

University of Southern Queensland
Faculty of Engineering & Surveying

**Real-Time Monitoring and Control of a
Pressure Control System**

A dissertation submitted by

Craig Struthers

in fulfilment of the requirements of

ENG4112 Research Project

towards the degree of

Bachelor of Engineering (Electrical / Electronic)

Submitted: October, 2005

Abstract

Constant pressure control within a hydraulic pumping system is conventionally performed by the use of an electric pump with a variable speed drive controlling the speed of the pump. A pressure transmitter is used as a feedback to control the speed of the pump which in turn, also controls the pressure in the system. The aim of this project is to develop and test an automated pumping controller, which is able to maintain constant pressure in a hydraulics system without any physical contact with the medium. The medium in this instance being a pressure transmitter.

This project was chosen in order to address the lack of products currently available within the industrial control field for pump pressure control. Personal experience has presented a number of instances where applications have required a pressure feedback but because the medium is so corrosive and/or dangerous the pressure transmitter becomes prohibitively expensive.

Along with the control system itself, a user interface was developed to operate over the ethernet ensuring the ability to utilise current WEB server interfaces such as windows Explorer and the like.

University of Southern Queensland
Faculty of Engineering and Surveying

ENG4111/2 *Research Project*

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Engineering and Surveying, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Engineering and Surveying or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled “Research Project” is to contribute to the overall education within the student’s chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Prof G Baker

Dean

Faculty of Engineering and Surveying

Certification of Dissertation

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

CRAIG STRUTHERS

Q97202949

Signature

Date

Acknowledgments

The author would like to thank DR Peng (Paul) Wen for his guidance and feedback during the course of this project work. The author also thanks his employer, TRITEC Electrical Controls and Automation PTY LTD, for support, both monetary and time-off, during the course of this project work and all previous subjects undertaken at the University of Southern Queensland towards the fulfillment of the Bachelor of Engineering degree program. The author would also like to extend a very big thank you to his wife Karen Struthers and son Jordan Struthers for both their patience and help throughout the studying process.

CRAIG STRUTHERS

University of Southern Queensland

October 2005

Contents

Abstract	i
Acknowledgments	iv
List of Figures	x
List of Tables	xiii
Chapter 1 Introduction	1
1.1 Introduction	1
1.2 Statement of the problem	1
1.3 Aims and Objectives	2
1.3.1 Develop and test an automated pumping controller	2
1.4 Significance of Study	3
1.5 Scope and Limitation of Study	4
1.6 Dissertation Layout	5

CONTENTS	vi
<hr/>	
Chapter 2 Literature Review	6
2.1 Current environment	6
2.2 Overview of pumping system	9
2.3 Operating Philosophy of a Pump	10
2.4 Variable Speed Drive (VSD)	12
2.4.1 Purpose of a VSD	12
2.4.2 Programming Software for the VSD	14
2.4.3 Serial Communications	18
2.4.4 MODBUS Protocol	19
2.5 WEB Server and Java Applications	20
2.5.1 WEB Server	20
2.5.2 Java Applications	21
2.6 Pump and Motor equations	22
2.7 Current direction in pump pressure control	27
2.8 Summary	28
 Chapter 3 Methodology	 29
3.1 Resource Planning	29
3.1.1 Equipment	29

CONTENTS	vii
3.1.2 Facilities	34
3.1.3 Computers	34
3.1.4 Software	34
3.2 Construction of Test Equipment	35
3.3 Programming of the VSD	38
3.4 Programming of the Web-interface	39
Chapter 4 Results and Discussions	40
4.1 Introduction	40
4.2 Data	41
4.2.1 Open Head System Tests	41
4.2.2 Closed Head System Tests	42
4.2.3 Fixed Speed Tests	42
4.2.4 Closed Loop System Tests with Pressure Transducer	49
4.2.5 Closed Loop System Tests using Algorithm	51
4.3 Comparison of Pressure Transducer and Algorithm	53
4.4 WEB Server - Interface	54
4.4.1 Software Development	54
4.4.2 Connection and Testing	56

CONTENTS	viii
4.5 Obstacles	57
Chapter 5 Future Works	59
Chapter 6 Conclusions	61
6.1 Overview and Obstacles	61
6.2 Pressure System Controlled without a Pressure Transmitter	62
6.3 Real-time Monitoring and Control System	63
6.4 Final Summary	63
References	65
Appendix A Project Specification	68
Appendix B Variable Speed Drive (VSD) Operating Manual	71
Appendix C MODBUS Protocol & VSD Registers	88
C.1 MODBUS Protocol	89
C.2 VSD Registers	93
Appendix D JAVA Program	99
Appendix E PDL Communications Software	116
E.1 Drivelink Interface	117

CONTENTS

E.2 Drivecom Software	130
Appendix F Vysta Screen List of the Algorithm Test Program	133
Appendix G DAVEY Pump Data Sheets	135

List of Figures

2.1	Illustration of existing Pressure Control System.	7
2.2	Illustration of proposed Pressure Control System.	8
2.3	Illustration of a typical Pump Curve.	10
2.4	Illustration of the components of a Centrifugal Pump.	11
2.5	Illustration of the system architecture of a VSD.	13
2.6	This is an example of a VYSTA Schematic Connection Diagram .	15
2.7	This is an example of VYSTA Schematic Oval Function Block Menu	16
2.8	This is an example of a Read Variable Dialog Box	16
2.9	This is an example of a VYSTA Screen Layout	17
3.1	Closed Loop Testing System PI Diagram	36
3.2	This Photograph is of Closed Loop Testing System's Pump and Pressure Transmitter	36
3.3	This Photograph is of the Closed Loop Testing System	37

LIST OF FIGURES

3.4	This Photograph is of the VSD	37
4.1	Plot of Pressure and Motor Current with speed being varied for 0% - 100% with the pump outlet valve fully open	41
4.2	Plot of Pressure and Motor Current with speed being varied for 0% - 100% with the pump outlet valve fully closed	42
4.3	Plot of Pressure and Motor Current when speed is held at 30% . .	44
4.4	Plot of Pressure and Motor Current when speed is held at 40% . .	45
4.5	Plot of Pressure and Motor Current when speed is held at 50% . .	45
4.6	Plot of Pressure and Motor Current when speed is held at 60% . .	46
4.7	Plot of Pressure and Motor Current when speed is held at 70% . .	46
4.8	Plot of Pressure and Motor Current when speed is held at 80% . .	47
4.9	Plot of Pressure and Motor Current when speed is held at 90% . .	47
4.10	Plot of Pressure and Motor Current when speed is held at 100% .	48
4.11	Plot showing closed loop system Pressure - with pressure trans- mitter feedback	49
4.12	closed loop system Vysta Program - with pressure transmitter feed- back	50
4.13	Plot showing closed loop system Pressure - with pressure trans- mitter feedback	51
4.14	closed loop system Vysta Program - with pressure transmitter feed- back	52

LIST OF FIGURES

4.15 Illustration of Control System JAVA GUI	54
--	----

List of Tables

2.1	MODBUS protocol subset utilised by the PDL Elite Series VSD .	19
4.1	GUI Control Functions	55

Chapter 1

Introduction

1.1 Introduction

This dissertation completes the final requirements for the project work undertaken in the combined semester 1 and 2 units ENG4111 and ENG4112 Research Project for the final year of the Bachelor of Engineering degree program at the University of Southern Queensland. It follows the submissions of the Project Appreciation, the Project Specification appendix A and an oral presentation conducted during week one of September 2005 Residential School. It is the formal outcome of the research, planning and completion of the project topic as chosen by the author.

1.2 Statement of the problem

Each year there are many more pumps in use than are actually being supplied new (Hydraulic Institute, Europump & the U.S. Department of Energys (DOE) Industrial Technologies Program 2004). Pumps are utilised in almost all aspects of industry and engineering with an almost endless assortment of pumping equipment available ranging in size, type and material of construction. Research

contributing to advancements in pumping technology can potentially create immense benefits for a large proportion of the engineering and industry sectors. With today's focus on energy efficiency and sustainability, the current pumping environment provides an ideal opportunity to modify current pumping systems in order to achieve a more efficient process. One way of achieving this goal is by reducing the number of components within the pumping system itself. Notwithstanding the environmental advantages that this would provide, the added benefits of removing a component from a conventional pump system include ease of installation, reduction in the labour content, cost savings, and improved reliability of the system.

Currently, within the industrial control field there appears to be a deficiency in products designed specifically for the control of pumping systems. This is particularly evident in the application of constant pressure in a hydraulics system. The current industry standard is to perform this process with multiple units and a pressure transmitter feedback. This project proposes to develop a pumping controller able to maintain constant pressure within the hydraulics system without utilising a pressure transmitter and controller. Thus removing the pressure transmitter component of the pumping system.

1.3 Aims and Objectives

1.3.1 Develop and test an automated pumping controller

The specific aim of this project as stated in the Project Specification appendix A is to develop and test an automated pumping controller, which is able to maintain constant pressure in a hydraulics system. The main purpose of this undertaking is to ensure that the controller will have no physical contact with the medium. The term medium refers specifically to the pressure transmitter component of the pumping system. In completing the functionality of this project, a real-

time monitoring, configuration and control system software package is also being developed. The purpose of this is to enable users to interface with the system and achieve a greater level of interactivity and functionality through the use of a web browser.

1.4 Significance of Study

This project was chosen in order to find a practical solution to achieving pressure feedback in pumping applications with highly corrosive or dangerous mediums. At present, this particular aspect of the pumping industry does not appear to be adequately addressed. Although pressure transmitters are available for mediums that are considered dangerous and corrosive the cost of this component is highly prohibitive. Furthermore, replacement or calibration of the pressure transmitter component can be difficult and again not cost effective under such severe conditions.

Research into current initiatives reveals that the main developments in sensorless pump control are being undertaken predominantly within the medical field. In common with most elements of business, the medical industry has been challenged to lower costs, improve efficiencies and comply with environmental requirements. Demand for improved technology that enables people to enjoy healthy and productive lives has provided opportunities for engineers to modify pumping systems such as artificial hearts and blood pumps. This has led to such developments as the sensorless control of pump flow. The objective being to remove the invasive pressure transmitter component from within the human body.

Despite the obvious benefits to the medical industry however, the foundation of this project is on the application of this technology within the industrial field.

1.5 Scope and Limitation of Study

The scope and limitations of the work undertaken for this project is as follows:

1. The work undertaken has been conducted only within the offices of TRITEC Electrical Controls & Automation PTY LTD.
2. TRITEC Electrical Controls & Automation PTY LTD have contributed resources to this project with the understanding that any developments resulting from this study will remain and become the property of this company.
3. The project outcomes are applicable for pumping systems within the industrial sector only.
4. The project focus is specifically towards single pump use rather than multi-stage or multiple pump systems.
5. All items and equipment utilised in the project have been selected on the basis of size and cost considerations.
6. The majority of work in this project is based largely on the author's experience within this field as minimal research on this particular topic could be identified.
7. The project is subject to limitations revealed through the process of undertaking the project itself which are identified in 4.5.

1.6 Dissertation Layout

This dissertation is organised into six chapters including a number of tables and appendices. The outline of the chapters is as follows:

Chapter 1 provides an introduction to the dissertation, describing briefly the current industry standard, the aims and objectives of the project, the significance of this study and the scope and limitations of the project itself.

Chapter 2 encompasses a literature review providing background on the current pumping environment, the operating philosophy of a pump system, the Variable Speed Drive, the WEB Server and JAVA applications, pump and motor equations and the current direction in pump pressure control.

Chapter 3 consists of the methodology undertaken to achieve the aims and objectives of the project. It provides technical specifications of the equipment used and discusses the procedures undertaken to construct the test equipment as well as those to program the VSD and WEB-interface.

Chapter 4 is highly significant in that it provides the results obtained from the various testing undertaken. Analysis of the results is provided along with discussion of the programming outcomes and obstacles encountered as part of the project process.

Chapter 5 examines areas intended to be addressed in future works. It includes the possibility of multiple VSD applications along with further fine-tuning to the WEB server Graphical User Interface.

Chapter 6 provides a conclusion examining the extent to which the project aims and objectives were achieved in relation to the data obtained and analysis of the final program written.

Chapter 2

Literature Review

2.1 Current environment

Every year there are a far greater number of pumps in use than are actually being supplied new (Hydraulic Institute et al. 2004). Pumps are utilised in almost all aspects of industry and engineering, and range widely from feeds to reactors, distillation columns in chemical engineering to pumping storm water in civil and environmental engineering. There is an almost endless assortment of pumping equipment available ranging in size, type and material of construction (Nelik 1999, Karassik, Krutzsch, Fraser & Messina 1986). With today's focus on energy efficiency and sustainability, the current pumping environment provides an ideal opportunity to modify current pumping systems in order to achieve an efficient process that utilises less components (Hydraulic Institute et al. 2004). This project proposes to remove the pressure transmitter component of pump pressure control in order to achieve the reduction of a component. In addition to environmental advantages, the added benefits of removing the pressure transmitter from a conventional pump system include ease of installation; overcoming the requirement to have a pressure transmitter signal mounted in the field at a remote distance from the pump; reduction in the labour content of installing a

pressure transmitter and the associated cabling which sometimes can be a long distance from the pump; cost savings; and improved reliability of the system as the new system would limit the components prone to failure.

Currently, within the industrial control field there appears to be a deficiency in products designed for the control of pumping systems. This is particularly evident in the application of constant pressure in a hydraulics system. The current industry standard is to perform this process with multiple units and a pressure transmitter feedback. Figure 2.1 demonstrates how pressure control in a hydraulic pumping system is currently achieved. Note in particular the use of a pressure transmitter in conjunction with a controller and a Variable Speed Drive.



Figure 2.1: Illustration of existing Pressure Control System.

This project proposes to develop a pumping controller able to maintain constant pressure within the hydraulics system without utilising a pressure transmitter and controller. Figure 2.2, represents the proposed pressure control system whereby all the pressure control is performed within the Variable Speed Drive itself, the WEB server is simply a remote user interface. In order to understand how pres-



Figure 2.2: Illustration of proposed Pressure Control System.

sure control will be achieved without using a pressure transmitter it is necessary to have a broad understanding of how a pump works.

2.2 Overview of pumping system

For the purposes of this project pumped material is described in terms of fluid. It must be acknowledged that some pumps can manage solids, however, the material must demonstrate an overall liquid behaviour to do so (Nelik 1999). Arguably the most fundamental means of categorising pumps is by the way in which energy is conveyed to the pumped fluid (Nelik 1999, Karassik et al. 1986). By this method all pumps can be separated into two major categories, either kinetic or positive displacement. Under kinetic displacement, the rotating element known as the impeller creates a centrifugal force which 'impels' kinetic energy to the fluid moving it from pump suction to the discharge (Nelik 1999). Alternatively, positive displacement uses the corresponding motion of the pistons or the mechanical action of the gears or other moving parts to move the fluid from suction to discharge (Nelik 1999). A pumping system essentially consists a supply or suction side, a pump with a driver and a discharge or delivery side (Nelik 1999).

There are two parameters within the pump system that are of primary interest to this project, that of pressure and flow. The term flow is used to describe how much of the fluid must be moved. Pressure indicates how much hydraulic resistance must be overcome to move the fluid (Nelik 1999). Given a situation where there are zero losses it could be assumed that all of the input power would be transferred into moving the flow against the given pressure, however, other factors must be taken into consideration such as pump speed, viscosity of the fluid and gravity which also have an effect on flow and pressure (Nelik 1999). A pump curve is a useful tool to demonstrate the relationship between pressure and flow. The following figure 2.3 illustrates a typical pump curve. The shape of the curve varies depending on the type of pump used (Nelik 1999). Pump curves such as this will be utilised within this project to provide information as to how the pump performs with respect to speed and pressure within the system.

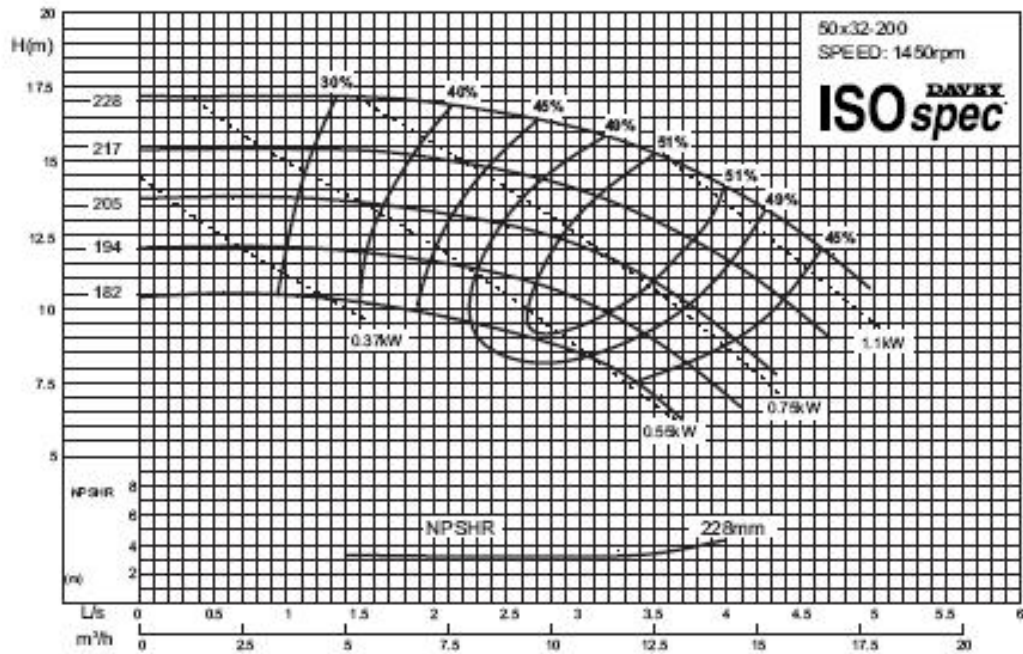


Figure 2.3: Illustration of a typical Pump Curve.

2.3 Operating Philosophy of a Pump

The concept of pressure and flow also provides a useful distinction between types of pumps. For instance, a positive displacement pump uses gears or pistons to move the fluid and does not impart velocity to the liquid it is pumping. It is therefore described as a flow generator (Nelvik 1999, Darby 2001). A centrifugal pump alternatively is known as a pressure generator because its rotating element transfers the energy to the fluid. The momentum transferred by the impeller increases both kinetic energy and the momentum of the fluid. The subsequent kinetic energy is then converted to pressure energy which is known as the head (Nelvik 1999, Darby 2001). The pressure developed by the pump depends upon a number of factors including the speed, shape and size of the impeller (Darby 2001). The following figure 2.4 provides an excellent illustration of the components of a typical centrifugal pump. It can be seen that the pump consists of three components, an inlet duct, an impeller and a volute. Fluid enter the inlet duct (D). As the shaft (A) rotates, the impeller (B) also rotates. The impeller

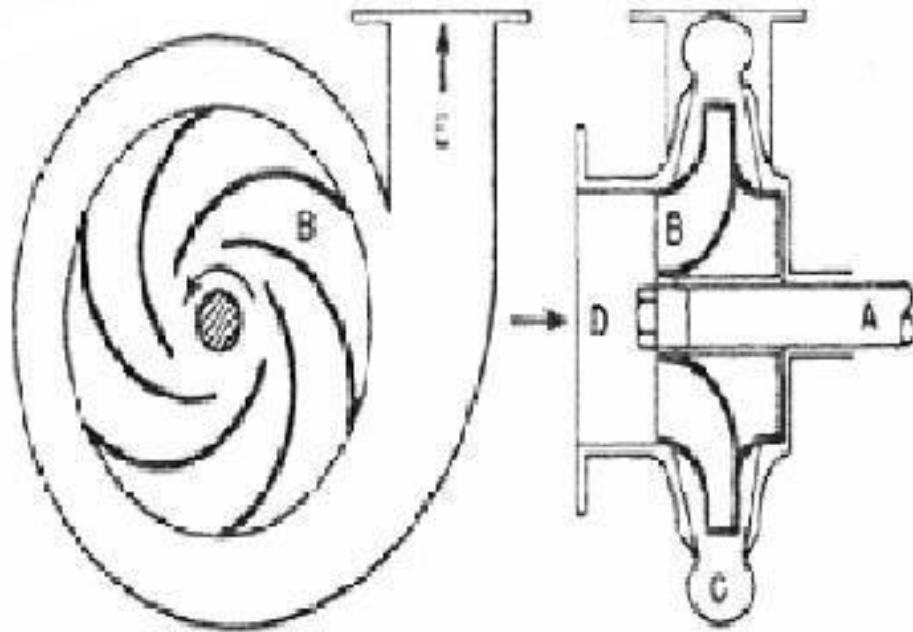


Figure 2.4: Illustration of the components of a Centrifugal Pump.

consists of a number of blades that project the fluid outward when rotating. The centrifugal force gives the fluid a high velocity. The moving fluid passes through the pump case (C) and into the volute (E). The volute chamber has a uniformly increasing area which decreases the fluid's velocity, which in turn converts the velocity energy into pressure energy (Davidson 2002).

Notwithstanding the fact that the centrifugal pump is one of the most widely used pumps for transferring liquids, it also has excellent ability to control pressure, is quiet in comparison to other pumps, has relatively low operating and maintenance costs, takes up minimal floor space and can create a uniform, non pulsating flow (Darby 2001, Davidson 2002). For these reasons the centrifugal pump has been chosen for the purposes of this project.

2.4 Variable Speed Drive (VSD)

2.4.1 Purpose of a VSD

The last part of the pumping system which has relevance to this project is the Variable Frequency Drive or VFD which is now more commonly known as a Variable Speed Drive or VSD. A VSD is an electronic controller that alters the speed of an electric motor by means of modulating the power to that motor (Five Star Electric Motors 2005). The VSD maintains constant pressure in a conventional pumping system by receiving a signal from a pressure transmitter, and corresponding the motor output relative to the feedback from the transmitter (Five Star Electric Motors 2005). VSDs provide a number of advantages over traditional methods of constant pressure control which have been taken into consideration as part of this major project. Not only do VSDs cost less than other alternatives to pressure control such as water towers for instance but they also provide accurate pressure control; minimise water leakage; provide an efficient delivery of power to the motor; are easily upgradeable; can increase pump life; and can provide considerable energy savings (Five Star Electric Motors 2005). Furthermore, and most significantly, a VSD allows for rapid adjustment of small variations which greatly enhances its value to this project (Hydraulic Institute et al. 2004). The VSD chosen for this project is a Microdrive Elite Series manufactured by PDL Electronics. This particular VSD has the capacity to run a user program in addition to its own operating system. Figure 2.5 illustrates the architecture of the VSD utilised in this project. Note that it has a display board which communicates to the main control board. This display board is capable of being utilised as a user interface for entering parameters and controlling the VSD.

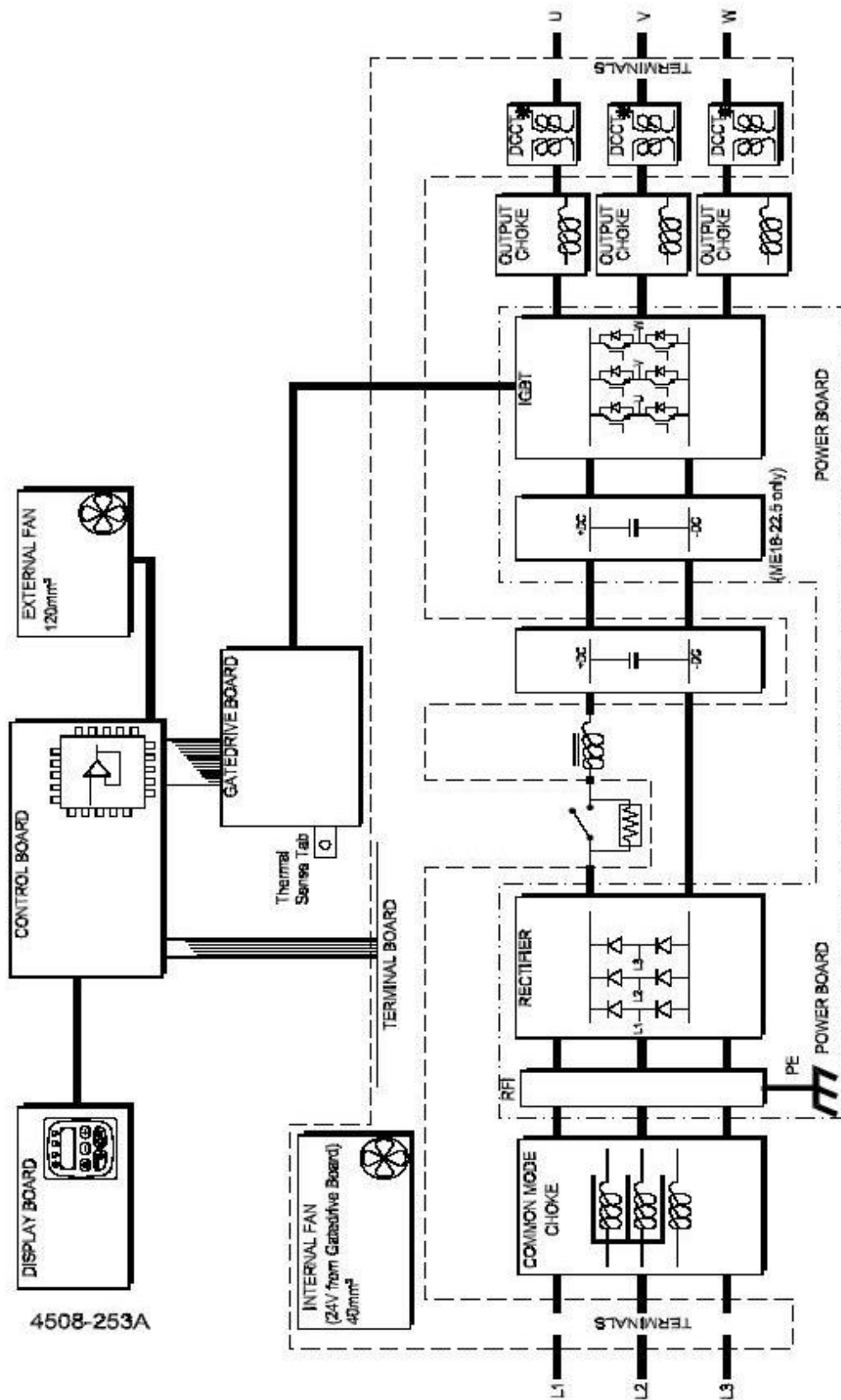


Figure 2.5: Illustration of the system architecture of a VSD.

2.4.2 Programming Software for the VSD

The VSD has a user programming software which was developed by PDL called Vysta. This software is a graphical icon based connection software. The VSD has 30 registers that can be used for a user application written in Vysta. The PDL Microdrive Elite Series are primarily motor controllers. Therefore any programming that is done in Vysta must not interfere with their ability to control the motor (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File 2002*). It is important that if the standard motor control functions like starting, stopping and fault resetting for instance are to be disabled within the motor controller, then these commands must be provided in the Vysta program (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File 2002*). The Vysta program execution cycle occurs at 4mS. The inputs and outputs of each of the function blocks are updated each cycle (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File 2002*). As a result it is imperative that the input of a function block must not in any way be dependent on its own output (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File 2002*). In order to program Vysta, the use of the Schematic Editor is required in addition to the screen lists and it is relevant to the objectives of this project to possess a broad understanding of both.

The Schematic

The Schematic Editor enables a function block based control configuration to be assembled as can be seen in figure 2.6. Function blocks are selected from Vysta's

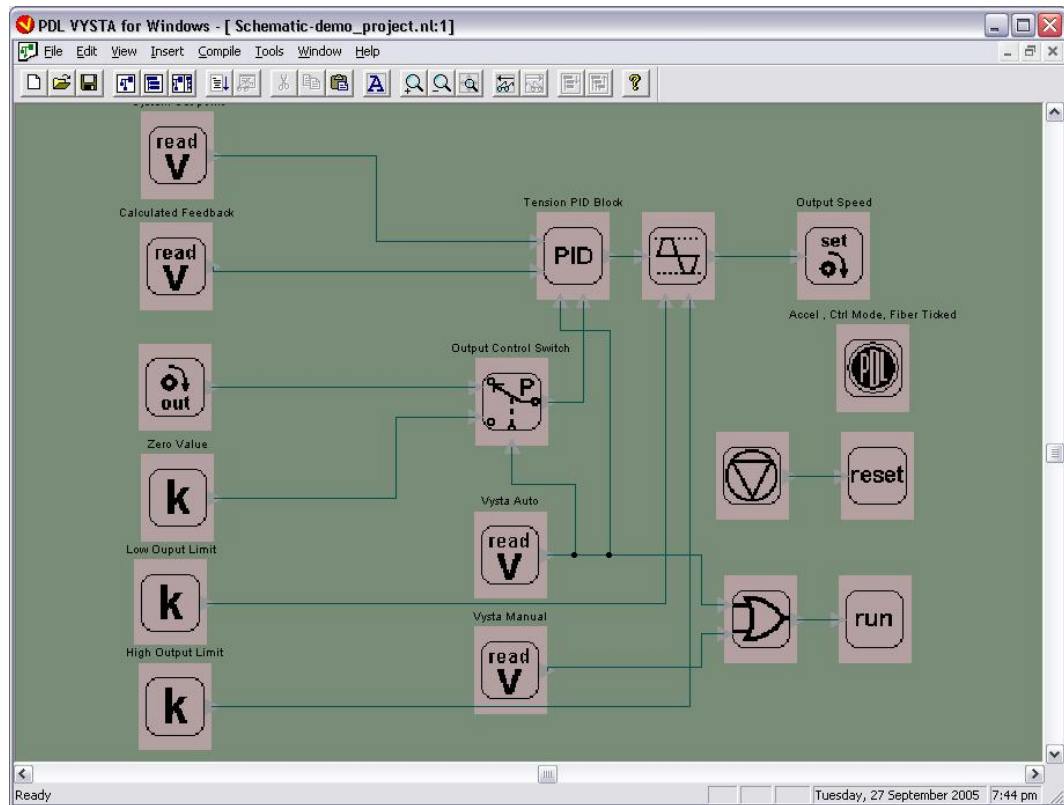


Figure 2.6: This is an example of a VYSTA Schematic Connection Diagram

Menu and interconnected using click and drag. The menu can be seen in figure 2.7.

Each Vysta program may only have one Schematic associated with it. A wide range of standard logic and process control function blocks are available such as PID controllers, comparators, logic gates and arithmetic functions for example. A function block may be selected from the Menu multiple times (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File 2002*). Each function block has its own configuration dialog box(es) for the various parameters associated with that function block as can be seen in figure 2.8.

It is possible to access Standard Elite Series system variables, however, custom

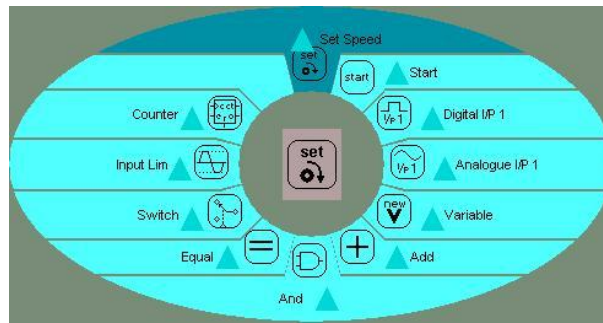


Figure 2.7: This is an example of VYSTA Schematic Oval Function Block Menu

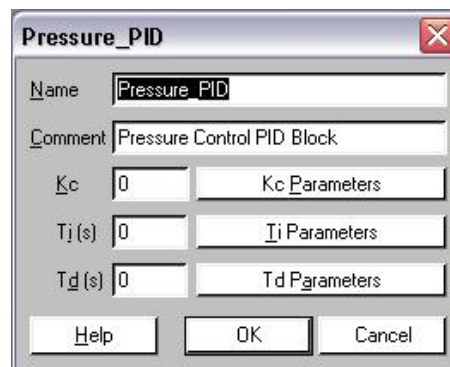


Figure 2.8: This is an example of a Read Variable Dialog Box

variables can be created for control purposes (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File 2002*). Variable values are both displayed and entered by way of the motor controller's Display Unit. When writing a Vysta program that will run on a PDL AC motor controller, the Schematic Program is required to interface with some of the standard motor controller functions (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File 2002*). The Standard Program function block is used once in each Vysta program's Schematic to select the control source for the various functions (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File 2002*). Understanding the method by which the VSD is programmed is critical to the overall project objectives. The programming of the VSD is highly significant in that it is the method by which the function of control (normally performed by the pressure transmitter) will be achieved in the new pump control system. As mentioned previously the programming of Vysta requires not only the use of a Schematic Editor but the Screen

Lists as well.

User Defined Screens

The PDL Microdrive Elite Series VSD has a two line Liquid Crystal Display (LCD) with a width of 16 characters (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File 2002*). Figure 2.9 shows the layout of the screen menus that have been programmed for this project.

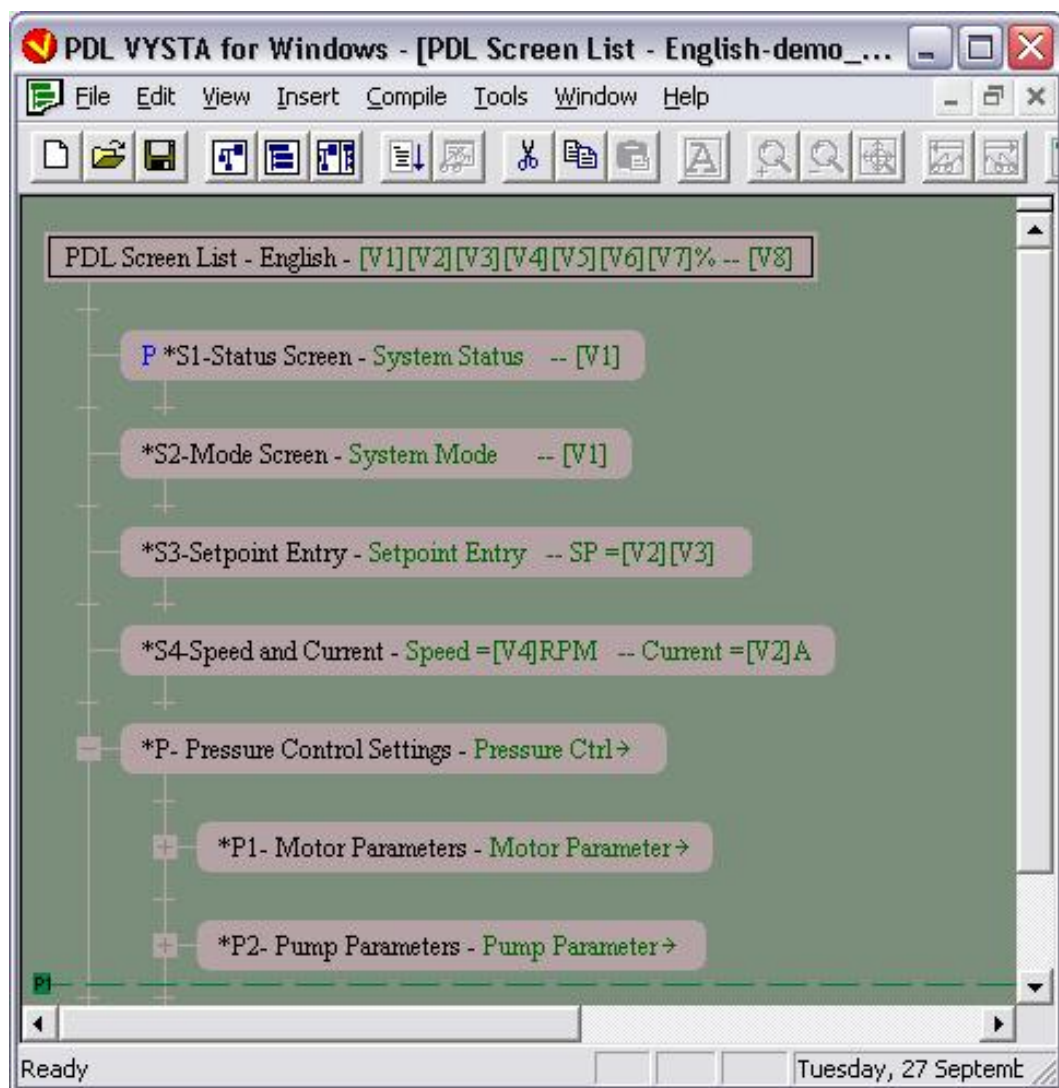


Figure 2.9: This is an example of a VYSTA Screen Layout

The Display Unit uses this Screen List when the program is active within the

VSD. By using the Display Unit buttons, the various screens can be accessed and displayed. Each screen can be used to display, for data entry or for parameter adjustment (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File* 2002). The Screen List is in the form of a tree structure with sub-screens in a branch known as children of the parent screen. The top screen in the hierarchy is the status screen which contains special motor controller related information and appears on the top line of the Display Unit while the other screens appear on the bottom line (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File* 2002). Within the Screen List it is possible to select one or more screens and perform typical editing operations for example cut, copy, paste, delete, insert and edit to name a few. A Vysta program may contain multiple Screen Lists which can be used to provide a multi-language support or to provide a function-specific Screen List or a short-menu function (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File* 2002).

2.4.3 Serial Communications

The PDL Microdrive Elite Series contains built-in Serial Communications circuitry which enables them to be linked onto a MODBUS communications network (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File* 2002). The two Serial Communications standards available in this Series are RS232 and RS485. The RS485 network has a Multi-drop capability, allowing up to 240 slave units to be linked to the one MODBUS master controller. The RS232 system permits the connection of one unit only on a line. The PDL VSD acts as a slave peripheral when connected on a MODBUS system. This means that the VSD does not initiate MODBUS messages, this is performed by a MODBUS master which in this project is the WEB server. The VSD can be controlled and/or monitored as a slave unit from the WEB server (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File* 2002). All the controls, parameters, and modes available on the VSD are available by using the MODBUS Serial

communications. In addition to the functions available via the Display Unit of the VSD, the MODBUS master can monitor and control a process by using the VSD's control board inputs and outputs (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File* 2002). The Vysta variables and associated functionality of the Vysta program can also be fully integrated into the MODBUS serial communications network. The Elite serial communications uses the hardware standard RS232 and RS485 for the physical link and the industry standard MODBUS protocol for the communications protocol (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File* 2002).

2.4.4 MODBUS Protocol

MODBUS is an open protocol which can be communicated over RS232 or any other serial connection (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File* 2002). This project is using MODBUS to communicate from the WEB server to and from the VSD. The serial communications protocol of the Elite Series complies with the industry standard MODBUS protocol (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File* 2002). The VSD supports a subset of the complete MODBUS function set listed in figure 2.1. MODBUS Function 3 and 16 refer to Holding Registers with addresses of

Table 2.1: MODBUS protocol subset utilised by the PDL Elite Series VSD

Function	Description
3	Read Multiple Holding Registers
16	Force Multiple Holding Registers

the form 4XXXX (*Vysta Virtual Automation Programming Platform Version 2.0 - Help File* 2002). The VSD System variables have MODBUS addresses already assigned and these are tabulated in Appendix C, which is the System Variable Data Table (*Vysta Virtual Automation Programming Platform Version 2.0 - Help*

File 2002).

2.5 WEB Server and Java Applications

2.5.1 WEB Server

The WEB server chosen for this project is a device which has a primary role of converting from ethernet to serial. A Serial Device allows for the 'transmission of intelligent information from computer to computer, or from computer to a peripheral, one bit at a time' (Angelfire 2005). Serial communications was and is the standard communications used within the industrial field. Serial devices can be categorised as, input only devices, output only devices and input/output devices (*Web Enabling Your Serial Device* 2002). Input only devices include measurement and monitor devices such as temperature gauges, weather stations and heart monitors. An output device include displays like sign boards and printers. Input/output devices include controls and interactive devices like robotics, PLCs and terminal sessions (*Web Enabling Your Serial Device* 2002). A technician interacts with these devices typically from another 'hard wired' serial device such as a computer or controller. With recent technological advances it is now possible to interact with these devices from a remote location using a WEB browser over an Ethernet network (*Web Enabling Your Serial Device* 2002). It is now possible to access low cost hardware to convert the RS232, RS422, or RS485 serial port into an Ethernet interface. This can be accessed by any Internet Protocol (IP) based application such as a WEB browser over an IP network from any place in the world (*Web Enabling Your Serial Device* 2002). The serial data is simply enclosed into TCP packets, which can travel through any IP based network. The hardware which enables this process is known as a Device Server (*Web Enabling Your Serial Device* 2002). Device Servers are inherently network aware. (*Web Enabling Your Serial Device* 2002). This means that they have the ability to add

functionality like web services, e-mail and network diagnostics (*Web Enabling Your Serial Device* 2002). Although the functionality of the device cannot be changed, it is possible to enhance the user interface. Regardless of the device functionality, a method to query and or control the device over the network is required (*Web Enabling Your Serial Device* 2002). To perform the required programming a browser-supported language like JAVA is needed. In order to connect to the Device Server with a JAVA applet communicating with a serial device attached to the Device Server requires familiarity with JAVA programming as well as a JAVA compiler (*Web Enabling Your Serial Device* 2002). Understanding the basic principles of JAVA is crucial to achieving the aims of this project.

2.5.2 Java Applications

The advent of the Internet and the World Wide Web fundamentally transformed the computing industry into what it is today. In today's world nearly all PCs are connected to the Internet. As part of this transformation a new way to program was developed which is known as JAVA (Schildt 2005). JAVA is the superior language of the Internet and is a critical tool for programmers worldwide.

JAVA has had a profound effect on programming. In a network, there are two categories of objects that can be transmitted between the server and a PC, that is passive information and active programs. JAVA enables both types of objects to be transmitted (Schildt 2005). An applet is a special form of JAVA program that can be transmitted over the Internet and automatically executed by a JAVA compatible WEB browser. A JAVA applet can be described as an intelligent program unlike for instance an animation or media file. In effect, this means it is a program that can respond to user input (Schildt 2005).

As a result of JAVA's programming abilities as well as its compatibility with the Lantronix embedded Web Server it is the programming tool chosen for this project. In order to compile and run JAVA programs it is essential to acquire a

JAVA development system. The one chosen for this project is a JAVA Development Kit available from Sun Microsystems (Schildt 2005).

2.6 Pump and Motor equations

The following equations have been selected as they provide a useful insight into how the parameters within a pump system are interrelated. For this reason these initial equations have been in the development of the mathematical model which will be used to develop a control algorithm later in the project.

The following equation (2.1) known as Bernoulli's equation is sourced from (Darby 2001). It provides a relationship between the work applied to the fluid by a pump and the difference in pressure, with respect to the density of the fluid. Furthermore it is also shown that the acceleration due to gravity and the product of fluid pressure is also equal to the work done by the pump.

$$-w = \frac{\Delta P}{\rho} = gH_p \quad (2.1)$$

Where:

$-w$ = Work by the Pump on the fluid

P = Pressure

ρ = The Density of the fluid

g = acceleration due to Gravity

H_p = Fluid pressure or Pump Head

Due to a pump not being 100% efficient, energy delivered from the motor is lost due to losses such as friction and heat. Equation (2.2) demonstrates that pump efficiency is equal to the work done by the pump divided by the work put into

the pump by the motor (Darby 2001).

$$\eta_e = \frac{-w}{-w_m} \quad (2.2)$$

Where:

η_e = Pump Efficiency

$-w$ = Work by the Pump on the fluid

$-w_m$ = Work put into the Pump by the Motor

The following equation (2.3) assists in correct pump selection in that it also provides consideration of system flow rate and determines the required brake horsepower (HP) (Darby 2001).

$$HP = -w_m m = \frac{\Delta P Q}{\eta_e} = \frac{\rho H_p Q}{\eta_e} \quad (2.3)$$

Where:

$-w_m$ = Work put into the Pump by the Motor

m = is the momentum of the motor

P = Pressure

Q = the flowrate of the fluid

η_e = Pump Efficiency

ρ = The Density of the fluid

H_p = Fluid pressure or Pump Head

The following equation (2.4) provides the ability to relate power to torque and angular velocity which in turn is also related to flow and head pressure (Darby 2001).

$$HP = \Gamma\omega = \frac{\rho H_p Q}{\eta_e} \quad (2.4)$$

Where:

Γ = Torque applied to the driving shaft of the pump

ω = angular velocity of the driving shaft

Q = the flowrate of the fluid

η_e = Pump Efficiency

ρ = The Density of the fluid

H_p = Fluid pressure or Pump Head

From Equation (2.5) it can be seen that the radius of the impeller of the pump can now be related to flow and torque (Darby 2001).

$$\Gamma = m\omega R_i^2 = \rho Q\omega R_i^2 \quad (2.5)$$

Where:

Γ = Torque applied to the driving shaft of the pump

m = is the momentum of the motor

ω = angular velocity of the driving shaft

R_i = the radius of the impeller

ρ = The Density of the fluid

Q = the flowrate of the fluid

With the angular momentum of the fluid entering the eye of the impeller being neglected, it can be shown from equations (2.4) and (2.5) and solving for H_p , that we can obtain an equation for Head Pressure which is equation (2.6)(Darby 2001).

$$H_p \cong \frac{\eta_e \omega^2 R_i^2}{g} \quad (2.6)$$

Where:

- H_p = Fluid pressure or Pump Head
 η_e = Pump Efficiency
 ω = angular velocity of the driving shaft
 R_i = the radius of the impeller
 g = acceleration due to Gravity

In order to calculate the power delivered to the pump shaft utilising the flow of current to the motor is as per Equation (2.7). The term power factor determines how much of the total power input to the motor is transferred to the pump shaft (Chaurette 2005). Likewise, this applies to motor efficiency.

$$P_{pump}(Hp) = \frac{1.34}{1000} \sqrt{3} V (Volts) A (Amps) \eta_{motor} P.F. \quad (2.7)$$

Where:

- P_{pump} = Power consumed at the pump shaft in Hp
 V = Motor Supply Voltage
 A = Motor Supply Current
 η_{motor} = Motor Efficiency
 $P.F.$ = Power Factor

From manipulating the above equations, it is possible now to derive a direct relationship of the pump parameters, which will in turn have a direct physical relationship to the electrical characteristics of the motor driving the pump. This is one of the foundation tasks of this project.

The following equations, equation (2.8), equation (2.9) and equation (2.10) are the most basic of pump equations showing the relationship between Pump Speed, Flow, Head (pressure) and Power. These three equations are of most interest to this project with the equations substituted and transposed we end up with an equation which has head pressure related to power within the system (Australian Pump Manufacturers Association Ltd 1987).

$$\left(\frac{Q_1}{Q_2}\right) = \left(\frac{N_1}{N_2}\right) \quad (2.8)$$

Where:

Q = Flow in the System

N = Speed of the Pump

$$\left(\frac{H_1}{H_2}\right) = \left(\frac{N_1}{N_2}\right)^2 \quad (2.9)$$

Where:

H = Head Pressure in the System

N = Speed of the Pump

$$\left(\frac{P_1}{P_2}\right) = \left(\frac{N_1}{N_2}\right)^3 \quad (2.10)$$

Where:

P = Power of the Pump

N = Speed of the Pump

The following equation (2.11) is the resultant equation of such a substitution and transposition. This equation lends itself to the application required in this project and will be utilised within the test stages of this project.

$$H_1 = \left(\frac{P_1}{P_2} \right)^{\frac{2}{3}} H_2 \quad (2.11)$$

Where:

P = Power of the Pump

H = Head Pressure in the System

2.7 Current direction in pump pressure control

Initial investigation into similar research as intended by this project revealed limited advancements in the area of pressure control without the use of a pressure transmitter. Interestingly however, examples of similar research seemed to be primarily concentrated within the medical industry. The primary reason for this appears to be the need to avoid invasive extra components such as pressure transmitters within the human body (Trinkl, Mesana, Havlik, Mitsui, Demunck, Dion, Candelon & Monties 1991, Minghua & Longya 2000*b*). One example that is of interest to this project is a ventricular assist device which can be permanently implanted within the human body. The computer modeling of the interaction of the electric motor and the blood pump within the circulatory system has parallels with objectives of this project (Minghua & Longya 2000*a*). It must be noted however, that this development is on a much smaller scale (ie physical flow and pressure rates) of the pump than is anticipated within this project. In addition, the application is highly specialised. Along similar lines is a project based upon non-invasive measurements of blood pressure and flow utilising a centrifugal pump (Kitamura, Matsushima, Tokuyama, Kono, Nishimura, Komeda, Yanai, Kijma &

Nojin 2000).

2.8 Summary

It may appear that a large proportion of the latter half of the Literature Review is devoted to the technical specifications of the VSD, however this section is crucial to understanding how the removal of the pressure transmitter component from the pumping system can realistically be achieved. Certainly the overview of the current environment and the pumping system provides a useful foundation for understanding what are essentially the basic elements of the project, that is; what is the current industrial situation in relation to the project; what is a pumping system; how does a centrifugal pump operate; what is the role of the pressure transmitter in terms of pressure control. The discussion of the VSD and the WEB server and JAVA applications however is a crucial element in the process that has been undertaken to complete the overall aims and objectives of the project. Furthermore, the pump and motor equation section, again although heavy in theory provides an essential foundation to the relationships between pressure, motor current and speed that are examined in great detail within the Chapter 4.

Chapter 3

Methodology

3.1 Resource Planning

3.1.1 Equipment

Following is the equipment that has been utilised for this project. Each item has been carefully chosen for its suitability for the purpose of the project. Although much of the theoretical background of this equipment has been provided within the Literature Review, this section provides the technical specifications and the reasoning behind the selection of the specific items utilised.

The Centrifugal Pump

As discussed in the Literature Review a centrifugal pump has been chosen for this project. This form of pump consists of a shaft mounted impeller(s) rotating unidirectionally within a casing. The liquid enters the impeller eye and acquires energy in the form of velocity as it passes through the impeller passages. The velocity head is converted into pressure head by the Volute which directs the liquid

from the outer perimeter of the impeller to the pump discharge (Australian Pump Manufacturers Association Ltd 1987).

There are 3 main types of Impellers

1. Radial Flow Impellers
2. Mixed Flow Impellers
3. Axial Flow Impellers

For this project the type of pump chosen is a Centrifugal pump with a Radial Flow Impeller driven by a suitably sized Squirrel cage induction motor. The pump is a manufactured by Davey Pumps Australia refer to appendix G.

The specifications for the pump are as follows:-

Manufacturer : Davey Pump Australia PTY LTD
Type : Centrifugal Pump
Model No. : ISO CM 50x32-200
Impeller Size : 200mm
Motor Power : 1.1KW
Voltage : 240VAC Three Phase
No. of Poles : 4
Speed : 1450RPM
Connection Type : Delta

Variable Speed Drive

Various brands of VSDs were considered, with final selection being a PDL Micro Drive Elite refer to appendix B. The other VSDs considered include Danfoss,

Telemecanique, Moeller and ABB, all of which were capable of controlling the speed of the motor but the PDL drive had the added feature of being able to write user software within the VSD thus eliminating the need for an additional controller. The PDL drive uses an icon function block based programming language called VYSTA which has been developed by PDL.

A Variable Speed Drive (VSD) is used to control the speed of a three phase squirrel cage induction motor. The way this is achieved is to modify the frequency of the supply to the motor, as this is the only non physical element that can be modified in order to change the speed of the motor. From Equation (3.1) the above can be seen to be the case.

$$N = \frac{120f}{P} - S \quad (3.1)$$

N = Speed in RPM

f = Frequency of Motor Supply

P = Poles of The Motor

S = The Slip of the Motor (The difference from synchronous Speed)

The current method of control utilises VVVF which stands for Variable Voltage Variable Frequency. This method of control also varies the voltage in proportion with the Frequency so that the V/HZ ratio is kept constant refer to (PDL 2002).

The specifications for the VSD are as follows:-

Manufacturer : PDL Electronics NZ

Model No. : ME-12

Amps : 12A

Voltage Input : 240VAC Single Phase

Voltage Output : 240VAC Three Phase

Web Server

A Lantronix Din Rail mounted Serial to Ethernet convertor / Web server was selected. This unit has the ability to contain a Web page and also HTML links to other servers and web sites if required. The unit can also be programmed to perform calculations and control functions for the VSD system as well as be able to be configured as an HMI (Human Machine Interface) from the VSD system to the real world.

Manufacturer : Lantronix
Model No. : LA-XSDRIN-01 XPress DR-IAP
Casing : DinRail mounted case
Communications Type : (RS-485 or RS-422 or RS-232) to Ethernet
Voltage Input : 24VDC

Pressure Transmitter

A pressure transmitter was utilised so that the pressure that the pump delivers for various speeds can be recorded. This enables calculations of the relationship between motor current and system pressure which will be required for the final system to be achieved. The Pressure transmitter was sized so that it could handle the highest deliverable pressure by the pump which was 17.5 metres of head which can be converted to pressure by using equation (3.2) as follows. By substituting the Head and the Specific Gravity (SG) of the water at 4 Degrees C of 1 a maximum pressure of 171.5 Kpa is achieved. The selected unit is capable of reading this range and is an off the shelf industrial unit.

$$p = 9.8 \times H \times SG \quad (3.2)$$

p = Pressure in Kpa

H = Head in metres

SG = Specific Gravity of Liquid

Manufacturer : Schneider Electric

Model No. : XMLE010U1C21

Casing : 40mm cylindrical case

Voltage Input : 24VDC

Output Signal : 4-20mA (current loop passed through a 500 Ohm resistor
to convert the signal into a Voltage signal)

Test Tanks

The Test Tank used is to be a 500 litre poly tank with a 2 inch outlet valve which will feed the pump and a return line which will be fed back into the top of the tank so that the system can maintain constant circulation

Manufacturer : Pine Crest Products

Model No. : ute500tnk

Capacity : 500 Litres

Material : polymer plastic

Piping and Fittings

Various fittings and lengths of pipe required to make the pump circuit include pipe work that starts at 2 inch (51mm) into the intake of the pump and one and one quarter inch (32mm) from the pump discharge through the valve and back to the tank.

3.1.2 Facilities

This Project was undertaken at the offices and workshops of Tritec Electrical Controls And Automation PTY LTD.

3.1.3 Computers

The computer that has been used for this project is a TOSHIBA Satellite Pro and the specifications are as per the following list.

Manufacturer : Toshiba
Model No. : PT831A-19MW8
Processor : Intel Pentium M
Memory : 512MB of Ram
CPU Speed : 1.69GHz
Hard Disk Size : 60GB

3.1.4 Software

The Computer Software to be utilised throughout this project is as per the following list:-

Operating System	: Windows XP Professional version 2002 : service pack 2
PDL VSD Communications	: PDL Drivelink Version 2.7
PDL Parameters Setup	: PDL Drivecom Version 3
PDL Programming	: PDL VYSTA Version 2.0.0.0
JAVA Compiler	: Sun Java version 7
Web Server Configuration	: Lantronix Device Installer
Web Pages Setup	: WEB2COB a DOS Application (converts Web : pages to COB files which can be downloaded : to the web server)
Documentation	: L ^A T _E X
General Usage	: Standard Windows and Microsoft Software : packages

3.2 Construction of Test Equipment

The culmination of the previous specified equipment resulted in the final test product as can be seen from figure 3.1. The figure shows a 500 litre water tank with the outlet of this tank feeding into the pump. The outlet of the pump feeds into the tank creating a closed loop system. On the output of the pump is installed a pressure transmitter. This pressure transmitter has then been connected back into the VSD and the VSD is then monitored for pressure, speed and motor current readings. Figures 3.2, 3.3 and 3.4 show photo's of the actual test equipment used. It differs from the PI Diagram in figure 3.1 in that a valve has not been installed in the outlet of the tank. This was not installed as the tank and system did not require the ability of isolation of the tank outlet for servicing as the system's sole purpose was for testing and testing only.

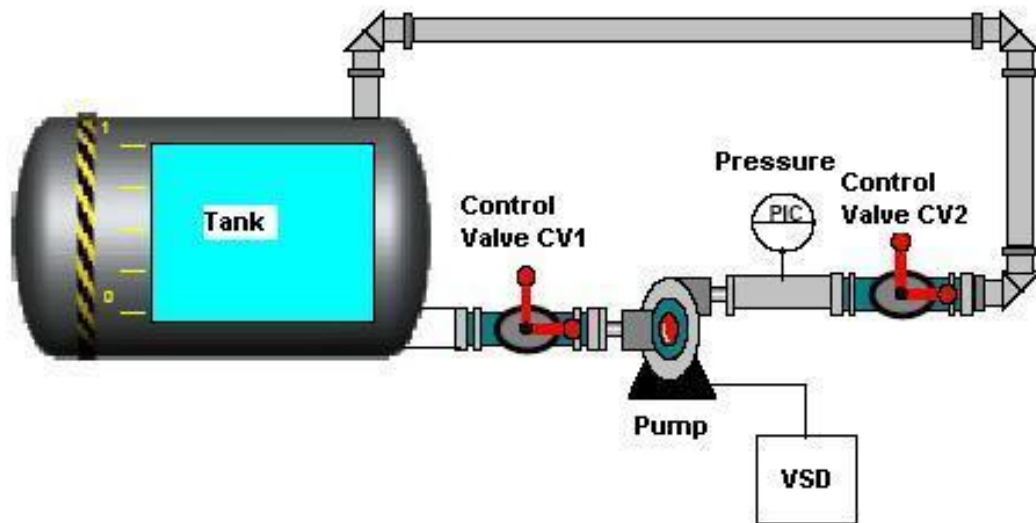


Figure 3.1: Closed Loop Testing System PI Diagram



Figure 3.2: This Photograph is of Closed Loop Testing System's Pump and Pressure Transmitter



Figure 3.3: This Photograph is of the Closed Loop Testing System



Figure 3.4: This Photograph is of the VSD

3.3 Programming of the VSD

Programming of the VSD involved utilising three softwares written and supplied by PDL Electronics. The first software used was Vysta which is the actual programming language that provides the platform to write the control applications. The control applications are for both the conventional control system for testing in addition to the end product of the control algorithm that does not require the use of a pressure transmitter. Subsection 2.4.2 explained how the programming software Vysta was used and how to configure and utilise it's capabilities.

The next software that is used is PDL Drivelink Version 2.7. This software is utilised to download the compiled Vysta program into the VSD. It also has the ability to delete the Vysta program in the VSD and return the VSD to its original configuration. This was required to be done throughout the project during the various testing stages. Appendix E section E.1 contains the operating instructions for using this software. It is imperative that this software is understood thoroughly so as correct operation of the VSD is achieved. This is true for all of the software utilised during the duration of this project.

The last software that is required to be used is PDL Drivecom Version 3. This software is used to read and write system variables within the VSD. Appendix E section E.2 shows an overview of this software. This software was used to set up variables used in the Vysta programs that have been written to achieve the end result of the project. Utilising this software enabled testing of the Vysta programs without writing additional display screens for the entering and viewing of variables within the Vysta programs.

3.4 Programming of the Web-interface

In order to develop the user interface, various types of WEB servers were investigated. A Lantronix unit was chosen. It is 24 volt DC powered and has the ability to communicate from Ethernet to RS-232, RS-422 or RS-485 all within the one unit. The unit has the ability to run user software and web pages. These user applications are required to be programmed in JAVA. The added benefit of using this type of device is that any person within an organisation who has access to their LAN or WAN can view and adjust parameters without the cost of further software other than windows explorer. The WEB page for the WEB server was written in JAVA and then compiled into an applet which was then implemented by a HTML document. Once the HTML document was written and tested, all the application programs and the HTML document were then downloaded to the WEB server via a DOS application WEB2COB.EXE which was supplied by Lantronix for this purpose. Once the WEB server has been downloaded to the application it can be accessed by simply typing in the IP address of the WEB server and the WEB page will then be displayed.

Chapter 4

Results and Discussions

4.1 Introduction

A program was written in Vysta which ramped the speed of the drive up and then held it at maximum speed and then ramped it down again. This was done repetitively so that a pressure, speed and motor current relationship in an open head and a closed head system could be obtained. Upon completion of the above, further fixed speed tests were carried out and the results are recorded in the following sections. When these results were recorded and analysed a control algorithm was then developed and tested. This control algorithm was then compared with the conventional control system in order to determine the extent to which the project aims have been achieved.

4.2 Data

4.2.1 Open Head System Tests

The first test that was conducted was the Open Head System Test. The reason for conducting this test and the Closed Head Test was to obtain an indication as to how the system would respond in terms of pressure and motor current when the speed was varied. Figure 4.1 illustrates the motor current and pressure variance in an open head system when the speed of the pump is varied from 0% to 100% and vice versa. It shows that the motor current at 3.75 amps at full speed results in a pressure of approximately 155 Kpa. This indicates that the motor current is linearly proportional to both speed and pressure in the Open Head System. The next test conducted was identical to this test apart from the output valve of the pump being fully closed in order to achieve a closed head system.

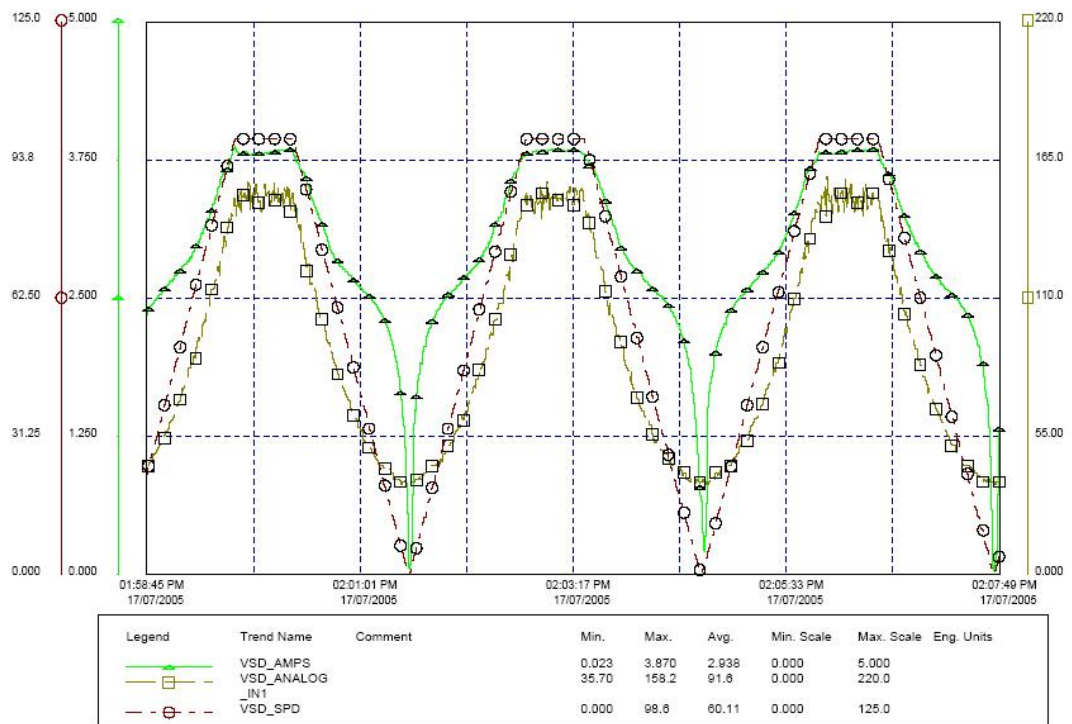


Figure 4.1: Plot of Pressure and Motor Current with speed being varied for 0% - 100% with the pump outlet valve fully open

4.2.2 Closed Head System Tests

Figure 4.2 illustrates the closed head pressure. This graph shows that the motor current drops down to approximately 3 amps whilst the pressure increases up to 200 Kpa or thereabouts. This data indicates that there are pressure limitations within the system. These limitations are that once the pressure of approximately 165 Kpa is reached, then the motor current begins to decrease until such time that a minimum current of approximately 3 Amps is reached and maintained. This current of 3 Amps will be held at this level as long as the motor pressure is greater than the 165 Kpa.

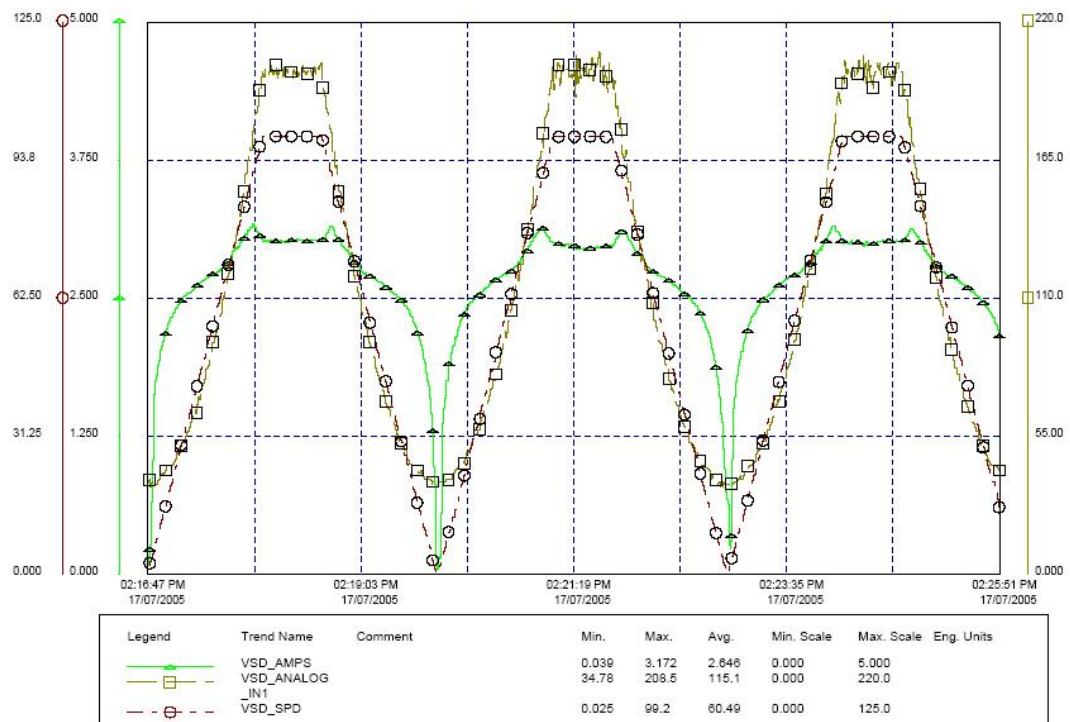


Figure 4.2: Plot of Pressure and Motor Current with speed being varied for 0% - 100% with the pump outlet valve fully closed

4.2.3 Fixed Speed Tests

After conducting the initial Open and Closed Head Tests, it was determined that further tests were required in order to better understand the relationships

between system pressure, pump speed and motor Amps.

The following tests were designed to identify the response of pressure and motor current only with a fixed speed. The speed initially was set at 30% due to the fact that a centrifugal pump of the type utilised, does not start moving the fluid until this point or at greater speed. Therefore, the lower speeds are irrelevant for these tests.

In figures 4.3 to 4.10 there are three individual plots with respect to time. These are as shown the plots with the units of amperes (A) for VSD_AMPS, Kilo Pascals (Kpa) for VSD_ANALOG_IN1 and percentage of full speed (%) for VSD_SPD. The data was obtained using a SCADA package called CITECT which used MODBUS to communicate with the VSD to record the data in the plots. The speed of the VSD was first set to 30% and then the outlet valve on the pump was turned slowly from fully opened to the fully closed position and the data was then recorded.

Figure 4.3 shows that the effect of closing the valve is very minimal as the pump is not running at a high enough velocity to move the fluid it is pumping.

Figure 4.4 shows that when the pump is running at 40% speed there is also minimal if any change in the motor current due to the position of the valve. This is also the case for figure 4.5 which shows the speed set at 50%.

From figure 4.6 it can be seen that the speed is now set at 60% and that the motor current is just beginning to deflect when the change in valve position occurs. Note in each of these initial test data, that the pressure within the system represents the position of the valve. The higher pressure being the closed position of the valve and subsequently the lower pressure being the fully open position.

From figure 4.7 it can be seen that as the speed of the system increases then the change in motor current with respect to pressure is greater and shows a more obvious change.

From figure 4.8 the motor current is of a lot larger swing than corresponding tests where the speed was less.

From figures 4.9 and 4.10 both show a considerably greater change in motor current in relation to pressure.

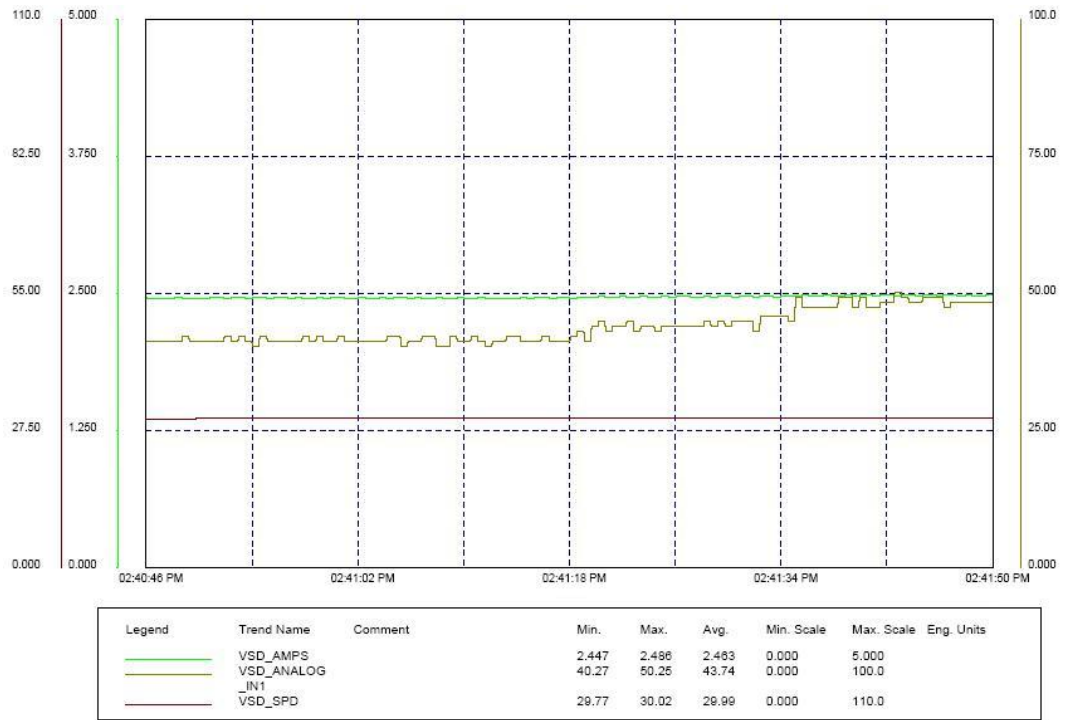


Figure 4.3: Plot of Pressure and Motor Current when speed is held at 30%



Figure 4.4: Plot of Pressure and Motor Current when speed is held at 40%



Figure 4.5: Plot of Pressure and Motor Current when speed is held at 50%

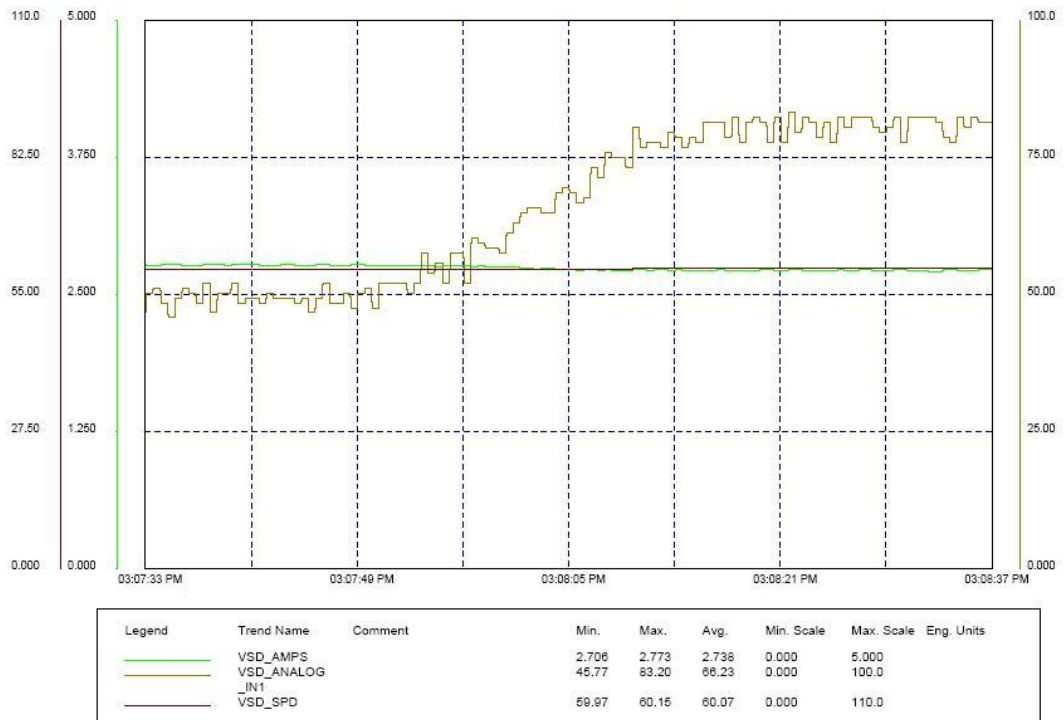


Figure 4.6: Plot of Pressure and Motor Current when speed is held at 60%

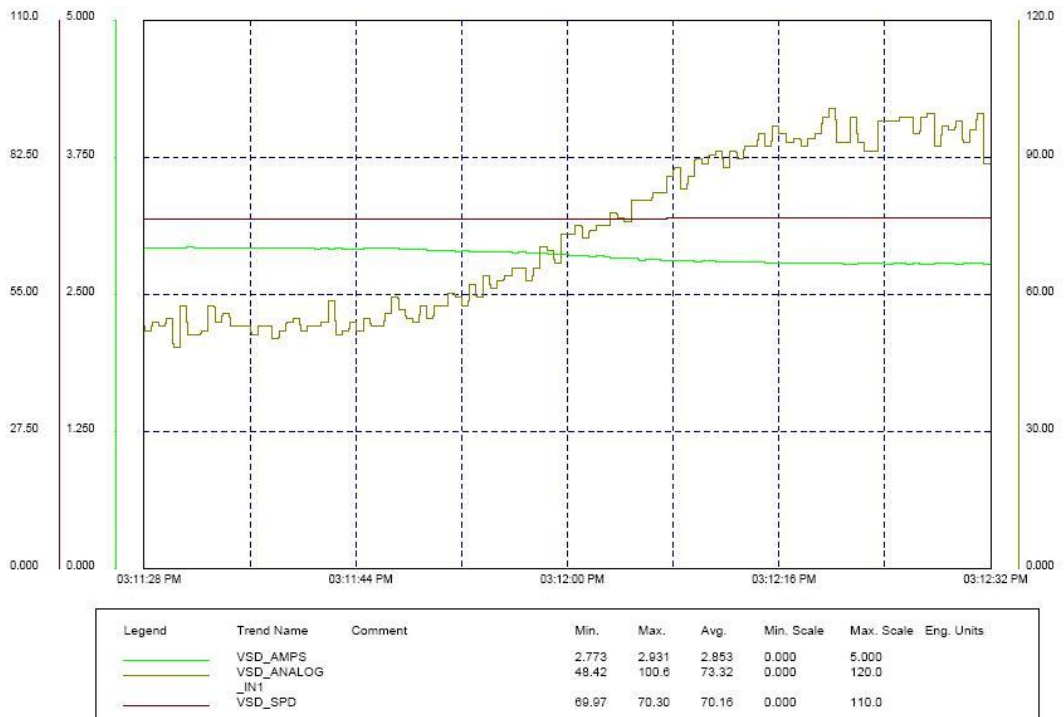


Figure 4.7: Plot of Pressure and Motor Current when speed is held at 70%



Figure 4.8: Plot of Pressure and Motor Current when speed is held at 80%

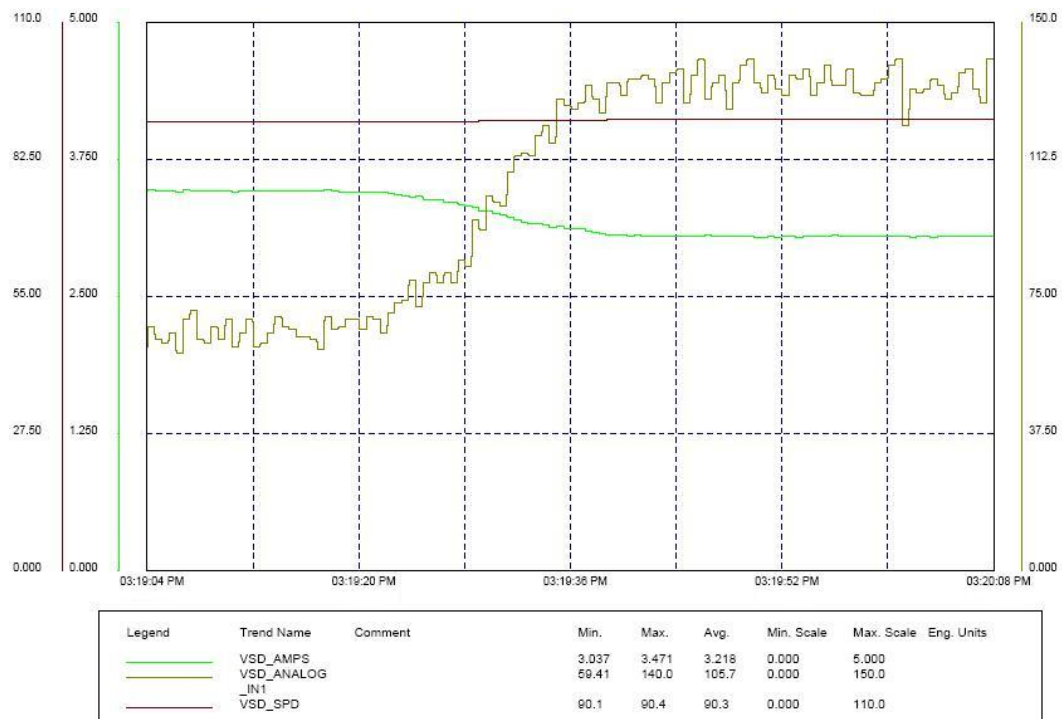


Figure 4.9: Plot of Pressure and Motor Current when speed is held at 90%

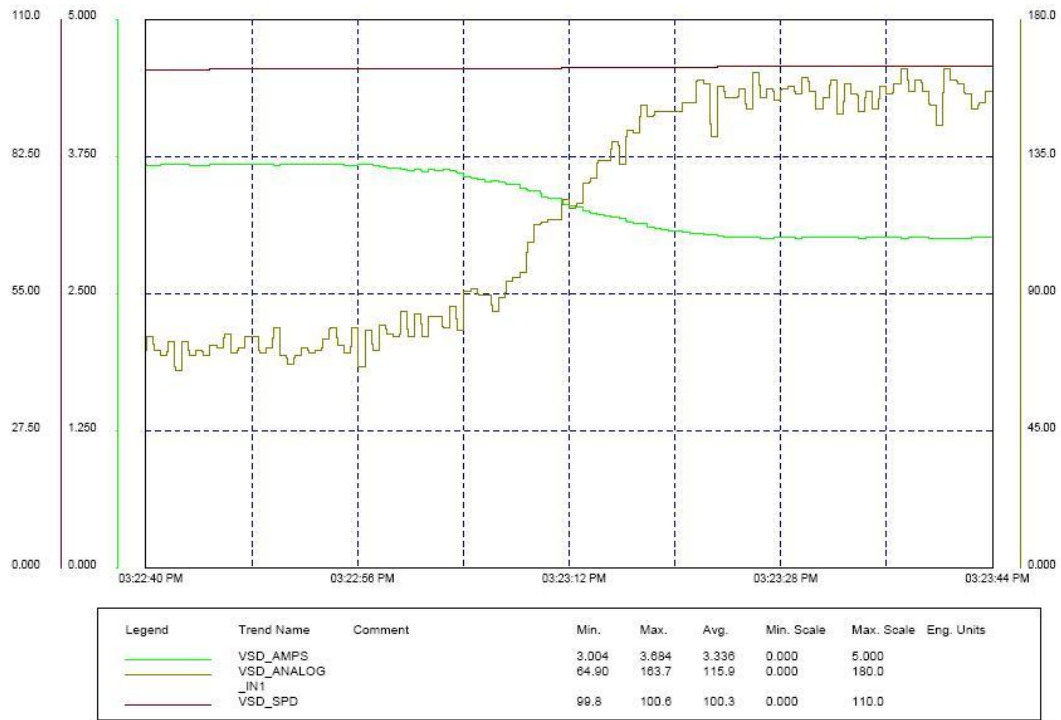


Figure 4.10: Plot of Pressure and Motor Current when speed is held at 100%

From the initial tests that were performed it is evident that below 60% of motor speed it would be virtually impossible to control pressure in the pumping system by modeling the motor current. This is due to the fact that the motor current does not vary with the change in pressure. This provides a distinct limitation to the project in that the pump selected for the system must have its pump curve appropriately matched to the duty and conditions. From the results obtained thus far it is evident that the control algorithm that is required to control pressure in a pumping system without the use of a pressure transmitter will require the system to operate around a known speed which can be calculated to approximately match the pressure required in the system. At this point, due to the test data revealing that the actual motor current decreases as the system pressure increases, the information can be used as a feedback which will be used to finely adjust the speed of the pump to control more closely the required pressure setpoint.

4.2.4 Closed Loop System Tests with Pressure Transducer

The closed loop control of a conventional system with the use of a pressure transmitter is achieved by utilising the internal PID capabilities of the VSD which has an output as shown in figure 4.11. It can be seen that the pressure is controlled constantly about the required setpoint of 65Kpa. From this data the best setpoint to use for the Closed Loop System using the Control Algorithm was to be 65Kpa as it can be seen from the conventional system, control is possible. Utilising this

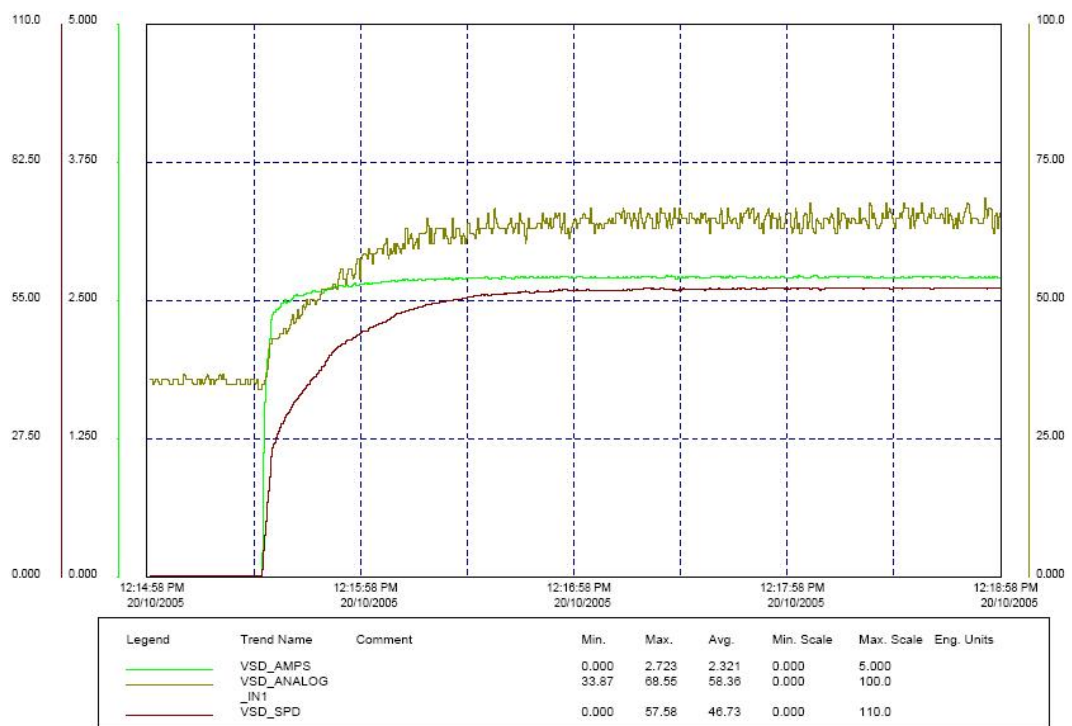


Figure 4.11: Plot showing closed loop system Pressure - with pressure transmitter feedback

data a program was created to test the conventional system. This program can be seen in figure 4.12. This program makes use of the built in PID Function Block within Vysta. The PID function block has two input variables which are connected to the left of the function block. The top variable is the setpoint that is being controlled to and the bottom variable is the feedback reference. The feedback reference is from the pressure transmitter and is connected into the analog input of the VSD. This input is then scaled from 0 - 750 Kpa as this is the range

of the pressure transmitter (the connection for the analog input on the VSD can be seen in appendix B page 22). During this test the setting of the setpoint and controlling of the drive was performed by entering and controlling from the VSD local display screen. The schematic shown in figure 4.12 provides the foundation for the final control algorithm VSD program.

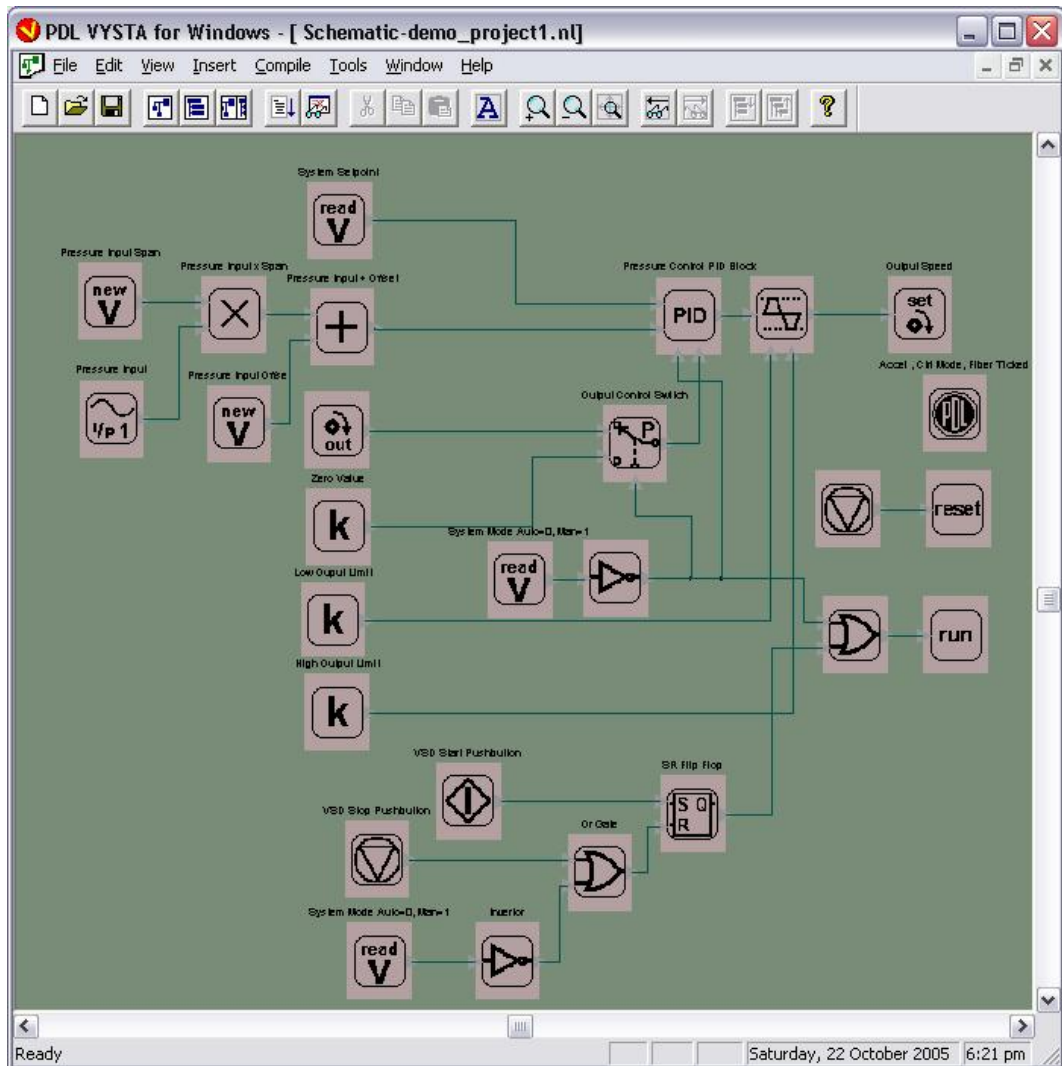


Figure 4.12: closed loop system Vysta Program - with pressure transmitter feed-back

4.2.5 Closed Loop System Tests using Algorithm

Testing of the system was carried out using the basis of the control algorithm which has been derived from the tests carried out earlier and the understanding of pumps that has been gained thus far through this project. Equation (2.11) was utilised within this control algorithm with the addition of operating limits set that allow the VSD to operate between a low speed and maximum speed. These are set as close as possible to the speed required for the pressure setpoint to be achieved.

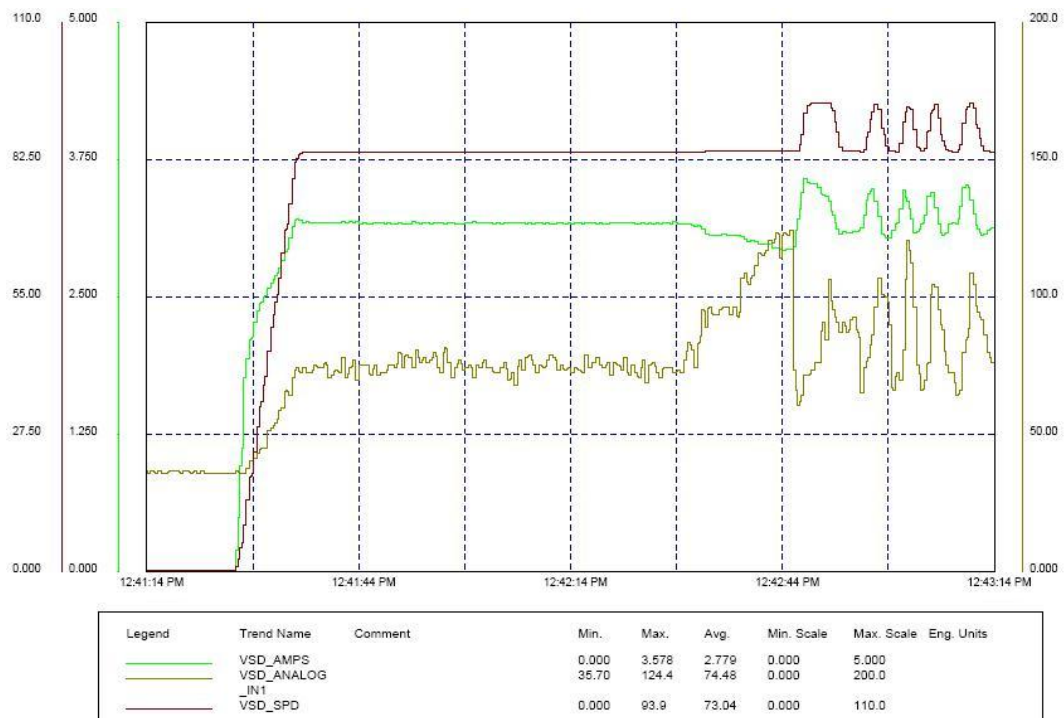


Figure 4.13: Plot showing closed loop system Pressure - with pressure transmitter feedback

From the graph obtained in figure 4.13 it can be shown that control around the setpoint was achieved, with the pressure fluctuating marginally. This was due largely to having to control system pressure by crude means of manually turning the outlet valve on the pump for open to closed and vice versa. It can be said therefore, that the control of the system pressure has been achieved. However, the control algorithm requires further refinement in order to obtain a more accurate

and smoother control response.

The program used to achieve the control with the control Algorithm as discussed earlier is shown in figure 4.14. For the purposes of the test, the control program has been written to only monitor motor current and control, with the calculated setpoint being entered into the VSD program from the VSD user control screen. This has been programmed in Vysta, with Appendix F providing the screen list of the user control screen. The VSD is then placed into Automatic mode via the user control screen and the previous data was obtained (the data in figure 4.13).

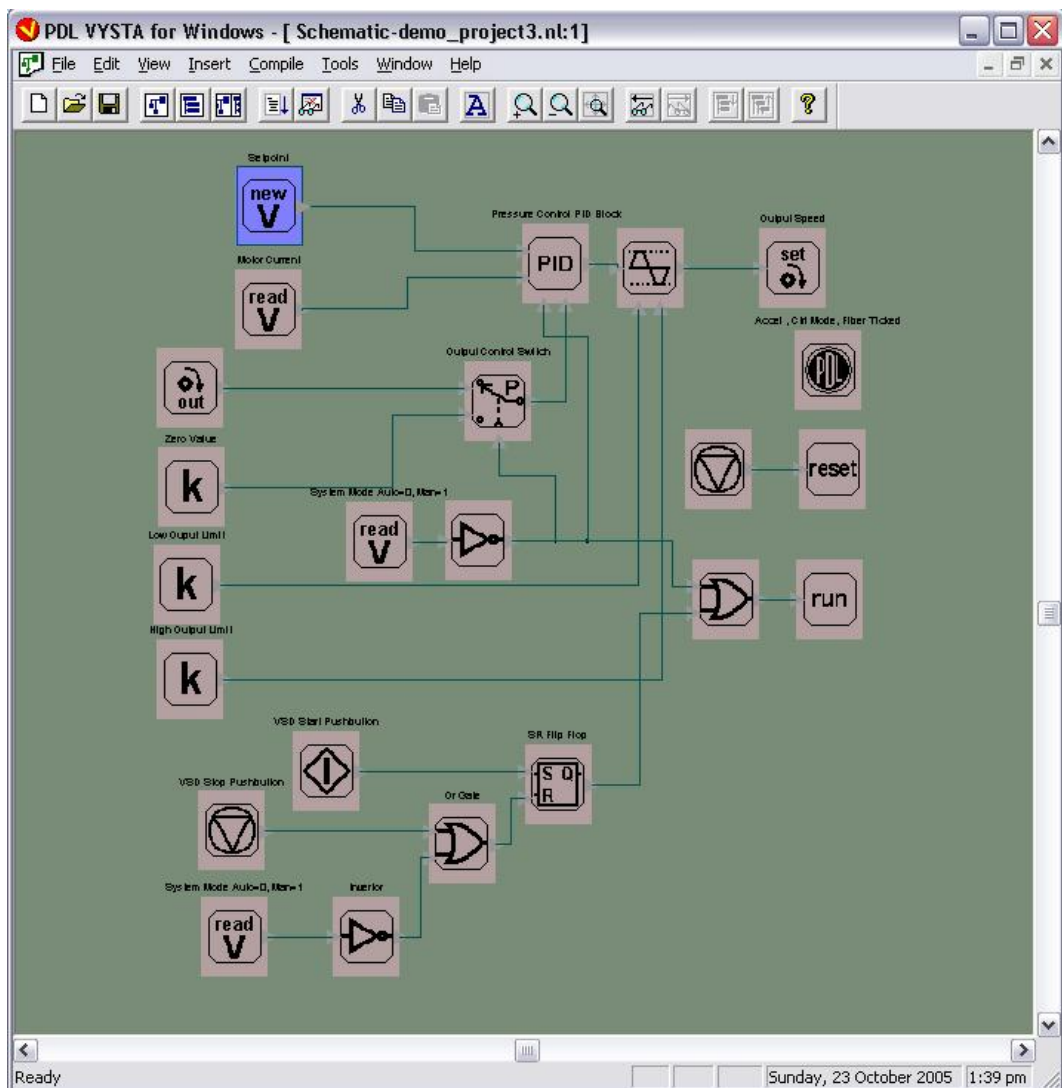


Figure 4.14: closed loop system Vysta Program - with pressure transmitter feedback

Equation (2.11) provides the relationship that the control and setpoint of the pump are based upon. The previous results obtained gave a set of limits from which the control points are calculated from. This was largely a trial and error process to achieve a good control point. Further trial and error needs to be undertaken to refine the control of the system further.

4.3 Comparison of Pressure Transducer and Algorithm

A comparison between the conventional pressure control system (the system utilising a pressure transmitter feedback) and the Algorithm controlled system from the test data obtained, revealed that the conventional system offers superior control of the pressure system. This is due to the fact that although the control algorithm does work, (that is it is able to control pressure circumventing the need for a pressure transmitter), it requires further refinement and testing to achieve an improved level of control. This level would need to meet the current level of control within industry. At this stage the control algorithm is unable to achieve this. It is envisaged further refinement will achieve the desired level of control and this is discussed in more detail in Chapter 5.

4.4 WEB Server - Interface

4.4.1 Software Development

Software for the WEB server was developed in JAVA. A copy of the source code produced and tested so far is included in appendix D. The software is only 70% complete as the focus of the project has primarily been on the development of a control algorithm. The JAVA program has three main parts, the first being the Graphical User Interface (GUI). This GUI has been developed to give the user of the system the ability to setup and monitor the system performance. Figure 4.15 illustrates the layout of the GUI and the parameters which are able to be set and viewed. As can be seen on the top left-hand side of the GUI there are a number

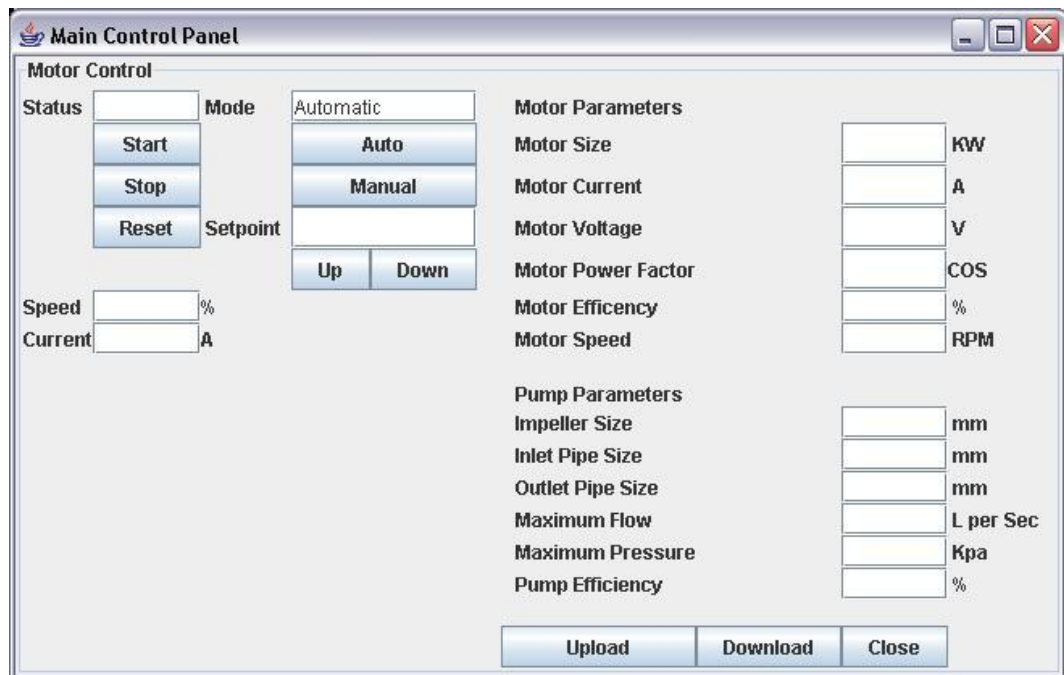


Figure 4.15: Illustration of Control System JAVA GUI

of different controls provided. These controls and their associated functions are listed in the table 4.1.

Table 4.1: GUI Control Functions

CONTROL NAME	FUNCTION
Status Field	Indicates the current Status of the VSD ie.Ready, Running,Fault,etc
Start Button	This button allows the user to Start the VSD when the mode is selected to Manual
Stop Button	This button allows the user to Stop the VSD when the mode is selected to Manual
Reset Button	This button allows the user to reset any fault within the VSD, it can be operated in any Mode
Speed Field	Indicates the current speed of the VSD in % of full speed
Current Field	Indicates the VSD motor current in Amps
Mode Field	Indicates the current operating mode of the system, it has two modes Automatic and Manual. Automatic mode control the speed of the VSD to control pressure and Manual mode runs the speed of the VSD to the setpoint that is entered as well as the System being able to be start/stopped from the start, stop buttons
Auto Button	This button changes the mode of the system into Automatic
Manual Button	This button changes the mode of the system into Manual
Setpoint Field	Indicates the current setpoint of the system and also allows the user to modify the setpoint
Up Button	This button increases the value in the setpoint field by 1% each time it is pressed, it is used for fine adjustment
Down Button	This button decreases the value in the setpoint field by 1% each time it is pressed, it is used for fine adjustment

The next part of the JAVA program was to write data from the JAVA application to the WEB server and from the WEB server serial port to the VSD. The commands sent from the JAVA application to the VSD were all MODBUS commands. The MODBUS address and the commands used are shown in appendix C section C.1 for the commands and section C.2 for the MODBUS addresses and variables available. The structure of the MODBUS is shown in appendix C section C.1. It shows that all the command strings have the requirement for a check sum. In this case it requires a CRC16 check sum to be calculated and then attached to the end of the command string. The code to develop this CRC16 check sum was a major component of the JAVA program and can be seen in appendix D. It is the last class within the JAVA program and it is appropriately named CRC16.

The final part of the JAVA program was to read data back from the VSD for display and verification. When referring to verification it is meant to verify the data that actually has been set to each critical parameter which is to be utilised in the VSD. These parameters reside in the VSD and are independent of the WEB server. This means that if the WEB server is not connected for instance or simply not functioning then the VSD would still be able to control to the desired setpoint and parameters. This means that the WEB server is only a window into the VSD system which makes the only device within the system that is critically required for the system to run being the VSD. At this stage, this section of the program has only been tested for retrieving data for motor speed and motor current. Discussion is provided in the later Chapter 5 regarding further developments planned for the WEB server interface.

4.4.2 Connection and Testing

The WEB server was connected to the VSD by the in built RS-232 terminal connections to the VSD RS-232 connections. Once it was connected the communications were checked for correct operation by sending a start signal from

the WEB server by pressing the Start Button, with the system in Manual mode. Operation and correct functionality of buttons was then checked and found to be functioning correctly. The source code had now been proved and further extension of the capabilities of the JAVA code is now a matter of adding additional commands when required.

4.5 Obstacles

Undertaking this project has presented a number of obstacles that have required a degree of compromise in the approach taken. The overall scope and limitation of the project was discussed earlier as can be see in Chapter 1 Section 1.5 in relation to the broad approach to this project. Those obstacles specifically encountered within later development and testing include the following:

1. Interpreting how the Vysta software worked within the VSD and then implementing this program to perform the desired functions proved time consuming. This software despite appearing to be a simple form of programming, in practice requires quite complex and involved understanding of the function blocks within the software. A further complication is that there are only three to four other companies Australia-wide that have access to this software therefore help desk resources are limited to almost non-existent!
2. Similar to the previous obstacle it was found that in order to enable the WEB server to interlink with the VSD and the WEB browser it was necessary to achieve an unexpectedly high level of knowledge of JAVA.
3. Due to the cost constraints faced by this project it was not possible to utilise larger equipment in VSDs and pumps. This would have enabled a larger current change for pressure which would have provided better results

for the relationships between pressure, motor current and speed.

4. The main obstacle encountered was limitations experienced on closed head. It became necessary to ensure that the system setpoint used for this application was well under the closed head motor current and head pressure. The reason for this is, if control is modeled on the power of the motor and the system is then close headed, then control will be lost. This is due to the fact that the pressure will be represented by a lower motor power than what would have been found normally in an open headed system. This means that the pump would want to speed up to reach the required setpoint thus producing a higher pressure than the setpoint. To be able to do this on various systems the pump and motor would generally need to be of a greater size than that used in a conventional system.

Despite encountering these obstacles, it was still possible to follow the essential aims and objectives of this project. The results obtained confirm that control of the pumping system can and has been achieved without using a pressure transmitter. However given a more ideal set of conditions (ie without the aforementioned obstacles) it is envisioned that a better level of control could have been achieved.

Chapter 5

Future Works

The outcomes of this project have presented a number of options to be pursued in the future. As indicated the basic aims and objectives of the project in terms of achieving control of a pumping system without the use of a pressure transmitter have been reached. However, there are a number of refinements to be completed before it could be a saleable system which are as follows:

1. To improve the control algorithm to achieve better and more accurate control over the whole range of the pump.
2. Perform more rigorous testing with a broader range of pumps and motors.
3. Complete the WEB server interface so that it has the ability to upload/download parameter information from and to the VSD application. In addition, the development of a monitoring mode which when entered will constantly update values in the GUI in Real-time.
4. At this stage the interface is designed to control one VSD only. In the future it is envisaged that this will be expanded to encompass multiple VSDs

in duty and standby applications.

5. Control has been achieved for single pump use only, in the future it is envisaged that control will encompass multiple and/or cascaded pumps.

Completion of these future objectives would enable this system to be utilised for commercial purposes in a wide variety of industrial applications.

Chapter 6

Conclusions

6.1 Overview and Obstacles

The primary objective of this conclusion is to examine the extent to which the project aims and objectives, as identified in the Introduction have been achieved. The specific aim of this project was to develop and test an automated pumping controller able to maintain constant pressure in a hydraulics system without the use of a pressure transmitter. In addition, a real-time monitoring, configuration and control system software package would be developed. The theory contained within the Literature Review outlined the current pumping environment, overview of a pumping system and relevant pump equations which contributes towards an overall understanding of what the project is attempting to achieve. Although highly technical in content, further discussion was given pertaining to the Variable Speed Drive, the WEB Server and JAVA applications. This provided a valuable insight into the process needed to be followed in order to achieve the project objectives. It must be acknowledged that undertaking this project has proven a challenge in terms of achieving a working knowledge of JAVA software as well as PDL Vysta software, while at the same time becoming familiar with the electromechanical relationships of the equipment chosen. To emphasise this

point a number of obstacles have been identified as influencing the overall outcomes of the project which includes cost constraints, limitations within the choice of components and the difficulty of achieving a high level of understanding of the software and programming languages. However, it was possible to proceed with project as planned taking into consideration these constraints.

6.2 Pressure System Controlled without a Pressure Transmitter

The Results and Discussions Chapter provides the significant outcomes from undertaking this project. Numerous testing was undertaken, and it was found that as the centrifugal pump does not start moving the fluid until 30% speed or greater that the results from tests at the lower speeds were irrelevant. It became evident that below 60% of motor speed it would be virtually impossible to control the pressure in the pumping system by modeling the motor current. This was explained by the fact the variance in motor current is minimal with the change in pressure. Therefore it was established that the control algorithm required to control pressure in a pumping system without the use of a pressure transmitter would require the system to operate around a known speed calculated to approximately match the pressure required in the system. Further testing revealed that the pressure is controlled constantly at the required setpoint of 65 Kpa. Utilising this data an algorithm was created to replicate the control process. Control around the setpoint was achieved with some pressure fluctuation but this was due in large part to having to control system pressure by means of manual operation of the outlet valve of the pump. It can be concluded that control of the system pressure was indeed achieved without the use of a pressure transmitter as identified in the initial aims and objectives of the project. However, further refinement is necessary to obtain a more accurate and smoother control response. This was further confirmed by comparison of the conventional pressure control system (ie

the system utilising a pressure transmitter) and the Algorithm controlled system which revealed that the conventional system clearly provides superior control of the pressure system.

6.3 Real-time Monitoring and Control System

Having established that control of the pumping system was achieved without the use of a pressure transmitter it follows that consideration then be given to providing a real-time monitoring, configuration and control system software package. As indicated previously, the level of understanding required of the programming tools influenced to some extent the quality of the resulting software package. The software package at this stage allows the user to view only motor current and pump speed and change the mode of the system, as well as starting/stopping and resetting a fault. Future works propose to complete the WEB server interface so that it has the ability to upload/download parameter information from and to the VSD application. In addition, the development of a monitoring mode which will constantly update values in real-time. It can be concluded that a very basic approximation of the aims and objectives of this project has in fact been achieved. In order for this system to be suitable for commercial purposes a number of additional refinements were identified as important future works. This includes improvement of the control algorithm, more rigorous testing with a broader range of pumps and motors and expansion of the system to include multiple VSDs, and multiple and/or cascaded pumps.

6.4 Final Summary

As a result of undertaking this project it became clear that equipment within the pumping industry is fast catching up to the IT age. The product/application pro-

posed in this project can conceivably provide immense benefits for the engineering and industry sectors alike. It is ultimately envisaged that the pressure control system identified in this project will, after further development, refinement and thorough testing be marketed and sold within the pumping industry.

References

- Angelfire (2005), *Online Definition - Serial Device*, [Online], Available: www.angelfire.com/ny3/diGi8tech/SGlossary.html, [Accessed 28 September 2005].
- Australian Pump Manufacturers Association Ltd (1987), *Australian Pump Technical Handbook*, third edn, APMA LTD, Canberra.
- Chaurette, J. (2005), *Fluide Design - Solved Problems*, [Online], Available: http://www.fluidedesign.com/solved_pumping_problems.htm, [Accessed 10 May 2005].
- Darby, R. (2001), *Chemical Engineering Fluid Mechanics*, second edn, Marcel Dekker Incorporated, New York, [Online], Available: Ebrary Books Online, [Accessed 30 April 2005].
- Davidson, G. (2002), *Centrifugal Pump: Parallel & Series Operation*, University of Pittsburgh School of Engineering, [Online], Available: www.engr.pitt.edu/chemical/undergrad/lab_manuals/centrifugal_pumps.pdf, [Accessed 10 August 2005].
- Five Star Electric Motors (2005), *Using Variable Frequency Drives on Pump Systems*, Five Star Electric Motors - Motors Controls Drives San Antonio, [Online], Available: <http://www.vfd.com/techarticles/vfd/Using/Variable/Frequency/Drives/on/Pump/Systems.pdf>, [Accessed 10 May 2005].

- Hydraulic Institute, Europump & the U.S. Department of Energy (DOE) Industrial Technologies Program (2004), *Variable Speed Pumping - A Guide to Successful Applications*, [Online], Available:<http://www.bpma.org.uk/Executive/Summary/-vsp.pdf>, [Accessed 9 May 2005].
- Karassik, I., Krutzsch, W., Fraser, W. & Messina, J. (1986), *Pump Handbook*, second edn, McGraw-Hill Book Company, New York.
- Kitamura, T., Matsushima, Y., Tokuyama, T., Kono, S., Nishimura, K., Komeda, M., Yanai, M., Kijima, T. & Nojin, C. (2000), *Physical Model-Based Indirect Measurements of Blood Pressure and Flow Using a Centrifugal Pump*, Vol. 24, [Online], Available:<http://www.blackwell-synergy.com/doi/abs/10.1046/j.1525-1594.2000.06605.x?cookieSet=1>, [Accessed 30 April 2005].
- Minghua, F. & Longya, X. (2000a), *Computer Modeling of Interactions of an Electric Motor, Circulatory System, and Rotary Blood Pump*, Vol. 46, Lippincott Williams and Wilkins, Inc, [Online], Available:<http://www.asaiojournal.com/pt/re/asaio/abstract.00002480-200009000-00020.htm;jsessionid=Caaaje1MmOuqUAUj8YsI7IVvbGm3kBRCza7cQ6bc4tMOaRdefqgxE1-4782852!-949856032!9001!-1>, [Accessed 30 April 2005].
- Minghua, F. & Longya, X. (2000b), *Computer Simulation of Sensorless Fuzzy Control of a Rotary Blood Pump to Assure Normal Physiology*, Vol. 46, Lippincott Williams & Wilkins, Inc., [Online], Available:<http://www.asaiojournal.com/pt/re/asaio/abstract.00002480-200005000-00006.htm;jsessionid=CaXYOa2dfWTxXiAR51AKzg710gyP2DbknAoBi8flARjdHXe1hG20!-4782852!-949856032!9001!-1>, [Accessed 23 April 2005].
- Nelik, L. (1999), *Centrifugal and Rotary Pumps - Fundamentals with Applications*, CRC Press, Boca Raton, Florida.

- PDL (2002), *Principles of Induction Motors Drives*, PDL Electronics, Napier, New Zealand. 4501-100 Rev c.
- Schildt, H. (2005), *Java - A Beginner's Guide, Third Edition*, McGraw Hill Osborne, Emeryville, California.
- Trinkl, J., Mesana, T., Havlik, P., Mitsui, N., Demunck, J., Dion, I., Candelon, B. & Monties, J. (1991), *Control of Pulsatile Rotary Pumps Without Pressure Sensors*, Vol. 37, [Online], Available: Compendex Database, [Accessed 13 May 2005].
- Vysta Virtual Automation Programming Platform Version 2.0 - Help File* (2002), PDL Electronics, Napier, New Zealand.
- Web Enabling Your Serial Device* (2002), Lantronix, Irvine, California.

Appendix A

Project Specification

**University of Southern
Queensland
Faculty of Engineering and
Surveying
ENG 4111 / ENG 4112 Research Project
Project Specification**

For: Craig Struthers
Topic: Real-Time Monitoring and Control of a Pressure Control System.
Supervisor: Dr Peng (Paul) Wen
Enrollment: ENG 4111 - S1, X, 2005
ENG 4112 - S2, X, 2005

Project Aim:

To develop and test an automated pumping controller which can keep constant pressure in a hydraulics system whilst having no physical contact with the medium (ie. no pressure transmitter). The development of a real time monitoring, configuration and control system software package to enable users to interface with the system with minimal training.

Programme: Issue A, 21st March 2005

1. Research current industrial standards.

-
2. Research equipment such as motors, VSD's and pumps for suitability.
 3. Research current programming languages for compatibility with the chosen equipment.
 4. Design control algorithms and functionality required to achieve the end result.
 5. Mathematically model the system so that physical attributes of the system can be obtained and reviewed.
 6. Investigate and develop an embedded WEB server communications system for interfacing of the controller.
 7. Build a prototype system and complete full functional testing.

As time permits:

1. Develop an off the shelf item for retail sale.

AGREED:

----- (Student) ____ / ____ / 2005

----- (Supervisor) ____ / ____ / 2005

Appendix B

Variable Speed Drive (VSD)

Operating Manual



PDL ELECTRONICS LTD

**ELITE SERIES
GETTING STARTED
MANUAL**

Head Office:
81 Austin Street
P.O. Box 741
Napier
New Zealand
Tel.: +64-6-843-5855
Fax.: +64-6-843-5185



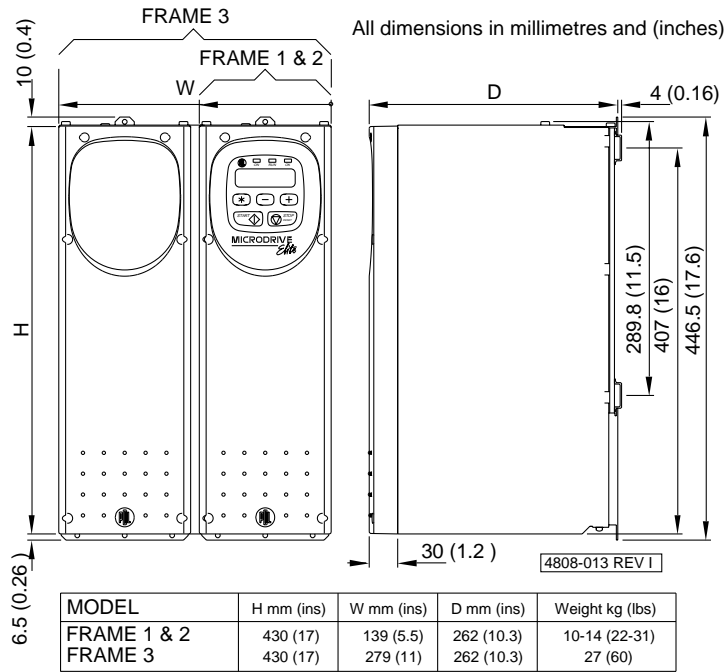


Figure 1.1: Microdrive Elite Frame 1 to 3 Dimensions

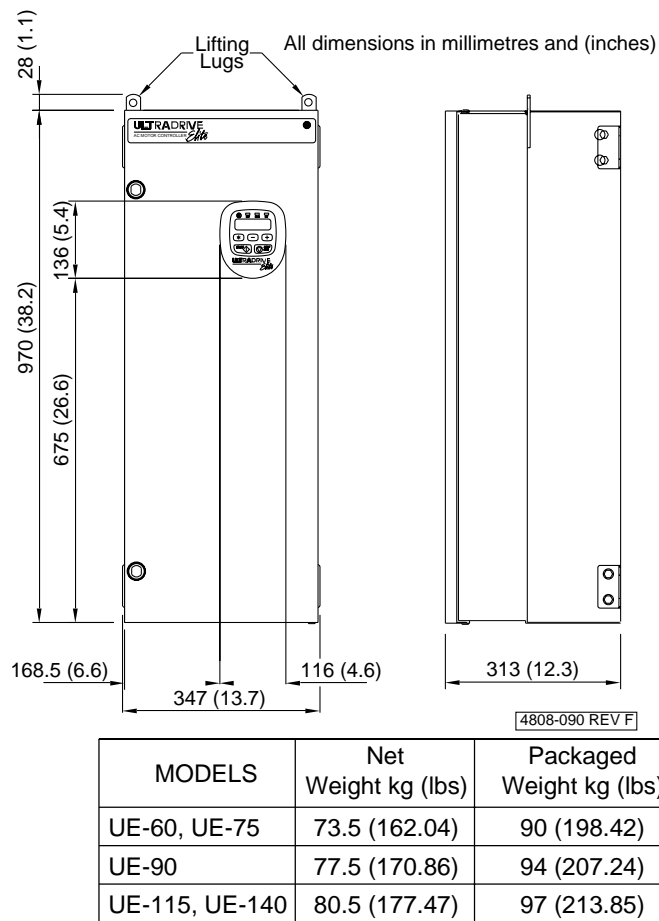


Figure 1.2: Ultradrive Elite Frame 4 Dimensions

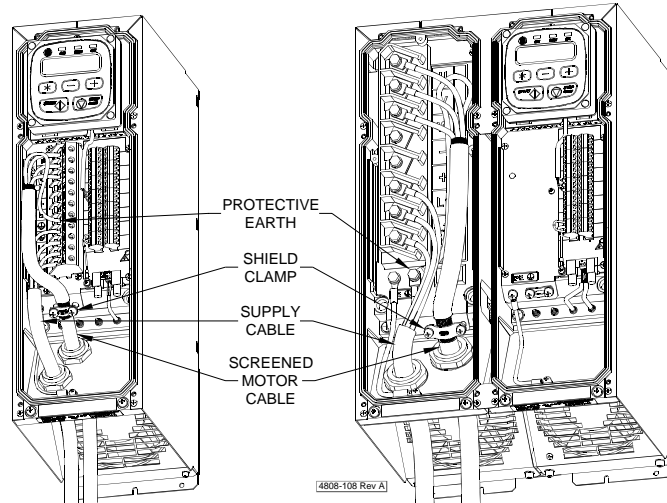


Figure 1.10: Microdrive Elite Frame 1 to 3 Screened Motor Cable Configuration

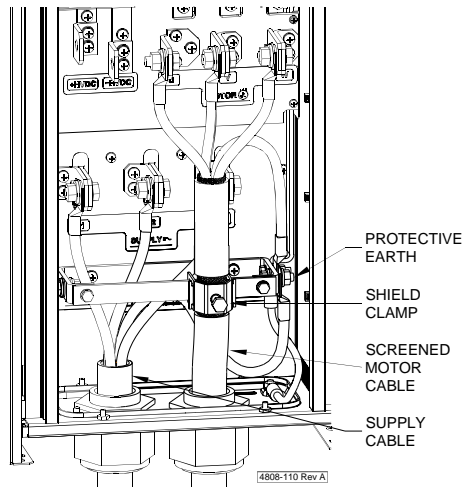


Figure 1.11: Ultradrive Elite Frame 4 Screened Motor Cable Configuration

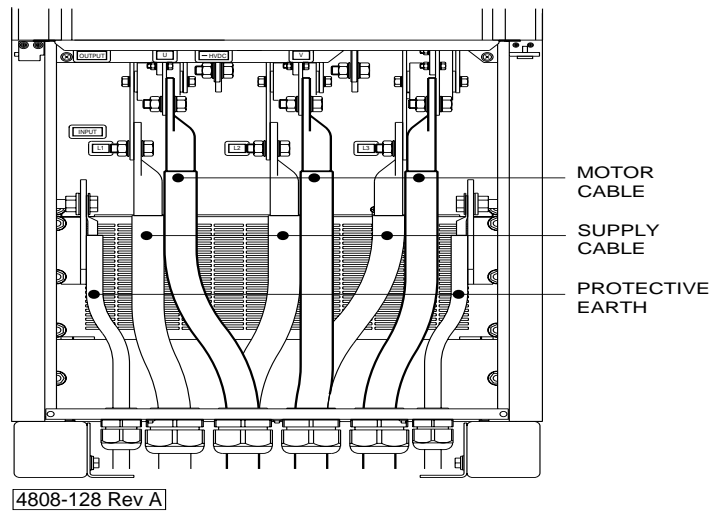
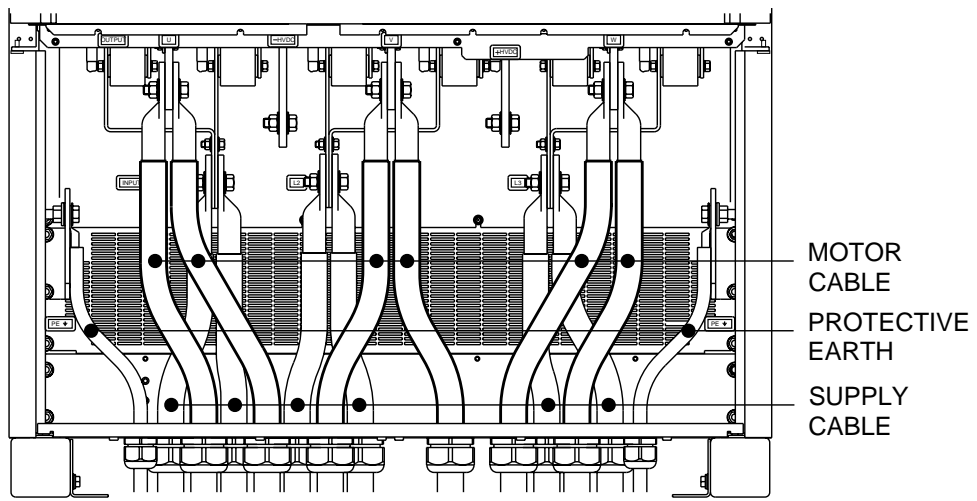
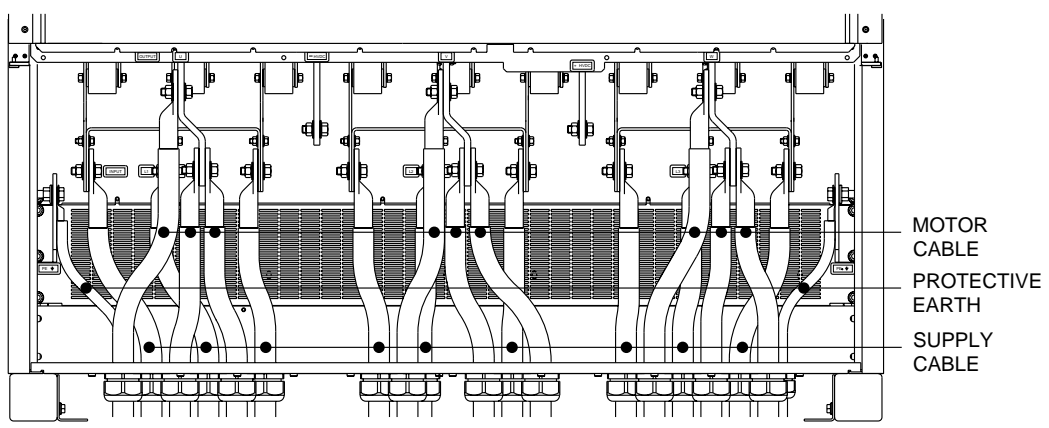


Figure 1.12: Ultradrive Elite Frame 5 Screened Motor Cable Configuration



4808-129 Rev A

Figure 1.13: Ultradrive Elite Frame 6 Screened Motor Cable Configuration



4808-130 Rev A

Figure 1.14: Ultradrive Elite Frame 7 Screened Motor Cable Configuration

1.3.2 MOTOR ROTATION

USE OF “+” AND “-”

“+” Speed is used to describe speed in the forward direction.

According to IEC34-7, the motor rotates clockwise when:

- viewed from the shaft end
- terminals U1, V1 and W1 or U2, V2 and W2 are connected to the Elite Series output phases U, V, W respectively
- the Elite Series is operating with “+” speed.
- “-” is used to describe speed in the reverse direction of the motor.

1.3.3 DISPLAY MOUNTING

The display unit may be rotated in 90° increments, to suit the mounting orientation of the Microdrive Elite Series. The display unit may also be mounted remotely from the drive, to a maximum of 3 metres.

1.3.4 CONTROL WIRING

Control Wiring Recommendations

Bring the control wiring into the enclosure through the gland plate, and install glands to maintain IP54 integrity. Loom control wiring and power wiring separately, at least 300 mm apart and crossing only at right angles. Control cables must be screened to ensure correct operation. Connect the screen only to the ground at the Elite Series to prevent ground loops.

Connection recommendations are:

Maximum tightening torque:	0.5 Nm (4.5 lb-in)
Maximum cable size:	1.5 mm ² appliance wire (26 - 14 AWG Cu)
Maximum number of cables per terminal:	Two
Cable stripping length:	7 mm (0.28 in)

The default configuration of the digital inputs is active high. i.e., the common of all multi-function input switches should be connected to +24Vdc (Terminal T21).

The External Trip/PTC input must be connected to +24Vdc (Terminal T21) (when set for active high) for the Elite Series to start and run a motor.

1.3.5 EARTHING OF CONTROL 0V

To comply with the requirements of a Class 1 earthing system, the Elite Series control 0V must be linked to earth at some point. Connection of multiple earth points may cause earth loops and should be avoided. An earth link is provided between Terminal T20 and the terminal surround plate and must be removed if not required. Removal will allow the 0V point to float up to ±50Vdc (30Vac) from chassis earth.

1.3.6 SHAFT ENCODER SELECTION AND MOUNTING

The encoder orientation shown in the drawings in this manual (i.e., the connection of the A and B outputs) assumes the encoder is to be connected directly to the non-drive end (non-shaft end) of the motor and that motor wiring orientation is normal (motor terminals U1, V1 and W1 are connected to Elite Series terminals U, V, W, respectively). In this case, an increasing count (Screen Z9) should correspond to rotation in the positive direction (motor shaft rotates clockwise when the motor is viewed from the drive end), in response to a positive speed reference.

If the encoder direction is inverted (e.g., by mounting at the drive end or using an inverting belt coupling), A and B, or for a differential encoder, A and A signals should be swapped. Refer Figure 5.4.

A shaft encoder will be needed if operating the Elite Series in closed loop vector control mode.

Choice of Encoder

If the Elite Series is to be used in Closed Loop Vector control mode, a shaft encoder will need to be connected to the motor. A specification for a suitable encoder for a 50 or 60Hz motor is:

Encoder type:

Incremental, quadrature (bi-phase), differential or single-ended output. Push-pull output preferred to maximise range.

Recommended ppr:

1000 to 2000 ppr per motor pole pair, for directly driven encoder

Minimum ppr:

500 ppr per motor pole pair (4 pole motor = 1000 ppr)

Supply requirement:

5Vdc, 100mA maximum

Alternative Specification:

Type:

Single ended push-pull - will cause a reduction in noise immunity.

Or:

Single ended open collector - pulses will be distorted by long cables. For this type of encoder the product of cable length (metres) x maximum frequency (kHz) should not exceed 1500. Absolute maximum cable length is 30 m.

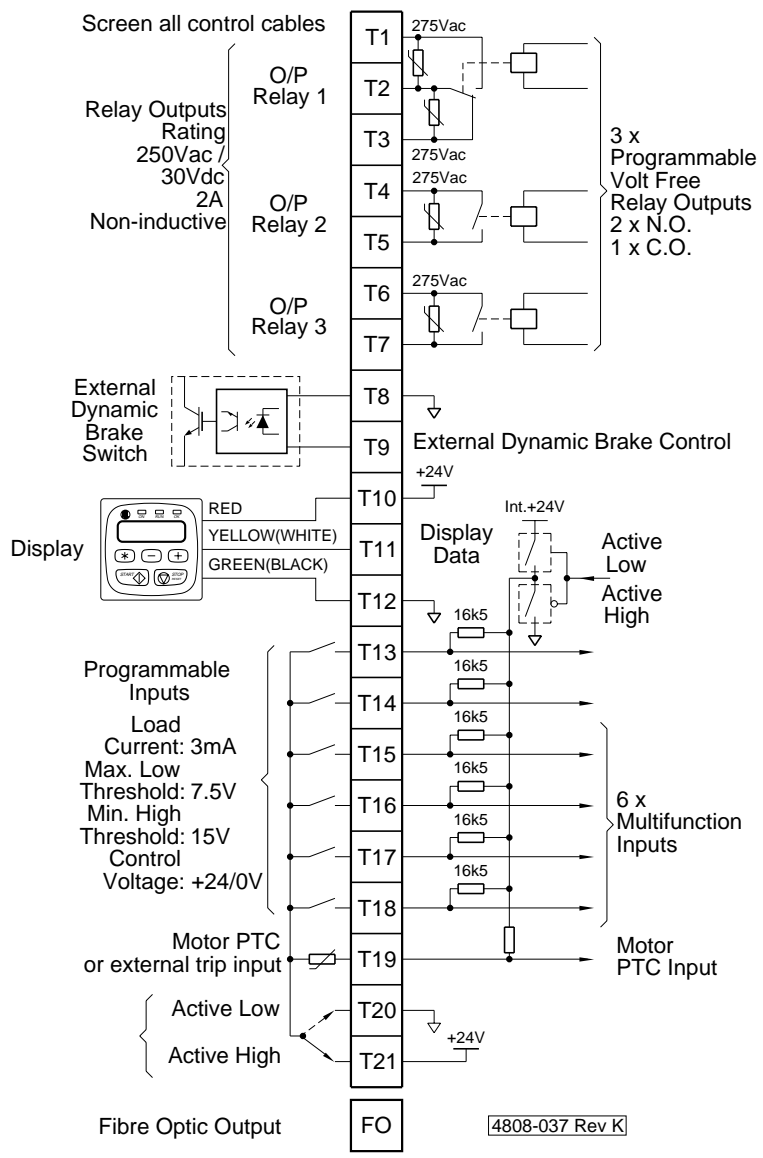


Figure 1.17: Control Inputs and Outputs

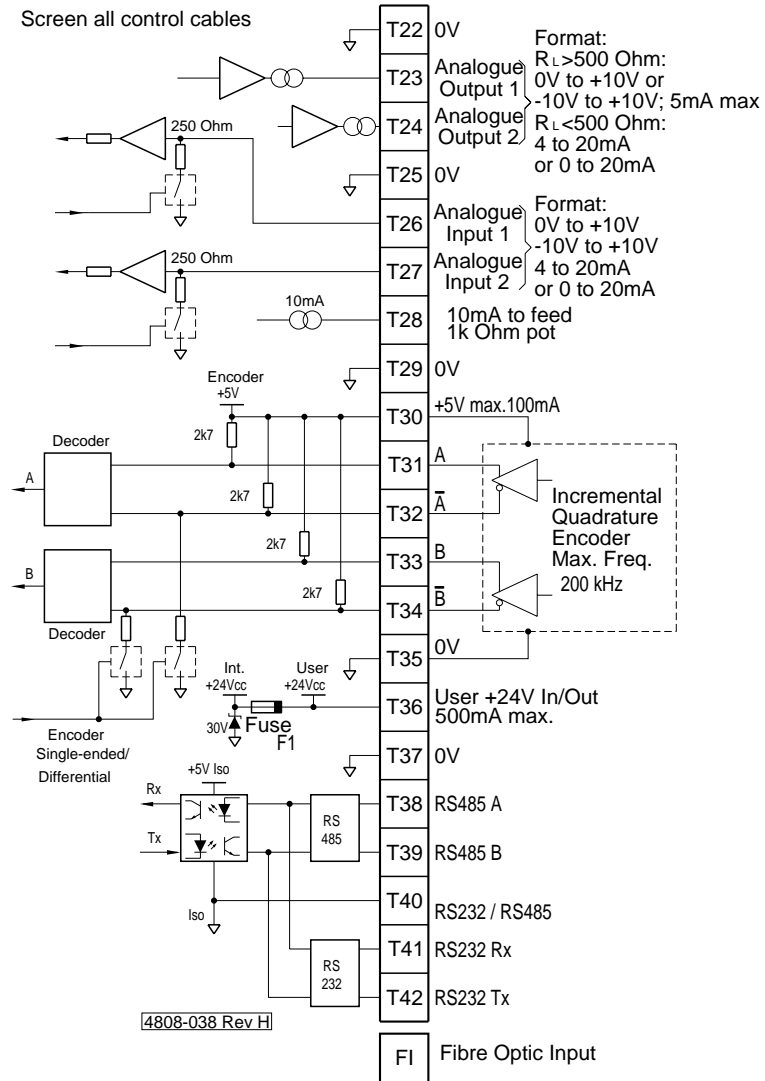


Figure 1.18: Control Inputs and Outputs

2.1 DISPLAY UNIT DESCRIPTION

2.1.1 THE DISPLAY UNIT AND KEYS

Refer to Figure 2.1 for Display Unit Details

STATUS LINE : Indicates drive status, overload status, output torque, output speed.

CONTROL LINE : Indicates screen number, screen description, parameter for adjustment.

SCREEN CONTROL KEYS

“+” and “-” keys enable scrolling between screen groups and subscreens.

“*” allows unfolding of screens if required.

By holding down the “*” key and using the “+” or “-” keys, individual modes or parameters can be adjusted, if allowed.

START / STOP-RESET PUSH-BUTTON

If keyboard control is enabled, these push-button allow starting or stopping/resetting of the Elite Series. This may be in conjunction with external START and STOP push-button.

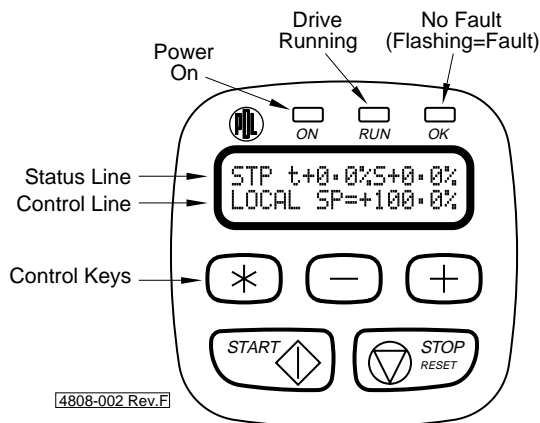


Figure 2.1: Elite Series Display Unit

2.1.2 SELECTION OF SCREENS

Screens are arranged in folded format. Each screen group has a main screen with the group identifying letter and description. Folded under this main screen can be a number of subscreens, each of which has a single parameter or mode for viewing or adjustment. These subscreens cannot be viewed until unfolded. Once unfolded, some subscreens have a numerical parameter which may be adjusted. Others may have a list of options with each option separately viewable and selectable. Extra screens or subscreens may become available when the Elite Series is in “Commissioning” mode.

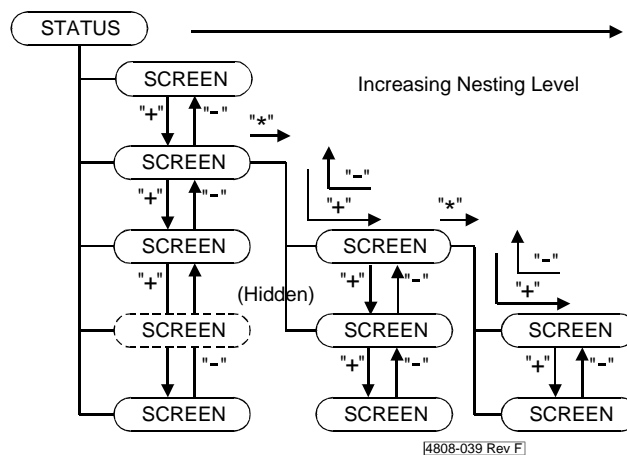


Figure 2.2: Control of Screen Folding

Referring to Figure 2.2, when “+” or “-” are used to scroll through the screens, no subscreens are shown. If a particular subscreen is required, scroll to the associated group, then press and release “*”. This will unfold all of the screens associated with that group. “+” will move down through the subscreens, stopping on the last subscreen in a group. “-” will move up through the subscreens, until the group title is reached. This will cause the screens to automatically refold.

2.1.3 PARAMETER AND MODE ADJUSTMENT

Once a screen group has been unfolded and a screen selected, the parameter or mode displayed on the control line may be adjusted. For a screen with access rights configured as "hidden" or "read only", this adjustment may only be made if the Elite Series is in COMMISSIONING mode.

Adjustment is done by depressing the "*" key and using "+" or "-" keys, to increase or decrease the parameter respectively.

2.2 CONFIGURING OF OPERATING MODE

Before livening the Elite Series motor controller, it is important that you know the intended operating mode and control configuration of the drive. These may have been preset into the Elite Series before dispatch. Alternatively this may have been predetermined by an Applications Engineer but still need to be programmed into the Elite Series. If this is the case, you as the installer may have to temporarily set up a mode and configuration, to allow livening and testing.

2.2.1 OPERATING MODES

OPERATION Mode

This is the normal operating mode of the drive. Each screen will have a pre-configured access right, controlling whether it is hidden, read only, or read-write. Thus operator access to screens can be controlled.

Read Only: The screen can be viewed, but not changed.

Read-Write: The screen can be viewed and the parameter changed when in OPERATION mode.

Hidden: The screen cannot be viewed or changed.

COMMISSIONING Mode

In this mode, each screen is visible and commissioning parameters may be adjusted, irrespective of the screen's access right. Some parameters are not adjustable while the drive is started or running.

Access to COMMISSIONING Mode may be controlled by a password.

2.2.2 SWAPPING BETWEEN OPERATION AND COMMISSIONING MODES

Selecting COMMISSIONING mode before a Password has been set:

Scroll to Main Screen Z.

Z COMMISSION=N

Hold down "*" key and use "+" or "-" keys and the status line should change to:

Z COMMISSION=Y

All screens will now be visible, and all parameters be adjustable.

Selecting COMMISSIONING mode after a Password has been set:

Figure 2.3 illustrates the procedure for swapping between OPERATION and COMMISSIONING modes using a password.

Scroll to Main Screen Z. The display's control (bottom) line will read:

Z COMMISSION=N

Hold down "*" and use "+" or "-" keys and the screen will automatically display:

PASSWORD= ZZZZZ

Hold down "*" key and use "+" or "-" keys until the correct password is reached. Then release the keys.

All screens will now be visible, and all parameters be adjustable.

Selecting OPERATION Mode:

To change from COMMISSIONING Mode to OPERATION Mode, scroll to Screen Group Z.

The display's control line will read:

Z COMMISSION=Y

Hold down "*" key and use "+" or "-" keys to toggle to :

Z COMMISSION=N

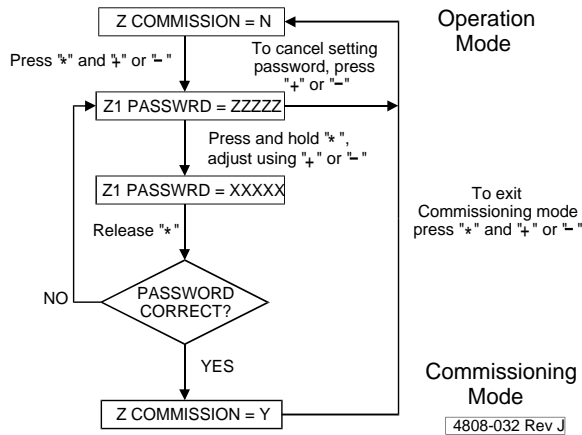


Figure 2.3: Setting Commission Mode after a Password has been set

2.2.3 SETTING A PASSWORD FOR THE FIRST TIME

Refer to Figure 2.4.

Once set to COMMISSIONING mode as described above, a password may be set up. Unfold Screen Group Z and scroll to Screen Z1. The display will read:
Z1 PASSWORD=OFF

Hold down "*" key and use "+" or "-" keys to set the required password.

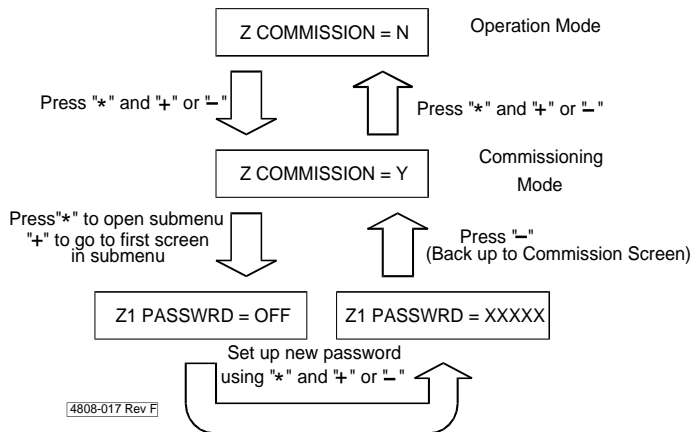


Figure 2.4: Setting a Password for the First Time

What happens if a password is unknown or forgotten?

Once a password has been entered, a special hashing number is displayed on Screen Z when trying to enter COMMISSIONING mode.

The display will read:
PASSWORD= ZZZZZ

Take a note of this number and contact a PDL Electronics Applications Engineer, who with suitable authority will be able to pass this code through an algorithm to reconstruct the original password.

4.1 OPERATION MODE AND CONFIGURATION

Figure 4.2 summarises the Screen List available by default. Full descriptions of all screens are given in the Elite Series Technical Manual, PDL Part No. 4201-180.

4.1.1 OPERATION MODES

The Elite Series may be set up to run in one of four operation modes. These are shown in Figure 4.1.

V/Hz Operating Mode:

For general-purpose speed control applications, e.g. pumps, fans, conveyors etc. A shaft encoder is not needed. This open loop speed control mode generates an output with a fixed voltage vs frequency profile. Suitable for running multiple parallel motors from one Elite Series. Select by setting Screen X1 Control Type = V/Hz.

Also use V/Hz mode when autotuning an Elite Series motor controller.

Closed Loop Vector Mode - Torque Control:

For use in torque control applications, e.g. winder systems, position control applications with an external speed-position controller. A quadrature shaft encoder will be required on the motor, to provide rotor position feedback.

To set up this mode of operation, set the encoder pulses per motor shaft revolution on Screen N8 and program Screen X1 to Closed Loop Vector. Then select torque control mode, either by appropriately configuring one of the multi-function inputs (Screen I7c to I7h, Selection 16 Speed/Torque Mode) and activating the switch, or by setting for torque control mode (Screen A1 LOCAL MODE=TQ).

Closed Loop Vector Mode - Speed Control:

Recommended for servomotor type applications, where fast dynamic response is required, and for crane hoists and other applications where full torque capability at zero speed is required. A quadrature shaft encoder is required on the motor, to provide rotor position and speed feedback.

To set up this mode of operation, set the encoder pulses per motor shaft revolution on Screen N8 and program Screen X1 to Closed Loop Vector. Then select speed mode, either by appropriately configuring one of the multi-function inputs (Screen I7c to I7h, Selection 16 Speed/Torque Mode) and deactivating the switch, or by setting for speed control mode (Screen A1 LOCAL MODE=SP).

When operating in closed loop vector mode, switching between speed control and torque control modes can be done without stopping the Elite Series.

Open Loop Operating Mode:

For general-purpose speed control applications, e.g. pumps, fans, conveyors etc. A shaft encoder is not needed. Configuration to this mode is set by programming Screen X1 to Open Loop Vector.

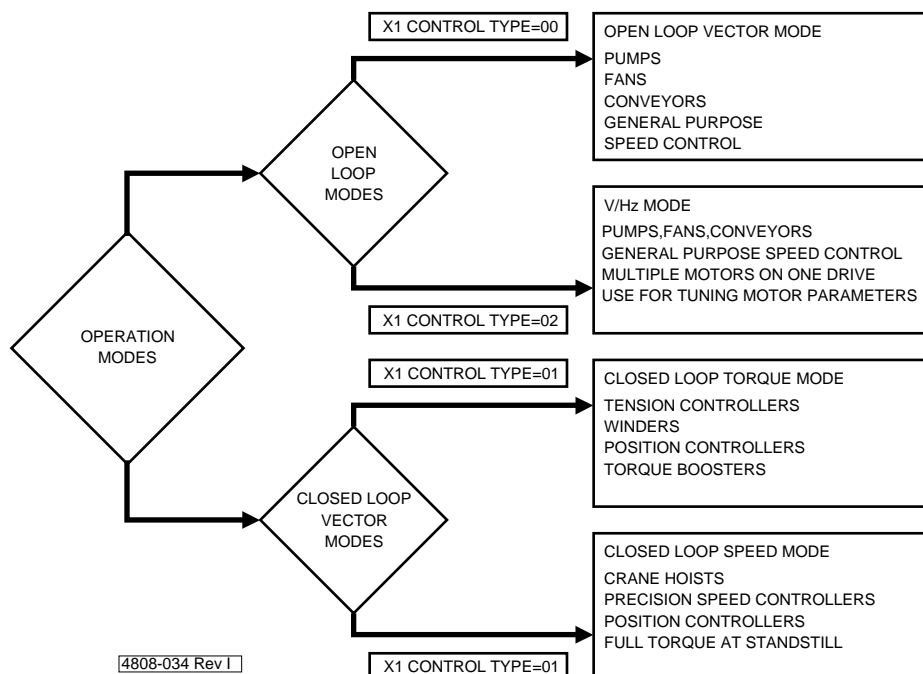


Figure 4.1: Elite Series Operation Modes

4.1.2 INPUT CONFIGURATION

If Start–Stop/Reset is not required from the display unit, set Screen I1. (Refer to Elite Series Technical manual 4201-180).

Select the required speed or torque reference source from Screen I2, I3. If an alternative source is required, e.g. for local/remote or auto/manual control, select from Screens I4, I5.

If Analogue Input 1 is to be used as a reference source, set format and scaling from Screens I6a, I6b, I6c. Similarly, Screens I6d, I6e, I6f set up Analogue Input 2.

If a zero band is required, set on Screen I6g. This sets a definite zero speed or zero torque region when using either analogue input.

If the fibre optic input is to be used as a reference source, set scaling from Screens I8a, I8b.

Configure the multi-function inputs (MFIs) from Screens I7. Screen I7a programs the MFIs in groups, while I7c to I7h programs each individually.

Configure the MFIs for active high or active low from Screen I7b.

4.1.3 OUTPUT CONFIGURATION

Select the function, format and scaling of Analogue Output 1 from Screens O1a, O1b, O1c, O1d. Similarly, Screens O1e, O1f, O1g, O1h set up Analogue Output 2.

Select the required output relay functions from Screens O2a, O2c, O2e, and their sense from Screens O2b, O2d, O2f.

If using the fibre optic output, set function and scaling from Screens O3a, O3b, and O3c.

4.1.4 ACCELERATION AND DECELERATION RATES

If operating the Elite Series as a speed controller, set required acceleration and deceleration rates from Screens R1, R2. Generally, set for the required response without torque limiting when accelerating (indicated by TLT on status line of display) and without excess regeneration on deceleration (indicated by VLT on status line). These rates active only when speed controlling.

If two rates are required, set alternative rates and break speed on Screens R3, R4, R5.

Set required deceleration rate when emergency stopping on Screen R6.

Set an appropriate Stop Timeout on Screen S11.

4.1.5 SPEED AND TORQUE LIMITS

Set speed limits by Screens L2, L3. Normally set outside the range of the reference speed input. Should be active only when in torque control mode on light load. Indicated by SLT on status line of display.

Set torque limits by Screens L4, L5. Normally set outside the range of the torque reference input. Should only be active when in speed control modes, on overloads (indicated by TLT on status line of display). Also torque limiting becomes active on loss of shaft encoder pulses when running in closed loop vector mode.

Set speed limit timeout on Screen L6. Drive will trip if speed limiting exceeds this time.

Set torque limit timeout on Screen L7. Drive will trip if torque limiting exceeds this time. Provides protection against loss of shaft encoder pulses.

4.1.6 MULTI-REFERENCES

Set Screens M1 to M7 in conjunction with certain input modes (Screens I7) as preset torque or speed references.

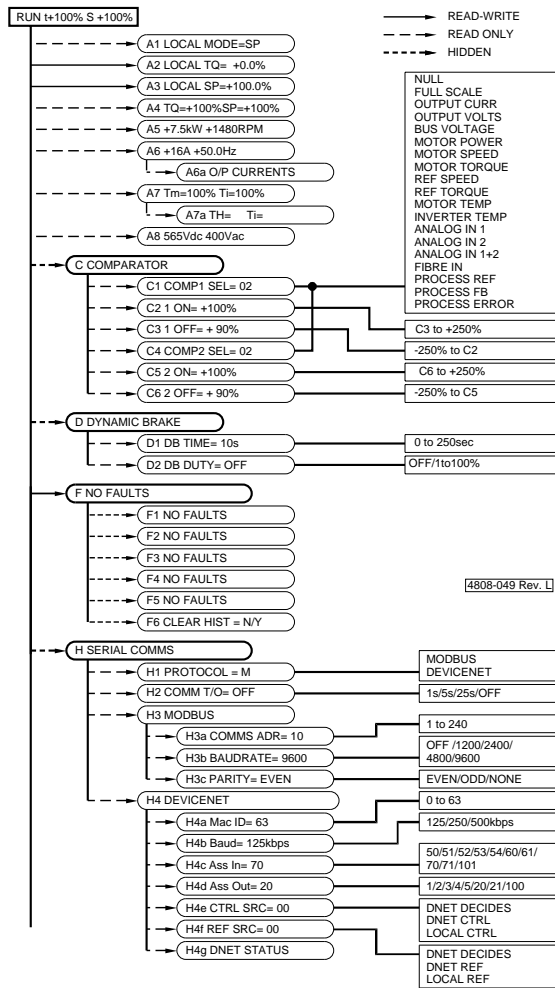


Figure 4.2: Default Screen Lists A-H

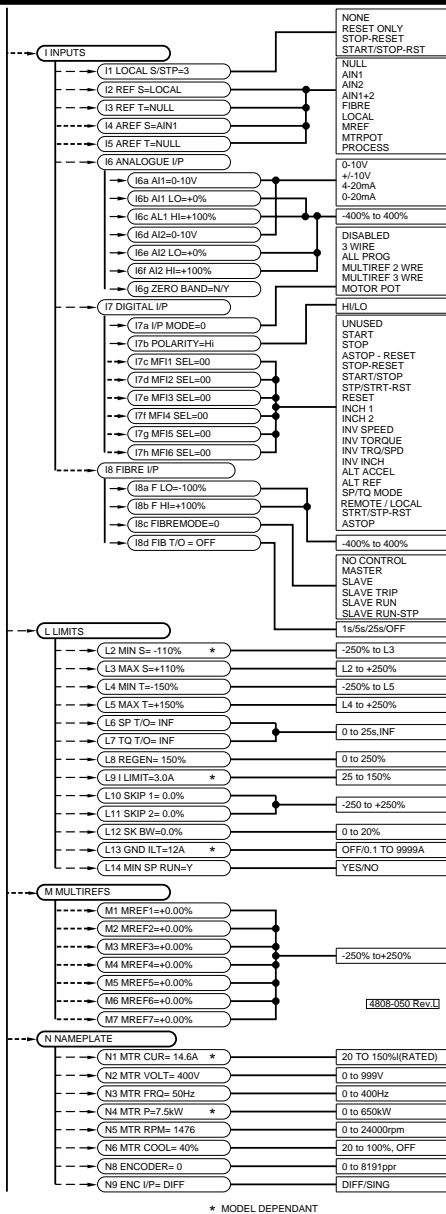


Figure 4.3: Default Screen Lists I-N

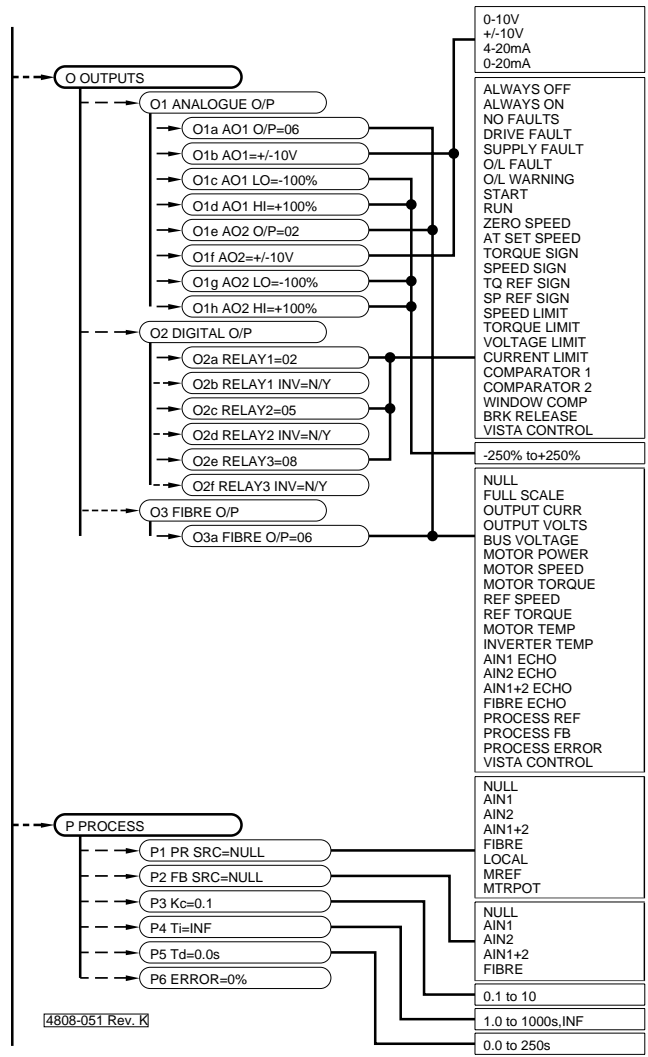


Figure 4.4: Default Screen Lists O-P

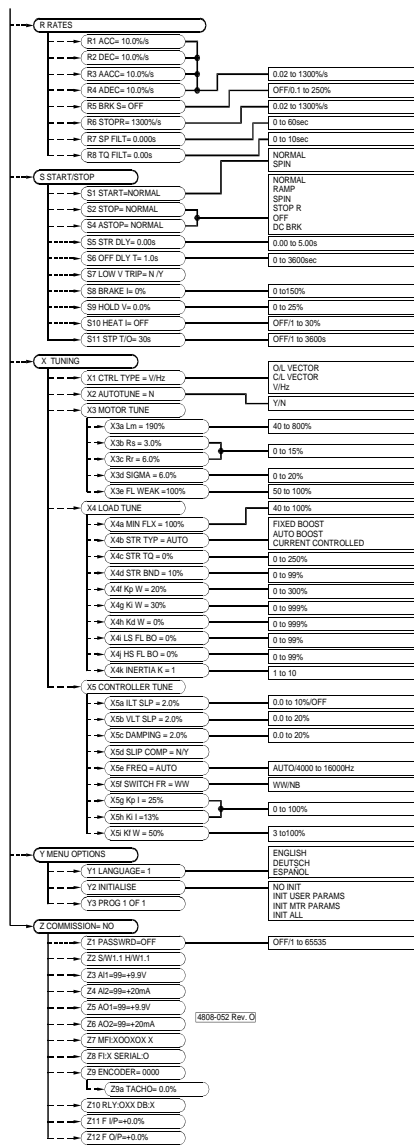


Figure 4.5: Default Screen Lists R-Z

Appendix C

MODBUS Protocol & VSD Registers

C.1 MODBUS Protocol

3 SUPPORTED MODBUS FUNCTIONS

3.1 INTRODUCTION

The Elite serial communications protocol adheres completely to the industry standard Modbus protocol. The Elite supports a subset of the complete Modbus function set, listed in Table 3.1.

Function	Description
3	Read Multiple Holding Registers
16	Force Multiple Holding Registers

4202-232 Rev A

Table 3.1 Modbus Functions supported by the Elite

Modbus Function 3 and 16 refer to Holding Registers with addresses of the form 4XXXX. All data address references are made relative to the first address of the particular Modbus Function. In the case of Functions 3 and 16, the first address is 40001(decimal), and this would be referenced as 00000.

An example Modbus Function 3 query message is shown in Fig. 3.1, and a typical response message is given in Fig. 3.2. A fuller explanation of the message contents is given in Section 3.2.

All word length(2 byte) variables are transmitted with the most significant byte first. The holding register contents are transmitted in a 16-bit format with the most significant byte first. Where appropriate, negative numbers will be transmitted in “two’s complement” format. If more than one register is requested, the lowest addressed register is transmitted first and the other holding registers will follow in sequential order.

3.2 MODBUS FUNCTION 3 - READ MULTIPLE HOLDING REGISTERS

This function allows a Modbus master to acquire the contents of a holding register from the addressed slave unit. This function will only access the individually addressed Elite and does not support global broadcast or group broadcast addressing modes.

The Elite implementation of this function allows up to 16 holding registers to be read in one message. All registers can be read through Function 3.

An example Modbus Query message showing the format of the Read Multiple Holding Registers is shown in Fig. 3.1.

The example shown in Fig. 3.1. reads from an Elite whose Modbus system address is 10(decimal), 0A(hexadecimal). As the data addresses use relative addressing (refer Section 3.1) the starting address is relative to 40001. This example reads the value of holding registers 40001 to 40003, so the starting address relative to 40001 is 0(decimal), 0000(hexadecimal).

Modbus System Address	Modbus Function Code	Data Address Start(40001)	Number of Holding Registers(=3)	CRC
0AH	03H	00H 00H	00H 03H	04H B0H

4202-228 Rev A

Fig. 3.1 Example Modbus Message using RTU Framing

A typical response to the example Function 3 Query shown in Fig 3.1 is shown in Fig 3.2.

The data byte count is the number of data bytes following in the message. This is computed as two bytes per register, giving a byte count of six.

The data returned is interpreted as follows. The Data Address 40001 is the rated (nameplate) motor current and has a value of $(100\% \times 7537/8192) = 92.0\%$ of the rated drive current (2.5A, so motor current is $0.92 \times 2.5 = 2.3A$). The Data Address 40002 is the rated (nameplate) motor voltage and has a value of 230V. The Data Address 40003 is the rated (nameplate) motor frequency and has a value of 50Hz.

Modbus System Address	Modbus Function Code	Byte Count	Data(40001) (=7537)	Data(40002) (=230)	Data(40003) (=50)	CRC
0AH	03H	06H	1DH 71H	00H E6H	00H 32H	4DH E1H

4202-229 Rev A

Fig. 3.2 Example Modbus Message Response using RTU Framing

The CRC value is calculated from all the bytes in the response including the Modbus system address, Modbus function code, the data address start, number of holding registers, the byte count and the data bytes. The method used for calculating the CRC value is discussed in Appendix B.5. The data byte count is the number of data bytes in the response message - 3 word length registers, each of 2 bytes gives 6 bytes of data.

When an error has occurred (e.g. by trying to read from a non-existent register), the Elite returns an exception response containing an exception code indicating the type of error. Refer to Appendix B.6 for the exception response format and an explanation of the exception codes.

3.3 MODBUS FUNCTION 16 - FORCE MULTIPLE HOLDING REGISTERS

This function allows a Modbus master to control the value of a number of holding registers in the addressed slave unit. Any holding register which is not read-only may be forced to a new value. Since the Elite will still have control over the value in each of the holding registers, the value may still be modified by the Elite after being set by the Modbus master.

The Elite implementation of this function allows up to 16 holding registers to be forced in one message.

This function supports individual addressing as well global broadcast and group broadcast addressing modes. Global broadcast and group addressing modes are discussed in more detail in Section 4.2 and Section 5.3.

An example Modbus Query message showing the format of the Force Multiple Holding Registers is shown in Fig. 3.3.

The example shown in Fig. 3.3 writes to an Elite whose Modbus system address is 10(decimal) 0A(hexadecimal). The data address 40001 is the rated (nameplate) motor current and is modified to $(100\% \times 1.5A / 2.5A) = 60\%$ of drive rated (nameplate current), so the actual value is $(0.6 \times 8192) = 4915$ (decimal), or 1333(hexadecimal). The data address 40002 is the rated (nameplate) motor voltage and is modified to 400V(decimal), 190(hexadecimal).

Modbus System Address	Modbus Function Code	DataAddress Start(40001)	Number of Registers(=2)	Byte Count	Data(40001) (=4915)	Data(40002) (=400)	CRC
0AH	10H	00H 00H	00H 02H	04H	13H 33H	01H 90H	23H FCH

4202-230 Rev A

Fig. 3.3 Function 16 - Force Multiple Holding Registers

A typical response to the example Function 16 Query shown in Fig 3.3 is shown in Fig 3.4.

The response contains the address of the first holding register and the number of registers modified.

Modbus System Address	Modbus Function Code	Data Address Start (40001)	Number of Registers (=2)	CRC
0AH	10H	00H 00H	00H 02H	40H B3H

4202-231 Rev A

Fig. 3.4 Function 16 Response - Force Multiple Holding Registers Response

It should be noted that a value modified over a Modbus message will not be stored in non-volatile memory (EEPROM); the value will be lost when the Elite is powered down. A separate message must be sent that causes a value to be saved in EEPROM.

An example Modbus Query showing the format of a message to cause a rated name plate current to be saved to EEPROM is shown in Fig. 3.5

Modbus System Address	Modbus Function Code	Data Address = 40885	Number of Registers	Byte count	Data = 40001	CRC
0AH	10H	03H 74H	00 01H	02H	9CH 41H	45H E4H

4202-240 Rev A

Fig. 3.5 Example of message that causes rated name plate current to be saved to EEPROM

The CRC value is calculated from all the bytes in the response including the Modbus system address, Modbus function code, the data bytes and the data byte count. The method used for calculating the CRC value is discussed in Appendix B.5. The data byte count is the number of data bytes in the response message - 2 word length registers, each of 2 bytes gives 4 bytes of data.

When an error has occurred (e.g. by trying to write to a read-only register), the Elite returns an exception response containing an exception code indicating the type of error. Refer to Appendix B.6 for the exception response format and an explanation of the exception codes.

C.2 VSD Registers

4 ELITE DATA REGISTERS

4.1 ELITE HOLDING REGISTERS

There are 188 holding registers in the Elite that can be accessed over the Modbus System. Section 4.3 summarises the accessible Elite holding registers in Modbus data address order. Section 4.4 cross-references the parameters as they appear on the Elite screens, to Elite holding registers. A more detailed explanation of each holding register is given below. The Modbus system designer should refer to the Microdrive Elite Instruction Manual (PDL Part No. 4201-180) for more information.

Note: Modbus Function 3 allows up to 16 registers to be read at once.
Modbus Function 16 allows up to 16 registers to be written to at once.

4.2 SHORT-FORM MODBUS REGISTER DETAILS ORDERED BY MODBUS ADDRESS

Address	Screen	Description	Range	Scaled Range
40001	N1	Rated (nameplate) motor current	20..150%	1638..12288
40002	N2	Rated (nameplate) motor volts	0..999Vac	0..999
40003	N3	Rated (nameplate) motor frequency	0..400Hz	0..400
40004	N4	Rated (nameplate) motor power	0..650kW	0..65000
40005	N6	Motor cooling at zero speed	20..101%	1638..8273
40006	N5	Rated (nameplate) motor speed	0..2400rpm	0..24000
40007	N8	Pulse per revolution of tach encoder	0..8191ppr	0..8191
40008	X2	Autotune motor	0..1	0..1
40010	L8	Regeneration limit	0..250%	0..20480
40011	L2	Minimum speed	-250%..Max Limit	-20480..Max Limit
40012	L3	Maximum speed	Min Limit..+250%	Min Limit..+20480
40013	L6	Speed limit timeout	0..25s,INFINITE	0..26000
40014	L4	Minimum torque	-250%..Max Limit	-20480..Max Limit
40015	L5	Maximum torque	Min Limit..+250%	Min Limit..+20480
40016	L7	Torque limit timeout	0..25s,INFINITE	0..26000
40017	X4c	Starting torque (boost) adjustment	0..250%	0..20480
40018	L9	Current limit	25..150%	2048..12288
40019	X4d	Start Band	0..100%	0..8192
40020	D1	Time constant of dynamic brake resistor	0..250s	0..250
40021	D2	Duty rating of dynamic brake resistor	OFF,1..100%	0..8192
40030	H3a	Modbus serial comms address	1..240	1..240
40031	H3b	Modbus serial comms baud-rate	0..3	0..3
40032	H2	Modbus serial comms timeout period	0..3	0..3
40040	-	Acceleration rate reference	0.1..6000%/s	1..60000
40041	-	Deceleration rate reference	0.1..6000%/s	1..60000
40042	R6	Decel(stopping) rate used when stopping	0.1..6000%/s	1..60000
40043	R7	Speed filter time constant	0..100s/(100%/s)*10	0..1000
40044	R8	Torque filter time constant	0..10s	0..10000
40050	S6	Off delay time	0..25s, INFINITE	0..2 6000
40051	S5	Start delay time	0..1s	0..1000
40052	-	Stop mode in use	0..5	0..5
40053	S7	Mains power loss response	0..1	0..1
40056	S8	Level of dc current used for braking	0..150%	0..12288
40057	S1	V/Hz starting mode	0..2	0..2
40058	S9	DC Hold level	0..25%	0..2048
40059	S10	DC Heat level	OFF, 1..30%	0..2458
40060	X3e	Field weakening point	50..100%	4096..8192

Note *** indicates that this parameter is a read only parameter.

Table 3.2: Elite Modbus Register Details

40061	X3a	Main inductance	40..800%	3276..65535
40062	X3b	Stator resistance	0..15%	0..1228
40063	X3c	Rotor resistance	0..15%	0..1228
40064	X3d	Total leakage	0..20%	0..1638
40066	X5g	Current PI loop proportional gain	0..100%	0..8192
40067	X5h	Current PI loop integral gain	0..100%	0..8192
40068	X4g	Rotor speed PID loop integral gain	0..4096	0..4096
40069	X4h	Rotor speed PID loop derivative gain	0..4096	0..4096
40070	X4f	Rotor speed PID loop proportional gain	0..300%	0..24576
40071	X5i	Rotor speed PID filter constant	3..100%	246..8192
40072	X5e	Modulation type	0..1	0..1
40073	X5f	Modulation frequency	AUTO,4000..16000	3999..16000
40080	-	Host reset control	0..1	0..1
40081	-	Host stop control	0..1	0..1
40082	-	Host start control	0..1	0..1
40083	-	Host trip control	0..1	0..1
40084	-	Speed / Torque Mode reference	0..1	0..1
40085	I1	Local start stop and reset control	0..3	0..3
40088	A3	Local speed reference	-250..+250%	-20480..+20480
40089	Status Line	, overload, speed/torque indication ***	0..128	0..128
40090	Status Line, A5	Motor speed ***	-400..+400%	-32768..+32767
40091	Status Line	Motor torque ***	-400..+400%	-32768..+32767
40092	A8	DC bus voltage ***	0..800%	0..65535
40093	A6	Current output ***	0..800%	0..65535
40094	-	Tacho calculated rotor speed ***	-32768..+32767	-32768..+32767
40095	F	Current fault status ***	0..39	0..39
40098	A5	Power output ***	-400..+400%	-32768..+32767
40099	A8	Voltage output ***	0..800%	0..65535
40100	A7	Estimated motor temperature ***	0..800%	0..65535
40101	A7	Estimated inverter temperature ***	0..800%	0..65535
40102	-	Estimated D/B resistor temperature ***	0..800%	0..65535
40103	-	Reserved ***	-50..100	-50..100
40104	-	Reserved ***	-50..100	-50..100
40107	I8c	Fibre optic control mode select	0..5	0..5
40108	-	Fibre reference input ***	-400..+400%	-32768..+32767
40110	-	Fibre reference output ***	-400..+400%	-32768..+32767
40113	I8d	Fibre optic communication timeout	0..3	0..3
40114	Z11	Fibre optic input value ***	-400..+400%	-32768..+32767
40116	I8a	Fibre optic input low setpoint	-400..+400%	-32768..+32767
40117	I8b	Fibre optic input high setpoint	-400..+400%	-32768..+32767
40120	Z3	Analogue input 1 value ***	-400..+400%	-32768..+32767
40121	Z4	Analogue input 2 value***	-400..+400%	-32768..+32767
40122	Z5	Analogue output 1 value ***	-400..+400%	-32768..+32767
40123	Z6	Analogue output 2 value ***	-400..+400%	-32768..+32767
40124	-	Analogue input 1+2 value ***	-400..+400%	-32768..+32767
40125	I6b	Analogue input 1 low setpoint	-400..+400%	-32768..+32767
40126	I6c	Analogue input 1 high setpoint	-400..+400%	-32768..+32767
40127	I6e	Analogue input 2 low setpoint	-400..+400%	-32768..+32767
40128	I6f	Analogue input 2 high setpoint	-400..+400%	-32768..+32767
40129	O1c	Analogue output 1 low setpoint	-400..+400%	-32768..+32767
40130	O1d	Analogue output 1 high setpoint	-400..+400%	-32768..+32767
40131	O1g	Analogue output 2 low setpoint	-400..+400%	-32768..+32767
40132	O1h	Analogue output 2 high setpoint	-400..+400%	-32768..+32767

40133	I6g	Zero band for analogue I/P sources	0..1	0..1
40134	I6a,Z3	Analogue input 1 format	0..3	0..3
40135	I6d,Z4	Analogue input 2 format	0..3	0..3
40136	O1b,Z5	Analogue output 1 format	0..3	0..3
40137	O1f,Z6	Analogue output 2 format	0..3	0..3
40138	I7b	Multi-function input logical inversion	0..1	0..1
40139	N9	Encoder type select	0..1	0..1
40140	Z2	Software version ***	0..25.5	0..255
40141	Z2	Hardware version ***	0..25.5	0..255
40150	Y1	Screen list select	0..255	0..255
40151	Y3	Current Vista configuration select	0..255	0..255
40152	Y3	Number of Vista configurations ***	0..255	0..255
40153	-	Vista block Error code ***	0..255	0..255
40161	A4	Reference speed	-400..+400%	-32768..+32767
40162	A4	Reference torque	-400..+400%	-32768..+32767
40170	Z7	Status of Multifunction input 1 (read only)	0..1	0..1
40171	Z7	Status of Multifunction input 2 ***	0..1	0..1
40172	Z7	Status of Multifunction input 3 ***	0..1	0..1
40173	Z7	Status of Multifunction input 4 ***	0..1	0..1
40174	Z7	Status of Multifunction input 5 ***	0..1	0..1
40175	Z7	Status of Multifunction input 6 ***	0..1	0..1
40176	Z7	Status of Multifunction I/P 7 / Ex.Trip ***	0..1	0..1
40180	-	Elite stop signal ***	0..1	0..1
40181	-	Elite start signal***	0..1	0..1
40182	-	Elite reset signal ***	0..1	0..1
40183	-	Elite run command	0..1	0..1
40190	L10	Skip Speed 1	-250..+250%	-20480..+20480
40191	L11	Skip Speed 2	-250..+250%	-20480..+20480
40192	L12	Skip Bandwidth	0..20%	0..1638
40200	F6	Clear Fault History	0..1	0..1
40201	F1	Fault History 1 ***	0..39	0..39
40202	F2	Fault History 2 ***	0..39	0..39
40203	F3	Fault History 3 ***	0..39	0..39
40204	F4	Fault History 4 ***	0..39	0..39
40205	F5	Fault History 5 ***	0..39	0..39
40210	X4a	Dynaflux minimum flux level	40..100%	3276..8192%
40211	X4b	Select torque boost mode	0..2	0..2
40212	X5a	Current limit slip value	0..10%	0..819
40213	X5b	Voltage limit slip value	0..10%	0..819
40214	X5c	No-load damping	0..20%	0..1638
40215	X5d	Slip compensation enable	0..1	0..1
40220	P6	Process control error signal ***	-400..+400%	-32768..+32767
40221	-	Process control enable ***	0..1	0..1
40222	-	Process control reference value ***	-400..+400%	-32768..+32767
40223	-	Process control feedback value ***	-400..+400%	-32768..+32767
40224	P1	Process control reference source select	0..7	0..7
40225	P2	Process control feedback source select	0..7	0..7
40226	P3	Process control gain factor	1..100	1..100
40227	P4	Process control integration time	10..10010	10..10010
40228	P5	Process control differential factor	0..2500	0..2500
40230	-	Inverter rated voltage	400,690Vac	400,690
40231	-	Inverter rated current	0..6553amps	0..65535
40613	-	Drive identification code ***	0..65535	-
40885	-	EEPROM Address***	40001..49999	40001..49999

			(excluding 40885)	(excluding 40885)
41001	I7c	Multi-function input 1 select	0..18	0..18
41002	I7d	Multi-function input 2 select	0..18	0..18
41003	I7e	Multi-function input 3 select	0..18	0..18
41004	I7f	Multi-function input 4 select	0..18	0..18
41005	I7g	Multi-function input 5 select	0..18	0..18
41006	I7h	Multi-function input 6 select	0..18	0..18
41007	I7a	Multi-function input mode select	0..5	0..5
41010	C2	Comparator 1 "ON" setpoint	-250..+250%	-20480..+20480
41011	C3	Comparator 1 "OFF" setpoint	-250..+250%	-20480..+20480
41012	C1	Comparator 1 source select	0..18	0..18
41013	-	Comparator 1 output ***	0..1	0..1
41014	I2	Speed reference source select	0..8	0..8
41015	I3	Torque reference source select	0..7	0..7
41016	I4	Alt speed reference source select	0..8	0..8
41017	I5	Alt torque reference source select	0..7	0..7
41019	-	Multi-reference select ***	0..255	0..255
41020	M1	Multi-reference 1 setpoint	-400..+400%	-32768..+32767
41021	M2	Multi-reference 2 setpoint	-400..+400%	-32768..+32767
41022	M3	Multi-reference 3 setpoint	-400..+400%	-32768..+32767
41023	M4	Multi-reference 4 setpoint	-400..+400%	-32768..+32767
41024	M5	Multi-reference 5 setpoint	-400..+400%	-32768..+32767
41025	M6	Multi-reference 6 setpoint	-400..+400%	-32768..+32767
41026	M7	Multi-reference 7 setpoint	-400..+400%	-32768..+32767
41027	O2a	Relay 1 source select	0..22	0..22
41028	O2c	Relay 2 source select	0..22	0..22
41029	O2e	Relay 3 source select	0..22	0..22
41030	O2b	Invert the logic of Relay 1	0..1	0..1
41031	O2d	Invert the logic of Relay 2	0..1	0..1
41032	O2f	Invert the logic of Relay 3	0..1	0..1
41033	O1a	Analogue output 1 source selection	0..18	0..18
41034	O1e	Analogue output 2 source selection	0..18	0..18
41039	O3a	Fibre optic output source select	0..18	0..18
41041	A2	Keyboard torque reference	-250..+250%	-20480..+20480
41042	A1	Keyboard speed/torque mode select	0..1	0..1
41043	R1	Acceleration rate	0.1..6000%/s	1..60000
41044	R2	Deceleration rate	0.1..6000%/s	1..60000
41045	R3	Alternative acceleration rate	0.1..6000%/s	1..60000
41046	R4	Alternative deceleration rate	0.1..6000%/s	1..60000
41047	R5	Break speed for Alt accel/decel	0..250%	0..20480
41048	S2	Usual stopping mode	0..5	0..5
41049	S4	Alternative stopping mode	0..5	0..5
41062	-	Motorised potentiometer speed	-400..+400%	-32768..+32767
41063	-	Motorised potentiometer torque ***	-400..+400%	-32768..+32767
41090	Status Line	Inverter and Motor overload warning ***	0..16	0..16
41091	X1	Control mode select	0..2	0..2
41110	C5	Comparator 2 "ON" setpoint	-250..+250%	-20480..+20480
41111	C6	Comparator 2 "OFF" setpoint	-250..+250%	-20480..+20480
41112	C4	Comparator 2 source select	0..18	0..18
41113	-	Comparator 2 output ***	0..1	0..1
41114	-	Window comparator output ***	0..1	0..1

Note *** indicates that this parameter is a read only parameter.

Table 3.2 Elite Modbus Register Details

Note 1: 40613 - Drive Identification Code

High Byte:	1 = Microdrive	Low Byte:	1 = ME-2.5, 400V	66 = ME-38, 400V
	2 = Microflo		33 = ME-6.5, 400V	2 = ME-46, 400V
	3 = Microvector		65 = ME-10.5, 400V	35 = UE-60, 400V
	4 = Elite Series		97 = ME-12, 400V	67 = UE-75, 400V
			96 = ME-16, 400V	129 = UE-90, 400V
			64 = ME-18, 400V	36 = UE-115, 400V
			128 = ME-22.5, 400V	68 = UE-140, 400V
			42 = ME-28, 400V	
			34 = ME-31, 400V	

Appendix D

JAVA Program

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;
import java.net.*;
import java.io.*;
import java.lang.*;
import java.lang.String.*;
import java.text.*;
import java.util.*;

import javax.swing.JPanel;
import javax.swing.BorderFactory;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import javax.swing.JTabbedPane;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JTextField;
import javax.swing.JLabel;
import javax.swing.JApplet;

import java.awt.Button;
import java.awt.event.*;

public class Test extends Applet implements Runnable{
    static private boolean isapplet = true;
    static private InetAddress arg_ip = null;
    static private int arg_port = 0;
    Thread timer;

    public tcpip gtp = null;;
    InetAddress reader_ip = null;
    int port = 14000;

    int ControlLoop1 = 0;
    //The below are the command strings for the pushbuttons
    int [] StartCMD_Main= {0x0A,0x10,0x00,0x51,0x00,0x01,0x02,0x00,0x01};
    int [] StartCMD_RST_Main= {0x0A,0x10,0x00,0x51,0x00,0x01,0x02,0x00,0x00};
    int [] StopCMD_Main= {0x0A,0x10,0x00,0x50,0x00,0x01,0x02,0x00,0x01};
    int [] StopCMD_RST_Main= {0x0A,0x10,0x00,0x50,0x00,0x01,0x02,0x00,0x00};
    int [] ResetCMD_Main= {0x0A,0x10,0x00,0x4F,0x00,0x01,0x02,0x00,0x01};
    int [] AutoCMD_Main= {0x0A,0x10,0x01,0x2C,0x00,0x01,0x02,0x00,0x01};
    int [] ManCMD_Main= {0x0A,0x10,0x01,0x2D,0x00,0x01,0x02,0x00,0x01};
    int [] UpCMD_Main= {0x0A,0x06,0x00,0x02,0x00,0x32};
    int [] DownCMD_Main= {0x0A,0x06,0x00,0x02,0x00,0x32};

    //The below are the command strings for collecting status information
    int [] MotorSpeed_Main= {0x0A,0x03,0x00,0x5F,0x00,0x01};
    int [] MotorAmps_Main= {0x0A,0x03,0x00,0x5C,0x00,0x01};

    //The below are the returned status information
    double Motor_Speed_Int_Status;
    double Motor_Amps_Int_Status;

    //Control Panel Variables.

    JButton btStart_PB;
    JButton btStop_PB;
    JButton btReset_PB;
    JButton btAuto_PB;
    JButton btManual_PB;
    JButton btUP_PB;
    JButton btDN_PB;
    JTextField tfSetpoint;
    JTextField tfMode;
    JTextField tfStatus;
    JLabel lbSetpoint_Label;
    JLabel lbStatus_Label;
    JLabel lbMode_Label;
    JTextField tfSpeed;
    JTextField tfCurrent;
    JLabel lbSpeed_LBL;
    JLabel lbCurrent_LBL;

```

```

JLabel lbPerc_LBL;
JLabel lbAmps_LBL;
//*****
JTextField tfMotor_KW;
JTextField tfMotor_FLC;
JLabel lbMotor_KW_LBL;
JLabel lbMotor_FLC_LBL;
JLabel lbMotor_V_LBL;
JTextField tfMotor_V;
JLabel lbLabel 7;
JLabel lbLabel 8;
JLabel lbLabel 9;
JLabel lbMotor_PF_LBL;
JLabel lbMotor_Eff_LBL;
JLabel lbMotor_SPD_LBL;
JTextField tfMotor_PF;
JTextField tfMotor_Eff;
JTextField tfMotor_SPD;
JLabel lbLabel 10;
JLabel lbLabel 11;
JLabel lbLabel 12;
JLabel lbLabel 13;
//*****
JTextField tfImpeller_Size;
JTextField tfInlet_Size;
JLabel lbImp_Size_LBL;
JLabel lbInlet_Size_LBL;
JLabel lbOutlet_Size_LBL;
JTextField tfOutlet_Size;
JLabel lbMaximum_FL_LBL;
JLabel lbMaximum_Press_LBL;
JLabel lbPump_Eff_LBL;
JTextField tfMaximum_FL;
JTextField tfMaximum_Press;
JTextField tfPump_Eff;
JLabel lbLabel 1;
JLabel lbLabel 2;
JLabel lbLabel 3;
JLabel lbLabel 4;
JLabel lbLabel 5;
JLabel lbLabel 6;
JLabel lbLabel 20;
JLabel lbLabel 21;
JLabel lbLabel 22;
JLabel lbLabel 23;
JButton btClose_PB;
JButton btUpload_PB;
JButton btDownload_PB;

```

```

public void init()
{
    gtp = null;
    reader_ip = null;
    port = 14000;
}

```

```

//*****
*****

```

```

public void run() {
    int i;
    byte[] in = new byte[100];
    //int Control Loop1=0;
    int j;
    int[] a = new int[100];
    int[] b = new int[100];

    int[] ReturnedValues = new int[10];
    int result=0;
    Thread me = Thread.currentThread();
    while (timer == me) {
        try {
            Thread.currentThread().sleep(1000);
        }
        catch (InterruptedException e) { }
        if ( (gtp != null) && ((i = gtp.available()) > 0) ) {

            in = gtp.receive();

```

```

        for (j = 0; j < in.length; j++) {
            a[j] = in[j];
            if (a[j] < 0){
                b[j]=256 + a[j];
            }
            else{
                b[j]=a[j];
            }
            //System.out.println("Integer value "+b[j]);
            //System.out.println("Integer value "+a[j]+" another
value "+b[j]);
        }
        result = (b[3]*256)+b[4];
        System.out.println("the result is "+result);
        ReturnedValues[1]=result;
    }
    System.out.println("The Motor Speed is "+ReturnedValues[1]);
    //System.out.println("The Motor Amps is "+ReturnedValues[2]);
    Motor_Speed_Int_Status = (Math.round((ReturnedValues[1] *
(40000))/32767));
    Motor_Amps_Int_Status = (Math.round((ReturnedValues[2] * (22.5 *
800))/65535));
    //System.out.println("The Motor Speed is
"+Motor_Speed_Int_Status/100);
    //System.out.println("The Motor Amps is
"+Motor_Amps_Int_Status/100);
    result=0;
}
}

public void start()
{
    timer = new Thread(this);
    timer.start();
    String st = new String(" ");
    setFont(new Font("Dialog", Font.BOLD, 16));
    setLayout(new GridLayout(6, 3));
    GridBagConstraints c = new GridBagConstraints();
    c.gridx = 0; c.gridy = 0; c.gridwidth = 1; c.gridheight = 10;
    c.anchor = GridBagConstraints.CENTER;
    c.fill = GridBagConstraints.BOTH;
    c.insets = new Insets(5, 5, 5, 5);
    setBackground(Color.yellow);
    setSize(200, 200);

    if (isApplet) {
        try{
            reader_ip =
InetAddress.getBy_name(getCodeBase().getHost());
        }
        catch (UnknownHostException){}
    }
    else {
        reader_ip = arg_ip;
        if (arg_port != 0) {
            port = arg_port;
        }
    }

    /* Open a socket to the Device Server's serial port */
    if (reader_ip != null) {
        if (gtp == null) {
            gtp = new tcpip(reader_ip, port);
            if (gtp.s == null) {
                st += "Connection FAILED! ";
                gtp = null;
            }
        }
    }

    if (gtp == null) {
        st += "Not Connected";
        add((new Label(st)), c);
        return;
    }
    st += "Connected";
    c.gridx = 10; c.gridy = 50; c.gridwidth = 3; c.gridheight = 1;
    c.weightx = 2; c.weighty = 10.0; c.anchor = GridBagConstraints.WEST;
    c.fill = GridBagConstraints.BOTH;
}
}

```

```

        add((new Label(st)), c);
//*****
**
        c.gridx = 500; c.gridy = 50; c.gridwidth = 3; c.gridheight = 1;
        c.weightx = 2; c.weighty = 10.0; c.anchor = GridBagConstraints.WEST;
        c.fill = GridBagConstraints.BOTH;
        add((new CP_Gen()), c);
//*****
**

    }
    public void destroy()
    {
        if (gtp != null)
            gtp.disconnect();
        gtp = null;
    }

    public void stop() {
    }

    public static void main(String[] args) {
        Frame frame = new Frame("TCP/IP Test");
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        if (args.length > 0) {
            try{
                arg_ip = InetAddress.getByName(args[0]);
            }
            catch (UnknownHostException e){}
            if (args.length > 1) {
                try {
                    arg_port = Integer.valueOf(args[1]).intValue();
                }
                catch (NumberFormatException e) {}
            }
        }
        Test ap = new Test();
        frame.add(ap);
        ap.init();
        isapplet = false;
        ap.start();
        frame.pack();
        frame.show();
    }

//*****
*****
/*
 * TextComponentTest
 * Tests out the java.awt.TextField and Java.awt.TextArea
 * Components
 */

public class TextComponentTest implements TextListener
{
    public TextComponentTest()
    {
        TextField tf= new TextField();
        tf.addTextListener(this);
        add(tf);
        setVisible(true);
    }
    public void textValueChanged(TextEvent event)
    {
        TextComponent src = (TextComponent)event.getSource();
        System.out.println(src.getText());
    }
}

//*****

```

```

****
/*
 * Routine for the interface panel.
 */
public class CP_Gen extends JFrame
{

class Control_Panel extends JPanel implements ActionListener
{

//*****
/**
 *Constructor for the Control_Panel object
 */
public Control_Panel ()
{
    super();
    setBorder( BorderFactory.createTitledBorder( "Motor Control" ) );

    GridBagLayout gbControl_Panel = new GridBagLayout();
    GridBagConstraints gbcControl_Panel = new GridBagConstraints();
    setLayout( gbControl_Panel );

    btStart_PB = new JButton( "Start" );
    btStart_PB.addActionListener( this );
    gbcControl_Panel.gridx = 1;
    gbcControl_Panel.gridy = 1;
    gbcControl_Panel.gridwidth = 1;
    gbcControl_Panel.gridheight = 1;
    gbcControl_Panel.fill = GridBagConstraints.BOTH;
    gbcControl_Panel.weightx = 1;
    gbcControl_Panel.weighty = 0;
    gbcControl_Panel.anchor = GridBagConstraints.NORTH;
    gbControl_Panel.setConstraints( btStart_PB, gbcControl_Panel );
    add( btStart_PB );

    btStop_PB = new JButton( "Stop" );
    btStop_PB.addActionListener( this );
    gbcControl_Panel.gridx = 1;
    gbcControl_Panel.gridy = 2;
    gbcControl_Panel.gridwidth = 1;
    gbcControl_Panel.gridheight = 1;
    gbcControl_Panel.fill = GridBagConstraints.BOTH;
    gbcControl_Panel.weightx = 1;
    gbcControl_Panel.weighty = 0;
    gbcControl_Panel.anchor = GridBagConstraints.NORTH;
    gbControl_Panel.setConstraints( btStop_PB, gbcControl_Panel );
    add( btStop_PB );

    btReset_PB = new JButton( "Reset" );
    btReset_PB.addActionListener( this );
    gbcControl_Panel.gridx = 1;
    gbcControl_Panel.gridy = 3;
    gbcControl_Panel.gridwidth = 1;
    gbcControl_Panel.gridheight = 1;
    gbcControl_Panel.fill = GridBagConstraints.BOTH;
    gbcControl_Panel.weightx = 1;
    gbcControl_Panel.weighty = 0;
    gbcControl_Panel.anchor = GridBagConstraints.NORTH;
    gbControl_Panel.setConstraints( btReset_PB, gbcControl_Panel );
    add( btReset_PB );

    btAuto_PB = new JButton( "Auto" );
    btAuto_PB.addActionListener( this );
    gbcControl_Panel.gridx = 3;
    gbcControl_Panel.gridy = 1;
    gbcControl_Panel.gridwidth = 2;
    gbcControl_Panel.gridheight = 1;
    gbcControl_Panel.fill = GridBagConstraints.BOTH;
    gbcControl_Panel.weightx = 1;
    gbcControl_Panel.weighty = 0;
    gbcControl_Panel.anchor = GridBagConstraints.NORTH;
    gbControl_Panel.setConstraints( btAuto_PB, gbcControl_Panel );
    add( btAuto_PB );

    btManual_PB = new JButton( "Manual" );
    btManual_PB.addActionListener( this );
    gbcControl_Panel.gridx = 3;

```

```

gbcControl_Panel.gridy = 2;
gbcControl_Panel.gridwidth = 2;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( btManual_PB, gbcControl_Panel );
add( btManual_PB );

btUP_PB = new JButton( "Up" );
btUP_PB.addActionListener( this );
gbcControl_Panel.gridx = 3;
gbcControl_Panel.gridy = 4;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( btUP_PB, gbcControl_Panel );
add( btUP_PB );

btDN_PB = new JButton( "Down" );
btDN_PB.addActionListener( this );
gbcControl_Panel.gridx = 4;
gbcControl_Panel.gridy = 4;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( btDN_PB, gbcControl_Panel );
add( btDN_PB );

tfSetpoint = new JTextField();
gbcControl_Panel.gridx = 3;
gbcControl_Panel.gridy = 3;
gbcControl_Panel.gridwidth = 2;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( tfSetpoint, gbcControl_Panel );
add( tfSetpoint );

tfMode = new JTextField();
gbcControl_Panel.gridx = 3;
gbcControl_Panel.gridy = 0;
gbcControl_Panel.gridwidth = 2;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( tfMode, gbcControl_Panel );
add( tfMode );
    tfMode.setText("Automatic");

tfStatus = new JTextField();
gbcControl_Panel.gridx = 1;
gbcControl_Panel.gridy = 0;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( tfStatus, gbcControl_Panel );
add( tfStatus );
    //tfStatus = gtp.receive();

lbSetpoint_Label = new JLabel( " Setpoint " );
gbcControl_Panel.gridx = 2;
gbcControl_Panel.gridy = 3;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;

```

```

gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lbSetpoint_Label, gbcControl_Panel );
add( lbSetpoint_Label );

lbStatus_Label = new JLabel( "Status " );
gbcControl_Panel.gridx = 0;
gbcControl_Panel.gridy = 0;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lbStatus_Label, gbcControl_Panel );
add( lbStatus_Label );

lbMode_Label = new JLabel( " Mode" );
gbcControl_Panel.gridx = 2;
gbcControl_Panel.gridy = 0;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lbMode_Label, gbcControl_Panel );
add( lbMode_Label );

tfSpeed = new JTextField();
gbcControl_Panel.gridx = 1;
gbcControl_Panel.gridy = 5;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( tfSpeed, gbcControl_Panel );
add( tfSpeed );
//tfSpeed.setText(Double.toString(Motor_Speed_Int_Status));

tfCurrent = new JTextField();
gbcControl_Panel.gridx = 1;
gbcControl_Panel.gridy = 6;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( tfCurrent, gbcControl_Panel );
add( tfCurrent );

lbSpeed_LBL = new JLabel( "Speed" );
gbcControl_Panel.gridx = 0;
gbcControl_Panel.gridy = 5;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.EAST;
gbcControl_Panel.setConstraints( lbSpeed_LBL, gbcControl_Panel );
add( lbSpeed_LBL );

lbCurrent_LBL = new JLabel( "Current" );
gbcControl_Panel.gridx = 0;
gbcControl_Panel.gridy = 6;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;

```



```
gbControl_Panel.setConstraints( IbCurrent_LBL, gbcControl_Panel );
add( IbCurrent_LBL );
```

```
IbPerc_LBL = new JLabel( "%" );
gbcControl_Panel.gridx = 2;
gbcControl_Panel.gridy = 5;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( IbPerc_LBL, gbcControl_Panel );
add( IbPerc_LBL );
```

```
IbAmps_LBL = new JLabel( "A" );
gbcControl_Panel.gridx = 2;
gbcControl_Panel.gridy = 6;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.WEST;
gbControl_Panel.setConstraints( IbAmps_LBL, gbcControl_Panel );
add( IbAmps_LBL );
```

```
//*****
```

```
tfMotor_KW = new JTextField();
gbcControl_Panel.gridx = 9;
gbcControl_Panel.gridy = 1;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( tfMotor_KW, gbcControl_Panel );
add( tfMotor_KW );
```

```
tfMotor_FLC = new JTextField();
gbcControl_Panel.gridx = 9;
gbcControl_Panel.gridy = 2;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( tfMotor_FLC, gbcControl_Panel );
add( tfMotor_FLC );
```

```
IbMotor_KW_LBL = new JLabel( " Motor Size" );
gbcControl_Panel.gridx = 7;
gbcControl_Panel.gridy = 1;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( IbMotor_KW_LBL, gbcControl_Panel );
add( IbMotor_KW_LBL );
```

```
IbMotor_FLC_LBL = new JLabel( " Motor Current" );
gbcControl_Panel.gridx = 7;
gbcControl_Panel.gridy = 2;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( IbMotor_FLC_LBL, gbcControl_Panel );
add( IbMotor_FLC_LBL );
```

```
IbMotor_V_LBL = new JLabel( " Motor Voltage" );
gbcControl_Panel.gridx = 7;
```

```

gbcControl_Panel.gridy = 3;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lbMotor_V_LBL, gbcControl_Panel );
add( lbMotor_V_LBL );

```

```

tfMotor_V = new JTextField();
gbcControl_Panel.gridx = 9;
gbcControl_Panel.gridy = 3;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( tfMotor_V, gbcControl_Panel );
add( tfMotor_V );

```

```

lbLabel7 = new JLabel( " KW" );
gbcControl_Panel.gridx = 10;
gbcControl_Panel.gridy = 1;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lbLabel7, gbcControl_Panel );
add( lbLabel7 );

```

```

lbLabel8 = new JLabel( " A" );
gbcControl_Panel.gridx = 10;
gbcControl_Panel.gridy = 2;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lbLabel8, gbcControl_Panel );
add( lbLabel8 );

```

```

lbLabel9 = new JLabel( " V" );
gbcControl_Panel.gridx = 10;
gbcControl_Panel.gridy = 3;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lbLabel9, gbcControl_Panel );
add( lbLabel9 );

```

```

lbMotor_PF_LBL = new JLabel( " Motor Power Factor" );
gbcControl_Panel.gridx = 7;
gbcControl_Panel.gridy = 4;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lbMotor_PF_LBL, gbcControl_Panel );
add( lbMotor_PF_LBL );

```

```

lbMotor_Eff_LBL = new JLabel( " Motor Efficiency" );
gbcControl_Panel.gridx = 7;
gbcControl_Panel.gridy = 5;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;

```

```

gbControl_Panel.setConstraints( lbMotor_Eff_LBL, gbcControl_Panel );
add( lbMotor_Eff_LBL );

lbMotor_SPD_LBL = new JLabel( " Motor Speed" );
gbcControl_Panel.gridx = 7;
gbcControl_Panel.gridy = 6;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( lbMotor_SPD_LBL, gbcControl_Panel );
add( lbMotor_SPD_LBL );

tfMotor_PF = new JTextField();
gbcControl_Panel.gridx = 9;
gbcControl_Panel.gridy = 4;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( tfMotor_PF, gbcControl_Panel );
add( tfMotor_PF );

tfMotor_Eff = new JTextField();
gbcControl_Panel.gridx = 9;
gbcControl_Panel.gridy = 5;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( tfMotor_Eff, gbcControl_Panel );
add( tfMotor_Eff );

tfMotor_SPD = new JTextField();
gbcControl_Panel.gridx = 9;
gbcControl_Panel.gridy = 6;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( tfMotor_SPD, gbcControl_Panel );
add( tfMotor_SPD );

lbLabel10 = new JLabel( "COS" );
gbcControl_Panel.gridx = 10;
gbcControl_Panel.gridy = 4;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( lbLabel10, gbcControl_Panel );
add( lbLabel10 );

lbLabel11 = new JLabel( " %" );
gbcControl_Panel.gridx = 10;
gbcControl_Panel.gridy = 5;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( lbLabel11, gbcControl_Panel );
add( lbLabel11 );

lbLabel12 = new JLabel( " RPM" );
gbcControl_Panel.gridx = 10;
gbcControl_Panel.gridy = 6;
gbcControl_Panel.gridwidth = 1;

```

```

gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lblLabel12, gbcControl_Panel );
add( lblLabel12 );

lblLabel13 = new JLabel( " Motor Parameters" );
gbcControl_Panel.gridx = 7;
gbcControl_Panel.gridy = 0;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lblLabel13, gbcControl_Panel );
add( lblLabel13 );
//*****

lblLabel20 = new JLabel( " Pump Parameters" );
gbcControl_Panel.gridx = 7;
gbcControl_Panel.gridy = 8;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lblLabel20, gbcControl_Panel );
add( lblLabel20 );

tfImpeller_Size = new JTextField();
gbcControl_Panel.gridx = 9;
gbcControl_Panel.gridy = 9;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( tfImpeller_Size, gbcControl_Panel );
add( tfImpeller_Size );

tfInlet_Size = new JTextField();
gbcControl_Panel.gridx = 9;
gbcControl_Panel.gridy = 10;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( tfInlet_Size, gbcControl_Panel );
add( tfInlet_Size );

lblImp_Size_LBL = new JLabel( " Impeller Size" );
gbcControl_Panel.gridx = 7;
gbcControl_Panel.gridy = 9;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lblImp_Size_LBL, gbcControl_Panel );
add( lblImp_Size_LBL );

lblInlet_Size_LBL = new JLabel( " Inlet Pipe Size" );
gbcControl_Panel.gridx = 7;
gbcControl_Panel.gridy = 10;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lblInlet_Size_LBL, gbcControl_Panel );

```

```

add( lbInlet_Size_LBL );

lbOutlet_Size_LBL = new JLabel( " Outlet Pipe Size" );
gbcControl_Panel.grid dx = 7;
gbcControl_Panel.grid y = 11;
gbcControl_Panel.grid width = 1;
gbcControl_Panel.grid height = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weight x = 1;
gbcControl_Panel.weight y = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lbOutlet_Size_LBL, gbcControl_Panel );
add( lbOutlet_Size_LBL );

tfOutlet_Size = new JTextField();
gbcControl_Panel.grid dx = 9;
gbcControl_Panel.grid y = 11;
gbcControl_Panel.grid width = 1;
gbcControl_Panel.grid height = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weight x = 1;
gbcControl_Panel.weight y = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( tfOutlet_Size, gbcControl_Panel );
add( tfOutlet_Size );

lbMaximum_FL_LBL = new JLabel( " Maximum Flow" );
gbcControl_Panel.grid dx = 7;
gbcControl_Panel.grid y = 12;
gbcControl_Panel.grid width = 1;
gbcControl_Panel.grid height = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weight x = 1;
gbcControl_Panel.weight y = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lbMaximum_FL_LBL, gbcControl_Panel );
add( lbMaximum_FL_LBL );

lbMaximum_Press_LBL = new JLabel( " Maximum Pressure" );
gbcControl_Panel.grid dx = 7;
gbcControl_Panel.grid y = 13;
gbcControl_Panel.grid width = 1;
gbcControl_Panel.grid height = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weight x = 1;
gbcControl_Panel.weight y = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lbMaximum_Press_LBL, gbcControl_Panel );
add( lbMaximum_Press_LBL );

lbPump_Eff_LBL = new JLabel( " Pump Efficiency" );
gbcControl_Panel.grid dx = 7;
gbcControl_Panel.grid y = 14;
gbcControl_Panel.grid width = 1;
gbcControl_Panel.grid height = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weight x = 1;
gbcControl_Panel.weight y = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lbPump_Eff_LBL, gbcControl_Panel );
add( lbPump_Eff_LBL );

tfMaximum_FL = new JTextField();
gbcControl_Panel.grid dx = 9;
gbcControl_Panel.grid y = 12;
gbcControl_Panel.grid width = 1;
gbcControl_Panel.grid height = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weight x = 1;
gbcControl_Panel.weight y = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( tfMaximum_FL, gbcControl_Panel );
add( tfMaximum_FL );

tfMaximum_Press = new JTextField();
gbcControl_Panel.grid dx = 9;
gbcControl_Panel.grid y = 13;
gbcControl_Panel.grid width = 1;
gbcControl_Panel.grid height = 1;

```

```

gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( tfMaximum_Press, gbcControl_Panel );
add( tfMaximum_Press );

tfPump_Eff = new JTextField();
gbcControl_Panel.gridx = 9;
gbcControl_Panel.gridy = 14;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( tfPump_Eff, gbcControl_Panel );
add( tfPump_Eff );

lblLabel1 = new JLabel( " mm" );
gbcControl_Panel.gridx = 10;
gbcControl_Panel.gridy = 9;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lblLabel1, gbcControl_Panel );
add( lblLabel1 );

lblLabel2 = new JLabel( " mm" );
gbcControl_Panel.gridx = 10;
gbcControl_Panel.gridy = 10;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lblLabel2, gbcControl_Panel );
add( lblLabel2 );

lblLabel3 = new JLabel( " mm" );
gbcControl_Panel.gridx = 10;
gbcControl_Panel.gridy = 11;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lblLabel3, gbcControl_Panel );
add( lblLabel3 );

lblLabel4 = new JLabel( " L per Sec " );
gbcControl_Panel.gridx = 10;
gbcControl_Panel.gridy = 12;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lblLabel4, gbcControl_Panel );
add( lblLabel4 );

lblLabel5 = new JLabel( " Kpa" );
gbcControl_Panel.gridx = 10;
gbcControl_Panel.gridy = 13;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( lblLabel5, gbcControl_Panel );
add( lblLabel5 );

```

```

l bLabel 6 = new JLabel ( " %" );
gbcControl_Panel.grid x = 10;
gbcControl_Panel.grid y = 14;
gbcControl_Panel.grid width = 1;
gbcControl_Panel.grid height = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weight x = 1;
gbcControl_Panel.weight y = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( l bLabel 6, gbcControl_Panel );
add( l bLabel 6 );

```

```

//*****

```

```

l bLabel 21 = new JLabel ( " " );
gbcControl_Panel.grid x = 9;
gbcControl_Panel.grid y = 15;
gbcControl_Panel.grid width = 1;
gbcControl_Panel.grid height = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weight x = 1;
gbcControl_Panel.weight y = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( l bLabel 21, gbcControl_Panel );
add( l bLabel 21 );

```

```

l bLabel 22 = new JLabel ( " " );
gbcControl_Panel.grid x = 6;
gbcControl_Panel.grid y = 0;
gbcControl_Panel.grid width = 1;
gbcControl_Panel.grid height = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weight x = 1;
gbcControl_Panel.weight y = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( l bLabel 22, gbcControl_Panel );
add( l bLabel 22 );

```

```

l bLabel 23 = new JLabel ( " " );
gbcControl_Panel.grid x = 7;
gbcControl_Panel.grid y = 7;
gbcControl_Panel.grid width = 1;
gbcControl_Panel.grid height = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weight x = 1;
gbcControl_Panel.weight y = 1;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( l bLabel 23, gbcControl_Panel );
add( l bLabel 23 );

```

```

btClose_PB = new JButton( "Close" );
btClose_PB.addActionListener( this );
gbcControl_Panel.grid x = 9;
gbcControl_Panel.grid y = 16;
gbcControl_Panel.grid width = 1;
gbcControl_Panel.grid height = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weight x = 1;
gbcControl_Panel.weight y = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( btClose_PB, gbcControl_Panel );
add( btClose_PB );

```

```

btUpload_PB = new JButton( "Upload" );
btUpload_PB.addActionListener( this );
gbcControl_Panel.grid x = 7;
gbcControl_Panel.grid y = 16;
gbcControl_Panel.grid width = 1;
gbcControl_Panel.grid height = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weight x = 1;
gbcControl_Panel.weight y = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbControl_Panel.setConstraints( btUpload_PB, gbcControl_Panel );
add( btUpload_PB );

```

```

btDownload_PB = new JButton( "Download" );
btDownload_PB.addActionListener( this );
gbcControl_Panel.gridx = 8;
gbcControl_Panel.gridy = 16;
gbcControl_Panel.gridwidth = 1;
gbcControl_Panel.gridheight = 1;
gbcControl_Panel.fill = GridBagConstraints.BOTH;
gbcControl_Panel.weightx = 1;
gbcControl_Panel.weighty = 0;
gbcControl_Panel.anchor = GridBagConstraints.NORTH;
gbcControl_Panel.setConstraints( btDownload_PB, gbcControl_Panel );
add( btDownload_PB );

```

```

//*****
}
//*****

```

```

/**
 */
public void actionPerformed( ActionEvent e )
{
    if ( e.getSource() == btStart_PB )
    {
        // Action for btStart_PB
        CRC16(StartCMD_Main);
        CRC16(StartCMD_RST_Main);
    }
    if ( e.getSource() == btStop_PB )
    {
        // Action for btStop_PB
        CRC16(StopCMD_Main);
        CRC16(StopCMD_RST_Main);
    }
    if ( e.getSource() == btReset_PB )
    {
        // Action for btReset_PB
        CRC16(ResetCMD_Main);
    }
    if ( e.getSource() == btAuto_PB )
    {
        // Action for btAuto_PB
        tfMode.setText("Automatic");
        CRC16(AutoCMD_Main);
    }
    if ( e.getSource() == btManual_PB )
    {
        // Action for btManual_PB
        tfMode.setText("Manual");
        CRC16(ManCMD_Main);
    }
    if ( e.getSource() == btUP_PB )
    {
        // Action for btUP_PB
        //CRC16(UpCMD_Main);
        CRC16(MotorAmps_Main);
        tfCurrent.setText(Double.toString(Motor_Amps_Int_Status/100));
    }
    if ( e.getSource() == btDN_PB )
    {
        // Action for btDN_PB
        //CRC16(DownCMD_Main);
        CRC16(MotorSpeed_Main);
        tfSpeed.setText(Double.toString(Motor_Speed_Int_Status/100));
    }
    if ( e.getSource() == btClose_PB )
    {
        // Action for btClose_PB
        //CRC16(DownCMD_Main);
        //Uploader ComIn = new Uploader();
        //byte [] feedback = ComIn;
        //System.out.println("This is the feedback from the Uploader "+ComIn);
    }
}
}

```



```

}

Control_Panel pnControl_Panel;

public CP_Gen()
{
    super( "Main Control Panel" );
    pnControl_Panel = new Control_Panel ();
    setDefaultCloseOperation( EXIT_ON_CLOSE );

    setContentPane( pnControl_Panel );
    pack();
    show();
}
}
//*****

/*
 * CRC16 Routine
 * This Routine Calculates the CRC frame and adds it to the command string,
 * It then send the command out the serial port on the web server.
 */
public void CRC16 (int [] Buffer)
{
    int Polynomial = 0xA001;
    int Length = Buffer.length;
    int CRC16 = 0xFFFF;
    int i, j;
    int Data = 0x0000;
    int Constant_K = 0x0001;

    for (i=0; i <= (Length-1); i++)
    {
        Data = Buffer[i];
        for (j=1; j <= 8; j++)
        {
            if (((Data ^ CRC16) & Constant_K) == 1)
            {
                CRC16 = ((CRC16 >> 1) ^ Polynomial);
            }
            else
            {
                CRC16 = (CRC16 >> 1);
            }
            Data = Data >> 1;
        }
    }

    char [] CDD = new char [20];
    CDD [(Length-1)+1] = (char)(0x00FF & CRC16);
    CDD [(Length-1)+2] = (char)(CRC16 >> 8);
    for (i=0; i <= (Length-1); i++)
    {
        CDD [i] = (char)(Buffer[i]);
    }
    String str = new String(CDD);
    gtp.send(str);
}
}

```

Appendix E

PDL Communications Software

E.1 Drivelink Interface

PDL ELECTRONICS LTD

Leaders in AC Motor Control



**INTRODUCTION TO
PDL
CONFIGURATION UTILITIES**

CONTENTS

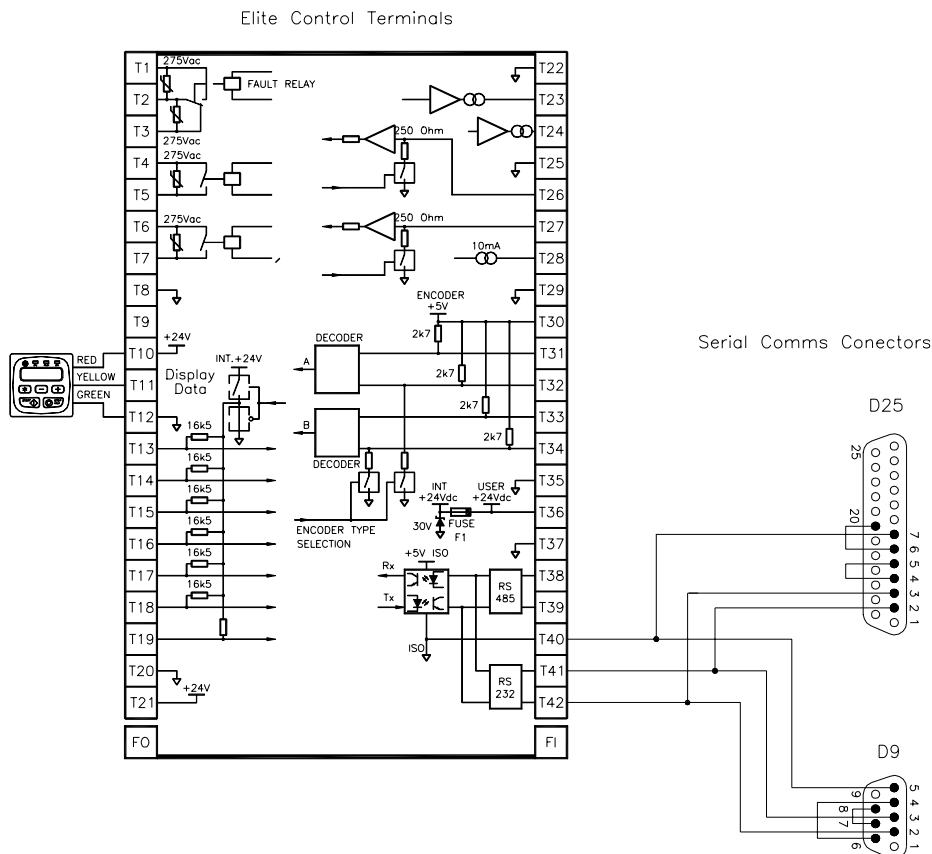
PDL DriveLink	5
Communications Setup	6
Loading a Vista program	8
Loading Elite Series System Code	9
PDL Modbus Logger	11
Setup	11
Operation	12
Reading and Writing System Variable Data	13
Reading and Writing Vista Variable Data	13
Logging Data to a File	14

PDL DriveLink

The PDL Electronics DriveLink software is used for loading into the Elite Series, a successfully compiled Vista program that is in the form of **filename.vlo**. It is also used for upgrading the Elite Series System Code that is in the form **0410aaXX.hex**, where **XX** refers to the software release version. To support a Vista program, the Elite Series software release version must be version 3.0 or greater.

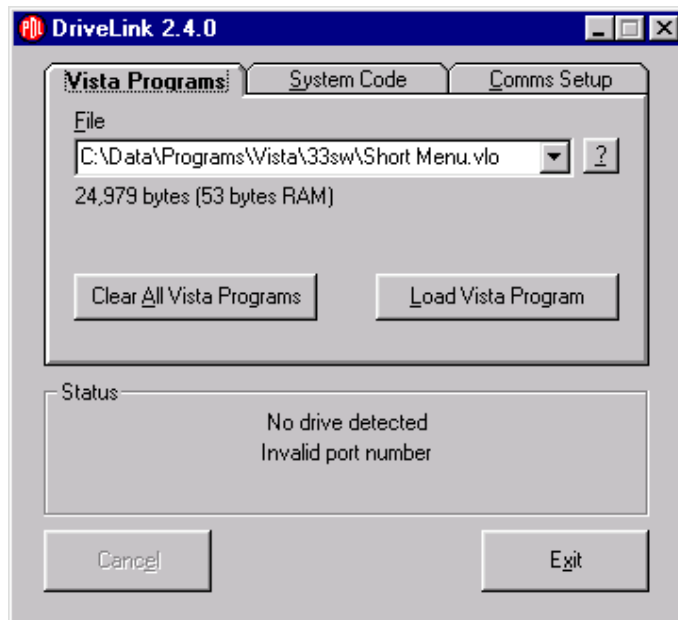
The Elite Series is connected to the PC via an RS232 serial communications link. Below is a circuit diagram showing connections between 9 and 25 pin "D" type plugs and the Elite Series communications terminals.

RS485 can be used for loading the Vista program (**filename.vlo**), but it is not recommended for loading the Elite Series System Code (**0410aaXX.hex**).

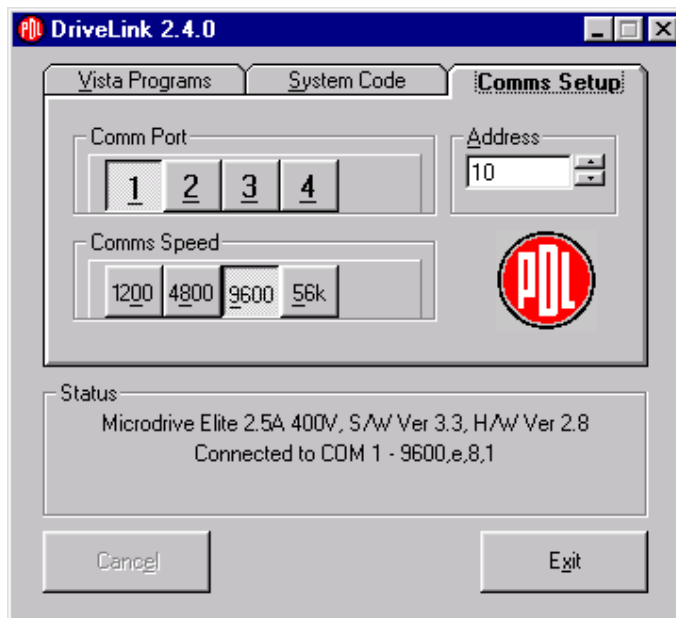


Communications Setup

Once the Elite Series has been connected to the PC and powered up, run DriveLink (DRVLINK.EXE) on the PC. If the status indicates "No drive detected, Invalid port number" as shown below, click the **Comms Setup** tab.



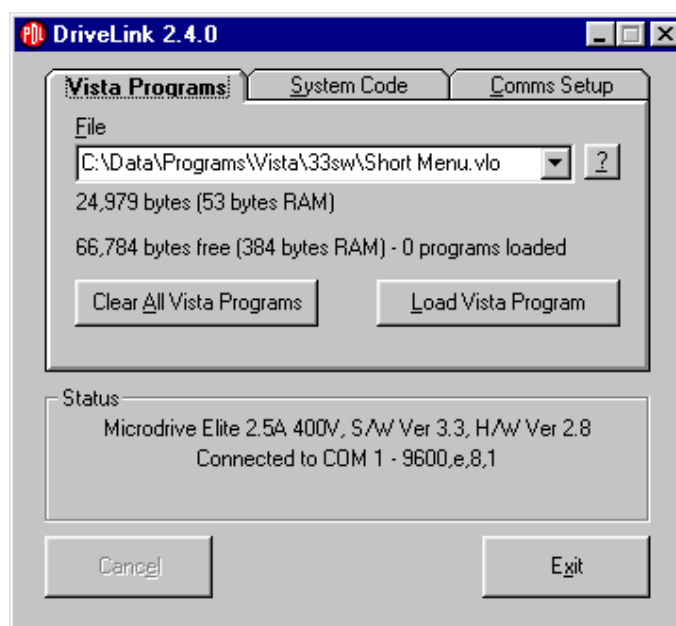
The **Comms Setup** tab allows the user to set the serial communication port used on the PC and the Comms Speed (baud rate). Note the same baud rate will have to be set in screen H3b of the Elite Series. The default baud rate is 9600.



The **Comms Setup** tab also allows the user to address individual drives if a multi-drop communications interface is being used (i.e. RS485). This address must be the same as the address in screen H3a of the Elite Series that is being communicated with. The default address in the Elite Series is 10. If the Elite Series has Screen H3c (Parity) ensure that this is set to Even.

An RS232 serial communications link is recommended if DriveLink is to be used to load the Elite Series System Code.

With the correct **Comm Port** and **Comms Speed** selected, the status should report the Elite Series that is connected, the software version, hardware version and the communications configuration. Clicking back on the **Vista Programs** tab will show the amount of free program memory and the number of Vista programs already loaded.




If communications has not been established, check the cable configuration and also ensure the Elite Series is powered up.

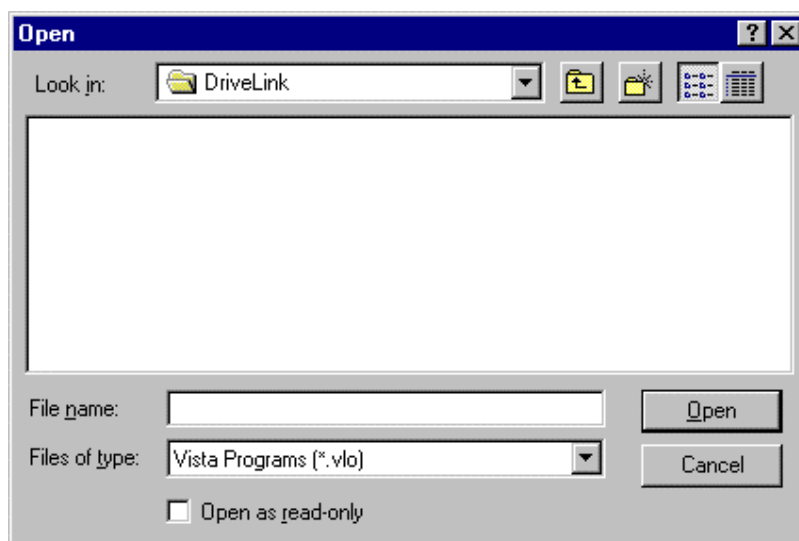
Once the Comms Setup procedure is complete and communications has been established, the Elite Series is ready to accept a Vista program or System Code.

Loading a Vista Program

The compiled Vista program is identified by the file extension **.vlo** (e.g. **filename.vlo** - visual language object file). After the Vista program is compiled the resultant .vlo file is placed in the same directory as the Vista program netlist file (**filename.nl**).

To load a Vista program into an Elite Series, click on the **Vista Programs** tab. The most recently loaded file name will be displayed in the text field under the tab. To load a different Vista program click the  button.

This will cause the **Open** dialog box to appear.



Select the Vista .vlo file to be loaded and click **open**. You will be returned to the **Vista Programs** tab where the program file name and path will be displayed together with the program size and RAM requirement. To load the Vista program, click the **Load Vista Program** button.

Once the program has finished loading, select the Vista program in screen Y3 of the Elite Series, typically program 2 of 2. The Elite Series will now be operating on the Vista program.

If there is insufficient free memory in the Elite Series, the Vista programs currently in the Elite Series will have to be cleared. To clear all the Vista programs in the Elite Series, click the **Clear All Vista Programs** button. Once this action is complete, new Vista programs can be loaded.

DriveLink clears all Vista programs by reloading System Code. If the DriveLink folder does not contain a copy of the System Code that is in the Elite Series, it

will extract it from the Elite Series. DriveLink will then reload this copy of the System Code to clear the Vista programs.

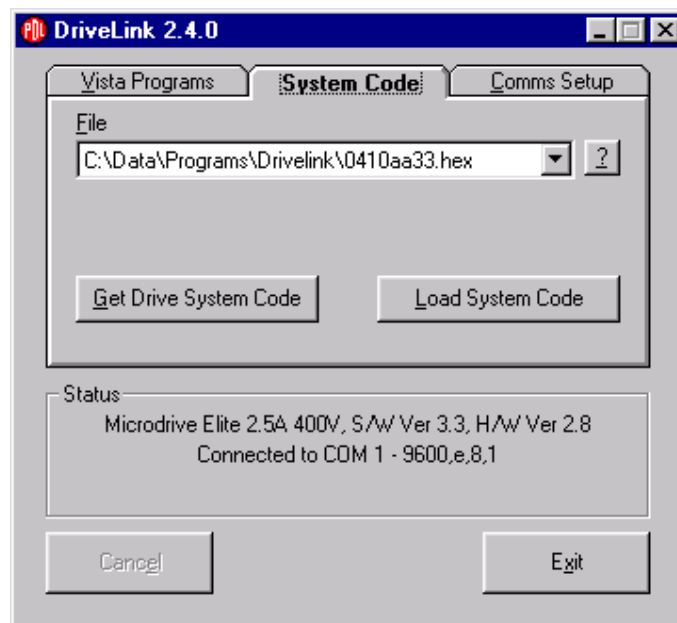
DriveLink does not extract Vista programs from the Elite Series and therefore it is essential that a copy of the Vista program is held external to the Elite Series.

Loading Elite Series System Code

DriveLink can also be used to load Elite Series System Code. This provides the ability to up-grade software as newer versions of Elite Series System Code are released. When up-grading Elite Series software, care has to be taken to ensure that the Elite Series is fitted with compatible hardware as reported in the Software release notes (4226 document series). The Elite Series must also have at least version 1.9 software already installed. The software version is reported in screen Z2.

If DriveLink is to be used to load the System Code an RS232 serial communications link is recommended.

To load Elite Series System Code, click on the **System Code** tab.



The most recently loaded System Code will be displayed in the text field under the tab.

To load a different System Code click the  button.

This will cause the **Open** dialogue box to appear.

Select the System Code file to be loaded and click **open**. You will be returned to the **System Code** tab where the System Code file name and path will be displayed.

To load the System Code, click the **Load System Code** button. The Elite Series System Code will now be loaded.

If the Elite Series contains a version of software that is not backed up in the DriveLink folder, the **Get Drive System Code** button will extract the System Code from the Elite Series.

Note: The communications must not be interrupted in any fashion during the System Code transfer. Power must not be disconnected from the PC or Elite Series during System Code transfer. This will result in an incomplete System Code and the Elite Series will not operate.

It is recommended an RS232 serial communications link be used for System Code transfer.

PDL Modbus Logger

The PDL Electronics Modbus Logger is a serial communications program that allows the user to monitor data stored in Vista variables that have been assigned a Modbus address, or Elite Series System Variables. The Modbus Logger message format is only compatible with Modbus function code 16. This limits data addresses to values within the 4XXXX range (decimal). Refer to the Elite Series Serial Communications Manual (4201-206) for a list of the System Variable addresses.

Modbus Logger can continually monitor the data at the selected addresses by using the **Poll** function. The **Poll Time** option sets the sampling rate (in seconds) of the data. If the **Log** function is used in conjunction with the **Poll** function the data will be logged to a file to allow further analysis. This file can be opened in an Excel spreadsheet and a trend graph can be easily created.

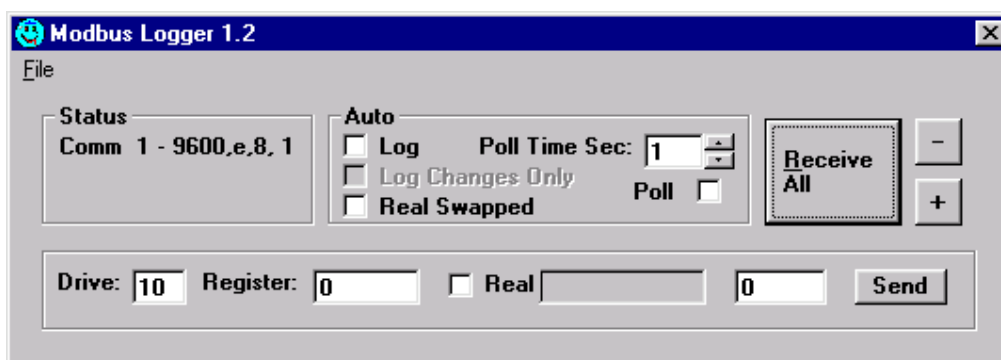
Modbus Logger can also be used to write data to a valid Modbus address. Care must be taken as the variable may be modified from another source (e.g. variable is controlled by a Vista program).

The two main functions of Modbus Logger is the debugging of Vista programs and the monitoring of system performance over time. To monitor Vista variables, valid Modbus addresses must be assigned to the variables during the Vista program creation.

When operating Modbus Logger ensure that no other serial communications application (e.g. DriveLink) is running at the same time within the computer, as this will cause incorrect program operation.

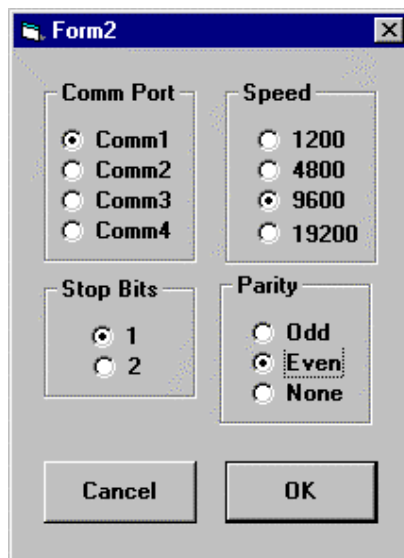
Setup

When Modbus Logger is run, the following dialog box appears.



Ensure the **Drive** selection box at the bottom left corresponds to the Elite Series address, shown in screen H3a of the Elite Series. The default address is 10.

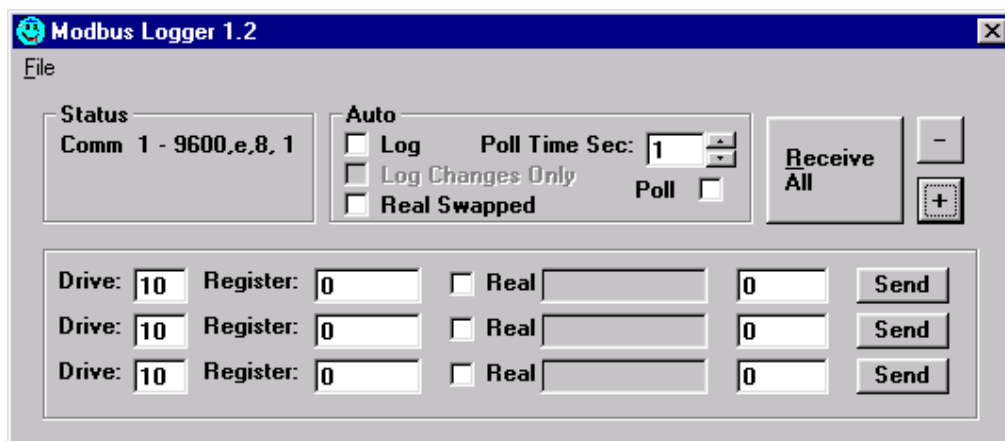
Using the **File** pull down menu select the **Comms Setup** option. The following dialog box will appear.



Configure the settings for the correct communications port on the PC and the correct Speed (Baud rate). The Stop Bits and Parity options should be set as above and not need adjustment. (Note Elite Series default baud rate is 9600 and default Parity is Even).

Operation

More than one data address can be read from or sent to by clicking the + button within Modbus Logger. Further additions to the dialogue box appear as shown below. Redundant address fields can be removed by clicking the - button.



More than one Elite Series may be monitored if RS485 is used and the appropriate drive addresses entered.

Reading and Writing System Variable Data

All System Variables are of a discrete data type, as opposed to a real number. To view a System Variable, enter the address and ensure the **Real** tick box is not checked. Refer to the Elite Series Serial Communications Manual (Document 4201-206) for a list of the System Variable addresses.

If **Poll** is activated the data contained at that address will be displayed in the centre box and automatically updated. The update rate is determined by the **Poll Time** setting (seconds). If **Poll** is not activated, clicking the **Receive All** button will initiate a one-shot read of the data.

Flags are displayed as either 00 or 01 and all other System Variables are displayed in hexadecimal.

Reading and Writing Vista Variable Data

Vista variables are defined as either Real or Flag data type. To accommodate the 32 bit real data type, valid Vista Modbus addresses must be a minimum of 2 significant bits apart. E.g. 42000, 42002, 42004 etc.

To view a Vista real variable, enter the Modbus address and click the **Real** tick box. Flags are viewed by entering the correct address and not checking the **Real** tick box.

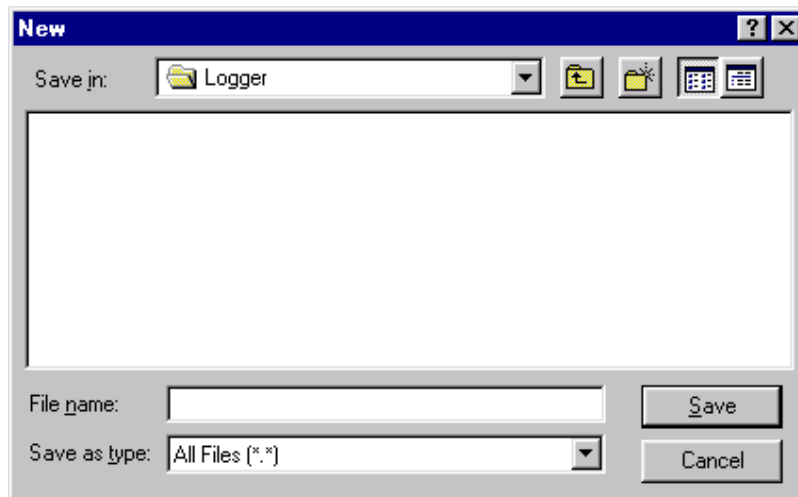
If the Elite Series System Code is version 3.0, then the **Real Swapped** tick box must be checked.

Due to a 16 bit limitation of the **Send** input box of Modbus Logger it is not possible to write new values into real Vista variables, however it is possible to write to a Vista flag variable.

Logging data to a file

If the **Poll** function is used in conjunction with the **Log** function the data will be logged in a Delimited file.

Before selecting the **Log** function, a (new) file must be created to store the data. To create a new file, click the **File** pull down menu and select **New**. The New dialogue box will appear.



In the **File name** field enter a name for the file. An extension is not necessary. Then click the **Save** button.

Click **Poll** and click **Log** to begin logging the data to the file. Alternatively **Log Changes Only** can be checked also, and this limits the size of the data file. Once monitoring is complete de-select the **Log** function. The data file can now be opened in an Excel spreadsheet for further analysis or graphing.

If logging is initiated before a file has been opened, Modbus Logger will prompt for a new file to be opened. If logging is subsequently stopped and started again, the last file used will be added to, unless a new file is created.

E.2 Drivecom Software



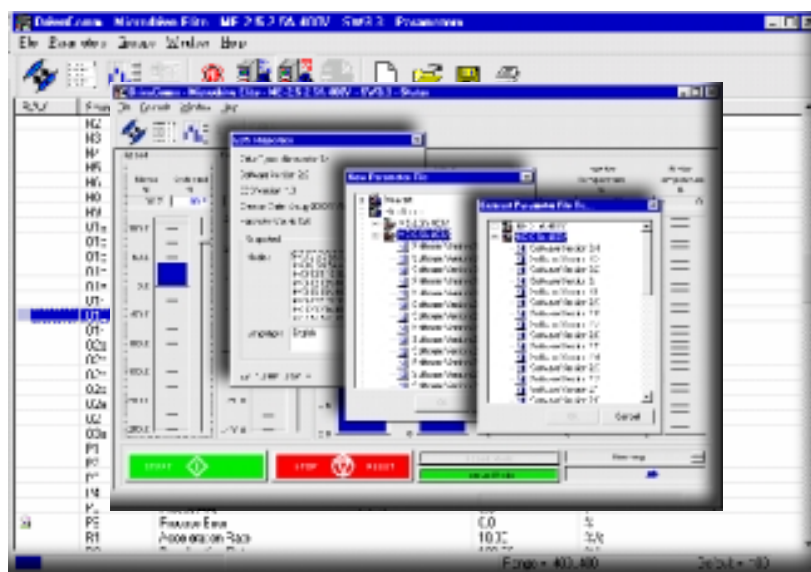
DriveComm3 is a Windows based software package that enables the commissioning, monitoring and parameter transfer between PDL Electronics motor control products and a PC.

With DriveComm3, the management of these parameters provides the user with the facility to inspect, manipulate and print the settings within our software configured motor control products. Conversion between software versions and also between sizes can be achieved easily through the file menu.

Detailed files can be built up for plant or machinery records. Tag names and item locations, as well as serial number information can be stored. Diagnostic analysis can be undertaken with the Status Tab. Monitoring parameters such as input and output voltages, frequency and current can provide a useful commissioning and monitoring tool.

Access to the software is available free from our website, where a simple registration must be completed to enable us to keep users advised of software updates and enhancements.

<http://www.pdl.co.nz/electronics/downloads.htm>



PDL Electronics Ltd, 81 Austin Street, P O Box 741, Napier, New Zealand
Phone: +64 6 843 5855 Fax: +64 6 843 5185 Email: electronics@pdl.co.nz

Specifications



Drive / Starter Compatibility:	Serial communications capable PDL Drives & Starters: Microdrive, Elite, Xtravert and RVSx
Serial Communications:	Modbus RTU
System Requirements:	Min. 32 Mb Pentium 100, Windows 9x, NT, 2000 screen resolution 800 x 600 or greater.

Performance Features



Parameter List	Key in new values for parameter; up-load or down-load all of, or a set of, parameters from a drive / starter; set all of, or a set of, parameters to their default values; save/restore parameters to/from disk.
Multiple Drive Software Versions	Support for multiple software versions/model sizes for drives or starters of the same type.
Status Window	Allows the user to continuously monitor various status parameters on the connected drive or starter and to control many of these from the PC.
Drive/starter Information	A dialog box allows entry of user information about a drive /starter, i.e. location, tag name etc.
Convert Parameter Files	Allows the user to convert any parameter file into a file compatible for use with a PDL AC motor controller of a different software version and/or model size. DriveComm performs the conversion while ensuring that all values in the destination file are legitimate with respect to value ranges and scaling.
Communications Setup	Manages the connection of the DriveComm software to the physical drive or starter, i.e. the communications port, baud rate, parity orientation and Modbus address.
Electronic Data Sheet Maintenance	Add, remove and check EDS files.
Multilingual Support	English, Spanish, German.
Printing Capabilities	Print all or a selection of parameters.

Appendix F

Vysta Screen List of the Algorithm Test Program

PDL Screen List - English - [V1][V2][V3][V4][V5][V6][V7]% -- [V8]

P *S1-Status Screen - System Status -- [V1]➤

*S2-Mode Screen - System Mode -- [V1]

*S3 - Setpoint Entry - Setpoint=[V1]

*S4-Speed and Current - Speed =[V4]RPM -- Current =[V2]A

*P- Pressure Control Settings - Pressure Ctrl➤

*P1- Motor Parameters - Motor Parameter➤

*P1A - Motor Size (KW) - Size =[V1]KW

*P1B - Motor Current (Amps) - Current =[V1]A

*P1C - Motor Voltage (Volts) - Voltage =[V1]V

*P1D - Motor Power Factor - P.F. =[V1]

*P1E - Motor Efficiency (%) - Efficiency =[V1]%

*P1F - Motor Speed (RPM) - Speed =[V1]RPM

*P2- Pump Parameters - Pump Parameter➤

*P2A - Pump Impeller Size (mm) - Impeller =[V1]mm

*P2B - Pump Inlet Pipe Size (mm) - Inlet =[V1]mm

*P2C - Pump Outlet Pipe Size (mm) - Outlet =[V1]mm

*P2D - Pump Maximum Flow (L/S) - Flow =[V1]L/S

*P2E - Pump Maximum Pressure (Kpa) - Press. =[V1]Kpa

*P2F - Pump Efficiency (%) - Efficiency =[V1]%

*T3 - Loop Controls - Loop Controls➤

*T3c - Loop Gain - Gain=[V1]

*T3d - Loop Integral - INT=[V1]

*T3e - Loop Derivative - Derv=[V1]

Y - Menu Options - Y MENU OPTIONS ➤

Y1 - Language Selection - Y1 LANGUAGE = [V1]

Y2 - Initialise Parameters - Y2 INITIALISE [V1]

Y3 Current Program - Y3 PROG [V1] OF [V2]

C Z - Commissioning Screens - Z COMMISSION=[V1] ➤ -- Z PASSWORD=[V2]

Z1 - Commissioning Mode Password - Z1 PASSWRD=[V1]

Z2 - Software/Hardware Versions - ELITE [V1]A[V2]V -- Z2 S/W[V3] H/W[V4]

Z2a Control Board Serial Number - Z2a Ser=

Z3 - Analogue Input 1 Value - Z3 AI1=[V1]=[V2][V3]

Z4 - Analogue Input 2 Value - Z4 AI2=[V1]=[V2][V3]

Z5 - Analogue Output 1 Value - Z5 AO1=[V1]=[V2][V3]

Z6 - Analogue Output 2 Value - Z6 AO2=[V1]=[V2][V3]

Z7 - Digital Input Values - Z7 MFI:[V1][V2][V3][V4][V5][V6] [V7]

Z8 - Fibre-Optic/Serial Communications Input Status - Z8 FI:[V1] SERIAL:[V2]

Z9 - Encoder Count - Z9 ENCODER=[V1]

Z9a Tacho Speed - Z9a TACHO=[V1]%

Z10 - Output Relay Status - Z10 RLY:[V1][V2][V3] DB:[V4]

Z11 - Fibre-Optic Input Value - Z11 F I/P=[V1]%

Z12 - Fibre-Optic Output Value - Z12 F O/P=[V1]%

Appendix G

DAVEY Pump Data Sheets

ISOspec ^{DAVEY}®

CF Series Bareshaft Pumps
CM Series Motor Pumps
ISO2858 Heavy Duty Industrial Centrifugal Pumps

DEPEND ON
DAVEY

PUMPS



TECHNICAL SPECIFICATIONS

The Davey ISOspec® range has been designed to international standard **ISO2858** ensuring a sturdy & reliable, long lasting, high performing product that consumers have come to depend on from Davey.

Suitable Applications

Water Supply

Commercial irrigation, booster stations, municipal water supply, flood irrigation, general transfer.

Industry

Cooling tower transfer, refrigeration systems, commercial fountains, condensate recovery, dairy wash down packages.

Environmental

Dust suppression in mining & quarry applications, fume scrubbers for odour control, water treatment transfer & filtration.

Leisure

Water circulation in large aquatic centres, backwash filtration in commercial pools, water features & commercial fountains.

Building Services

Commercial heating, air conditioning systems, pressure boosting, cooling tower & fire service applications.

Pumped Liquids

Non aggressive & non combustible liquids. Clean low viscous liquids free of fibres or particles.

Operating Conditions

CF Series Bareshaft Pumps

Maximum flow	900m ³ /hr
Maximum total heads	160m
Liquid temperature*	-15 to 140°C
Operating pressure	16 Bar

* With optional high temperature seal.

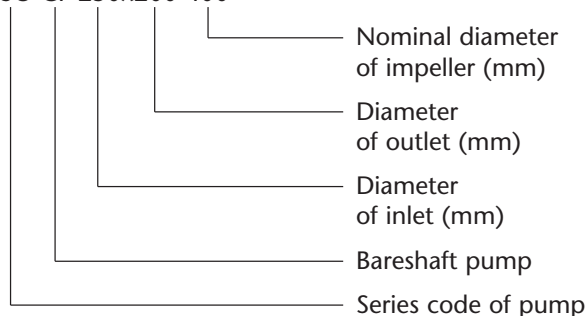
CM Series Motor Pumps

Maximum flow	410m ³ /hr
Maximum total heads	160m
Liquid temperature*	-15 to 140°C
Operating pressure	16 Bar

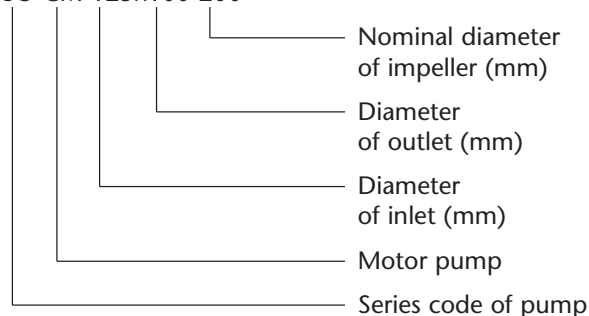
* With optional high temperature seal.

Model Designation (Examples)

ISO CF 250x200-400



ISO CM 125x100-200



The Davey ISOspec® Series is comprised of 29 standard sizes. Pumps with different impeller diameters and speeds are available to satisfy various performance ranges. Performance curves within this catalogue are in compliance with ISO2858 standards.

Design Features

Davey ISOspec® design is in accordance with international standards of **ISO2858**, which means ISOspec® Series pumps are interchangeable with other similar pumps, conforming to the same standards. This ensures a robust, long lasting, high performing product consumers have come to depend on from Davey.

Bronze wear rings - fitted as standard, replaceable front & rear wear rings with optional materials, for a trouble-free lifecycle.

Bronze impeller - in a closed design is fitted as standard to prevent corrosion in stationary or inactive situations. Cast 304SS or 316SS are available on request. The use of 3-D solid model (CAD) Computer Aided Design and (CFD) Computational Fluid Dynamics ensures **high efficiencies**, reducing overall running costs. Impeller diameters can be trimmed to suit specified performance. ISOspec® impellers are dynamically balanced, providing smooth, vibration free operation, preventing premature bearing failure.

Pump casing - highly efficient cast iron volute casings, with flanges rated to PN1.6MPa (16bar), drilled to AS2129, Table E. Material options: 304SS or 316SS.

Casing o-ring - re-usable o-rings in Nitrile for ease of re-assembly (optional materials available).

Shaft seal - single, high quality John Crane or approved equivalent mechanical seal with **carbon vs ceramic** fitted as standard to all ISOspec® Series pumps with other options such as Silicone vs Silicone or high temp also available.

Tappings - convenient suction & discharge pressure gauge tappings plus volute drain, fitted as standard to all ISOspec® Series pumps.

Back pull-out design - allowing for easy removal of rotating element without disturbing the pipework, lagging or pump volute casing. This is proven to reduce downtime whilst performing routine maintenance.

CM Series adaptor housing - heavy duty, flanged to IEC accepting AS 1359 standard designed motors.

Enlarged shaft - reduces shaft deflection. Standard in 420SS & 316SS as an option. Tapered & keyed shaft design allowing ease of removal in maintenance & positive locking whilst in operation (CF Series only).

Stub shaft - standard in 316SS, attached via a heavy duty muff coupling assembly. Tapered & keyed shaft design allowing ease of removal in maintenance & positive locking whilst in operation (CM Series only).

Bearings - Heavy duty SKF or approved equivalent, greased for life, reducing maintenance. Protected by a quality manufactured lip seal reducing ingress of moisture or foreign matter. Housed within removable bearing cap cover assembly (CF Series only).

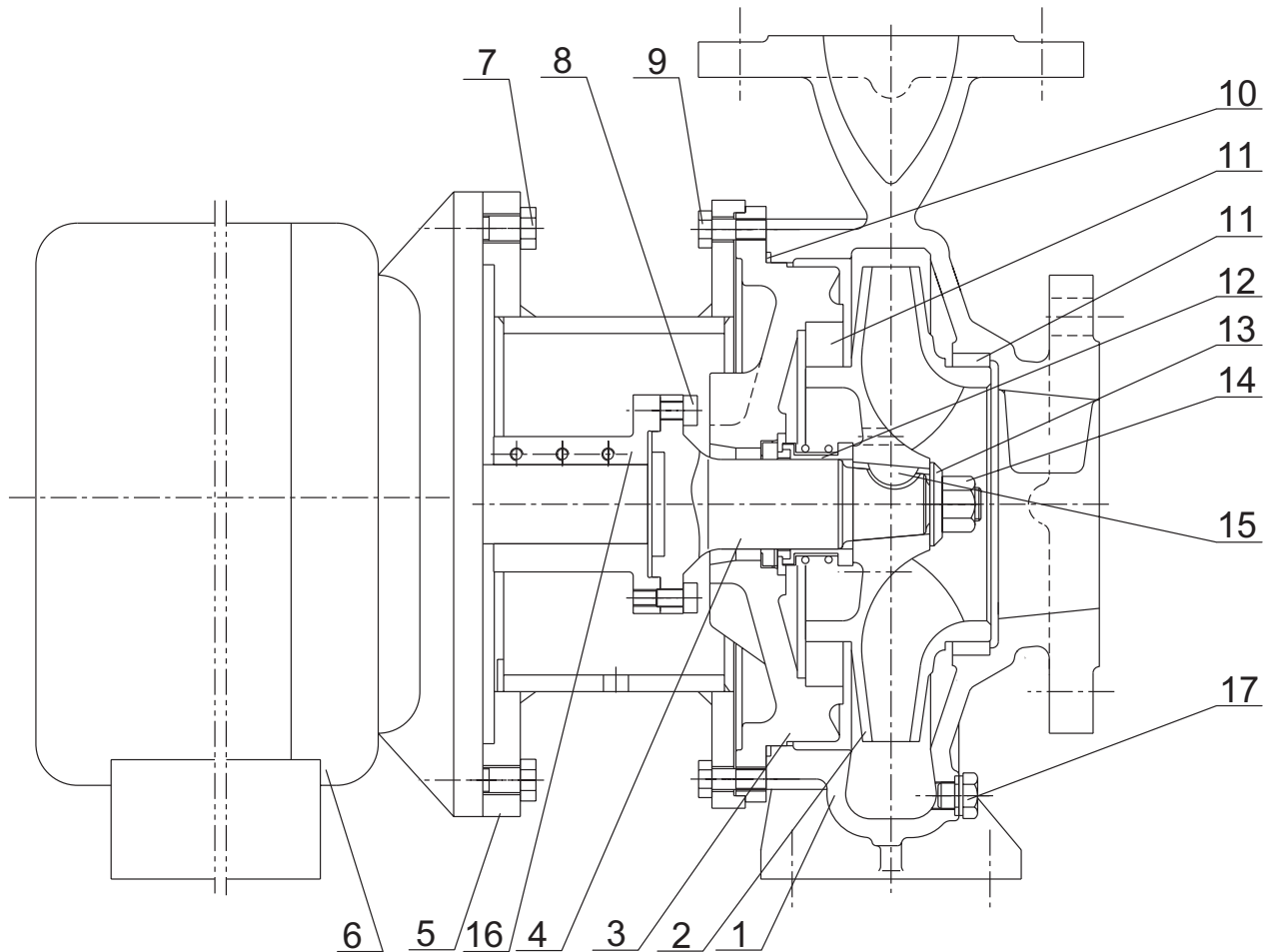
CF Series bearing housing - Robust / heavy duty, manufactured in high strength cast iron providing trouble-free life cycle.

CM Series Motors - All motors are MEPS compliant ensuring minimum efficiency requirements to AS/NZS 1359.5.2000. IP55, TEFC (totally enclosed fan cooled), insulation class F, temperature rise class B, 415V, 50Hz. Other voltages available upon request.

TECHNICAL SPECIFICATIONS

ISO DAVEY
spec

CM Series Arrangement

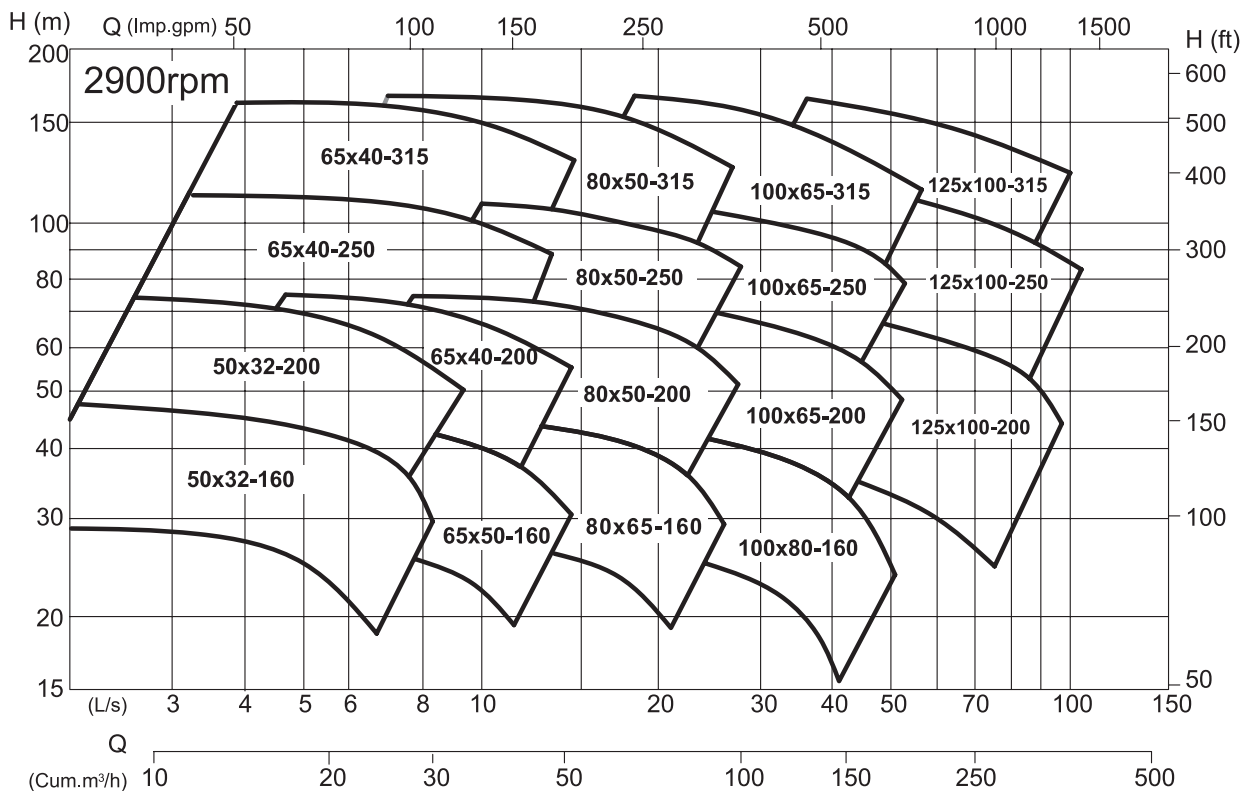
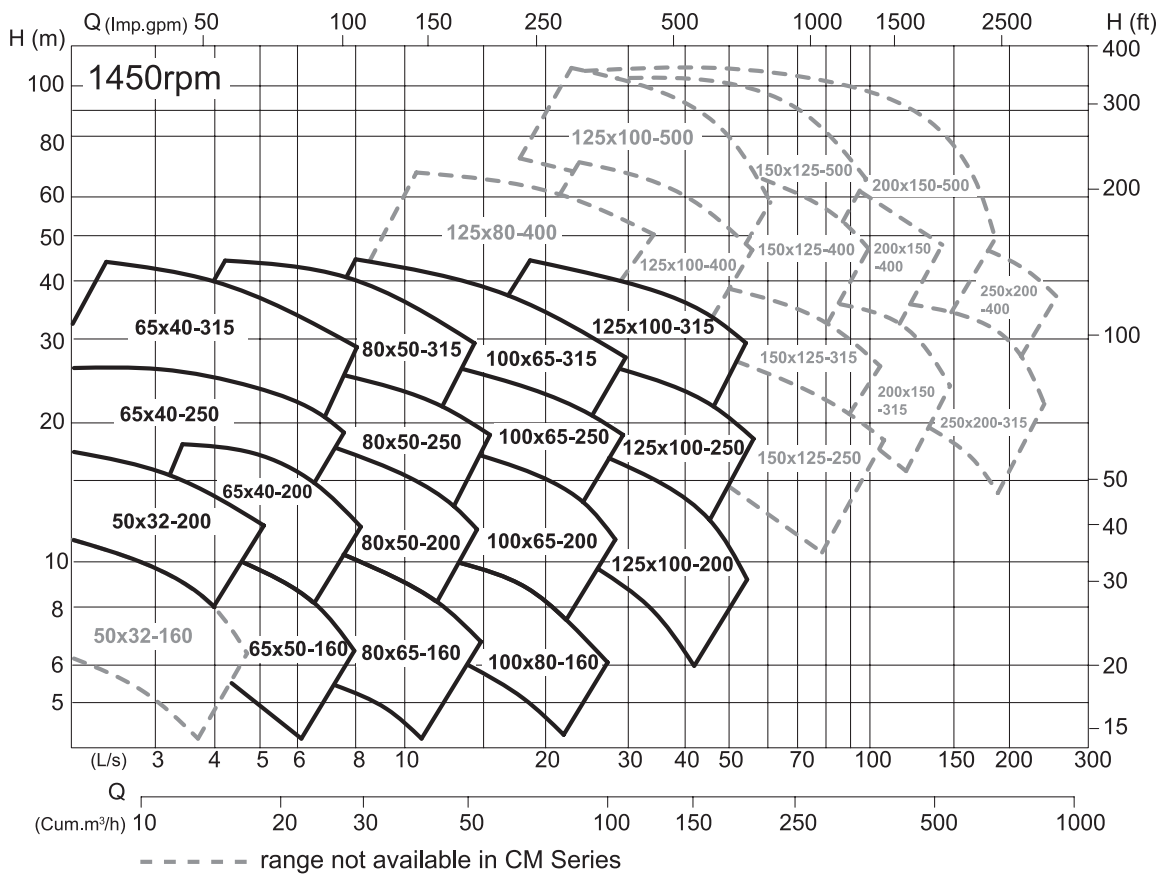


- | | | |
|---------------------|-----------------------|---------------------------|
| 1. Pump Casing | 2. Impeller | 3. Rear Casing Cover |
| 4. Stub Shaft | 5. Bell Housing | 6. Motor |
| 7. Bolt | 8. Bolt | 9. Bolt |
| 10. O-ring | 11. Bronze Wear Ring | 12. Mechanical Shaft Seal |
| 13. Impeller Washer | 14. Impeller Nut | 15. Impeller Key |
| 16. Muff Coupling | 17. Volute Drain Plug | |

TECHNICAL SPECIFICATIONS

ISO spec ^{DAVEY}

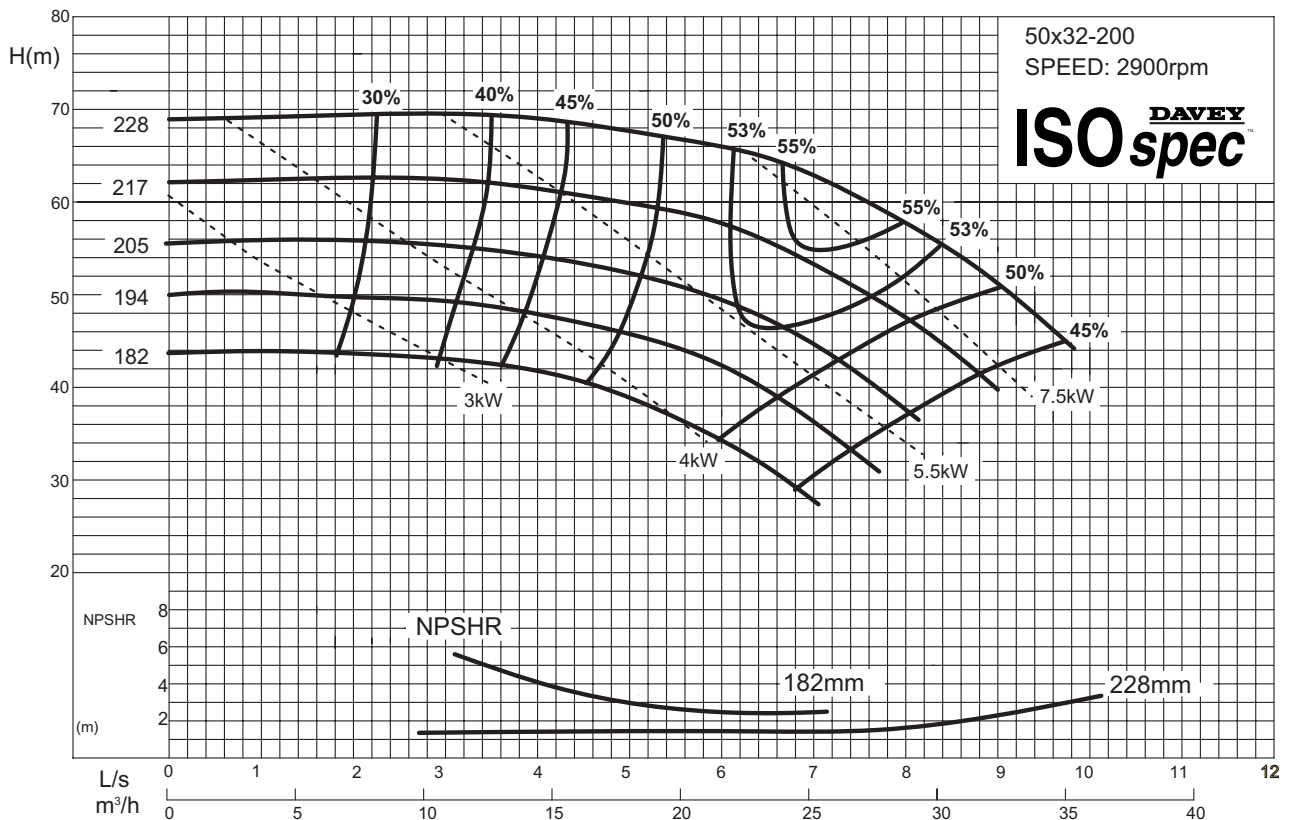
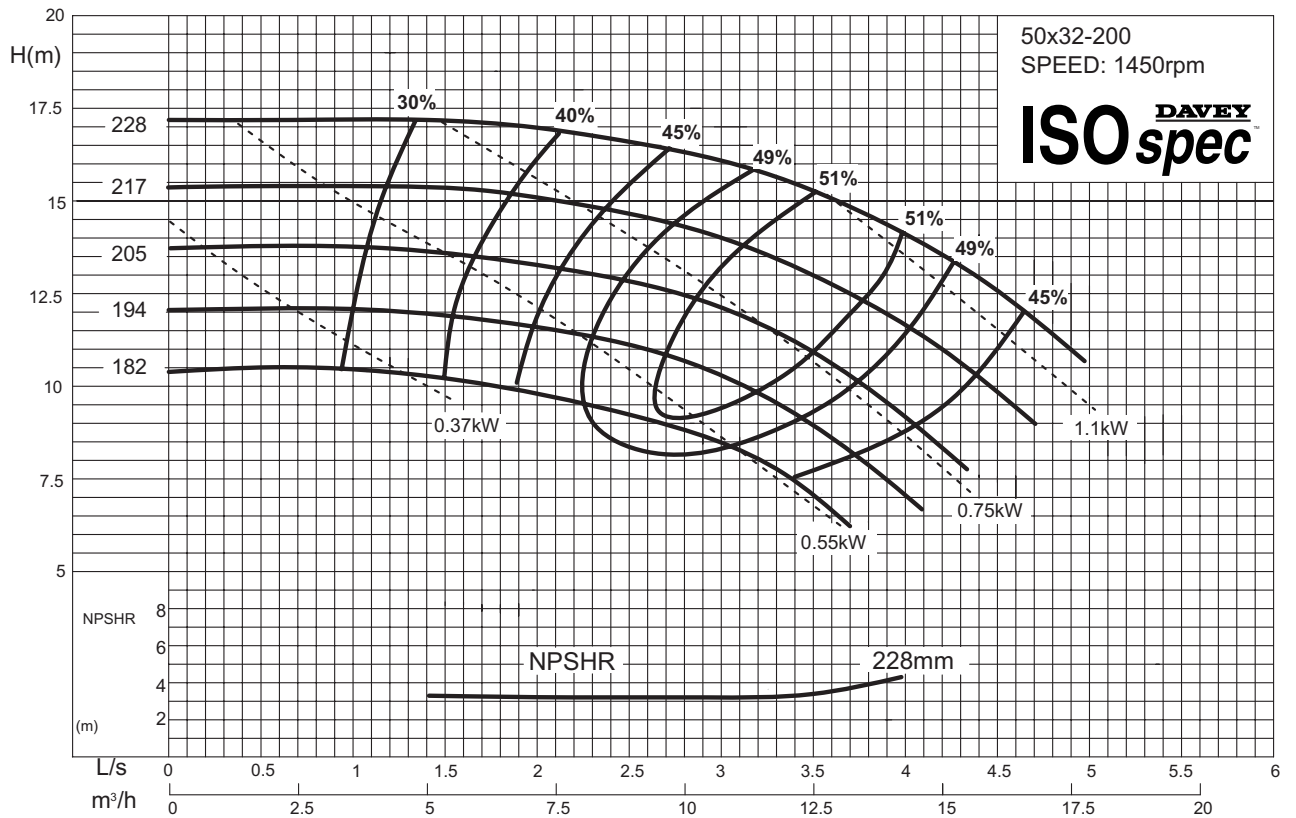
CM Series Performance Range



TECHNICAL SPECIFICATIONS

ISO spec DAVEY

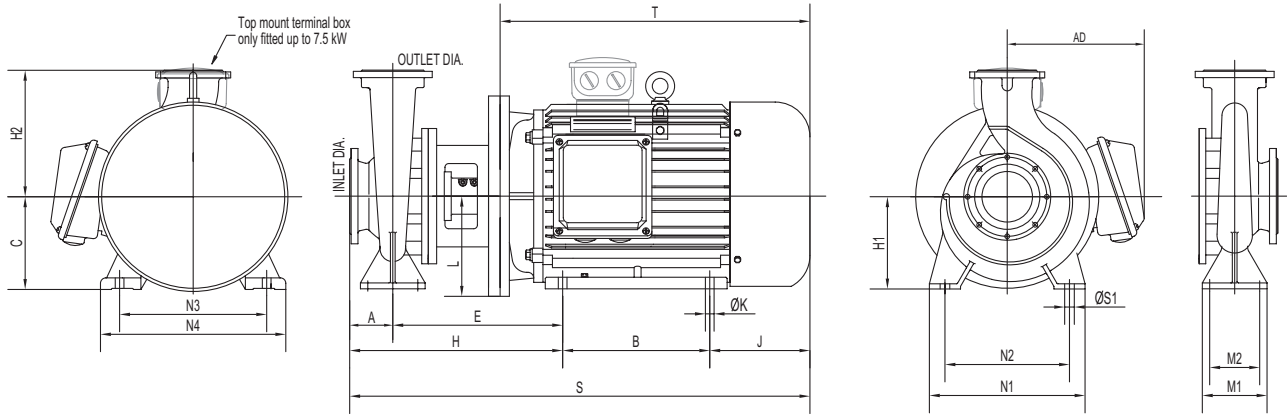
CF & CM Series Performance Curve 50x32-200



TECHNICAL SPECIFICATIONS

ISOspec DAVEY®

CM Series Dimensions - 4 Pole



Pump End In	Pump End Out	Pump End Imp	kW	Motor Frame	N3	B	C	E	H2	H	J	K	L	H1	N2	N1	S1	N4	S	T	M2	M1	AD	Kg
50	32	200	1.1	90S	140	100	90	209	180	289	104	10	100	160	190	240	14	180	493	260	70	100	N/A	54
			1.5	90L	140	125	90	209	180	289	104	10	100	100	160	190	240	14	180	518	285	70	100	N/A
65	50	160	1.1	90S	140	100	90	209	160	289	104	10	100	132	190	240	14	180	493	260	70	100	N/A	47
65	40	200	1.1	90S	140	100	90	209	180	309	104	10	100	160	212	265	14	180	513	260	70	100	N/A	55
			1.5	90L	140	125	90	209	180	309	104	10	100	160	212	265	14	180	538	285	70	100	N/A	56
			2.2	100L	160	140	100	231	180	331	117	12	125	160	212	265	14	205	588	320	70	100	N/A	67
65	40	250	1.5	90L	140	125	90	218	225	318	104	10	100	180	250	320	14	180	547	285	95	125	N/A	77
			2.2	100L	160	140	100	239	225	339	117	12	125	180	250	320	14	205	596	320	95	125	N/A	89
			3	100L	160	140	100	239	225	339	117	12	125	180	250	320	14	205	596	320	95	125	N/A	91
65	40	315	3	100L	160	140	100	240	250	365	117	12	125	200	280	345	14	205	622	320	95	125	N/A	100
			4	112M	190	140	112	247	250	372	130	12	125	200	280	345	14	245	642	340	95	125	N/A	105
			5.5	132S	216	140	132	286	250	411	166	12	150	200	280	345	14	280	717	395	95	125	N/A	125
80	65	160	1.1	90S	140	100	90	209	180	309	104	10	100	160	212	265	14	180	513	260	70	100	N/A	54
			1.5	90L	140	125	90	209	180	309	104	10	100	160	212	265	14	180	538	285	70	100	N/A	55
80	50	200	1.5	90L	140	125	90	209	200	309	104	10	100	160	212	265	14	180	538	285	70	100	N/A	59
			2.2	100L	160	140	100	231	200	331	117	12	125	160	212	265	14	205	588	320	70	100	N/A	69
			3	100L	160	140	100	231	200	331	117	12	125	160	212	265	14	205	588	320	70	100	N/A	71
80	50	250	3	100L	160	140	100	239.5	225	364.5	117	12	125	180	250	320	14	205	621.5	320	95	125	N/A	91
			4	112M	190	140	112	246.5	225	371.5	130	12	125	180	250	320	14	245	641.5	340	95	125	N/A	95
			5.5	132S	216	140	132	285.5	225	410.5	166	12	150	180	250	320	14	280	716.5	395	95	125	N/A	114
80	50	315	4	112M	190	140	112	247	280	372	130	12	125	225	280	345	14	245	642	340	95	125	N/A	108
			5.5	132S	216	140	132	286	280	411	166	12	150	225	280	345	14	280	717	395	95	125	N/A	128
			7.5	132M	216	178	132	286	280	411	168	12	150	225	280	345	14	280	757	435	95	125	N/A	138
			11	160M	216	178	132	286	280	411	168	12	150	225	280	345	14	280	757	435	95	125	255	206
100	80	160	1.5	90L	140	125	90	219	200	319	104	10	100	160	212	280	14	180	548	285	95	125	N/A	76
			2.2	100L	160	140	100	240	200	340	117	12	125	160	212	280	14	205	597	320	95	125	N/A	86
			3	100L	160	140	100	240	200	340	117	12	125	160	212	280	14	205	597	320	95	125	N/A	88
			2.2	100L	160	140	100	240	225	340	117	12	125	180	250	320	14	205	597	320	95	125	N/A	87
100	65	200	3	100L	160	140	100	240	225	340	117	12	125	180	250	320	14	205	597	320	95	125	N/A	89
			4	112M	190	140	112	247	225	347	130	12	125	180	250	320	14	245	617	340	95	125	N/A	92
			5.5	132S	216	140	132	286	225	386	166	12	150	180	250	320	14	280	692	395	95	125	N/A	110
100	65	250	4	112M	190	140	112	246.5	250	371.5	130	12	125	200	280	360	18	245	641.5	340	120	160	N/A	94
			5.5	132S	216	140	132	285.5	250	410.5	166	12	150	200	280	360	18	280	716.5	395	120	160	N/A	113
			7.5	132M	216	178	132	285.5	250	410.5	168	12	150	200	280	360	18	280	756.5	435	120	160	N/A	123
100	65	315	7.5	132M	216	178	132	311	280	436	168	12	150	225	315	400	18	280	782	435	120	160	N/A	186
			11	160M	254	210	160	360	280	485	177	15	175	225	315	400	18	325	872	495	120	160	255	238
			15	160L	254	254	160	360	280	485	178	15	175	225	315	400	18	325	917	540	120	160	255	259
125	100	200	4	112M	190	140	112	247	280	372	130	12	125	200	280	360	18	245	642	340	120	160	N/A	102
			5.5	132S	216	140	132	286	280	411	166	12	150	200	280	360	18	280	717	395	120	140	N/A	121
			7.5	132M	216	178	132	286	280	411	168	12	150	200	280	360	18	280	757	435	120	160	N/A	131

TECHNICAL SPECIFICATIONS



CM Series Dimensions - 4 Pole continued...

Pump End			kW	Motor Frame	N3	B	C	E	H2	H	J	K	L	H1	N2	N1	S1	N4	S	T	M2	M1	AD	Kg
In	Out	Imp																						
125	100	250	7.5	132M	216	178	132	311	280	451	168	12	150	225	315	400	18	280	797	435	120	160	N/A	173
			11	160M	254	210	160	360	280	500	177	15	175	225	315	400	18	325	887	495	120	160	255	246
			15	160L	254	254	160	360	280	500	178	15	175	225	315	400	18	325	932	540	120	160	255	267
125	100	315	11	160M	254	210	160	360	315	500	177	15	175	250	315	400	18	325	887	495	120	160	255	252
			15	160L	254	254	160	360	315	500	178	15	175	250	315	400	18	325	932	540	120	160	255	273
			18.5	180M	279	241	180	373	315	513	198	15	175	250	315	400	18	355	952	560	120	160	270	314
			22	180L	279	279	180	373	315	513	200	15	175	250	315	400	18	355	992	600	120	160	270	322

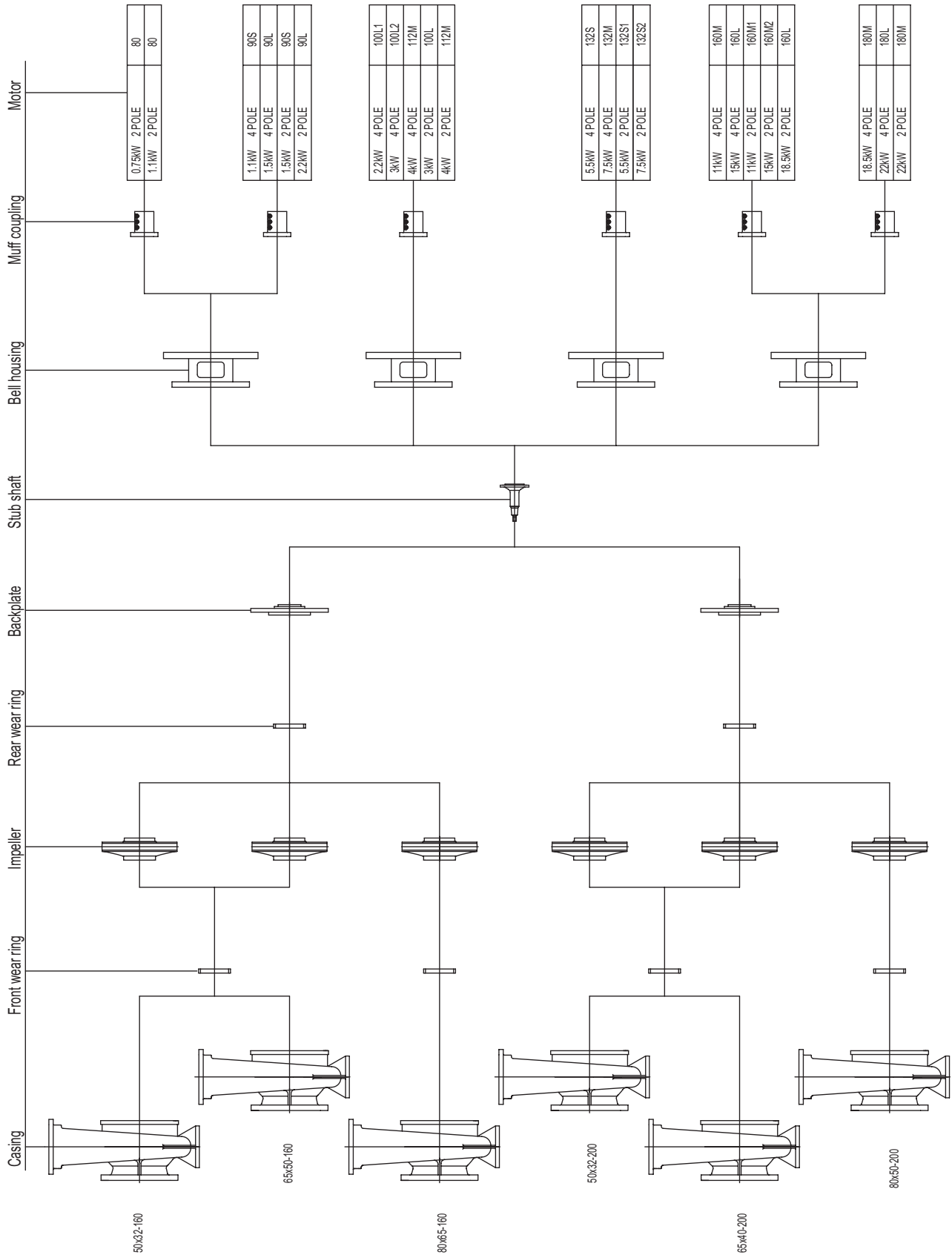
- Notes:
1. Standard flange drilling to AS2129 Table E: 16 bar rating.
 2. Dimensions in mm.
 3. All motors listed are 415V.
 4. All motors up to 7.5kW have top mounted terminal box.

N/A = Not Applicable

TECHNICAL SPECIFICATIONS

ISO spec **DAVEY**[®]

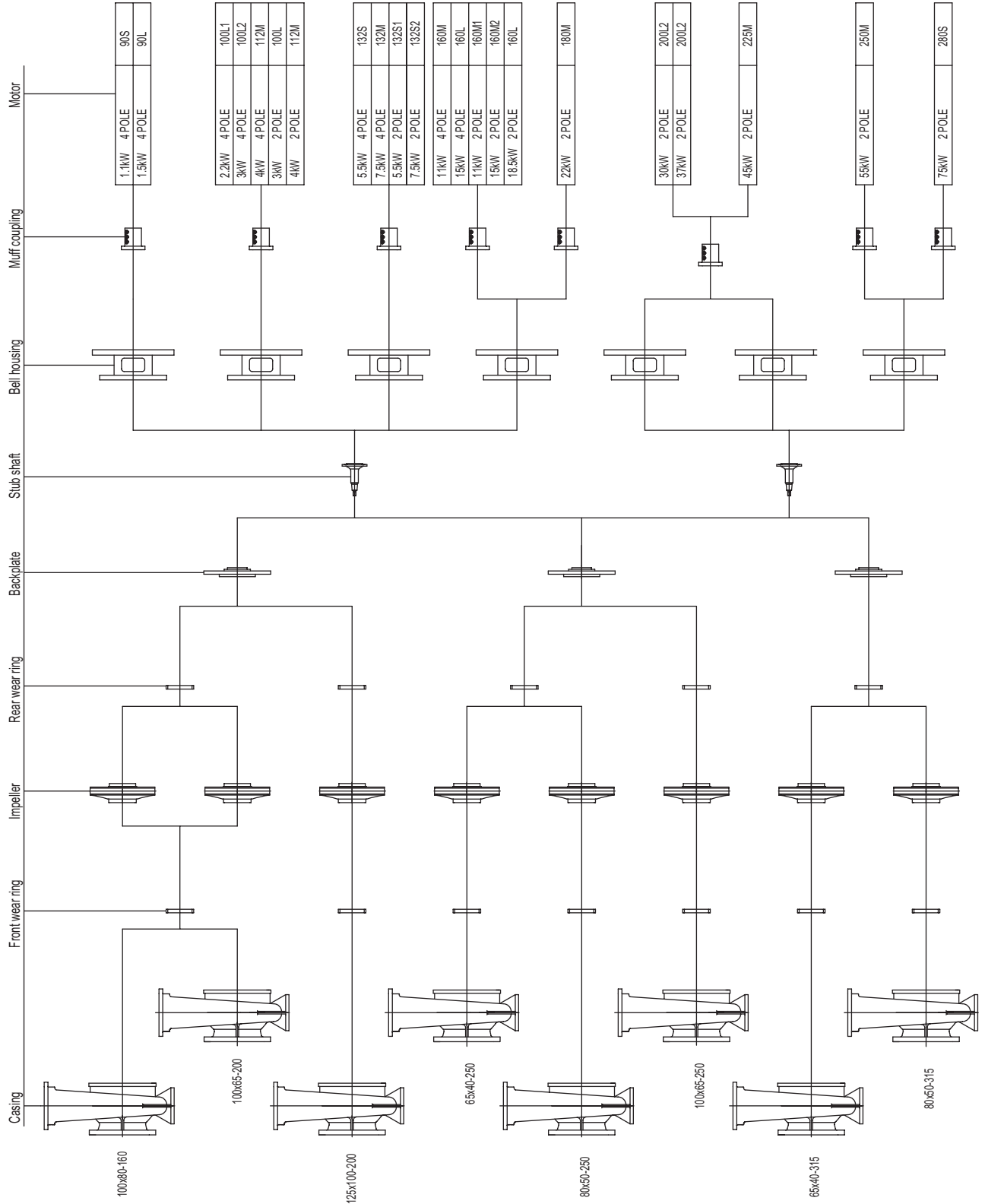
CM Series Shaft Module No. 1



TECHNICAL SPECIFICATIONS

ISO DAVEY *spec*

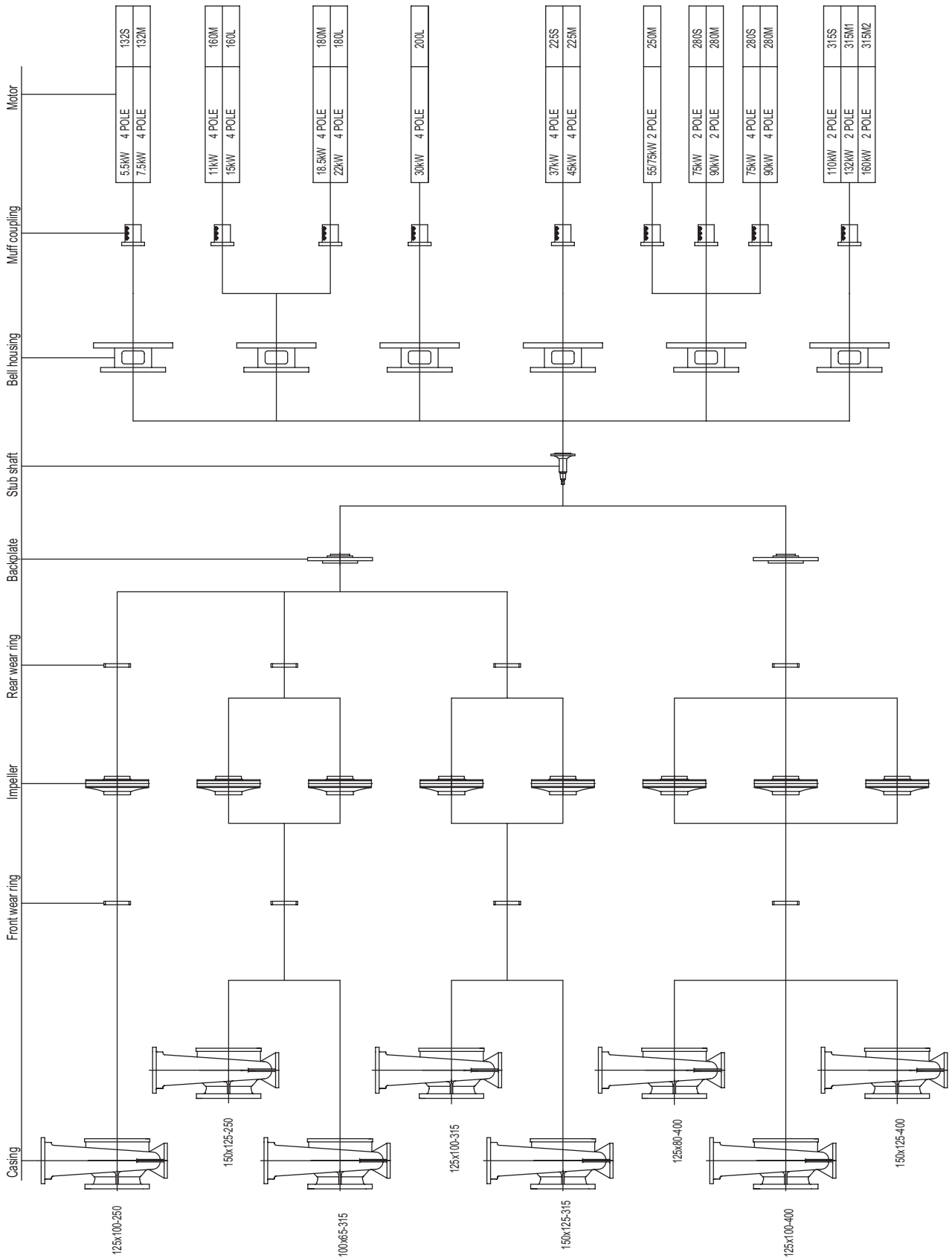
CM Series Shaft Module No. 2



TECHNICAL SPECIFICATIONS

ISO spec **DAVEY**[®]

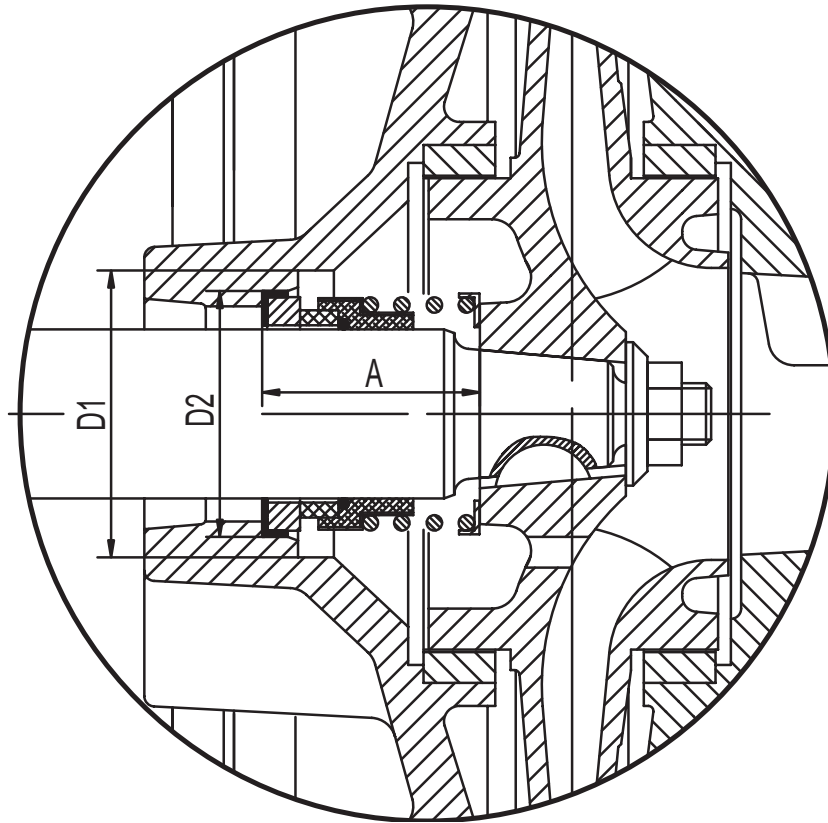
CM Series Shaft Module No. 3



TECHNICAL SPECIFICATIONS



CF & CM Series Seal Cavity Dimensions



Module	Seal size	D1	D2	A
No. 1	32	56	48	42.5
No. 2	43	68	61	45
No. 3	53	90	73	47.5
No. 4 *	60	92	80	52.5

All dimensions in mm, unless otherwise stated.

*CM Series not available.

TECHNICAL SPECIFICATIONS



Material Specifications

STANDARD CONSTRUCTION				
Component	Standard Fitted Pump	Australian Standard	British Standard	ASTM Standard
Casing	Cast iron GG25	AS1830/T260	BS1452 - Gr260	A48-Class 35
Back Plate	Cast iron GG25	AS1830/T260	BS1452 - Gr260	A48-Class 35
Bearing Housing (CF)	Cast iron GG20	AS1830/T220	BS1452 - Gr220	A48-Class 30
Casing Wear Ring	Bronze	AS1565/836	BS1400 LG2	B584-C83600
Impeller	Bronze	AS1565/836	BS1400 LG2	B584-C83600
Impeller Key	Stainless steel 420	AS1444/420	BS970 - Gr420-S37	A276 - type 420
Impeller Lock Nut	Stainless steel 420	AS1444/420	BS970 - Gr420-S37	A276 - type 420
Shaft (CF) (CM)	Stainless steel 420 Stainless steel 316	AS1444/420 AS2074/H6B	BS970 - Gr420-S37 BS1504 - Gr316	A276 - type 420 A351-CF-8M

OPTIONAL MATERIAL CONSTRUCTION				
Component	Optional Materials	Australian Standard	British Standard	ASTM Standard
Casing	Ductile iron Stainless steel 304 Stainless steel 316	AS 1831/400 AS 2074/H5C AS 2074/H6B	BS2789-Gr500/7 BS1504-Gr304 BS1504-Gr316	A536-84 70-50-05 A351-CF-8 A351-CF-8M
Back Plate	Ductile iron Stainless steel 304 Stainless steel 316	AS 1831 AS 2074/H5C AS 2074/H6B	BS2789-Gr500/7 BS1504-Gr304 BS1504-Gr316	A536-84 70-50-05 A351-CF-8 A351-CF-8M
Casing Wear Ring	Stainless steel 304 Stainless steel 316	AS 2074/H5C AS 2074/H6B	BS1504-Gr304 BS1504-Gr316	A351-CF-8 A351-CF-8M
Impeller	Stainless steel 304 Stainless steel 316	AS 2074/H5C AS 2074/H6B	BS1504-Gr304 BS1504-Gr316	A351-CF-8 A351-CF-8M
Impeller Lock Nut	Stainless steel 304	AS 2074/H5C	BS1504-Gr304	A351-CF-8
Shaft (CF)	Stainless steel 316	AS 1444/316	BS970-Gr316-S16	A276 type 316

Other materials available upon request, please contact Davey.