bayer Process plant has been a popular and very useful part of the learning process at Murdoch. The existing plant is used in both third and fourth year units from both the Instrumentation and Control degree and the Industrial Computer Systems degree. Students initially operate the plant at a high-level from preconfigured operator screens, the level of complexity increases over time and students then move on to constructing their own controllers/expert systems to implement on the plant. Students also delve into the inner workings of the equipment of the Pilot Plant, making modifications to the software and underlying hardware.

With increasing student numbers, the pilot plant is in ever-increasing demand and a second process has been constructed to spread the load. This second process is the Universal Water System (UWS). The Universal Water System has been designed to give students hands-on access to a multivariable, complicated system. As with the Pilot Plant, students can gain familiarity with the system using pre-existing controllers, then move on to implementing more advanced control schemes. The basic control will be carried out by regular PID controllers, but more advanced
schemes can be implemented. These advanced control schemes include feed-forward control, adaptive control, decoupling control, fuzzy logic control, model-based control schemes such as generic model control, and model predictive control schemes such as dynamic matrix control.

To allow for these requirements a large amount of flexibility must be included in the programmatic structure of the system. Students should be able to use a range of different systems to implement their controllers, including Mathworks’ MATLAB, National Instruments LabVIEW, and Microsoft Office Excel.

The hardware in the UWS consists of 6 large tanks, 5 positive displacement pumps, 3 centrifugal pumps and 1 bellows pump, these are interconnected by a series of pipes. In the pipe work there are 4 proportional control valves, 13 solenoid control valves and 20 manual valves. A range of analogue level and flow meters and digital (float-type) high and low level indicators are used for each tank. The instrumentation and actuators are not all of the same type, with many types of flow meter, level measurement devices and pumps/valves. This is done to give students exposure to a wider variety of styles and types of equipment that are used in industry.

The UWS was located in the Murdoch University Pilot Plant for the duration of this project. During November 2010, the UWS was moved to the new Murdoch Engineering building on the Northern side of the South Street campus. Figure 6-1 shows the UWS in place at the old South Street Pilot Plant. The important pieces of equipment have been highlighted in order to show the relative locations of each.
The Piping and Instrumentation Drawing for the UWS is also included as Figure 6-2. This drawing was originally created during Rick Janosz’ thesis (ref (1)), and has since been updated to match the current state of the system. A higher resolution version of this P&ID is included in the project documentation as Appendix 0001:

UWS-0000  Drawings and Diagrams\UWS-0001  UWS P&ID.pdf
FIGURE 6-2 MURDOCH UWS P&ID

Legend:

FM Flow Meter
FW Flow Control Valve
LSH High Level Switch
LSLL Low Level Switch
LT Level Transmitter
LIT Level Indicating Transmitter
PJ Pump
BV Solenoid Valve
V Manual Operated Ball Valve

MURDOCH UNIVERSITY
UNIVERSAL WATER SYSTEM

Universal Water System Piping and Instrument Drawing

E  Document Created  B Jansen, J.Kue  12/5/2013
F  Name/Document Updated  H Bill  09/10/2010

Scale: Sheet 1 of 1  Date: 09/10/2010

P 10
7 EARLIER WORK ON THE UWS

Past students have been responsible for the early design and construction of the UWS. At the beginning of this project, the hardware was in place and the instrumentation, actuators and motors wired and tested. The most recent students to work on the UWS were Jamie Kurz and Rick Janosz. These two students worked on the Pilot Plant simultaneously as a part of their respective Theses. Both of these Thesis papers, and the technical drawings generated during their projects, were useful resources throughout the course of this project. The Paper by Janosz (1) focuses on the hardware installation and wiring, Kurz’s paper (2) covers the coding, CompactRIO interface and a simple user template. Jamie and Rick worked together during the installation and commissioning of the Hardware-cRIO interface and both are responsible for the state the system was in at the start of the project. This state can be represented graphically as shown in Figure 7-1.

The instrumentation and actuators were installed and commissioned. The IO for the cRIO controller was configured and commissioned. The cRIO was effectively being used as a ‘remote IO’ module and passing data back and forth to the Server using the NI ‘Shared Variable’ engine. The Server machine ran a VI named ‘UWS DSC SCAN’ which monitored these variables on the cRIO, logged them, and performed level and motor-protection interlocking. There was a ‘Student Template’ LabVIEW VI which allowed students to read and write data to the cRIO (through the ‘UWS DSC SCAN’ vi). It was intended to use this as the building point for the current project, extending the client interface.
The projects completed by Kurz and Janosz were concerned primarily with commissioning the hardware and cRIO interface, rather than designing the control system. The control system in place was an extension of the simple test-VIs used when commissioning the hardware of the UWS. This system was constructed with a 'ground-up' approach, adding extras as the need became apparent. This led to a lack of coherence and an overly complicated software system. It was clear that a 'top-down' design approach would be needed to ensure consistency and standardize the interface between elements in the system.

Major problems existed with the previous system's safety mechanisms and client interface. The code that controlled the safety systems for the UWS (pump and valve interlocking) was executing on the host machine (Windows XP based) and not on the CompactRIO device. This meant that if communications dropped out between the server and the cRIO, or if the program was stopped on the server, the UWS hardware would continue on in its current state, without any safety systems in place. In order to rectify this, the code would be embedded in the CompactRIO, which could then continue to ensure the safety of the process and equipment entirely unattended.

The client template of the previous UWS control system was configured to use NI shared variables. This client template however, was configured on the UWS Server and used local shared variables instead of using remote shared variables. This meant that the client would not work unless it was running on the UWS Server, with the UWS DSC SCAN program also running on the server. So while the Hardware side of the project was operational the control system software needed a complete re-design.

Another early issue discovered with the cRIO system, was that the communication link with the cRIO controller was unreliable. The connection from the server to the
cRIO would regularly drop-out, and after these drop-outs, the cRIO controller required a restart before it could begin communicating again. In order to find the root-cause of the communication problem the server machine was shifted down to a position on the plant floor and connected to the same switch as the cRIO. This did not resolve the issue. Further checking into the IP address of the cRIO controller provided a solution. The ‘Static’ IP configured for the cRIO was actually an IP out of Murdoch’s ‘Dynamic’ IP block.

Most student computers at Murdoch University are randomly assigned an IP out of the Murdoch Dynamic IP block when they connect to the network. If a machine connects and is assigned the same IP as the CompactRIO, an IP conflict is created. This IP conflict means that packets addressed to the CompactRIO could be mistakenly sent to the other machine, and vice-versa. This problem was intermittent as the IP was not always in use by a Windows machine on the network. When there was no other machine assigned the same IP as the cRIO, the problem did not occur.

This issue was solved with the help of Will Stirling, who noticed the IP was in the incorrect range and assigned an IP from the School of Engineering’s static IP range. The cRIO system IP is now 134.115.133.141. The TCP/IP configuration details for the UWS control system are listed in the project documentation as Appendix 1006:

UWS-1000 Datasheets and Reference\UWS-1006 Network Details.pdf.
8 PROJECT OBJECTIVES

The scope of work for this project was to design and implement a robust and flexible control system, which allows for the interaction by students of varying ability levels. A National Instruments CompactRIO controller has been used in the project and the underlying software constructed in the NI LabVIEW coding environment. The overall system will have a level of complexity greater than that present in the Instrumentation and Control (IC) Laboratory used in ENG267 and ENG304, yet less complex than a full-blown DCS system, such as the Honeywell Experion system used in the Bayer Process Pilot Plant.

8.1 MULTIPLE LEVELS OF INTERACTION

The conceptual ‘levels’ of interaction by students range from a very high (superficial) level of interaction required when simply running the plant, right down to the very low level interaction required when adding or removing equipment from the Plant. These levels are presented in Figure 8-1. Students would start at the top level and progressively delve deeper into the workings of the UWS.
8.1.1 Zeroth Level

At the lowest level, the student would be interacting directly with the hardware and electrical components of the UWS. This includes things such as adding/replacing instruments or actuators, the wiring and termination inside the cabinet, the physical connections to the cRIO controller and its modules. This is the most technically complicated of student interaction and would require knowledge of all of the upper levels. The bulk of the work to complete this level was completed by previous students and reported on in references (2) and (1).

8.1.2 First Level

Level one consists of all of the code and procedures carried out by the cRIO embedded controller. This includes Interlocking, Signal Processing, Motor Protection and Input/Output Signal Conversion as required by the different equipment. For safety purposes, this level will be restricted and not accessible to students interacting at higher levels. Users interacting with the UWS at this level will need to have strong LabVIEW coding experience and knowledge of low level networking and hardware topics. Examples of interactions at this level would be changing the I/O code to handle new instruments, re-scaling motor outputs, or adding/modifying interlocks in the system.

8.1.3 Second Level

At the second level, the complexity of interaction is designed to be approximately on par with that currently experienced by Instrumentation and Control students in units ENG346 and ENG420. The server machine hosts a shared library of all the current variables in the system. The server passes variables back and forth to the cRIO controller. The variables on the server reside in a LabVIEW shared variable Library, which is a background process making use of the LabVIEW shared variable
engine. This variable engine is configured to communicate with clients using both the NI Shared Variable protocol, and the more universal OPC protocol.

The ability to connect any OPC compliant software package to the system means that when implementing advanced control schemes, the user can choose the most appropriate package. For example, the user could choose to implement a complicated model-based controller, such as Generic Model Controller, using Mathworks’ MATLAB software, or the user could choose to implement a matrix-heavy controller such as a Dynamic Model Controller in Microsoft Excel rather than LabVIEW.

8.1.4 THIRD LEVEL

Interaction at level three is the simplest¹ form of interaction with the UWS System. At this high level, a set of pre-built, static operator screens provide an operator interface for the UWS System. These operator screens are designed to follow ASM guidelines and include a simple area-specific alarm system. If desired, more advanced users can customize the provided UWS Client program in order to add functionality to the client, including more advanced controllers.

¹ Simplest from a coding and technical standpoint, the system itself is designed to be difficult to control.
8.2 CONTROL LOOP FLEXIBILITY

In a highly interactive system such as this, a single piece of equipment could be used to control multiple variables in a system. A simple example of this is shown in Figure 8-2.

In this example, the pump could be used to control the level in the source tank, the level in the output tank, or the flow rate through the flow meter. With this in mind, the UWS Client interface allows for the changing of control loop selection ‘on the fly’. The example shown in Figure 8-2 is taken directly from the real UWS system, PU09 and PU08 have this configuration.

8.3 DOCUMENTATION AND USER GUIDES

As this project would create a new control system, an important part of the project was the generation of the supporting documentation. Technical drawings, data reference sheets, standard procedures, functional descriptions, and user guides are all required to support the UWS control system. Additionally, the ‘Chad’ website for the UWS was to be updated to allow students easy access to the documentation.
8.4 PROJECT COMPONENTS

In order to understand the full scope of this project, and allow for the initial design phases, a mind-map diagram was generated using the (open source) software package Xmind (http://www.xmind.net/). This project scope diagram is shown in Figure 8-3, which presents the detail required from each of the ‘levels’ of design and other project elements to be completed.

FIGURE 8-3 PROJECT SCOPE

Each of these major elements constitutes a different phase of the project. The project was broken down into five phases: Standards and Design; cRIO Controller (Level 1); UWS Server (Level 2); Client Interface (Level 3); and the final Phase is to Document the System and update the Chad website. For the scope of this project, it is only necessary to test and prove the OPC connectivity. A master’s student Kan Sumontha is constructing an Excel plugin for OPC integration, as well as constructing proof of concept controllers for MATLAB.
9 SYSTEM DESIGN

The design phase was one of the most important parts of the project. Over a period of two weeks, the structure for the system was mapped out and the behaviour of each element identified. During this time the bulk of the data reference documentation was created. By following the top-down design methodology, it was easy to keep consistency across the project.

All of the data reference sheets created in this phase are included in the project documentation as Appendix 1000:

UWS-1000 Datasheets and Reference

9.1 EQUIPMENT BEHAVIOUR

Before starting to populate the variable lists for the system, it was necessary to identify how the equipment was to behave. Except the MacNaught flow meters, all the field instruments were calibrated to return a 4-20mA signal, which is then scaled back to a % level reading (for level instruments) or a flowrate in litres per minute (flow instruments). This calibration was done by Janosz and Kurz in the first half of 2010. These calibrations were re-tested and it was found that only tank 03 required re-calibration. The problems with tank 03’s level measurement will be discussed in Section 11. The MacNaught flow meters returned a pulse to the control system, as they are positive displacement meters there are exactly 36 pulses per litre. Some simple calculations were required to convert these pulses into a L/m flow rate.

The pumps and solenoid valves required slightly more investigation. Steps needed to be taken in order to protect the system and the users. The system had to be designed with this in mind, and have the ability to stop pumps and close valves if
required. For the valves, the interlocking is simple, the user sends an open request to the valve (MV) and if the tank that the valve is discharging into is not full the valve will open. If the discharge tank fills up while the valve is open, the system will close the valve, regardless of what the user is requesting.

Pumps have a more complicated system of interlocks, as they have to do more than preventing tank overflows. A full explanation of the interlocks for this system is available in Section 9.62. For pumps, it is not acceptable for the user to be able to hold the run request active. In these systems, if a pump is tripped due to an interlock, the pump will re-start as soon as the interlock clears. This could lead to an unattended pump continually starting in an unsafe condition.

In order to avoid this situation, the pumps will have a START and a STOP command. If the pump is safe and ready to start, a rising edge on the start command will start the pump. This will send a run request (MV) to the field. At any time, if an interlock condition becomes unhealthy, or a stop command is received from the user, the pump will be stopped. The pump control block will also return a TRIPPED tag to the user. This is TRUE if the pump has been stopped by an interlock, and will be used for the area alarming.

2 The interlock listing is also available in Appendix 1005:
UWS-1000 Datasheets and Reference\UWS-1005 Pump Interlocks.pdf
9.2 Platform & Variable Passing

As the CompactRIO used in the UWS control system is coded using the National Instruments LabVIEW (Real-Time) package, there is a range of possible methods to pass data between elements in the system. It is possible to code an entirely custom variable passing and database system in LabVIEW using the TCP/IP and database VIs. This is unnecessary for this project, as LabVIEW already has a module specifically designed for these objectives.

The LabVIEW Data-logging and Supervisory Control (DSC) module is designed specifically to handle the sharing of information between distributed systems. From the DSC user manual:\n
\[\text{The LabVIEW Datalogging and Supervisory Control (DSC) Module adds features and capabilities to LabVIEW to help you create monitoring applications and data logging applications. The DSC Module provides solutions for supervisory control of a wide variety of distributed systems using the flexibility of graphical LabVIEW programming.}\]

Specifically, the DSC module allows for the use of National Instruments Shared Variables, and the Shared Variable Engine. This engine runs on the Host device and shares data with any other devices that request it. The shared variable engine uses the National Instruments Publish-Subscribe protocol, where a client device ‘subscribes’ to the variable on the host device, the host then updates the client whenever the value of the variable changes.

Using the DSC module for the variable passing also allows the use of the Citadel data-logging database. The citadel database is the database configured by the DSC

---

3 The DSC user manual is published by National Instruments, and is available as Appendix 1999: UWS-1000 Datasheets and Reference; UWS-1999 User Manuals and Guides
module, and allows for the simple logging of historical data in the system. Using the DSC module allows for a simple integrated system of data hosting and logging, all configuration for the shared variable’s and logging is done through the LabVIEW Library editor.

### 9.3 Data Structure

The next consideration of the design phase was to itemise what pieces of data would be available for each piece of equipment. Each piece of data would follow the naming structure XXXX.YYY, where XXXX is the name of the equipment and YYY denotes the nature of the data. For example, the ‘Ready to Start’ (RDY) input for the Pump PU01 would be identified as *PU01.RDY*, while the ‘Interlock’ (IL) status of the pump would have the tag *PU01.IL*. By standardizing the naming structure in this fashion it is easy to identify from the tag name which piece of equipment it relates to and what data is represented. An example of this data structure is included in Table 9-1, showing the data points related to the Bellows Pump (PU07).

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag</th>
<th>Data Type</th>
<th>Writeable</th>
<th>OPC Desc</th>
<th>Description.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equipment Identifier</td>
<td>PU07</td>
<td>Bellows</td>
<td>-</td>
<td>-</td>
<td>Unique identifier for this piece of equipment.</td>
</tr>
<tr>
<td>Start Pulse</td>
<td>PU07.STT</td>
<td>Boolean</td>
<td>TRUE</td>
<td>PU07 Start Pulse</td>
<td>If all interlocks are clear the equipments MV goes true on the rising edge from this bit.</td>
</tr>
<tr>
<td>Stop Pulse</td>
<td>PU07.STP</td>
<td>Boolean</td>
<td>TRUE</td>
<td>PU07 Stop Signal</td>
<td>Clears the MV for this equipment whenever this is TRUE. Unlike STT this is not only on a rising edge.</td>
</tr>
<tr>
<td>Run Request</td>
<td>PU07.MV</td>
<td>Boolean</td>
<td>FALSE</td>
<td>PU07 Run Request</td>
<td>‘Run’ Command output from the controller to the motor.</td>
</tr>
<tr>
<td>Interlock</td>
<td>PU07.IL</td>
<td>32 bit</td>
<td>FALSE</td>
<td>PU07 Interlock</td>
<td>Each interlock (Boolean) makes up a bit of this integer. If it is 0 all the interlocks are clear, otherwise an interlock is active.</td>
</tr>
<tr>
<td>Speed</td>
<td>PU07.SY</td>
<td>Double</td>
<td>TRUE</td>
<td>PU07 Speed</td>
<td>Speed reference (0-100%), cRIO software converts into pulse period for bellows pump solenoid.</td>
</tr>
<tr>
<td>Tripped</td>
<td>PU07.TR</td>
<td>Boolean</td>
<td>FALSE</td>
<td>PU07 Tripped</td>
<td>TRUE if the Pump has been stopped by an interlock. Cleared with a STT or STP pulse.</td>
</tr>
</tbody>
</table>

In Table 9-1, the **name** field is to identify the nature of the data, the **tag** field (where XX is replaced with a number) is the specific tag for that piece of data related to that specific piece of equipment. The **writeable** field is true if the cRIO Library will allow that value to be changed by operators. Many of the fields are
read-only as they are either inputs from the field or generated on the cRIO. The **OPC description** is the short description included in the LabVIEW Library and propagated onto the OPC server. The more verbose **description** field is only used in this reference document.

Different types of pumps require different sets of data. The positive displacement VSD pumps have a ‘READY’ feedback from the VSD (digital input), but do not have a running feedback signal from the field. The centrifugal pumps do not have a ready signal but do have a pump running feedback signal. The different data sets for each type of pump are included in the project documentation as Appendix 1002:

[UWS-1000 Datasheets and Reference\UWS-1002 Pump Data Structure.pdf](UWS-1000 Datasheets and Reference\UWS-1002 Pump Data Structure.pdf)

Other equipment in the system also have sets of data assigned and follow the same XXXX.YYY data structure as for the pumps. Solenoid Valves have the designation SVxx, Flow Valves are FVxx, Tank levels are LIxx, flow indicators are FIxx. Manual valves in the system are not included in the UWS control system, but do follow the same naming structure HVxx and GVxx\(^4\).

There are three points of data associated with tanks, the analog value (from the field instrument), and two digital values (from float-switch digital inputs) for very high and very low levels in the tank. The analog reading is the ‘process variable’ (LIxx.PV), and the very high and very low digital inputs are set up to be the HiHi (LIxx.HH) and LoLo (LIxx.LL) alarm values for the tank. This structure is shown in Table 9-2.

\(^4\) This can be seen in the P&ID for the system, Appendix 0001:
[UWS-0000 Drawings and Diagrams\UWS-0001 UWS P&ID.pdf](UWS-0000 Drawings and Diagrams\UWS-0001 UWS P&ID.pdf)
TABLE 9-2 TANK 01 DATA STRUCTURE

<table>
<thead>
<tr>
<th>Name</th>
<th>Tag</th>
<th>Data Type</th>
<th>Writeable</th>
<th>OPC Desc</th>
<th>Description.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equipment Identifier</td>
<td>LI01</td>
<td>Type</td>
<td>-</td>
<td>-</td>
<td>Unique identifier for this piece of equipment.</td>
</tr>
<tr>
<td>Reading</td>
<td>LI01.PV</td>
<td>Double</td>
<td>FALSE</td>
<td>LI01 Level Reading</td>
<td>Instrument reading (0-100%)</td>
</tr>
<tr>
<td>HiHi Alarm</td>
<td>LI01.HH</td>
<td>Boolean</td>
<td>FALSE</td>
<td>LI01 HiHi Alarm</td>
<td>Digital HiHi Alarm (Float Toggle)</td>
</tr>
<tr>
<td>LoLo Alarm</td>
<td>LI01.LL</td>
<td>Boolean</td>
<td>FALSE</td>
<td>LI01 LoLo Alarm</td>
<td>Digital LoLo Alarm (Float Toggle)</td>
</tr>
</tbody>
</table>

Tanks 05 and 06 (reservoir) do not follow the data structure in Table 9-2 exactly, in that 05 does not have a LL float switch, and 06 does not have an analogue reading, instead it has an additional low level float switch. This is assigned the Lo tag, and is a warning to the user that the level in the reservoir is getting low, before the LoLo alarm is triggered, which will interlock various pumps.

Flow transmitters follow the naming convention F1xx, the only variable associated with the flow measurements is the process variable (PV) reading, for example the flow meter 01 measurement has the tag name 'F101.PV'. Valves have a simple set of data points, flow valves have only their manipulated variable setting, which is the open % requested of the valve. For flow valve 01 the tag for the open % value sent to the valve is 'FV01.MV'. All of these data structures are included in the project documentation and are included as appendices 1002-1004:

UWS-1000 Datasheets and Reference\UWS-1002 Pump Data Structure.pdf
UWS-1000 Datasheets and Reference\UWS-1003 Valve Data Structure.pdf
UWS-1000 Datasheets and Reference\UWS-1004 Instrument Data Structure.pdf
9.4 SYSTEM LIBRARY

Once the data sets for each type of equipment were specified, these were collated into a single spreadsheet and populated with the tag names of all the plant equipment. This spreadsheet is the library reference for the UWS data library. This total data library is a very useful reference and is part of the project documentation as Appendix 1001:

UWS-1000 Datasheets and Reference\UWS-1001 cRIO Library.pdf

The total library for this system consists of over 100 tags and this would be rather time consuming to enter manually. The data library reference was later used to create a file which could be imported directly into a LabVIEW shared variable library. This is explained in Section 10.4. Examples of these Library imports and exports, showing the necessary format, are included in the project documentation as Appendix 1998:

UWS-1000 Datasheets and Reference\UWS-1998 Library Backups

9.5 cRIO AND SERVER INTERACTION

Early in the design phase, some simple test libraries were generated on the CompactRIO and the UWS Server to see how the 2-library system would operate. It was discovered that having 2 ‘master’ copies of a variable on the system could bring about a ‘loop’ effect if the value was changed on both libraries simultaneously. If this happened, each library would think it’s value was the most current value, and the values in each would oscillate back and forth. In order to avoid this (and for variable security), every variable only existed in one library as a master copy. The other library would contain a read-only copy of the variable. The tasks carried out by of each of these machines and the direction of data flow is shown in Figure 9-1.
A specific example of the data flow for a single pump is shown in Section 11.4. The location of the ‘master’ copy for each tag is listed in the master variable list document, included as Appendix 1001:

[UWS-1000 _ Datasheets and Reference\UWS-1001 _ cRIO Library.pdf]
9.6 INTERLOCKS

The interlocks in this system have been designed with two objectives in mind: process safety and equipment protection. Process safety means ensuring there are no overflows from the tanks onto the surrounding area and equipment. An overflow is undesirable as it wastes water from what is otherwise a closed system\(^5\). More importantly, there is electrical equipment and cables around the lower levels for the pumps, so spillage of water from the higher tanks could lead to an unsafe environment or electrical short. Equipment protection is limited to protecting the pumps. A simplified ‘HAZOP’ (Hazards and Operability) procedure was conducted to identify which interlocks were required for each piece of equipment.

The interlocks surrounding Pump 01 are considered as an example, a simplified PFD for Pump 01 is shown in Figure 9-2.

Pump 01’s output flows through solenoid valve SV01, then flow meter FI01, this flow can be used to fill either Tank 01 or tank 03 by opening the appropriate solenoid (SV18 or SV19). The interlocks associated with Pump 01 are listed in Table 9-3.

\(^5\) Some water is still lost to evaporation of course, though this is minimised by cover on each tank.
### TABLE 9-3 PUMP 01 INTERLOCKS

<table>
<thead>
<tr>
<th>Equipment</th>
<th>IL Number</th>
<th>Description</th>
<th>Condition</th>
<th>Delay</th>
<th>Tested and Approved</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUMP 01</td>
<td>1</td>
<td>Outlet Valve Closed</td>
<td>NOT SV01</td>
<td>0</td>
<td>k.blay, 16/9/2010</td>
</tr>
<tr>
<td>PUMP 01</td>
<td>2</td>
<td>Neither Discharge Valve Open</td>
<td>SV18 NOR SV19</td>
<td>0</td>
<td>k.blay, 16/9/2010</td>
</tr>
<tr>
<td>PUMP 01</td>
<td>3</td>
<td>Tank 1 Selected and Full</td>
<td>SV19 AND LAHTK01</td>
<td>5</td>
<td>k.blay, 16/9/2010</td>
</tr>
<tr>
<td>PUMP 01</td>
<td>4</td>
<td>Tank 3 Selected and Full</td>
<td>SV18 AND LAHTK03</td>
<td>5</td>
<td>k.blay, 16/9/2010</td>
</tr>
<tr>
<td>PUMP 01</td>
<td>5</td>
<td>Pump Running and No Flow</td>
<td>PU01.MV AND FI01 &lt; 1</td>
<td>5</td>
<td>k.blay, 16/9/2010</td>
</tr>
<tr>
<td>PUMP 01</td>
<td>6</td>
<td>No Power to VSD</td>
<td>NOT MIPU01</td>
<td>0</td>
<td>k.blay, 16/9/2010</td>
</tr>
<tr>
<td>PUMP 01</td>
<td>7</td>
<td>Supply Tank Lo Lo</td>
<td>LALLTK06</td>
<td>5</td>
<td>x.blay, 16/9/2010</td>
</tr>
</tbody>
</table>

The justification for these interlocks is as follows.

1. **Outlet Valve Closed (Equipment Protection)** – It is unsafe to operate a positive displacement pump against a closed valve. This will generate very high pressures at the pump outlet and can cause the pipes/joins to fail, or cause the pump motor to burn out.

2. **Neither Discharge Valve Open (Equipment Protection)** – For the same reason as Interlock 1, if both of the discharge valves (SV18 and SV19) are closed, the pump is at risk of burnout or leaking.

3. **Tank 01 Selected and Full (Process Safety)** – The UWS control system will not allow the pump to over-fill a tank and risk an overflow/spillage.

4. **Tank 03 Selected and Full (Process Safety)** – As before, the control system will not allow the pump to over-fill a tank.

5. **Pump Running and No Flow (Equipment Protection)** – If the manual valve HV01 is closed, there is no feedback to let the control system know. Pump 01 could start and be sucking against a closed valve. The lo-flow detection interlock is in place to avoid this situation. If the pump has been running for 5 seconds and no flow is detected (<1 l/min) the pump will be shut down and locked out for 20 seconds.

6. **No Power to VSD (User Notification)** – The VSD sends back a ‘ready to start’ signal to the cRIO, if the VSD is not ready, or it becomes unhealthy while running, the pump will be stopped. This interlock is in place not to
stop/hold out the pump, but to let the user know why the pump was stopped.

7. **Supply Tank Lo Lo (Equipment Protection)** – If the supply tank LoLo alarm is active, it means that the level in the tank is lower than the intakes of pumps 01-07. This interlock avoids the dangerous situation that could be caused by sucking air through the pumps.

As pumps can have up to 7 individual interlocks, it was necessary to condense this data in order to use it in later calculations (in the cRIO) and return the data to the operator. In order to do this, the individual Boolean status of each interlock is concatenated into a single binary integer (in this case a 32-bit unsigned number). That way, if all of the interlocks are clear, the integer is zero. If interlock 1 is true, the interlock is 1. If interlock 2 is true, the interlock integer is 2. If interlock 3 is true, the interlock integer is 4, and so on.

Using this method, the pump control logic only has to check the integer, if it is not zero, one of the interlock conditions has occurred and the pump is stopped. The operator graphics can show the fact that the pump is interlocked, and can even let the operator know exactly which interlock(s) caused the lockout, by interpreting the integer.

### 9.7 **Security**

The final consideration in the design phase was the security of the system. The UWS control system had to be designed so that only users that were present in the Pilot Plant could operate the system. Users in other areas of the University could read the system variables, and extract historical data, but not make changes to the system. This was to be achieved by the use of Library passwords, and DSC security settings on the system libraries. The specifics of these settings are discussed in Sections 10 and 11.
The CompactRIO (cRIO) controller is produced by National Instruments. It has two main components: a real-time controller, with a processor, and communication via USB, Ethernet, or RS-232 port; and a backplane for the installation of I/O modules. The backplane also has a field-programmable gate array (FPGA) which executes low-level high-speed functions.

The installation of the Controller and I/O modules, and the wiring and termination of the system, was completed before the start of this project. The hardware
connection was the focus of the Master’s Thesis by Janosz which was an excellent resource throughout this project.

The CompactRIO series of controllers has two different methods of code execution. A field-programmable gate array (FPGA) allows for simple code to be executed very rapidly (in excess of 100kHz scan rate). This FPGA is programmed using the LabVIEW software with a graphical interface similar, but not exactly the same, as the classic LabVIEW VI coding interface.

The other method of executing code on the CompactRIO is on the processor of the Controller module directly. This controller module can execute standard LabVIEW coded VIs and includes a LabVIEW shared variable engine, allowing the cRIO to host and read from other shared variable libraries.

To configure the CompactRIO controller, the National Instruments Measurement and Automation eXplorer (MAX) program is used. Using this program the CompactRIO’s basic configuration can be carried out. The only changes necessary in this project were the updating of the TCP/IP settings (IP/Gateway/Subnet Mask) as discussed in Section 7.

10.2 PROGRAMMING THE COMPACTRIO

With the IP conflict issue resolved as mentioned in Section 7, the creation of the code for the CompactRIO controller could be started. The code for the controller is constructed as a LabVIEW Project, which encapsulates all the VIs, Libraries and I/O modules for the cRIO. When creating a new ‘Real-Time’ project in LabVIEW, the

-----------------------

6 From Page 5 of the CompactRio Developers Guide (3). Included in Appendix 1999: UWS-1000 Datasheets and Reference\UWS-1999 User Manuals and Guides

7 The normal LabVIEW VIs do have to be compiled into a Real-Time executable file before the CompactRIO can execute them.
user is able to choose if they wish to code in the FPGA or regular LabVIEW environment.

For this project, the regular LabVIEW environment is used. This means that the code for the low level interlocking and filtering will be executed on the cRIO controller’s processor. A single shared variable library can be constructed on the cRIO to synchronize the system variable with the server. This option is selected over the FPGA environment for three main reasons.

Firstly there is no need to use the FPGA interface, even though the FPGA has scan times over 1000 times faster than the cRIO controller scan time. The scan rate of the cRIO controller (many times per second) is more than fast enough for any of the elements in this system. None of the pump controllers have any need to be executed at higher scan rates. The wider system will always be limited by the cRIO scan time, as this defines the fastest possible rate at which data can be updated in the Shared Variable Library.

The second reason for using the classic LabVIEW programming interface for this project was familiarity. While the FPGA programming interface is similar to the classic LabVIEW interface, there are differences. The FPGA does not use the classic LabVIEW VIs or the shared variable library. To simplify any future interaction with this code, it was better to keep it in LabVIEW classic style.

The third is for simplicity, in order to use the FPGA interface. An additional ‘interposing’ Library had to be created, and some code still generated for the cRIO Controller. The shared variable library for network hosting, and initialization functions, would still have to be executing on the controller, with the Library updating at the same rate as the Controller scan rate.
For these reasons, the classic LabVIEW coding interface is used to generate code that will execute on the CompactRIO controller processor. The FPGA functions are not used in this Project.

10.3 Tasks of the CompactRIO Program

The program for the CompactRIO controller is responsible for the most basic low-level tasks in the control system. If the communication link between the CompactRIO and the UWS Server is broken, the cRIO must still be able to operate the system in a safe manner and continue to monitor the interlocks and system conditions. The tasks to be performed by the cRIO controller are shown in Figure 10-3.

![Compact RIO Tasks](image)

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary I/O</td>
<td>Reading from Field Instruments, Sending signals to Field equipment</td>
</tr>
<tr>
<td>Input Processing</td>
<td>Filtering Analog Signals, MacNaught flow meters: convert period to l/m</td>
</tr>
<tr>
<td>Pump Control</td>
<td>Starting and stopping of Pumps as requested by user</td>
</tr>
<tr>
<td>Interlocking</td>
<td>Monitor System to ensure safe state, Removal of run permissive on pumps if unhealthy</td>
</tr>
<tr>
<td>Output Processing</td>
<td>Output analog values to field, Bellows Pump: Convert speed to Period</td>
</tr>
<tr>
<td>Hosting Process Data</td>
<td>Processed field inputs hosted in Shared Variable Library, Host status variables generated by cRIO Code, Read user inputs from UWS Server library</td>
</tr>
</tbody>
</table>

**FIGURE 10-3 TASKS CARRIED OUT BY THE CRIIO CONTROLLER**
10.4 SHARED VARIABLE LIBRARY

The first step to constructing the program for the CompactRIO controller was to implement the full shared variable library. The details for each tag had been compiled into a single datasheet\(^8\) during the design phase. As mentioned in Section 9.4, the tag information spreadsheet consists of over 100 tags. To save time when populating the shared variable library in the CompactRIO project, the tags were added by the use of the export to csv and import from csv options in the LabVIEW project explorer. One tag was entered in the cRIO shared variable library and configured as desired. The library was then exported to a .csv file. The format for the library csv file could now be copied and the system library arranged to match that format. This formatted system library was then imported back into the LabVIEW project, populating the system library. Examples of these Library imports and exports are included in the project documentation as Appendix 1998:

\[UWS-1000\] Datasheets and Reference\[UWS-1998\] Library Backups

When the shared variable library is configured on a real-time device, the DSC security settings are not available in LabVIEW 2009. The security design for this project required the cRIO controller to be visible on the Murdoch Network, but only respond to requests from the UWS Server machine. The Measurement and Automation Explorer (MAX) was used to configure the CompactRIO so that the UWS Server is the only device able to change the configuration settings, but the Shared Variable library on the cRIO was still accessible to the entire network.

\[\]

\(^8\) Included as Appendix 1001:
\[UWS-1000\] Datasheets and Reference\[UWS-1001\] cRIO Library.pdf
The physical I/O readings had previously been configured to be ‘network published’, meaning any client on the Murdoch network could read and write directly to the I/O cards on the cRIO backplane. To remedy this, the I/O variables themselves were reconfigured to not be published to the network, with the cRIO Library behaving as a more tightly-managed interposing library.

All CompactRIO documentation suggested that it was capable of using the DSC security module to control access to the shared variable library, but LabVIEW 2009 does not allow for security configuration of a Real-Time Library. The security settings are visible from the National Instruments Distributed System Manager, but not in the LabVIEW project itself. To enable the security settings in the cRIO library, a small hack was used to force the security settings down to the cRIO library:

As the library can be imported from a .csv file, it was possible to find the syntax for the DSC security settings, then add these new settings to the CompactRIO library. A dummy variable was set up on another machine (hostname: PP2001-05), and configured to only allow the UWS Server access to the variable (via IP filter). This security enabled variable was then exported to csv and the format for the security settings found. These additional columns for the security settings were then added to the file to be imported to the cRIO library. With the security settings added to the cRIO library, other users on the network could no longer connect to variables on the cRIO directly. This is per-design as the UWS Server is designed to be the gatekeeper between other areas of the network and the CompactRIO controller. With these changes in place some care has to be taken when opening the cRIO project for editing. The relevant procedures have been written up in the project documentation and are included as appendices 3002 and 3003:

UWS-3000 Standard Operating Procedures\UWS-3002 Opening cRIO Project.pdf
UWS-3000 Standard Operating Procedures\UWS-3003 Editing and Deploying cRIO Program.pdf
10.5 SETTING THE cRIO TIME

As well as the actual value of each variable, the NI shared variable protocol also includes a timestamp as part of the information transferred over the shared variable PSP protocol. When a variable is copied onto the UWS Server, the UWS shared variable engine will use this timestamp\(^9\). It was noticed early in the project that the time on the CompactRIO was approximately 4 hours behind Western Standard Time (Perth Local Time). This was not foreseen to be a problem and there is no easy way to set the time in the CompactRIO administration program (MAX). The issue later became a problem when the logging was configured on the UWS Server (see Section 11.5), as the logging engine stored the timestamp value from the cRIO in the log database. This meant that when the ‘current’ trends were viewed, the most recent 4 hours were always blank.

A solution was found by searching the NI forums\(^4\). The solution was to configure the CompactRIO to use Murdoch’s Simple Network Time Protocol (SNTP) server to synchronise with WST. This configuration is quite complicated as it requires logging in to the cRIO using an FTP client and directly editing a configuration file on the cRIO itself.

This configuration resolved the timestamp problem and will ensure that it does not happen in the future due to slight differences in the clocks of the respective systems. A detailed procedure was generated for this and is included as Appendix 3004:

UWS-3000 Standard Operating Procedures\(\)UWS-3004 Configuring cRIO to use SNTP Server.pdf

\(^9\) The OPC server on the UWS Server also uses this timestamp information.
10.6 PROGRAM EXECUTION

The code in the cRIO has been set up to execute in time with the scan refresh rate of the cRIO controller, configured to be 200ms. The code for the main program was broken down into sub-VIs. This is done for two reasons, simplicity and re-usability. LabVIEW code, with its graphical nature, is very clean and easy to understand for small chunks of code and simple functions. As code gets larger however, the graphical nature of the code becomes somewhat of a hindrance with a large number of connecting ‘wires’ which are difficult to follow. Breaking the code down into smaller blocks removes this problem and the user can understand each block individually.

The main loop of the program is contained in the VI UWS cRIO Code.vi. This contains a sequence which is executed every time the cRIO starts, the first panel of the sequence locks the cRIO with the default password uws3!swu. The second panel contains a loop that continues executing until the cRIO is restarted. This inner loop is executing at 200ms and contains 4 sub-VIs that house the cRIO program.

This loop also contains the Period VI which measures the time (in ms) that has passed between scans, as the loop is a deterministic synchronous loop structure, this should always be 200. This number is used as the time step for sub-vi’s in the system, and used by the Client program to monitor the health of the cRIO.
10.7 cRIO Code

This section will briefly cover the workings of the code on the CompactRIO controller. This will be a very brief explanation of what is occurring ‘behind the scenes’ on the controller. A much more in-depth explanation of the code has been generated as a ‘Functional Description’ of the CompactRIO System. This Functional Description has been included in the project documentation as Appendix 4001:

UWS-4000 Functional Descriptions\UWS-4001 cRIO Functional Description.pdf

10.7.1 Input Processing

The first piece of code (sub-VI) in the cRIO program is the Map and Filter VI. This reads in the physical inputs, processes them as required and saves the processed variables into the Shared Variable library, to be read up to the UWS Server library.

The analogue signal (from the level transmitters) is fed through a digital filter with a 1s time constant. The digital float switches (HiHi and LoLo level alarms) are fed through an ‘on-delay’ block to debounce the signal and avoid jitter.
The flow measurements are also filtered with a first order digital filter with time constant 1s. The MacNaught flow meters need some additional processing, as they return ‘pulses’ instead of a 4-20mA signal. This is done using the custom ‘pulse to flow’ VI.

### 10.7.2 Interlocks

The Valve control system is very simple, most valves have no interlock and thus the requested open/close status (SV0X.MV) can be written directly out to the valve. The only valves with interlocks are valves that feed into a tank (for example SV16, the dump valve from Tank01 to Tank03). In these cases, the valve can only be opened if the HiHi float switch in the outlet tank is not active. If the outlet tank is full the valve is interlocked closed and the interlock tag (SV0X.IL) reads ‘1’.

![Valve Interlock Diagram]

**FIGURE 10-6 VALVE INTERLOCK**

Another function performed in the valve interlock VI is the limiting of the value sent to the flow valves, the requested output (FVOX.MV) is limited to a value between 0 and 100 before being sent out to the valves.
The Pump Interlock VI is more complicated than the Valve interlock code. The pumps require many more interlocks than the valves, both for process safety and motor protection. The full list of interlocks for each pump is available in Appendix 1005:

UWS-1000   Datasheets and Reference\UWS-1005   Pump Interlocks.pdf

The pump interlocks VI does not control the pumps directly, it merely generates the integer value for the pump interlocks. For a pump to be given a run permissive, all interlock conditions have to be clear (FALSE). The Pump interlocks.vi takes the Boolean status of all interlocks, then concatenates those into a binary integer (Unsigned 32 bit) where the interlock status constitute the bits of the integer (starting from LSB).

This means that the exact interlock conditions that are stopping the pump can be read back to the client as a single integer instead of a series of bits, and that the pump control VI merely needs to look at the interlock integer (PU0X.IL) and remove the run permissive if it the integer is non-zero.

The ‘Pump Running and No Flow’ interlock (present on all pumps) requires the pump to be running in order to be true. Without an off delay the interlock would clear as soon as it stopped the pump. An off delay of 20s ensures the interlock holds the pump out for a minimum period of time to stop a user immediately re-starting the pump.

10.7.3 PUMP CONTROL

The final VI called in each scan of the cRIO program is the Pump Logic VI. Shown in Figure 10-7, this VI turns pumps on and off, checks the interlock conditions, and passes speed references to the VSD and Bellows pumps.
The VI is set up in a flat sequence, all of the VSD blocks are together, the centrifugal blocks are together, and the bellows pump is by itself. The Pump Logic VI makes use of three important sub-VI’s: *Pump_VSD.VI*, *Pump_Centrifugal.VI* and *Pump_Bellows.VI*. All of these VI’s are re-entrant to allow for multiple callers, although with only one Bellows pump in the system, that VI is only called once.

### 10.7.3.1 Pump Specific Sub-VIs

As set out in the design phase, each pump has a standard set of variables attached to it. These sets differ slightly between different types of pumps, for the main part of the pump control logic is the same (STT, STP, TRP, MV, IL tags). The pump specific VIs use shift registers to ‘latch’ the MV. In order to access the shift registers, the code is encapsulated in a run-once while loop.

As defined in the pump control specifications, for a pump to start, the VI must see a rising edge on the STT command, AND the STP command must be FALSE, AND
the VSD must be powered (RDY), AND the INTERLOCK integer equal to 0 (all interlocks clear). When these conditions are met, the pump MV will switch to TRUE.

This MV will remain true unless one of the STOP conditions occur. These conditions are a TRUE on the STP command, an interlock being activated (IL integer != 0), or the VSD Ready bit switching to false. When these happen, the MV is latched to FALSE until the pump can be restarted.

The pump control VIs also have the task of setting the TRP tag to indicate if a pump was tripped by an interlock. If the pump MV is true and one of the interlocks is activated, the pump TRP tag will be set to TRUE. This will remain TRUE until a start or stop command is received, when it will be cleared.

Some extra logic is incorporated into the VI for the Bellows pump. The output from the Bellows pump block is an open/close command to the Bellows solenoid. In order to calculate the period between these open/close pulses, the Bellows pump sub-VI converts the 0-100% speed request into a pulse frequency between 0 and 100 pulses per minute.
10.8 Compiling the cRIO Code

The CompactRIO real-time controller can execute code that has been generated in the NI LabVIEW coding environment, but the code has to be compiled. The LabVIEW project for the Real-Time project includes a ‘Build Specifications’ element. A new set of build specs was generated, and the Main Program Loop VI compiled into a real-time executable (.rtexe) file.

This real-time executable file includes all of the sub-VI dependencies and once deployed, can execute natively on the cRIO controller. For this project, the Main Program Loop VI was set to be the start-up application for the cRIO controller. After restarting the controller (or power loss) the main program loop first initialises the controller then continues executing indefinitely.

A detailed explanation of this procedure has been generated and is included in the project documentation as Appendix 3003:

UWS-3000 Standard Operating Procedures\UWS-3003 Editing and Deploying cRIO Program.pdf

10.9 Testing the cRIO Code

As discussed in Section 10.6, One of the reasons for breaking the code into smaller functions (sub-VIs) was to simplify the testing of the code. Each of these sub-VI functions could be tested in isolation to make sure the code was behaving true to the design. An arbitrary front panel control could be used and the sub-VIs tested on the test machine, instead of having to compile and deploy the cRIO code to test each modification. This isolated testing was carried out throughout the construction process for each sub-VI.

With the code in place, the cRIO code and interlocks could be commissioned. The commissioning process is a painstaking, individual testing sequence for every
variable and piece of code in the system. The commissioning was done on the compiled and deployed cRIO code to make sure it was performing exactly to the design specifications.

The testing phase of the CompactRIO code lasted 2 weeks. A simple test program was generated on the UWS Server\textsuperscript{10} consisting of a simple front panel made entirely of \textit{bound variables}. This allowed the user to monitor the cRIO library and perform rudimentary testing of the system.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{monitor_vi.png}
\caption{MONITOR.VI TESTING INTERFACE}
\end{figure}

Using this testing VI, every input to the system was tested and checked to ensure that the tagging in the control system matched the values in the field, and that the

\textsuperscript{10} Included as Monitor.vi in the UWS Server Project
field value was correctly read into the DCS. During this phase the calibration on the analogue instruments was checked and confirmed to still be accurate.

Every interlock in the system was tested and approved during this commissioning phase. The conditions necessary for each interlock to be active were created\textsuperscript{11} and the pumps were confirmed to stop on the conditions listed in the design specifications. The approval status for the pump interlocks was then integrated into the pump interlock list.

This testing procedure was an exhaustive process, but vital to ensure the cRIO is going to behave as expected. Multiple bugs were detected and fixed during this process, including interlocks that looked at the incorrect level or flowrate, and pump speed references pointing to the wrong pump. The only way to ensure all of these bugs are removed is to follow the testing procedure. Once this process was complete, the system was reliable for the rest of the project, and at no point was there any need to make further changes to the cRIO code.

10.9.1 PERFORMANCE

During the first set of tests, the scan rate for the cRIO was brought down to 100ms to see what effect this faster scan rate would have on the system. The first results were positive, while there was no noticeable difference in the user experience, the code was executing and the values updated correctly at the new scan rate. After about 5 minutes, the system would start to become sluggish. The UWS would sometimes take a few seconds to respond to changes made by the user. To

\textsuperscript{11} The exception here is the supply tank Lo-Lo interlock, this was tested by forcing the value of the LoLo toggle, instead of draining the entire supply tank, and wasting the water. The supply tank interlock conditions were created (and successfully enabled the interlocks) at the end of the project while emptying the system prior to removal.
diagnose what was causing this the NI *Distributed System Manager* was opened to view the performance information for the cRIO controller.

The distributed system manager showed that the cRIO controller was operating at 100% CPU load. The code for the UWS was executing in a high-priority timed structure and taking up ~60% of the CPU time. In this small embedded system the same CPU has to handle the communications, and manage the shared variable library, this is only a ‘normal’ priority task and was taking up the remainder of the CPU time.

What was found to be happening was the CPU was over-loaded, and giving priority to the execution of the system code at the expense of the shared variable library management. The library does have a buffer (set to 50) for the variables to allow for small delays, but as this problem was persistent the buffer would eventually fill up and lead to ~5 second delays in the system.

To remedy this, the scan-time was set back to 200ms (in the Measurement and Automation Explorer). The lag induced in the system disappeared and the UWS began to respond properly again. Figure 10-10 shows the CompactRIO’s performance information after the change back to 200ms.

![Figure 10-10 CRIO Performance](image)
10.9.2 **Tank 03 Level Meter**

During the commissioning of the interlocks, it was noticed that when the VSD pump PU09 was running, the reading from the capacitive level instrument in Tank 03 (LI03) would start to drift up and down. This drift was significant (±20%) and continued until the pump was stopped. With help from Graeme Cole and John Boulton, it was suggested that the cause could be the non-conductive material which made up the body of the tank. Placing a metal strip in the tank seemed to help the problem. This was confirmed by finding the Operating Manual for that model of instrument (FMI51) from Endress & Hauser(5).

On page 48, the measured variable of the transmitter is defined as:

> Continuous measurement of change in capacitance between probe rod and container wall or ground tube, depending on the level of the liquid.

The instrument in place in the UWS does not have a ground tube installed (example shown on page 26 of the manual), thus the instrument was measuring the change in capacitance between the probe and nearby metal objects, one of which was Pump 09 itself. Thus as PU09 was switched on, the level measurement experienced significant disturbances. It was necessary to provide the other side of the capacitor to complete the circuit for the level instrument. John Boulton replaced the plastic bracket that held the float-switches with a grounded stainless steel bracket. This runs the length of the tank and provides the other half of the capacitor circuit. While not as effective as a metallic tank or ground tube, it made the oscillations manageable (±2%). The instrument required re-calibrating afterwards, this was achieved by performing a 2-point (full/empty) calibration from the transmitter head directly, without the need for a HART communicator.
11 THE UWS SERVER

The UWS Server is the point of interaction for a ‘Level 2’ type interaction as defined in Section 8.1. Users interacting at this level would be able to take full control of the UWS system, and implement custom controllers through a range of interfaces. Users at this level cannot however bypass interlocks or force variables on the CompactRIO.

A detailed functional description of the UWS Server is included with the project documentation as Appendix 4002:

UWS-4000  Functional Descriptions\UWS-4002  UWS Server Functional Description.pdf

11.1 BASIC CONFIGURATION

The generation of code for the CompactRIO was carried out on the machine that was to become the UWS Server. The UWS Server is a Windows XP machine in the Murdoch University Pilot Plant. Initially the machine that was to become the UWS Server had a dynamic IP and a Murdoch Standard Hostname (PP-2001-06). Some checking with Murdoch IT uncovered the fact that the Hostname rules for the student machines could change without any notice. In order to enable the security on the CompactRIO configuration page (MAX) and Shared Variable Library, the server would require a static IP and/or unchanging Hostname. With the help of Will Stirling, the UWS Server machine has been assigned the Hostname UWSServer1 and the (static) IP 134.115.133.136. This subnet is used for all Engineering projects requiring static-IPs. All TCP/IP configuration for the system has been compiled into a reference sheet and is available in the project documentation as Appendix 1006:

UWS-1000  Datasheets and Reference\UWS-1006  Network Details.pdf
No LabVIEW VIs are required to execute on the UWS Server to operate the system. The hosting and data logging for the server are configured to run as background services. These services automatically start with Windows and will be running whenever the UWS Server is powered up. No user is required to be logged in to start these processes. The project library and services for the UWS Server, along with two utility programs, are encapsulated in a LabVIEW project file.

11.2 **UWS Server Tasks**

The system has been designed with the UWS server sitting as the gateway between the CompactRIO controller and the Client machines. The UWS Server hosts data through both an NI Shared Variable engine and an OPC Server. The OPC server is an extension of the shared variable engine so the same set of data is hosted no matter which interface is accessed. The server also logs the historical data in an SQL compliant database. This historical data can be extracted for later trending and analysis. These tasks are shown in Figure 11-1.

<table>
<thead>
<tr>
<th><strong>UWS Server Tasks</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Real-Time Data Transfer</strong></td>
</tr>
<tr>
<td><strong>Data Logging and Historian</strong></td>
</tr>
<tr>
<td><strong>OPC Server</strong></td>
</tr>
<tr>
<td><strong>Security</strong></td>
</tr>
</tbody>
</table>

**FIGURE 11-1 TASKS CARRIED OUT BY THE UWS SERVER**
11.3 The Need for the UWS Server

The UWS Server acts as a buffer between client machines and the UWS cRIO controller, if the UWS Server was not in place, the cRIO controller would have to communicate directly with client machines, sending multiple copies of the system variables to each client machine that requests them.

![Diagram](image)

**Figure 11-2 System Without Server, Multiple CRIO Connections**

The UWS Server has a Shared Variable Library that takes a copy of any variables set by the cRIO, and hosts the variables set by the client machines so that the cRIO has a single source of variables to read from, and write to.

With the UWS Server acting as a host, the cRIO only needs to write one set of values, and read in values from a single point. The UWS server handles the additional communication load and centralises the system variables. This is a job much more suited to an Intel-based PC rather than the relatively low-powered cRIO controller.
There are other bonuses of this network configuration. The Server can further ease the communication load by setting an *update dead band*, wherein the new values are not transmitted to the clients unless the variable has changed by a defined amount. This update dead-band is configured in the UWS Sever project file (LabVIEW) and the UWS Server shared variable library. In this system the update dead band has been set to 0.01 for all variables. The update dead-band and other settings related to the variables (resolution, logging dead-band etc) can be configured in the UWS project file. A procedure has been generated to edit this file and this procedure is included in the project documentation as Appendix 3005:

UWS-3000  Standard Operating Procedures\UWS-3005  Editing the Server Library.pdf

### 11.4 Library Behaviour

As discussed in Section 10.4, both the cRIO and UWS Server contain shared variable libraries with all system variables included. To avoid any conflict between these libraries, each system variable only exists as a ‘master’ copy in one of these system libraries. The user-editable variables are stored on the UWS Server library (the cRIO Library reads in copies of these variables) and the variables returned
from the cRIO are stored on the cRIO Library. Each variable is read-only in at least one of the system libraries. This ensures no conflicts between the system and client machines. In the UWS Library reference (Appendix 1001) the location of the Master copy is listed for every variable.

In Figure 11-4, the direction of flow of the variables for Pump 01 is displayed. The 'Ready' bit, MV, Trip, and Interlock variables are generated on the cRIO and are as such can only be 'read' accessed by the rest of the system. The user editable variables (start bit, stop bit, speed request) exist as master copies in the UWS Server library. The clients can read and write to these variables, but the cRIO library can only read these variables.

Making the user-editable variables read/write accessible is particularly useful when combined with the ability of LabVIEW to 'bind' a control to a shared variable12. If multiple users have client programs open, changes made by one user will be reflected in all other user’s interfaces.

---

12 Extensively used in the UWS Client program included with this project.
11.5 LOGGING AND EXTRACTION

Another critical function performed by the server is the logging of process data for later extraction. The logging is set to log 30 days of data at the same resolution as the update dead-band (0.01). The log files are stored on the UWS Server in the C:\uws shared variable database directory.

To configure this data historian, the dead-band, resolution, and location of the log files can be configured in the shared variable properties. These properties are accessible in both the LabVIEW project library, and online via the Distributed System Manager. It is preferable to use the Distributed System Manager as a read only interface, or to make online changes for testing. Permanent changes should be made using the LabVIEW project as this file is used as a backup for the Library should it have to be re-deployed.

The historical database itself can be viewed (and administered) by using the Measurement and Automation Explorer (MAX) program (On the Desktop). On the Server, open MAX and select Historical Data | Citadel 5 Universe and browse to My Computer | UWS Database | UWSServer1 | UWS Server.

![FIGURE 11-5 UWS DATABASE ADMINISTRATION](image)

From here, each point can be viewed and the historical data either viewed or extracted. Databases can also be deleted through this interface to clean up old files.
The MAX interface for the historical data is somewhat sparse, and designed primarily for debugging and administration purposes. Many LabVIEW VIs exist that allow for the use of Historical Trends and data extraction programmatically. The UWS Client generated for this project has made use of these and includes both trending and extraction functions. The historical trends can be customised to show any desired trace. The UWS Client comes with a default set of 5 ‘groups’ with preset traces. The data extraction function in the UWS client allows the user to select the desired variables, time period, and sampling rate, and extract that data to a spreadsheet file. A more detailed discussion of this process will be included in Section 12.

11.6 TESTING PAGE

To test and commission both the CompactRIO and the UWS Server, a small testing utility was generated and populated with bound controls. This test program contained all system variables on a single page. While not being particularly user-friendly, it was a highly useful debugging and testing tool. This program is included with the final UWS Server project. Further information about Monitor.vi can be found in Section 10.9.

11.7 UWS SECURITY

The UWS Server is a key part of the security for the UWS control system. The cRIO controller has been configured to only communicate with the UWS Server and the UWS Server has in turn been configured only to allow certain machines read/write access. Only machines in the Pilot Plant control room (hostnames PP2001-01 to PP2001-05) can write to the UWS Server Library. Other machines on the network can view the current data as read-only, and extract historical data. This configuration means that no one can operate the plant from other areas of the campus, but students can extract data at a later time without the need to go to the
Pilot Plant. This security is configured in the Library setup in the UWS Server project. The DSC Security settings are set library-wide, instead of different settings for each variable. If Murdoch IT change the hostnames of the pilot plant machines sometime in the future, or additional machines need to be given access, a procedure has been generated for adding/modifying the list of machines with write access. This procedure is included in the project documentation as Appendix 3005:

UWS-3000  Standard Operating Procedures\UWS-3005  Editing the Server Library.pdf

11.8 SYSTEM ADMINISTRATION

The CompactRIO is locked in a sealed cabinet and will be difficult for higher level users to Access. If the event ever arises wherein the CompactRIO itself needs to be restarted, a utility has been generated for use (on the UWS Server) that can remotely manage the CompactRIO. This program is included with the UWS Server LabVIEW Project and has a simple interface that shows some basic details about the current state of the cRIO. Further information about this program and restarting the cRIO is available in the procedure included as Appendix 3001:

UWS-3000  Standard Operating Procedures\UWS-3001  Restarting the cRIO.pdf

The UWS Library can be monitored and administered using the NI Distributed System Manager in Start Menu | Program Files | National Instruments | Distributed System Manager. If the machine is unresponsive, a restart may be required. As this is a regular student-login accessible Windows XP machine, executing other processor-intensive programs on the server might also slow the system performance.
11.9 OPC CONNECTIVITY

The NI shared variable PSP protocol is an excellent tool and has been extensively used in this project. It is not however, a widely used protocol in industry and many third party clients do not have the ability to communicate via PSP. To allow for further flexibility in this system, the UWS Server had to also have the ability to communicate using the OPC protocol. OPC is a much more common protocol and is more widely used in industry than the NI PSP system.

National Instruments do provide an extension to the Shared Variable engine which will add OPC capability. With the help of Will Stirling, the National Instruments OPC plugin license was installed on the UWS Server. Once this was enabled the variables that are hosted by the NI engine can be accessed by any client with OPC capability. Will was also able to add the OPC client license to MATLAB to allow connection to the OPC server through the MATLAB interface.

The operation of the OPC server was confirmed by connecting to the server using both the NI Simple OPC client and MATLAB’s OPC interface. The OPC interface operates as anticipated and inherits the point health information and timestamp from the Shared Variable server. Changes made to either server are reflected in the other.

While this project was being completed, a Masters student Kan Sumontha was working on a parallel project which made use of the OPC connectivity of the system. Kan’s project is to generate an Excel plugin using VBA and make use of MATLAB’s real-time capabilities while integrating OPC. Kan’s work will set the foundation for future students to use the OPC server to connect to the UWS. The use of third party applications like Excel or MATLAB will allow for more advanced controllers to be implemented in the most appropriate software package.
12 CLIENT SOFTWARE

12.1 UWS CLIENT BASICS

The UWS Client program is a LabVIEW 2009 file which is available from the UWS Chad website or from the UWS folder on the MyLab drive on the Murdoch network.

The UWS Client contains all required files and code to communicate with and control the Universal Water System at Murdoch University. The UWS Client has an ASM Standard graphical interface, and also includes classic PID controllers for any analog outputs to the plant. The PID controllers allow the user to select the desired PV and choose their own control loops from the graphical interface, rather than having to re-code the program.

For more advanced users, the client is designed to be used as a template to implement more advanced control structures. The PID controller can be overridden with ‘Expert’ or MIMO systems by making use of the provided code areas and switches.

Two functional description documents have been generated for the UWS Client interface. These are a Code description and a UWS Client User Guide. These files have more specific and detailed information about the Client than is included in this report. These documents can be found in the project documentation as appendices 4003 and 4004:

UWS-4000  Functional Descriptions\UWS-4003  Client Software Template.pdf

UWS-4000  Functional Descriptions\UWS-4004  UWS Client User Guide.pdf
12.2 **LOOK AND FEEL (ASM COMPLIANCE)**

In generating the graphical interface for the Universal Water System, industry standard practices are followed with respect to the look and feel of the user interface. Released in 1998, the best practice guidelines constructed by the ASM (Abnormal Situation Management®) consortium define the appearance of most modern operator interfaces. The overview page for the graphics is shown in Figure 12-1.

![System Wide Overview](image)

**FIGURE 12-1 SYSTEM OVERVIEW PAGE**

The key to the ASM guidelines with operator interfaces is the ability to quickly identify and quantify any abnormal situation in the plant. The guidelines suggest shades of grey for the 'normal' appearance of the plant, with bright colours only used for alarms or other indications of unusual conditions. This allows operators to

---

13 ASM and 'Abnormal Situation Management' are registered trademarks of Honeywell International.
14 The ASM guidelines have a much larger scope that covers Alarm Management, Control Room Layout, and many other Human Factors that are not entered into in this report. A good synopsis of these and their impact is available in reference (12).
instantly recognise when something out of the ordinary is occurring in the system. The particular colour used to indicate an abnormal situation is different depending on the ‘priority’ of the situation.

12.2.1 **CUSTOM LABVIEW CONTROLS**

The standard set of LabVIEW controls (graphic interface objects) is insufficient for this sort of advanced graphic interface. *Customized* LabVIEW controls were generated to be used in the operator interfaces. To do this the templates for icons were first created in *Microsoft Visio*, this template is then imported into LabVIEW in the *Controls Editor* and configured to behave as desired. The icon template generated in Visio for the standard UWS pump icon is shown in Figure 12-2.

![FIGURE 12-2 PUMP ICON IN VISIO](image)

This icon was then used in the NI Controls Editor to make a custom control in LabVIEW. In this case the pump is an indicator, when fed a Boolean FALSE value the light grey body is shown, when the pump indicator is fed a Boolean TRUE value the dark grey body is shown. The full procedure for creating custom controls in LabVIEW has been documented and is available as Appendix 3006:

[UWS-3000 Standard Operating Procedures\UWS-3006 Customising LABView Controls.pdf](UWS-3000%20Standard%20Operating%20Procedures\UWS-3006%20Customising%20LABView%20Controls.pdf)

Also available as part of the project documentation is the library of custom controls generated for the UWS. This library includes all variants of the controls used in this
project, as well as the Visio template for the controls. The custom controls library is Appendix 2004:

UWS-2000  Software\UWS-2004  Custom Controls.zip

12.3 SIMPLE ALARMING

One of the intentions of the design phase of this project was to generate a system with a level of complexity more than that of the Instrumentation and Control laboratory, but less than that of the Experion DCS system used on the Pilot Plant. One element that is introduced in the UWS system is the idea of Alarming. It was observed by the author that in the ENG346 class of 2009, and the ENG420 class of 2010, very little use was made of the Alarm system in the Pilot Plant\textsuperscript{15}. This seemed to be largely because students were unaware of, or unfamiliar with the concept of relying on alarms, and instead tried to keep track of the system manually, leaving alarms unacknowledged.

In order to ease the transition to the full DCS interface, simple alarming has been configured on the UWS. There are only 24 alarms in the UWS system, these have been implemented in a simple, area specific manner. They show up on the operator graphics and let the user know if an abnormal situation or even has occurred. It is hoped that by introducing a simple form of alarming it will ease the transition for students into more comprehensive alarming systems such as those used in the Experion DCS interface.

The alarms for the UWS fall into three categories, process alarms, trip alarms, and system alarms. Three alarming levels are used, Low, High, and Critical. Low Priority

\textsuperscript{15} At that stage it was operating Honeywell’s Scan3000 system, similar in its alarms to Experion.
alarms are Cyan, High Priority alarms are Yellow, and Critical Priority alarms are Red.

FIGURE 12-3 EXAMPLES OF ALARMS IN THE UWS

Process alarms include all HiHi, Lo, and LoLo float switches in the system. The majority are set to High Level, with the exception of the Hi level alarm in the Supply Tank (Low Priority, no immediate danger) and the LoLo level alarm in the supply tank (Critical, will trip almost all pumps in the system). TRIP alarms become active if a pump is stopped by an interlock condition, these are all set to critical priority as they require the user's immediate response. System alarms are used to let the user know if there is a communication or program error between the client and server, or server and controller. These are also critical alarms as they require immediate user action. The full list of alarms in the UWS Client is compiled in the project documentation as Appendix 1007:

UWS-1000   Datasheets and Reference\UWS-1007   Alarm Listing.pdf

12.4 UWS CLIENT CODE
This Section is a brief overview of the behaviour of the UWS Client code. A more comprehensive explanation of this code is available in Appendix 4003:

UWS-4000 Functional Descriptions\UWS-4003 Client Software Template.pdf

The main UWS program executes inside a timed loop structure. This loop (by default) is configured to execute every 500ms. Inside this timed structure a *stacked sequence* is used to contain each sub-section of code. As in the CompactRIO, sub-VIs are used extensively to hold code that is re-used throughout the project. Great lengths have been taken, and many comments placed throughout the code, to make it as simple as possible for future users to extend or modify the code. Templates have been put in the code for advanced users that wish to implement their own custom controller models instead of the in-built PID controllers.

Outside of this main executing loop, a ‘data extraction’ loop is fired off whenever the user presses the ‘Extract’ button to copy historical data to a spreadsheet file. This is placed in a separate loop to allow any controllers to continue executing synchronously in the main loop, while the extraction loop is waiting for user input.

12.4.1 **BOUND VARIABLES**

To get the current system values, the UWS Client uses *bound variables* to bind the front-panel controls to variables on the UWS Server. Both the read-only system process variables (such as tank levels, flows, etc) and the user-defined variables (pump speeds, valve open/close status, etc) are bound to front panel controls. In this manner, multiple copies of the UWS Client can be operating at once and any change made by one instance is reflected in all other instances of the client.

It was discovered during the course of this project that bound variable settings are *inherited from a sub-VI*. That is, if a sub-VI is generated that includes bound
variables as *input* variables, any time this sub-VI is used and the ‘create control’ method is used in the upper program, then this generated control is *automatically bound to the appropriate variable*. This greatly simplifies the generation of new programs that make use of the UWS Server interface. A document specifying this procedure is included as Appendix 3007:

UWS-3000  Standard Operating Procedures\UWS-3007  Controlling Motors using Pre-Built VIs.pdf

### 12.4.2 Interlock Descriptions

An additional function of the custom motor control VIs that have been generated for this project is to convert the *integer* value of the interlock (IL) tag into a *string* that is understandable by the user. The CompactRIO concatenates all of the interlock conditions for a piece of equipment into a single integer value for logging and communication. This could be used with a lookup-table (such as the one included in Appendix 1005), but that is not ideal for users less familiar with the UWS. For this reason, code was put in place to generate the descriptive string (from the interlock description) so that the user can read which interlocks are active. This interlock string is one output from the motor control VI. The other output is a ‘cluster’ of variables that generates the pump display, shown in Figure 12-4.

![Pump Display](image)

**FIGURE 12-4 PUMP DISPLAY**

This motor display in Figure 12-5 includes the run status of the Pump, the speed at which the pump is running (Bar) and a large black cross that is displayed on top of
the motor if the motor is interlocked. Using a ‘Tip Strip’ property node for the output cluster, it is possible to set the string containing the active interlocks on the pump to the ‘hover’ tip for the display. This means that if a pump is interlocked, the user just needs to hover the mouse cursor over the pump and the active interlocks will be displayed.

![Diagram of interlocked pump showing tip strip](image)

**FIGURE 12-5 INTERLOCKED PUMP, SHOWING TIP STRIP\(^\text{16}\)**

This lets the user know immediately what conditions need to be met before the pump can start, or what conditions have occurred causing the pump to stop.

### 12.4.3 System Health and Alarming

The UWS Client program makes use of the timestamp property of the ‘ScanTime’ variable to determine the current health of the system. The ‘ScanTime’ variable should be being updated every scan of the CompactRIO controller (200ms). All of the computer elements in the system are synchronised to the Murdoch SNTP server. Knowing this, and allowing for a slight deviation and/or network latency, this timestamp can be used by the UWS client program to monitor the health of the CompactRIO controller. If communication is lost to the server, an error is generated when trying to read the timestamp. This error is captured, and the ‘Server Comms’ alarm is activated. If no error is generated, but the timestamp returned is *null*, then the server has lost communication with the cRIO controller and the ‘cRIO Comms’

---

\(^{16}\) Mouse cursor not visible in this screen-capture.
alarm is generated. Finally, if no error is returned, the timestamp is not null but the timestamp is older than 10 seconds, the ‘cRIO Program’ alarm is generated as the program on the cRIO has for some reason stopped executing.

12.4.4 CUSTOM CONTROLLER MODELS

More advanced users might not wish to use the in-built PID controllers that have been provided with the UWS client template. These users can take use of a designated ‘Custom Controller’ panel in the UWS Client code. In frame 8 of the main program sequence, an area has been left for users to build custom controllers apart from the included PID controller. The custom controller output writes the value directly to the Pump speed (or valve setting) on the front panel using a ‘Value’ property node. This means that when the custom controller is enabled it overrides the manual and PID output, and the custom controller output is shown on the front panel.

There is no reason to keep the stacked sequence structure for the custom controllers, these could be placed outside of the main program loop altogether and just use ‘value’ property nodes to write back out to the desired values. A detailed procedure for adding custom controllers to the UWS Client has been generated and is included as Appendix 3008:

UWS-3000 Standard Operating Procedures\UWS-3008 Adding Custom Controllers to UWS Client.pdf
12.5 User Interface

The UWS Client program is made up of 6 screens, the user can move between screens by using either the tab strip along the top of the screen, or by using the white navigation arrows.

<table>
<thead>
<tr>
<th>System-wide Overview</th>
<th>Shows all System Values, Read-Only System Alarms and Supply Tank Level Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area A</td>
<td>Alarms and control for Pumps 01, 02 and 09; tanks 01 and 03 Associated Valves and Flows</td>
</tr>
<tr>
<td>Area B</td>
<td>Alarms and control for Pumps 05, 06 and 07; tank 05 Associated Valves and Flows</td>
</tr>
<tr>
<td>Area C</td>
<td>Alarms and control for Pumps 03, 04 and 08; tanks 02 and 04 Associated Valves and Flows</td>
</tr>
<tr>
<td>Historical Trends</td>
<td>Embedded NI Hypertrend to view Historical Trends Pre-configured trend 'Groups'</td>
</tr>
<tr>
<td>Data Extraction</td>
<td>Exporting of Historical Data to Excel Format Allows user to select variables, time range and sampling time</td>
</tr>
</tbody>
</table>

**FIGURE 12-6 PAGES OF THE UWS CLIENT INTERFACE**

The different Areas that the plant is broken up into are chosen based upon the placement of the manual isolation valves. These areas can operate as a continuous unit with flow going from one to another, or they can operate individually, with each area only taking water from, and returning water to, the supply tank. A more detailed explanation of the UWS client and an introduction to each of these pages is included in the project documentation as the UWS Client user guide, Appendix 4004:

UWS-4000   Functional Descriptions\UWS-4004   UWS Client User Guide.pdf
12.5.1 CONTROLLING THE PLANT

As well as allowing users to manually control the plant, the UWS Client template includes basic PID loops to allow for flow and level control. To put a pump or flow valve into PID control mode, the user can click the mode button to toggle the system from MAN (manual) to AUT (automatic) mode. The PID tuning parameters are available by clicking on the PID tab at the top of the device controller block.

A design requirement for the system was to be able to dynamically change the control loop pairing of the system. When setting the tuning parameters for a device, it is possible to configure any output to control any of the process variables. This is achieved by using the PV selection drop down list on the PID tuning parameters tab of the device control block.
This PV selection allows users to test the system with a range of different control loop pairings, without having to modify the code. The default selection for each control loop is the level of the tank immediately downstream from the element.

12.5.2 Trends

The UWS Client program has a trending page, which allows the user to view trends (plots) of the system variables. The trends can be viewed either in real-time (scrolling plots) or the user can view any point in time that is logged in the UWS Server database. Trends are highly useful and allow the user to interrogate the performance of their system and detect poorly tuned control loops more easily.

The ‘Trends’ tab on the UWS Client program uses the NI Hypertrend trending system, part of the NI DSC Module. Hypertrend allows the user to configure custom groups and add/edit the traces visible in each group. Help is available by right-clicking on the Hypertrend interface and selecting ‘Help..’. A detailed procedure for adding and modifying trends has been generated and is included as Appendix 3009:

UWS-3000 Standard Operating Procedures\UWS-3009 Adding and Modifying Trends.pdf
To edit the visible traces on the current trend group, right click on the trend and select Properties. To switch between groups of traces, right click on the trend and choose Change Current Group.
The UWS Client comes with the following standard set of trends. These trends are saved in the local copy of the UWS Template file. A user can customise the trends and groups and save the changes.

**TABLE 12-1 DEFAULT TRENDS FOR UWS CLIENT**

<table>
<thead>
<tr>
<th>Area A</th>
<th>Name</th>
<th>Variable</th>
<th>Description</th>
<th>Units</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LI01</td>
<td>LI01.PV</td>
<td>Tank 01 Level</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>LI03</td>
<td>LI03.PV</td>
<td>Tank 03 Level</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>FI01</td>
<td>FI01.PV</td>
<td>Flow Meter 01</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
<tr>
<td></td>
<td>FI04</td>
<td>FI04.PV</td>
<td>Flow Meter 04</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
<tr>
<td></td>
<td>FI07</td>
<td>FI07.PV</td>
<td>Flow Meter 07</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
<tr>
<td></td>
<td>PU01</td>
<td>PU01.SY</td>
<td>Pump 01 Speed</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>PU09</td>
<td>PU09.SY</td>
<td>Pump 09 Speed</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>FV01</td>
<td>FV01.MV</td>
<td>Flow Valve 01 Setting</td>
<td>%</td>
<td>0-100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Area B</th>
<th>Name</th>
<th>Variable</th>
<th>Description</th>
<th>Units</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LI05</td>
<td>LI05.PV</td>
<td>Tank 05 Level</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>FI03</td>
<td>FI03.PV</td>
<td>Flow Meter 03</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
<tr>
<td></td>
<td>FI06</td>
<td>FI06.PV</td>
<td>Flow Meter 06</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
<tr>
<td></td>
<td>PU05</td>
<td>PU05.SY</td>
<td>Pump 05 Speed</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>PU07</td>
<td>PU07.SY</td>
<td>Pump 07 Speed</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>FV03</td>
<td>FV03.MV</td>
<td>Flow Valve 03 Setting</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>FV04</td>
<td>FV04.MV</td>
<td>Flow Valve 04 Setting</td>
<td>%</td>
<td>0-100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Area C</th>
<th>Name</th>
<th>Variable</th>
<th>Description</th>
<th>Units</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LI02</td>
<td>LI02.PV</td>
<td>Tank 02 Level</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>LI04</td>
<td>LI04.PV</td>
<td>Tank 04 Level</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>FI02</td>
<td>FI02.PV</td>
<td>Flow Meter 02</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
<tr>
<td></td>
<td>FI05</td>
<td>FI05.PV</td>
<td>Flow Meter 05</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
<tr>
<td></td>
<td>PU03</td>
<td>PU03.SY</td>
<td>Pump 03 Speed</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>PU08</td>
<td>PU08.SY</td>
<td>Pump 08 Speed</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>FV02</td>
<td>FV02.MV</td>
<td>Flow Valve 02 Setting</td>
<td>%</td>
<td>0-100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>All Flows</th>
<th>Name</th>
<th>Variable</th>
<th>Description</th>
<th>Units</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FI01</td>
<td>FI01.PV</td>
<td>Flow Meter 01</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
<tr>
<td></td>
<td>FI02</td>
<td>FI02.PV</td>
<td>Flow Meter 02</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
<tr>
<td></td>
<td>FI03</td>
<td>FI03.PV</td>
<td>Flow Meter 03</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
<tr>
<td></td>
<td>FI04</td>
<td>FI04.PV</td>
<td>Flow Meter 04</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
<tr>
<td></td>
<td>FI05</td>
<td>FI05.PV</td>
<td>Flow Meter 05</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
<tr>
<td></td>
<td>FI06</td>
<td>FI06.PV</td>
<td>Flow Meter 06</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
<tr>
<td></td>
<td>FI07</td>
<td>FI07.PV</td>
<td>Flow Meter 07</td>
<td>Litres/min</td>
<td>0-50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>All Levels</th>
<th>Name</th>
<th>Variable</th>
<th>Description</th>
<th>Units</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LI01</td>
<td>LI01.PV</td>
<td>Tank 01 Level</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>LI02</td>
<td>LI02.PV</td>
<td>Tank 02 Level</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>LI03</td>
<td>LI03.PV</td>
<td>Tank 03 Level</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>LI04</td>
<td>LI04.PV</td>
<td>Tank 04 Level</td>
<td>%</td>
<td>0-100</td>
</tr>
<tr>
<td></td>
<td>LI05</td>
<td>LI05.PV</td>
<td>Tank 05 Level</td>
<td>%</td>
<td>0-100</td>
</tr>
</tbody>
</table>
12.5.3 Data Extraction

The final task of the UWS Client is to allow for simple extraction of historical data into a spreadsheet file. The Client runs a secondary program loop (executing in parallel with the main program loop) that generates the trends using the LabVIEW shared variable extract to spreadsheet VI. The UWS Client can be used to extract Historical data to a .CSV spreadsheet format. This can be achieved in the Extract tab of the UWS Client program.

![Data Extraction Interface](image)

**FIGURE 12-11 DATA EXTRACTION INTERFACE**

The user then just needs to click on the variable names to choose variables to extract, then use the **sampling times** panel to choose the start/stop and sampling rate of the data. When the **Extract** button is pressed, the user will be prompted for a location to save the data and the data will be saved into standard CSV format. A detailed procedure for this data extraction is included as Appendix 3010:

[UWS-3000 Standard Operating Procedures](#) | [UWS-3010 Extracting Historical Data to Excel.pdf](#)
13 Testing and Operating the UWS

Throughout the project, each section of the code went into operation almost immediately after generation. Particularly in the case of the UWS Client program, where early revisions of this program were in operation even as later revisions were being created. The ‘feel’ of running the plant was tested and on most days the plant was under PID control by the UWS Client. On top of this general operation testing, many specific tests were carried out after each section of the project was completed.

13.1 Tests Performed

The list of tests that the UWS control system needed to pass was compiled during the design phase, with some additions throughout the project. This testing allowed for many bugs to be found and was necessary to fully commission the UWS control system. The UWS control system was able to successfully pass the following tests, performed throughout the project:

1. Recover from Power Loss, for any individual component, or a system-wide power loss.

   This was tested many times (both intentionally and un-intentionally) and the UWS could recover to an operating state after any power loss to a Client or the Server. If the cRIO loses power, when the system comes back up it will have all the pumps off and the valves closed, but be ready for the user to restart/open them.

2. Allow for multiple Client connections.

   This was tested with multiple machines operating LabVIEW 2009, and with multiple copies of the UWS Client open on single machines. In both cases, the system performs as designed and Users can control the system from any instance of the client.
3. **Provide Interlocking for System and Process Protection.**

All interlocks and alarms were *specifically* tested, by creating the condition that would cause the alarm or interlock to occur. This procedure is time consuming, but ensures that the system in its entirety is behaving as designed. A detailed list of the interlocks including their testing date is included as Appendix 1005:

UWS-1000  Datasheets and Reference\UWS-1005  Pump Interlocks.pdf

4. **Operate in a Safe Manner after Communication Loss.**

This was tested by unplugging the network cable that led from the CompactRIO to the rest of the system. The cRIO would continue to operate as it was prior to the loss of communication. If any of the interlock conditions became true, the pump/valve in question would be stopped/closed.

5. **Operate continuously for an extended period of time**

The UWS cRIO and Server ran continuously for weeks at a time on many separate occasions over the course of the project. The only restarting that occurred was after changes were made to one of the elements. The UWS Client on two separate occasions operated for over 10 hours continuously while controlling the plant with PID controllers, this was only limited by the fact that Murdoch regulations don’t allow for the plant to be operated unattended. The UWS Client itself ran for many days while just monitoring the (idle) plant. There is no reason to believe that the UWS Client | Server | Controller system could not run indefinitely\(^\text{17}\) if required.

\(^{17}\) Ignoring hardware or power breakdowns.
6. Provide for data logging, trending and extraction.

This has been tested and was used regularly while operating the plant. During this testing it was discovered that the trend groups for NI Hypertrend were saved in the local LabVIEW program, and thus the default groups were generated. During this testing the data extraction loop was moved outside of the main program as the controller code would otherwise have to wait for the user to select their save location and filename.

![Trend Showing Tank Levels](image)

**FIGURE 13-1 TREND SHOWING TANK LEVELS**

13.2 Alternate Configurations

The system can be broken up into separate areas, or combined together in a continuous system. In order to do this, the manual isolation valves in the system must be used. The manual isolation valves are the black lever-type valves that are on the lowest level of the UWS. These include valves to isolate each pump (for maintenance) as well as valves to allow or remove flow between sections of the system. Figure 13-2 shows the UWS configured to be a single continuous system.
Figure 13-3 shows the UWS configured as three discrete systems, isolated from each other.

Care must be taken to ensure that all of the *suction* valves for a pump are not closed at the same time. This can cause an unsafe situation and could damage the pump and the pipe-work. The manual valves *should never be changed while the system is running*. A reference sheet showing the manual valve configuration for each of these systems has been generated, and is available as Appendix 1008:

[UWS-1000 Datasheets and Reference\UWS-1008 Manual Valve Configuration.pdf](UWS-1000 Datasheets and Reference\UWS-1008 Manual Valve Configuration.pdf)
14 Documentation

The UWS control system has been entirely custom created for this project, large amounts of new code and interfaces have been constructed. Due to this, a critical part of the project was the thorough documentation of all aspects of the system. While great care has been taken throughout this project to include useful and descriptive comments throughout the LabVIEW code, a set of structured and comprehensive documents have also been generated for future users of the UWS.

Proper documentation of a project allows future users to familiarise themselves with the control system without having to study every aspect of the system. The documentation for this project consists of five main categories.

<table>
<thead>
<tr>
<th>Project Documentation Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drawings and Diagrams (0xxx)</td>
</tr>
<tr>
<td>Piping and Instrument Drawings</td>
</tr>
<tr>
<td>Wiring and Termination Drawings</td>
</tr>
<tr>
<td>Electrical Power Schematics</td>
</tr>
<tr>
<td>Datasheets and Reference (1xxx)</td>
</tr>
<tr>
<td>Variable Library Reference</td>
</tr>
<tr>
<td>Interlock Listing + Testing Dts</td>
</tr>
<tr>
<td>Network Details and Manual Valve Configuration</td>
</tr>
<tr>
<td>UWS Software (2xxx)</td>
</tr>
<tr>
<td>CompactRIO and UWS Server Project Backups</td>
</tr>
<tr>
<td>UWS Client Program</td>
</tr>
<tr>
<td>Custom Controls Library and Templates</td>
</tr>
<tr>
<td>Standard Operating Procedures (3xxx)</td>
</tr>
<tr>
<td>Step by step procedures for tasks around the UWS</td>
</tr>
<tr>
<td>Short, specific procedures</td>
</tr>
<tr>
<td>Can be used without previous knowledge</td>
</tr>
<tr>
<td>Functional Descriptions (4xxx)</td>
</tr>
<tr>
<td>In-depth documentation of different parts of the system</td>
</tr>
<tr>
<td>Code description for cRIO, UWS Server, and UWS Client</td>
</tr>
<tr>
<td>UWS Client User Guide</td>
</tr>
</tbody>
</table>

**FIGURE 14-1 PROJECT DOCUMENTATION**

These categories form the directory structure for the documentation, every document in the project has been assigned a 4-digit numerical reference. The
number ranges for these references reflect the nature of the document. That is, the reference number for any drawing or diagram follows the pattern 0xxx, datasheets have the pattern 1xxx, and so on.

14.1 DRAWINGS AND DIAGRAMS (0XXX)

The documents in this category relate to the physical hardware of the system, these documents were created by Rick Janosz during the construction of the system hardware. With the exception of the system P&ID, all the documents in this category remain unchanged from those generated by Rick (1).

UWS-0000  Drawings and Diagrams

14.2 DATASHEETS AND REFERENCE (1XXX)

These documents were generated during the design phase of this project. The datasheets contain the details of all tags in the system, the interlocks, the alarming and network details. There is also a manual valve configuration listing. Creating these reference sheets so early in the project meant that they could be used for the remainder of the project as a basis for coding and testing. The use of these datasheets was greatly responsible for keeping continuity and coherence across the project.

UWS-1000  Datasheets and Reference

14.3 UWS SOFTWARE (2XXX)

This folder contains the current revisions of all of the LabVIEW software and configuration files for this project. The main program, sub-VIs, library and build specifications for the CompactRIO controller; the library, and utility programs for the UWS Server; the UWS Client LabVIEW project, with associated sub-VIs; and
finally the full collection of custom controls (with templates) generated for this project.

14.4 STANDARD OPERATING PROCEDURES (3xxx)

The standard operating procedure documents (SOPs) contain simple, step by step instructions for a range of procedures that might need to be carried out by future users of the UWS. The purpose of the UWS is to allow users to carry out procedures and respond to specific situations that may arise. Without these SOPs the user would have to read the full code description (4xxx series) and/or study and problem solve the system in order to achieve these (relatively) simple goals. The user can use these SOPs as a simple reference instead of having to remember/learn all of the procedures.

14.5 FUNCTIONAL DESCRIPTIONS (4xxx)

The functional description documents are the longest and most in-depth of the project documentation, the length of these documents ranges from ~1.5k words to ~3k words. The functional descriptions include full description of the tasks, method, and code for the CompactRIO, UWS Server, and UWS Client. An additional FD is included which is an Operator User-Guide to the UWS Client interface. The user-guide is a starting point for any new user of the UWS, and the FDs for the other parts of the system would be required reading for anyone wishing to make major modifications to the system.
14.6 CHAD WEBSITE

The final stage of this documentation process was the updating of the ‘chad’ website for the Universal Water System. The chad site (chad.murdoch.edu.au) contains web-based references for engineering projects at Murdoch. Prior to this project the chad website for the UWS had not been updated since 2007, the content was vastly out-dated as at that time only the basic hardware design and early construction had started. This website has been completely rebuilt and now contains up-to-date information about the UWS and all of the project documentation generated for this project.

http://chad.murdoch.edu.au/~uws

All of the project documentation and software has been published to the UWS folder on the student shared ‘MyLab’ machine. This is accessible by all engineering students on the Murdoch network.

\MyLab\EngShared\UWS
15PROJECT CLOSE-OUT AND FUTURE DIRECTIONS

All of the requirements specified in the exploration and design phases for this project have been achieved. The project has been thoroughly documented and a handover package generated for future users of the project.

This project highlights the importance of clear and comprehensive design. Before any code was generated for the project, all of the data reference sheets had been constructed and the behaviour of each element in the system had been planned. The advantages of this are many, during the coding phases, all focus could be placed on the code itself, rather than spending time trying to figure out how to go about things. This made the coding much easier and the datasheets could be printed and always at-hand while working on the UWS. Also during testing, the system could be verified against the design specifications at the completion of every stage of development. This ensured consistency and meant that each stage of the project could build directly upon the previous without having to go back and re-test or modify earlier stages. The project design datasheets were given to Kan Sumontha, a Masters student who is working on building sample controllers for the UWS in Excel and MATLAB, as well as an Excel plugin for OPC.

The Universal Water System is in the process of being moved to the new Murdoch University Engineering building. The system is expected to be re-commissioned in January 2011. Once this is complete, the UWS is ready to be used as part of the engineering curriculum for undergraduates. The UWS can be used in a range of the offered Engineering units across 2nd, 3rd and 4th years.

It is the hope of the author that future engineering students at Murdoch can build the same hate/love/hate relationship with the UWS that the current students have with the Pilot Plant.
## 16 Appendices

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Name</th>
<th>Type</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UWS-0000</strong></td>
<td>Drawings and Diagrams Directory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UWS-0001</td>
<td>UWS P&amp;ID</td>
<td>Zipped Directory</td>
<td>Janosz, R; Blay, K</td>
</tr>
<tr>
<td>UWS-0002</td>
<td>Module 1 DI Sheet 1 I/O Termination</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0003</td>
<td>Module 1 DI Sheet 2 I/O Termination</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0004</td>
<td>Module 2 DO I/O Termination</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0005</td>
<td>Module 3 DO I/O Termination</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0006</td>
<td>Module 4 AI I/O Termination</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0007</td>
<td>Module 5 AO I/O Termination</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0008</td>
<td>Module 6 AO I/O Termination</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0009</td>
<td>Module 7 AO I/O Termination</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0010</td>
<td>Module 8 HSD</td>
<td>I/O Termination</td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0011</td>
<td>Pump 01 Electrical Schematic Electrical Schematic</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0012</td>
<td>Pump 03 Electrical Schematic Electrical Schematic</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0013</td>
<td>Pump 05 Electrical Schematic Electrical Schematic</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0014</td>
<td>Pump 08 Electrical Schematic Electrical Schematic</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0015</td>
<td>Pump 09 Electrical Schematic Electrical Schematic</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0016</td>
<td>Load Cell Schematic Electrical Schematic</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0017</td>
<td>Pump 02 Electrical Schematic Electrical Schematic</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0018</td>
<td>Pump 04 Electrical Schematic Electrical Schematic</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0019</td>
<td>Pump 06 Electrical Schematic Electrical Schematic</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td>UWS-0020</td>
<td>UWS Power Schematic Power Schematic</td>
<td></td>
<td>Janosz, R</td>
</tr>
<tr>
<td><strong>UWS-1000</strong></td>
<td>Datasheets and Reference Directory</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-1001</td>
<td>cRIO Library Tag Library</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-1002</td>
<td>Pump Data Structure Data Structure Reference</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-1003</td>
<td>Valve Data Structure Data Structure Reference</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-1004</td>
<td>Instrument Data Structure Data Structure Reference</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-1005</td>
<td>Pump Interlocks Interlock Listing</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-1006</td>
<td>Network Details TCP/IP Configuration</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-1007</td>
<td>Alarm Listing Alarm Listing</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-1008</td>
<td>Manual Valve Configuration Manual Valve Configuration</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-1009</td>
<td>Library Backups Directory</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-1019</td>
<td>User Manuals and Guides Directory</td>
<td></td>
<td>Various</td>
</tr>
<tr>
<td><strong>UWS-2000</strong></td>
<td>Software Directory</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-2001</td>
<td>cRIO Software LabVIEW Project</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-2002</td>
<td>UWS Server Software LabVIEW Project</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-2003</td>
<td>UWS Client Software LabVIEW Project</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-2004</td>
<td>Custom Control Library Zipped Directory</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-2005</td>
<td>Superseded Software Zipped Directory</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td><strong>UWS-3000</strong></td>
<td>Standard Operating Procedures Directory</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-3001</td>
<td>Restarting the cRIO Standard Operating Procedure</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-3002</td>
<td>Opening cRIO Project Standard Operating Procedure</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-3003</td>
<td>Editing and Deploying cRIO Project Standard Operating Procedure</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-3004</td>
<td>Configuring SNTP on cRIO Standard Operating Procedure</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-3005</td>
<td>Editing the Server Library Standard Operating Procedure</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-3006</td>
<td>Customising LabVIEW Controls Standard Operating Procedure</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-3007</td>
<td>Using custom Motor Vis Standard Operating Procedure</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-3008</td>
<td>Adding Custom Controllers to Client Standard Operating Procedure</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-3009</td>
<td>Adding and Modifying Trends Standard Operating Procedure</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-3010</td>
<td>Extracting Historical Data to Excel Standard Operating Procedure</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td><strong>UWS-4000</strong></td>
<td>Functional Descriptions Directory</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-4001</td>
<td>cRIO Functional Description Code Functional Definition</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-4002</td>
<td>UWS Server Functional Description Code Functional Definition</td>
<td></td>
<td>Blay, K</td>
</tr>
<tr>
<td>UWS-4003</td>
<td>Client Software Template Code Functional Definition</td>
<td></td>
<td>Blay, K</td>
</tr>
</tbody>
</table>
17 References


8. **National Instruments** LabVIEW In-Program Context Help. - 2010.


