Development and Implementation of a Modbus Based Wireless Air Protocol

Mulalo Phillip Ramalata

Thesis presented in partial fulfillment of the requirements for the
Masters of Science in Engineering Sciences
at the Stellenbosch University

Supervisor: Dr. Riaan Wolhuter

Department of Electrical and Electronic Engineering

October 2004

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and has not previously in it's entirety or in part been submitted at any university for a degree.

Abstract

The recent performance improvements of wireless communication systems are presenting possibilites for the use of wireless networks for industrial applications, which typically impose severe restrictions in terms of response time and flexibility.

Traditionally most of the industrial protocols employ cable connections, which have limitations in terms of flexibility and development oportunities. The industrial protocol used for this project, is the Modbus protocol. This protocol is developed and implemented on a wireless environment by using data radio modems. The Modbus protocol is a master/slave protocol which provides an industry standard for industrial data transfer.

A Modbus driver is designed for radio networks, so that it can function with different PLCs and SCADA packages, supporting Modbus protocol. This enables control and monitoring to be excercised over a long distance, and enables control equipment to be placed as required and not with particular wiring restrictions.

Opsomming

Die onlangse verbetering in radiokommunikasiestelsels bied moontlikhede vir die gebruik van radionetwerke vir industriële toepassings, wat gewoonlik streng beperkings plaas in terme van responstyd en buigbaarheid. Tradisioneel is meeste van die industriële protokolle kabelgebaseer, wat redelik onaanpasbaar is.

Die Modbus protokol is as basis in hierdie projek gebruik. Dit is 'n meester/slaaf industriestandaard, wat hier aangepas is vir toepassings met radiomodems. Die Modbus drywer is ontwerp om funksioneel te wees met 'n verskeidenheid van PLC's en SCADA pakette, wat die Modbus protokol ondersteun. Dit stel beheer- en meetaksies in staat oor lang afstande en laat relatief vrye plasing toe van toerusting sonder bedradingsbeperkings.

Acknowledgment

I wish to thank the following people:

- My supervisor Dr. R. Wolhuter for your support, I was no going to reach this point if it was not his support and words of encouragement.
- My mother, father, brothers and sisters for all the support you have given me.
- Institute for Satellite and Software Applications (ISSA) for all their support.
- All the friends and colleagues who have contributed to the progress of this project.

Glossary

ASCII	-American Standard Code for Information Interchange
CRC	-Cyclic Redundancy Check
CPFSK	-Continous Phase Frequency Shift Keying
CRLF	-Carriage Return Line-Feed
GND	-Ground
GUI	-Graphic User Interface
HEX	-Hexadecimal
HMI	-Human Machine Interface
IR	-Infrared
I/O	-Input or Output
ICASA	-Independent Communications Authority of South Africa
IP	-Internet Protocol
LSB	-Least Significant Bit
MODEM	-Modulator/Demodulator
MSB	-Most Significant Bit
OSI	-Open Systems Interconnection
PC	-Personal Computer
PLC	-Programmable Logic Controller
RF	-Radio Frequancy
RTU	-Remote Terminal Unit
RXD	-Receive Data
Transceiver	-Transmitter/Receiver
TXD	-Transmit Data
SCADA	-Supevisory Control and Data Acquisition
TCP	-Transport Control Protocol
UHF	-Ultra High Frequency

Contents

1 Introduction **1**

1.1 Goals 2

1.2 Project Overview 3

2 Wireless Telemetry **5**

2.1 Introduction 5

2.2 Wireless Transmission Media 6

2.2.1 Radio Transmission 6

2.2.2 Microwave Transmission 6

2.2.3 Infrared 7

2.3 Wireless Telemetry System 7

2.3.1 Typical Wireless Telemetry System 7

2.3.2 Telemetry Transactions 7

2.3.3 Radio Modem and PLC Protocols 9

2.3.4 Frequency Bands 10

2.4 Summary 10

3 Industrial Protocols **11**

3.1 Introduction 11

3.2 Some Industrial Protocols and their Applications 12

3.2.1 Modbus 12

3.2.2 Profibus 13

3.2.3 Foundation Fieldbus 13

3.2.4 DeviceNet 14

3.3 Summary 14

4 Modbus Protocol **16**

4.1 What is Modbus? 16

4.2 When do we use Modbus? 16

4.3 How does Modbus device communicate with other devices? 17

4.3.1 Physical Interconnection Standards 17

4.4 Modbus serial transmission mode 18

4.4.1 ASCII Transmission Mode 18

4.4.2 RTU Transmission Mode 19

4.5 Modbus Message Framing 20

4.5.1 ASCII Framing 20

4.5.2 RTU Framing 20

4.5.3 Address Field 21

4.5.4 Modbus Function Field 21

4.5.5 Modbus Data Field 22

4.5.6 Modbus Error Checking Field 22

4.6 Modbus Functions Format 23

4.6.1 Coil 24

4.6.2 Input Status 24

4.6.3 Input Register 24

4.6.4 Holding Register 24

4.7 Modbus Function Codes 24

4.8 Modbus Exception Responses 25

4.9 Summary 27

5 General Systems Design **28**

5.1 Hardware consideration 29

5.1.1 Wireless Data Transceiver 29

5.1.2 Programmable Logic Controller 30

5.2 Software Considerations 31

5.2.1 Modbus ASCII Master Software 31

5.2.2 Slave (PLC) 33

5.3 Summary 34

6 Implementation **36**

6.1 Hardware Configuration 36

6.1.1 Cable Connections 36

6.1.2 Overall system configuration 37

6.2 Slave (LG PLC GLOFA GM7) 38

6.2.1 GMWIN and Communication 38

6.3 Modbus ASCII Master Software 40

6.3.1 Serial Communication Driver 40

6.3.2 LRC Implementation 41

6.3.3 Function Codes Implementation 42

6.4 System Evaluation 54

6.5 Timing and Delay 55

6.6 Summary 55

7 Graphic User Interface **56**

7.1 Feature of Modbus ASCII window 56

7.2 Main Window (Menu) 57

7.2.1 Communication Menu 58

7.2.2 Configure Menu 58

7.2.3 Polling Menu 59

7.3 Reading PLC Coils/Registers Contents 61

7.4 Writing to the Output of a PLC 61

7.4.1 Writing to Output Coils 62

7.4.2 Writing to the Holding/Output Registers 62

7.5 Other Parameters 64

7.5.1 Registers Grid 64

7.5.2 Coils Grid 65

7.5.3 Monitoring Transactions 65

8 Conclussion and Recomendations **66**

8.1 Conclusion 66

8.2 Recommendations 67

A LRC Implementation **70**

A.1 Appending the LRC Value into the Message 71

B Delphi Code **72**

C Radio Modem and Its Specifications **73**

D Diagrams **75**

E Fieldbus Comparison Chart **77**

F PLC Data Sheet **78**

List of Figures

1.1 General Communications Model 2

2.1 Typical Wireless Telemetry System [17] 8

4.1 General Modbus ASCII Message Frame 20

4.2 General Modbus RTU Message Frame 21

5.1 General System Design 29

5.2 Master Station Query/Reasponse Transmission Flow Diagram 32

5.3 Modbus Slave Query/Response Flow Diagram 34

6.1 Overrall System Configuration 38

6.2 Communication Parameters Dialog 39

6.3 Window of Query/Response of Read Coils Function Code 44

6.4 Read Input Status Query/Response Graphical Illustration 46

6.5 Read Input Registers Graphical Interpretation 49

7.1 Modbus ASCII Main Window 57

7.2 Communication Settings 58

7.3 PLC Addressing Window 60

7.4 Poll Time and Response Time Settings 60

7.5 Window for Writting to the Outputt Coils of the PLC 62

7.6 Force Coil 1 of Slave Device 1 to OFF 63

7.7 Writing to the Multiple Holding Registers of the PLC 63

7.8 Preset Single Register Window 64

7.9 Moniroring Transactions Window 65

C.1 MDS Data Transceiver 74

D.1 MDS OEM 705 Data Transceiver to LG GM7 PLC RS232 Cable 75

D.2 LG PLC with A/D Inputs and Outputs Configuration 76

List of Tables

4.1 Modbus Function Codes as Supported by LG G7M DR10A PLC [1] 25

4.2 Modbus Exception Codes [10] 26

6.1 RS232 Cable Connections for PLC Programming 37

6.2 RS232 Connection For PLC To Modem 37

6.3 Memory allocation for Input/Output area for Modbus protocol communication 39

6.4 Read Coils Query/Response Example 43

6.5 Read Input Status Query-Response Example 45

6.6 Read Holding Register Query-Response Example 47

6.7 Read Input Register Query-Response Example 49

6.8 Force Single Coil Query-Response Example 50

6.9 Preset Single Register Query-Response Example 51

6.10 Query/Response to Force 8 Output Coils Starting at %MX00 53

6.11 Preset Multiple Register Query-Response 54

6.12 Maximum Query/Response Parameters 54

A.1 Placing LRC into the Message 71

Chapter 1

Introduction

Wireless technology has made some remarkable advances recently, and virtually everyone now uses it in some aspect of their life. Wireless technology can also be used with great success and at significant cost savings in industrial applications. The objective of this project is to explore the application of this technology to a particular industrial application area. The industrial realm includes a broad set of applications that require a higher level of capability and reliability than attainable from consumer devices meant for home or office use. For example, water treatment, process control, oil and gas, mining, railroad, utility, and supervisory control and data acquisition (SCADA) applications, all fit within this grouping. In these applications it is necessary to communicate with a programmable logic controller (PLC), an actuator, a flow transducer, a temperature or pressure sensor, or a human machine interface (HMI) system.

Industrial communications involves a wide range of hardware and software products and protocols, used to communicate between standard computer platforms and devices for industrial automation applications. Industrial radios replace wires and cables.

In order for the devices on the networks to understand each other, they have to use the same protocol. It is wise to choose an open protocol, supported by most vendors, like Modbus, Profibus and Fieldbus. In this project the implementation of a wireless air based Modbus protocol is the one of the objectives. The Modbus protocol is a master/slave protocol, supported by many Programmable Logic Controller (PLC) manufactures.

The simplified communications model consist of three main blocks i.e. transmitter,receiver and transmission media, as shown in Figure 1.1.

Figure 1.1: General Communications Model

1.1 Goals

The purpose of this project is:

- To design a wireless communication network suitable for use in modern SCADA systems.
- To base the network communications on a commonly available, industry standard protocol to enable a wide range of applications.
- To design a network with hardware and software, interoperability, collision avoidance, high transmission efficiency and the ability to add new devices on the network without disrupting traffic flow.
- The development of the above lead to an additional objective, i.e creation of a basic, but fully functional application layer resident, user interface and data management facility.

To achieve this objectives, various disparate components had to be integrated, as is usual for many designs. This include sensors and metering devices, a Programmable Logic Controller (PLC), a communications network to link it all together, a host computer and HMI software.

1.2 Project Overview

This document is structured as follows :

Chapter 2 : Wireless Telemetry
Telemetry is a process by which an objects characteristics are measured (such as velocity, voltage, current and pressure), and the results transmitted to a distant station where they are displayed, recorded and analyzed. The transmission media may be air and space for satellite applications, or copper wire and fiber cable for static ground environments like power generating plants. In this chapter different aspects of telemetry will be discussed, and some advantages and disadvantages of different transmission media.

Chapter 3: Industrial Protocols
When communication is desired among the industrial/automation devices from different vendors, software development can be a nightmare. Different vendors use different data formats and data exchange protocols. As the use of computer communications and industrial networking proliferates, a one-at-a-time special purpose approach to communication software development is time consuming and too costly to be acceptable. The alternative is for vendors to adopt and implement a common set of conventions. For this to happen, different standards are set for industrial communications and automation industries like, Modbus, Profibus, DeviceNet and Fieldbus. This chapter will discuss some of the popular industrial protocol standards.

Chapter 4: Modbus Protocol
This chapter will cover the detailed background of Modbus protocols such as when to use it? How does it communicate with other devices (topology), message framing and error checking methods.

Chapter 5: General System Design
This chapter indicates factors which where considered in the design of the whole system, this are factrors like hardware and sotware considerartion.

Chapter 6: Implementation
This chapter explains all the functions code implemented, it also explains how the hardwares were configured.

Chapter 7: Graphic User Interface
This chapter explain how the system parameters are configured on the master station and

how the response are displayed on the screen. This allows the user not to be concerned about the raw of bits and bytes, which are being transmitted or received. Basic windows and settings are explained in this chapter.

Chapter 8: Conclusion and Recommendations
Project conclusions and some future recomendations.

Chapter 2

Wireless Telemetry

2.1 Introduction

Telemetry can be defined as the sensing and measuring of data at some remote location and then transmitting that data to a central or host location were it can be monitored and used to control a process at the remote site.

The first known use of telemetering was by Shilling in Russia in 1812 for the firing of missiles [2]. Since then telemetering has been developed toward industrial applications. Various mediums or methods of transmitting data from one site to another have been used e.g. Infrared (IR),Radio Frequency (RF), optical, etc.

A telemetry system consists of two or more telemetry devices. One device is typically located in the remote site where equipment such as pumps, valves, gauges, generators, lights or sensors are located. This remote telemetry device interfaces with instruments that measure and control temperature, pressure, flow, power, security, voltage, current, etc. of the equipment. The host devices are typically used to monitor and control the state of the remote site.

The physical layer is the basis of all networks. The purpose of the physical layer is to transport the raw bit stream from one machine to another. Various physical media can be used for the actual transmission. Transmission media can be guided or unguided. The principal guided media are twisted pair, coaxial cable, and fiber optics. Unguided media include radio, microwaves, infrared, and lasers through the air. The unguided (wireless) transmission media will be discussed in the next section.

2.2 Wireless Transmission Media

Digital wireless communication is not a new idea. As early as 1901, the Italian physicist Guglielmo Marconi demonstrated a ship-to-shore wireless telegraph, using Morse Code (dots and dashes are binary, after all). Modern digital wireless systems have better performance but the basic idea is the same.

Different types of unguided media are discussed in the next sections.

2.2.1 Radio Transmission

Radio waves are easy to generate, can travel long distances, and some frequencies can penetrate buildings easily. Nowadays they are widely used for industrial communications. Radio waves could also be omni directional (i.e they travel in all directions from the source) so the transmitter and receiver do not have to be aligned physically. Due to radios ability to travel long distances, interference between users is a problem. For this reason, all governments tightly license the use of radio transmission, except for the ISM bands [9, 12].

2.2.2 Microwave Transmission

Above 1000 MHz, radio waves nearly travel in a straight lines and can therefore, be narrowly focused. Concentrating all the the energy into a small beam by means of a parabolic antenna gives a much higher signal to noise ratio, but the transmitting and receiving antenna must be accurately aligned with each other. Since microwaves travel in a straight line, the earth curvature is a problem if the towers are too far apart. Consequently, repeaters are needed periodically [9, 12].

Microwaves do not penetrate buildings very well as compared to radio waves a lower frequencies. Microwave is widely used for long-distance communications.

2.2.3 Infrared

Unguided infrared and millimeter waves are widely used for short range communication. The remote controls used on televisions, VCRs, and stereos all use infrared communication. They are relatively directional, cheap, and easy to build, but have a major drawback in that, they do not pass through solid objects. In general, as we go from long-wave radio toward visible light, the waves behaves more and more like light and less and less like radio.

Advantages of IR:

- No license required to operate an infrared system, in contrast to most Radio Frequency bands.
- Little or no interference from other devices around.
- Security for infrared systems against eavesdropping, is better than that of radio systems, precisely for the above-mentioned reasons [9].

Disadvantages of IR:

- Not suitable for communication over a long distance.
- Can easily be obstructed (It is directional).

2.3 Wireless Telemetry System

2.3.1 Typical Wireless Telemetry System

As discussed in Chapter 1, a typical telemetry system consists of a host site and one or more remote sites (slaves stations). A typical system is as shown in Figure 2.1:

2.3.2 Telemetry Transactions

At a remote site, sensors act as typical data source. The output of the sensor is converted to digital data by Programmable Logic Controllers (PLCs) or Remote Terminal Units

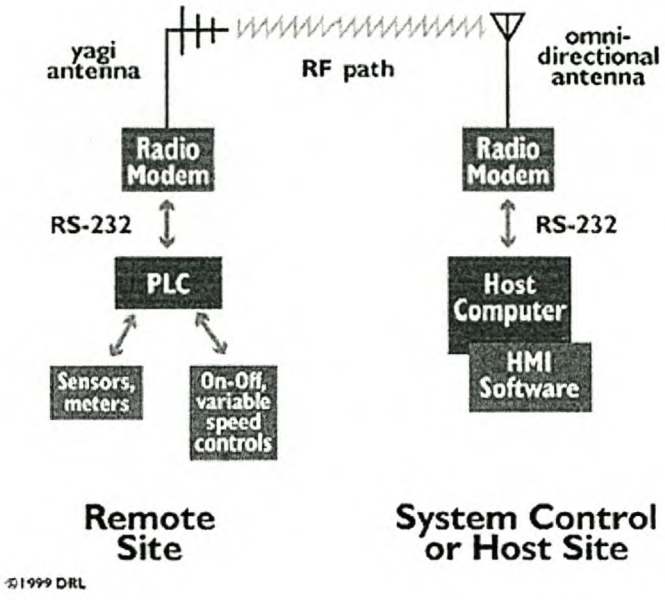


Figure 2.1: Typical Wireless Telemetry System [17]

(RTUs). The PLC is interfaced to a radio modem device (through RS-232 or RS- 485 cable) that converts the digital data into an analog signal that can be transmitted over the air. The radio transmitter then transmits the signal to the host site (master station) receiver. When the signal is received by the receiver the process is reversed. The modem takes the analog signal received and converts it back to a digital form that can be processed by the data recovery equipment [17].

In one commonly employed configuration, the host site requests data from the remote sites (master/slave communication). The master station transmit a request to the remote unit telling it to send its data. The master station reverts to a receive mode and awaits the transmission from a remote site (slave device). After the remote sends its data, it goes back to a receive mode waiting for a further instructions to come from the host. Once the base receives the remote site information, it may send additional instructions to that site or continue to the next remote site. This polling process continues until all the remotes sites in the system have sent their data [17].

2.3.3 Radio Modem and PLC Protocols

When configuring wireless telemetry system, there are two options for integrating the components:

1. Separate components (separate analog transceiver and voice band modem)
2. Integrated unit (wireless transceiver and modem)

Using integrated radio modems has some advantages such as [17]:

- Easier interfacing.
- High data rates and polling rate
- Digital interconnection is by standard serial connection, like RS-232
- Well documented interface with all popular PLCs or RTUs, and
- It is easy to configure.

Separate components are:

- Difficult to configure
- Could exhibit very high latency, and
- Low data rate and polling rate.

When choosing radio modem components it is important to note that PLC and RTU devices have their own communication protocols or languages that encapsulate the data stream in Packets or Frames. This frame encodes the data within a message start and end marker, an originating and destination address, and a CRC or checksum, as a minimum.

Modbus is probably the most widespread serial protocol in industry [19]. Most devices, beingable to communicate serially, talk Modbus. The Modbus communication is master/slave driven. Modbus is capable to run full-duplex RS232 lines and half-duplex RS485 solutions.

Newer implementation variants run via Ethernet and TCP/IP, but generally not over radio communications, as implemented in this project.

There are other protocols that operate similarly. Generally master/slave protocols that are framed, and employ message addressing, error checking specifically designed for master/ slave polling, work well in the wireless environment.

2.3.4 Frequency Bands

Telemetry radio systems are normally configured as one or more fixed base stations that obtain information from other fixed stations at remote sites. ICASA has allocated certain frequencies to be used for fixed operation. There are certain frequencies available in UHF band(450 - 470 MHZ) for radio telemetry operations.

2.4 Summary

A properly designed telemetry network saves time and cost by eliminating the service personnel necessary to visit each remote site for inspection, and data logging adjustment. The benefits of using wireless telemetry are easy troubleshooting and increase in equipment life, among others.

As discussed in previous sections, main hardware items to consider when designing radio telemetry systems, are as listed below:

1. Device configuration.
2. Radio modem hardware.
3. PLC (and/or RTU) communication protocols.
4. System configuration (Number of slave devices) and layout.

Other factors to be considered, by design engineers would be link budgets, antenna design and telemetry radio range as incorporated in the general signal propagation modeling. These aspects are however not covered in this thesis.

Chapter 3

Industrial Protocols

3.1 Introduction

There is growing momentum toward more multi-vendor connectivity through standardized versions of the sophisticated industrial networking protocols. This chapter will introduce some of the popular industrial protocol standards.

There are some important reasons that users are looking for to move to standardized industrial networks:

- Open Systems - It is difficult and costly to integrate systems with instrumentation from different vendors because of the multitude of communication protocols. With standard protocols, devices from many suppliers can co-exist on the same network and communicate with one another.
- Cost Reduction in Wiring - Many systems are still using 4-20 mA analog instrumentation, requiring extensive point-to-point wiring. Multi-drop means lower installation cost.
- Increased Information Need - In todays manufacturing and monitoring environment, industries are required to gather more information about their processes and the instrumentation connected to the processes. Traditional 4-20 mA instrumentation provides only one value. On a digital network, instruments can provide maintenance and diagnostics information for better tracking of instrument performance. This depends on the communication capabilities of the devices connected in a network.

- Intelligent Devices - Device vendors are putting more intelligence into their devices to satisfy customers demands for more functionality at lower costs. The increased information available with a digital network is necessary for capitalizing on the extra capabilities made possible by intelligence in the devices.
- Response Time - Real time requirements and message prioritizing make extra demands.
- Network topology - Flexibility in configuration is an advantage.

There are many different kind of industrial protocols and some of these are discussed in the next section.

3.2 Some Industrial Protocols and their Applications

Most of the industrial bus types work over RS-485 which is the physical layer which defines pinouts, cable characteristics and signal levels. Just because two devices have a RS-485 connection, does not mean that they can talk to each other, as they have to comply with certain standards as discussed below.

3.2.1 Modbus

Modbus RTU/ASCII is probably the most popular serial protocol in instrumentation, automation and process control. It provides everything from short serial linkage of smart devices to wide area networking of many devices. It is a simple way of encapsulating analog and digital I/O and parameters in registers.

Modbus devices communicate over a serial network in a master/slave (request/response) type relationship using one of two transmission modes: ASCII (American Standard Code for Information Interchange) mode or RTU (Remote Terminal Unit) mode.

Enhancements to Modbus include Modbus Plus and Modbus/TCP protocols, both of which allow Modbus information to be encapsulated in a network structure to support peer-to-peer communications. Modbus Plus communicates via a single twisted pair of wires and uses a token passing sequence for peer communication sequences. Modbus/TCP

is an open standard designed to facilitate Modbus message transfer using TCP/IP protocol and standard Ethernet networks.

Modbus can link as many as 250 devices on an RS-485 cable. Further more, there are many gateway devices that link Modbus and other networks, so if your product has a Modbus Protocol on a serial port you can communicate to almost any network via some converter [22, 23, 20].

A few disadvantages of Modbus is that transmission speed is very slow on standard serial media and it does not offer peer-to-peer capabilities. In this project standard wireless media is investigated to improve the transmission speed and general flexibility, obviously over much increased distances.

3.2.2 Profibus

Profibus is an industrial bus designed for deterministic communication between computers and PLCs. It is one of the most widely accepted international networking standards. Profibus can handle large amounts of data at high speed and serves the needs of large installations. Based on a real time asynchronous token bus principle, Profibus defines multi-master and multi-slave communication relations, with cyclic and acyclic access, allowing transfer rates of up to 500 kbits/s. The physical layer, the data link layer 2 and the application layer are all standardized. Profibus distinguishes between confirmed and unconfirmed services, allowing process communication, broadcast and multi-tasking. Multiple masters are possible in Profibus, but the outputs of any device can only be assigned to one master. There is no power in the bus physical layer distribution [23].

3.2.3 Foundation Fieldbus

Foundation Fieldbus has finally come into its own and is rapidly establishing itself as the future standard for process industry networking. Since its introduction in 1997, many distributed control system vendors have embraced the protocol, developing and certifying devices that conform to its specifications.

Fieldbus is a flexible, sophisticated protocol with many capabilities. It holds great appeal, because it is intrinsically safe and provides an integrated device level device-level/plantlevel approach. Broader adoption of Foundation Fieldbus has been slowed by

the protocols process-industry-centric nature, the limited availability of compatible devices, and the slow process of standardization. The Foundation Fieldbus standard is typically used in distributed control, continuous process control, batching and oil processing operations [23].

3.2.4 DeviceNet

DeviceNet is designed to connect simple devices from multiple vendors that comply with the DeviceNet network standards. DeviceNet device profile standards provide interoperability between device manufacturers.

Each DeviceNet segment can connect up to 64 devices. It is a four-wire system capable of delivering 8 Amps at 24VDC, sufficient for field devices such as solenoid valves. The four wires carry signal and power typically on a single cable. Multiple power supplies can be used for redundancy and additional power requirements.

DeviceNet uses a trunk (bus) line with drop cables to connect to devices. The trunkline requires 121 ohm terminating resistors at each end of the trunk.

DeviceNet supports Master/Slave, Peer-to-Peer, and Multi-Master network models. Data can be transferred on a cyclic or change of state basis using a Producer/Consumer paradigm that conserves network bandwidth. DeviceNet is very commonly used for communications from host systems to motor control centers and variable speed drives [23].

3.3 Summary

Some of the industrial bus standards have been discussed in Section 3.2, the selection of the exact protocol to depends on particular industrial application, number of devices, integrability with the future equipment and response time. In this project Modbus protocol is considered due to the advantages mentioned above (Section 3.2.1), and also the following prime considerations:

1. Modbus is well accepted and well understood by many in the world of industrial communications
2. Well documented and openness

3. Seen to be an ideal vehicle to improve functionality by porting it to a wireless implementation

Appendx E shows the comparison table of different industrial protocols.

Chapter 4

Modbus Protocol

4.1 What is Modbus?

Modbus is an application layer protocol, positioned at Level 2 of the OSI Model, that provides master/slave communication between devices connected on different types of buses or networks.

Modbus was developed by MODICON Inc. in 1979 [10]. It is a standard which is truly open and the most widely used network communication protocol in the industrial automation field. SCADA and HMI software can easily integrate serial devices via Modbus protocol.

4.2 When do we use Modbus?

The use of Modbus is recommended when:

1. When a facility has to be implemented in accordance with a well supported standard.
2. Device populations are primarily discrete.
3. End users already have an existing control system that support Modbus.
4. End users have legacy control systems that do not support other common protocols

Using Modbus also offer the following benefits, are factors made Modbus to be chosen among other protocols:

1. Openness, no license fees.
2. Widely supported by most SCADA and HMI software.
3. Easy to use.
4. Low development cost.
5. Wide knowledge resource.

4.3 How does Modbus device communicate with other devices?

Modbus devices communicate using a master/slave technique in which only one device (the master) initiates transactions (called queries). The other devices (slaves) respond by supplying the requested data to the master, or by taking the action requested in the query. A slave can be any peripheral device (I/O transducer, value, network drive, or other measuring device), which processes the information and sends its output to the master using Modbus Protocol.

Master can address individual slaves, or can initiate a broadcast message to all slaves. Slaves return a response to all queries addressed to them individually, but do not respond to broadcast queries.

4.3.1 Physical Interconnection Standards

4.3.1 (a) RS232

RS232 is a very old standard which still persists as it is very popular and generally adequate. It specifies the physical interconnection between just two devices for pointto-point serial communications over a distance of up to 10m. Its separate send and receive wiring paths generally allow simultaneous communications in both directions (full duplex).

4.3.1 (b) RS485

Modern networks require that any device be able to send to any other device. For any SEND output to be able to access any RECEIVE input, all SEND outputs and all RECEIVE inputs must be connected together electrically, and all the SEND outputs must be able to be switched on and off on demand, leaving one (the current transmitter) switched on. This is the basis of a common party-line network, including the coaxial Ethernet networks used to interconnect office PCs. Some are half-duplex because only one device can send at a time, but some support full-duplex operation. The RS485 standard generally supports half-duplex networking, and is widely used in industry, commerce and elsewhere, as the balanced format provides a reliable physical connection over long distances.

4.4 Modbus serial transmission mode

The transmission mode defines the bit content of the message bytes transmitted along the network, and how message information is to be packed and decoded.

Standard Modbus Networks employ two types of transmission mode:

1. ASCII Mode
2. RTU Mode

The mode of transmission is usually selected along with other serial port communication parameters (baudrate, parity, stop bits, etc.) as part of the device configuration.

4.4.1 ASCII Transmission Mode

In ASCII Transmission mode, each character byte is sent as two ASCII characters. This mode allows time intervals of up to a second between characters during transmission without generating errors.

The format of each byte in ASCII transmission mode is outlined below:

Coding System: Hexadecimal of ASCII characters 0-9 and A-F. One hexadecimal character contains 4-bits of data within each ASCII character of message [1].

Bits per Byte:
1 Start bit, and
7 data bits, with least significant bit sent first.
1 bit for Odd/Even parity or no bit for no parity.
1 Stop bit if parity is used or 2 bits with no parity.

Error Check Field :
Longitudinal Redundancy Check (LRC) this field check the message frame for errors.

4.4.2 RTU Transmission Mode

When controllers are set up to communicate on a Modbus network using RTU (Remote Terminal unit) mode, each 8-bit byte in a message contains two 4-bit hexadecimal characters(0-9, A-F Hex). The main advantage of this mode is that its greater character density allows better data throughput than ASCII for the same baud rate. Each message must be transmitted in a continuous stream [1, 7].

The format of each byte in ASCII mode is outline below:

Coding System:
8-Bit binary, hexadecimal 0-9, A-F Two hexadecimal characters contained in each 8-bit field of the message.

Bits per Byte:
1 Start bit, and
8 data bis, with least significant bit sent first.
1 bit for Even/Odd, no bit for no parity.
1 Stop bit if parity is used, 2 bits if no parity.

Error Check Field:
Cyclic Redundancy Check (CRC) this field chek the entire message frame for errors by calculating CRC value and append to a message or compare with the received CRC value.

4.5 Modbus Message Framing

In both RTU and ASCII mode the Modbus Transmission modes (RTU or ASCII), a message is placed by a transmission device into a frame with a known beginning and ending point. This allows the receiving device to begin at the start of the message, read the address portion and determine which device is addressed, and to know when the message is completed. Partial messages and errors can be detected as a result.

4.5.1 ASCII Framing

In ASCII mode, each message start with a colon ':' character (ASCII 3A Hex) and end with a carriage return-line feed CRLF pair (ASCII 0D and 0A Hex). The allowable characters transmitted for all other fields are hexadecimal 0-9, A-F. The network device monitor the network bus continuously for the colon character. When a colon is received, each device decodes the next field (address field) to find out if it is the addressed device [1, 7].

Intervals of up to a one second can elapse between characters within the message. If a greater interval occurs, the receiving device assumes an error has occurred.

A typical Modbus ASCII message frame is shown in Figure 4.1 (Each field will be discussed in the next sections):

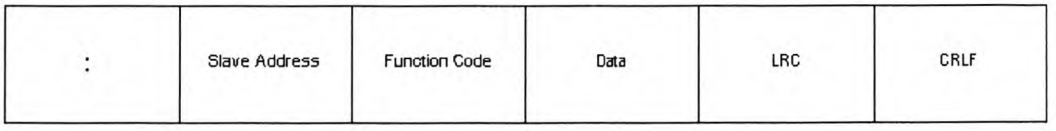


Figure 4.1: General Modbus ASCII Message Frame

4.5.2 RTU Framing

In RTU mode the messages starts with a silent interval of at least 3.5 character times. This is easily implemented as a multiple of character times at the baud rate that is used on the network (represented as 3.5 character time in Figure 4.2). The first field transmitted is the device address.

The entire message frame must be transmitted as a continuous stream of characters. If a silent interval of more than 1.5 character time occurs between two characters, the message frame is declared incomplete and it is discarded by the receiver which assume that the next byte will be the address field of a new message.

Similarly, if a new message begins earlier than 3.5 character times following a previous message, the receiving device will consider it a continuation of the previous message. This will set an error, as the value of CRC field will not be valid for the combined messages [1, 7].

A typical Modbus RTU Message Frame is shown in Figure 4.2 (Each message field will be discussed in the next section):

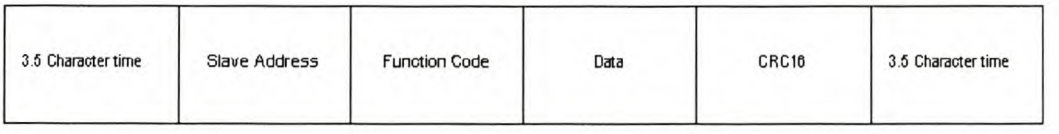


Figure 4.2: General Modbus RTU Message Frame

4.5.3 Address Field

The address field of a message frame contains two characters (ASCII) or 8-bits (RTU). Valid slave addresses are in a range of 0 - 247 decimal. The individual slave devices are assigned addresses in the range of 1 - 247, and 0 is reserved for a broadcast message which all the slave devices on the network recognizes. A master addresses a slave by placing a slave address in the address field of the message. When the slave sends its response it places its own address in the address field of the response of the message to let the master know which slave is responding.

4.5.4 Modbus Function Field

The function code field of a message frame contains two characters(ASCII) or 8-bits (RTU). Valid codes ranges from 1 to 255 decimal. Out of these function codes some codes are applicable to all controllers while some function codes apply to certain models of controllers (PLC, RTU) and others are reserved for future use.

When a message is sent from a master to a slave device the function code field tells the slave what kind of action to perform. Examples are:

- To read ON/OFF states of a group of discrete coils or inputs.
- To read the data contents of a group of registers.
- To read the diagnostic status of the slave.
- To write to designated coils or registers, or allow loading or verifying the program within the slave.

When the slave responds to the master, it uses the function code field to indicate either normal (error-free) response or that some kind of error occurred (called an exception response). For a normal response the slave simply echoes the original function code. For exception response, the returns a code that is equivalent to the original function code with its most-significant bit set to a logic 1. Some of the Modbus Function codes are described in the next chapter.

4.5.5 Modbus Data Field

The data field is constructed using sets of two hexadecimal characters in the range 00 to FF hexadecimal.

The data field of messages from a master to slave devices contains additional information which the slave must use to take action defined by the function code. This can include things like discrete and register addresses, the quantity of items to be handled and the count of the actual data bytes in the field.

4.5.6 Modbus Error Checking Field

Modbus networks employs two methods of error checking:

1. Parity checking of the data character frame (even, odd, or no parity)
2. Frame Checking within the message frame (Cyclical Redundancy Check in RTU Mode, or Longitudinal Redundancy Check in ASCII Mode).

4.5.6 (a) Parity Checking

A Modbus device can be configured for even or odd parity, or no parity as discussed in Section 4.4. This determines how the parity of the bit of the characterss data frame is set.

4.5.6 (b) Frame checking

The error checking field contents depend upon the Modbus transmission mode that is being used.

ASCII
When ASCII mode is used for character framing, the error checking field contains two ASCII characters. The error check characters are the result of a longitudinal Redundancy Check (LRC) calculation that is performed on the message contents, exclusive of the beginning colon and terminating CRLF characters. The LRC characters are appended to the message as the last field preceding CRLF characters. Appendix 6 explains how the LRC value is appended to the message.

RTU
When RTU mode is used for character framing, error checking field contains a 16-bit (2 bytes) value implemented as two 8-bit bytes. The error check value is the result of a Cyclical Redundancy Check (CRC) calculation performed on the message contents. The CRC Field is appended to the message as the last field in the message. When this is done, the low order byte of the field is appended first, followed by the high order byte. The CRC high order byte is the last byte to be sent in the message.

4.6 Modbus Functions Format

Data Addresses [1] are used in Modbus query messages when reading or modifying data. Four types of data are used: Coil, Input Status, Input Register and Holding Register.

[1]The Addresses differ from controller to controller, in this chapter the Address range will be referred to the one of Modicon Controllers (PLCs)

4.6.1 Coil

Coils are used to the ON/OFF state of discrete outputs to the Function Field, or to modify the status of slave devices (Read-Write). Coil data is either ON or OFF, which can be both read and modified. Valid address are in a range of 1 - 9999.

4.6.2 Input Status

Input Status is used to read the ON/OFF state of discrete inputs from the field, or the status of slave devices. The input status is either ON or OFF, which can be read-only. Valid address in the range of 10001 - 1999.

4.6.3 Input Register

Input registers are used for the value of analogue inputs from the information of the slave devices. The input register is 16 bits long, which can be read only. Valid addresses are in the range 30001 - 39999. Floating point or double-floating point data can be handled when the consecutive addresses are assigned.

4.6.4 Holding Register

Holding registers are used for the value of analogue outputs to the field, or set the information of slave devices. A holding register is 16-bits long, which can be both read or modified (read-write). Valid addresses are in the range of 40001 - 49999. Floating point or double-floating point data can be handled when consecutive address are assigned.

4.7 Modbus Function Codes

The function field of the Modbus message frame will contain two characters (in ASCII Mode), or 8 binary bits (in RTU Mode) that tells the slave (PLCs) what kind of action to take. Valid function codes are from 1 - 255, but not all codes are supported by all PLCs e.g LG G7M DR10A, and some codes are reserved for future use. Table 4.1 shows

the function codes supported by the PLC module as utilised in this project. The detailed description of each Function code is discussed in the implementation chapter.

CODE	FUNCTION CODE	NAME RAMARKS
01	Read Coil Status	Read bits (output)
02	Read Input Status	Read bits (input)
03	Read Holding Registers	Read Words (output)
04	Read Input Registers	Read Words (input)
05	Force Single Coil	Write bit (output)
06	Preset Single Register	Write word (output)
15	Force Multiple Coils	Write words (output)
16	Preset Multiple Registers	Write words (output)

Table 4.1: Modbus Function Codes as Supported by LG G7M DR10A PLC [1]

When the slave device responds to the master, it uses the function code field to indicate either a normal (error free) response, or that some kind of error has occurred (an exception response). Modbus exception codes are discussed in the next section.

4.8 Modbus Exception Responses

When a master station sends a request to a remote slave device it expects a normal response. One of four possible events can occur from the masters query:

- If the slave device receives the request without a communication error and can handle the query normally, it returns a normal response.
- If the slave device does not receive the request due to a communication error, no response is returned. The master program will eventually process a time out condition for request.
- If the slave device receives the request, but detects a communication error (parity, LRC, CRC), no response is returned. The master program will eventually process a time out condition for the request.
- If the slave device receives the request without a communication error, but cannot handle it (for example, if the request is to read a non-existent output or register), the slave device will return an exception response informing the master program of the nature of error.

A list of possible exception codes which are supported by the LG G7M PLC are shown in Table 4.2.

Code	Error Type	Meaning
01	Illegal Function	The function code received in the query is invalid or not allowed
02	Illegal Data Address	Error when exceeding the area limit of reading/writing on the slave station
03	Illegal Data Value	Error when the data value to be read from or write on the slave is not allowed
04	Slave Device Failure	Error status of the slave device
05	Acknowledge	The slave is engaged in processing a long duration program command. The master should retransmit the message later when the slave is free
06	Slave Device Busy	Error when request command processing takes too much time. The master should request again
07	Time Out	Error when processing time exceeds the time limit of the communication parameter as it communicates
08	Number Error	Error when data length is 0 or more than 256 bytes, when the data size is bigger than the allowed size.
09	Parameter Error	Error of setting parameters (Mode, Master/Slave).

Table 4.2: Modbus Exception Codes [10]

The exception response has two fields that differentiate it from a normal response:

Function code:
In a normal response, the slave echoes the function code of the original query in the function field of the response. All the function codes have the most-significant bit (MSB) of 0 (their values are all below 80 hexadecimal). In an exception response, the slave sets the MSB of the function code to 1. This makes the function code value in an exception response exactly 80 hexadecimal higher than the value would be for normal response.

With the function codes most significant bit set, the masters application program can recognize the exception response and examine the data of the function field.

Data Field:
In a normal response, the slave may return data or statistics in the data field (any information that was requested in a query). In an exception response, the slave returns an exception code in the data field. This defines the slave condition that caused the

exception.

4.9 Summary

Modbus function codes on the function field will be discussed in the next chapter. The two Modbus transmission modes (ASCII and RTU) discussed above, have individual advantages. The main advantages of Modbus ASCII mode, is that it allows time intervals of up to one second to occur between characters without causing errors. Unlike the ASCII mode, the main advantage of Modbus RTU mode is that its greater character density allows better data throughput, but the message has to be transmitted in a continuous stream of characters.

Chapter 5

General Systems Design

The first challenge when faced with the implementation of radio telemetry, is the number of disciplines involved, spanning basic electrical design through to radio frequency technology. In some instances many people will be involved with such a system and will most likely include electrical process, computer(software), and radio engineers. Radio telemetry involves the relaying of voice, digital/analogue process signals, and/or serial data.

A telemetry network consists of number of different elements, each with a distinct role to play in enabling the network as a whole to meet its objectives. In the simplified case a data communications facility consists of some form of input or output unit, a communication link and a host processor. The other element to be investigated when designing a telemetry network, is the software and communication protocol.

Figure 2.1 show the basic elements of a telemetry system. Choosing hardware for the network, depends on the design specifications. A block diagram of the system design for this project, is shown in Figure 5.1. The individual components are discussed in subsequent sections.

In this project, a EL705 OEM data transceiver was chosen because of its high bandwidth and good data latency. Separate radio transceivers and modems have their own disadvantages as mentioned in Section 2.3.3 [15].

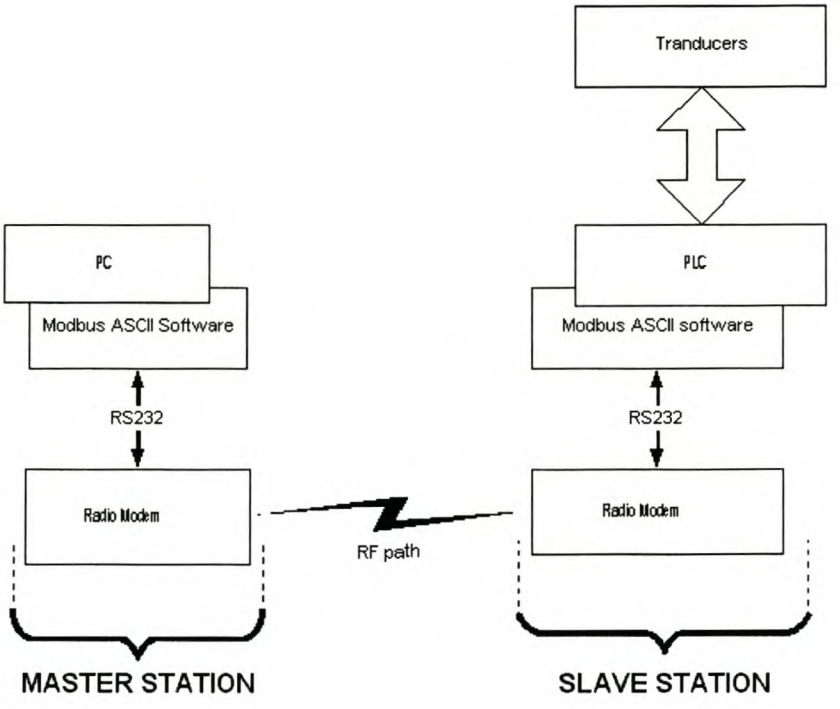


Figure 5.1: General System Design

5.1 Hardware consideration

In real situations, the majority of data communications systems employ mixed hardware, because two major elements of the system (communication link and the modems) are often supplied by different vendors. There are some things to consider when choosing hardware from different vendors, such as economical factors, technical performance, user base and miscellaneous other practical issues. By using wireless links, system expansion is easier without affecting other stations connected to the same network.

5.1.1 Wireless Data Transceiver

In this project, Modbus serial communications will be implemented over radio links. The MDS EL705 OEM Data Transceiver is a secure radio data modem used for a variety of applications, including process monitoring and control. The use of radio for Modbus ASCII wireless telemetry can permit a more cost effective and flexible solution than with cabling. Combining the flexibility of the radio data transceiver with many devices sup-

porting RS232 communications on Modbus protocol allows the construction of a system with intelligence distributed to plant level, without depending upon RS232 cables.

The MDS EL705 OEM Data Transceiver was chosen because low latency (< 15ms), operation in a point-to-point, or point-to-multipoint environment, provision of reliable communications even in adverse conditions, high bandwidth, and over the air baudrate of up to 9600 bps. The MDS EL705 OEM data transceiver use CPFSK modulation [15].

Other factors to consider when choosing hardware for telemetry network are:

- What is the network used for, and which communication protocol it is using?
- Message response time - Should the data be in a real time or not?
- How many messages required when sending read and write command to a Device (PLC)
- What is the maximum allowed length of a transmission string.
- The hardware and maintenance cost.

5.1.2 Programmable Logic Controller

PLC's are used to replace the necessary sequential relay circuits for machine and other controls. The PLC works by looking at its input and depending upon their state, exercising control over outputs. The PLC is pre-programmed to get the desired results. PLCs eliminate the cost of wiring and replace complicated relay based machine control. It also reduces downtime. By choosing the right PLC, one can communicate with any other PLC supporting Modbus communication.

Automatically controlled equipment yields more consistent quality than manually operated equipment. PLCs are the cornerstone of automation nowadays. PLC are fast, flexible and highly accurate devices that perform a chain of commands. These general purpose controls accept sensor inputs, evaluate them according to users program, generate outputs for different types of actuators and operator interface devices. Their microprocessor technology allows them to communicate with each other and with PC based SCADA and monitoring systems [18].

Due to economy, availability and excellent built-in features, the LG G7M DR10A PLC was selected for this project F.

5.2 Software Considerations

5.2.1 Modbus ASCII Master Software

In this project a network is established by interfacing a PC with a PLC, through a radio link. The Modbus ASCII Protocol is used, which is implemented using Borland Delphi 5. Borland Delphi was chosen for this project because it is object oriented, resulting in easy to build user interfaces.

The master station configuration is shown in Figure 5.1. The Modbus ASCII driver code for the master station, was also developed in Delphi 5.

Since Modbus is master/slave protocol, the system requires master and slave based software. The LG G7M DR10A PLC has built-in Modbus software, which only needs assigning register addresses for different Modbus functions as explained in the next section.

The network is controlled by any Modbus Master station, and it is responsible for polling each separate node in turn for, its response. A Comport component written by Dejan was used to conveniently interface developed software with the COMM port driver. This component is available for free on the internet.

The master(PC)station is designed such that it will be able to:

1. Prepare Modbus ASCII messages for transmission, adding appropriate control and address characters.
2. Initiate the transmission.
3. Accept acknowledgment (response) from the remote station.
4. Retransmit messages which are received with errors or not received.

The flow chart as per Figure 5.2 below, indicate howModbus Query-Response transactions in the Master station are handled:

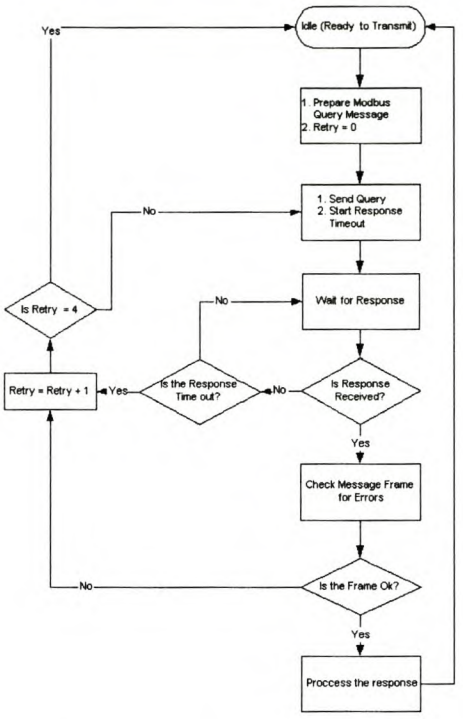


Figure 5.2: Master Station Query/Reasponse Transmission Flow Diagram

Flow Diagram explanation:

- Idle state is the normal state where there is no transaction taking place.
- When the query message is sent to the slave device the response timeout is started.
- When a reply is received, the master check the reply for error before processing the data. The frame is checked by comparing the calculated LRC with the received LRC. If the checking results in error, for example reply from an unexpected slave, or an error in the received frame then, if the response is from unexpected slave timeout keep on running, when timeout expires the master retry until number of retries are finished.
- If no reply is received, the response timeout expires, the master retries sending the request to a maximum of 4 retries.
- If the master retry 4 times, without succeeding the master returns to idle state.
- If the reply message comes without error, the master process the message and displays the response.

5.2.2 Slave (PLC)

Like PCs, PLCs can be programmed in many languages, although the most popular PLC language is still the traditional Ladder Logic, which allows the programmer to simulate a collection of relays, timers, etc. PCs can handle larger volumes of data, but for many reasons (mostly to do with their Microsoft operating systems), timing is imprecise and communications are inefficient. A PC/SCADA is ideal to create historical logs, display plant diagrams, values, statuses and trends. A PLC is used for all the logic, interlocking, safety systems and general control of industrial plant, and usually, to report back an overall picture to the PC/SCADA system. The programming of LG G7M DR10A PLC, is covered in Chapter 6.

LG G7M DR10A PLC has a built-in Modbus functionality which allows them to communicate with any network device that supports Modbus functions. The PLC has to be configured for Modbus ASCII transmission mode and other parameters, such as baudrate, slave station number, etc., as shown in Figure 6.3.

The basic steps followed by the slave station (LG PLC) are in such a way that they will be able to:

1. Receive a Modbus query from the Master i.e. it must be able to detect the start : and end CRLF of message frame
2. Assemble characters of the message into messages
3. Check message for errors and discard the message if error is detected
4. Prepare Modbus ASCII response message and send it to the Master

The flow chart indicating the data flow on the slave device when the query has been received is shown in Figure 5.3.

Explanation of flow diagram is as follows:

- The slave device waits for Modbus ASCII message indication and receives the message until the end of message frame is detected.
- Check if the function code is valid; if valid, proceed to check data address. If function code is incorrect, send Modbus exception response 1.

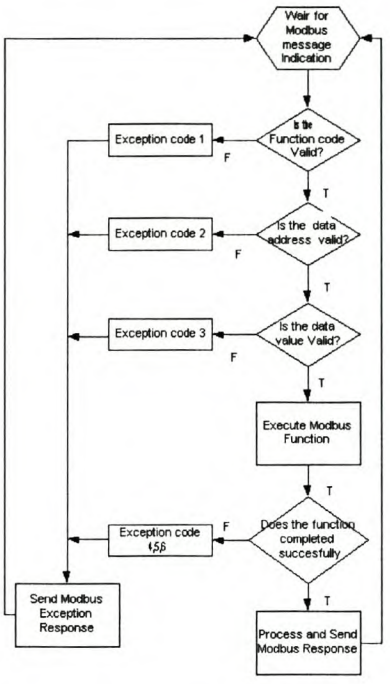


Figure 5.3: Modbus Slave Query/Response Flow Diagram

- Check the validity of data address, if valid proceed to check the data value. If the data address is invalid, send Modbus exception response 2.
- Check the the validity of data value, if valid execute the Modbus function. If the data value is invalid, send Modbus exception response 3 to the Master.
- Execute Modbus function, if completed successfully send Modbus ASCII response to the Master. If error occurs send Modbus Exception code (4 , or 5 or 6).
- After executing Modbus function and sending a response to the master, returns to the idle state and waits for the next Modbus query message.

5.3 Summary

Radio modems are usually found where the link required between two sites would normally be done with telephone modems, such as the linking of PLCs to each other (typically using MODBUS protocol) or to a computer running a SCADA software package or some other software supporting Modbus protocol. Others uses include any remote device usually

connected to a computers serial port but where the device is too remote to have a hardwire link.

Defining the radio telemetry requirements also involves ascertaining whether or not the system being planned, assembled, and installed now, is to be, in part or whole, incorporated into a larger system in the future. Using Modbus protocol over the radio link will assist with the achievement of all these objectives.

Chapter 6

Implementation

It is well known that Borland Delphi is a relatively new language which offers convenient user interfacing. In this project, Modbus ASCII software was implemented in a Delphi environment running on a Windows Me platform. Different hardware components for the network were integrated. Cable connections, hardware parameter settings and software components, are discussed in the next sections.

6.1 Hardware Configuration

6.1.1 Cable Connections

6.1.1 (a) PLC Cable Connections

PLC implemented in this project use two different RS232, one for programming the PLC and the other for communications with other devices. Cables connections are discussed below.

PLC programming cable

In order to set all the internal Modbus and communications parameters, the PLC is connected to a PC running GMWIN programming software and programmed using cable connections as shown in Table 6.1 below. PLC programming and internal configuration is discussed in Section 6.2.

PC end (DB9 Female)	PLC end (DB9 Male)
2	3
3	2
5	5

Table 6.1: RS232 Cable Connections for PLC Programming

PLC Cable for Modbus communications

When PLC is finally programmed, it is then connected to other devices using RS232 cables connections a shown in Figure B.1.

6.1.1 (b) PC to EL705 OEM Data Transceiver

The RS232 cable used for communication between the PLC and Radio data Modem is shown in Table 6.2.

PC side (DB9 Female)	MDS EL705 (DB25 Male)
RXD - 2	3 - RXD
TXD - 3	2 - TXD
GND - 5	7 - GND

Table 6.2: RS232 Connection For PLC To Modem

6.1.1 (c) PLC Input/Output Connections

The PLC was connected to switches, variable resistors, and LEDs as shown in Figure 6.1 for simulation purpose. Section 6.2 explains how the inputs/outputs of the PLC are assigned.

6.1.2 Overall system configuration

The MDS EL705 OEM Data Transceiver and PLC communications parameter settings were set as follows:
Baudrate : 1200 bps
Parity : Even

Data bits : 7
Stop bits : 1

The Data Transceiver set up can be found on **'EL705 OEM Series installation and operation guide'** manual.

The overall system hardware configuration for point-point connection is connected as in Figure 6.1.

Figure 6.1: Overrall System Configuration

Cable connections between the device were as discussed above, i.e the connections in Table 6.2 for connecting PC and MDS EL705 data transceiver, and Figure B.1 is used to connect PLC to MDS EL705 data transceiver.

6.2 Slave (LG PLC GLOFA GM7)

This section explains and give examples to setup the communication parameters and internal setup for the PLC, as well as the implementation of the A/D extension. The setup is done to achieve Modbus ASCII communication to the PLC with any other Modbus software.

6.2.1 GMWIN and Communication

The GMWIN software supplied with the PLC is used to program the PLC and set up communication parameters. Once loaded, a new project can be created from scratch, or a current project on the PLC can be downloaded to the PC and saved as a new project/program.

Using a RS232 cable as described in Table 7.2, Modbus communication parameters are set as shown in Figure 7.2:

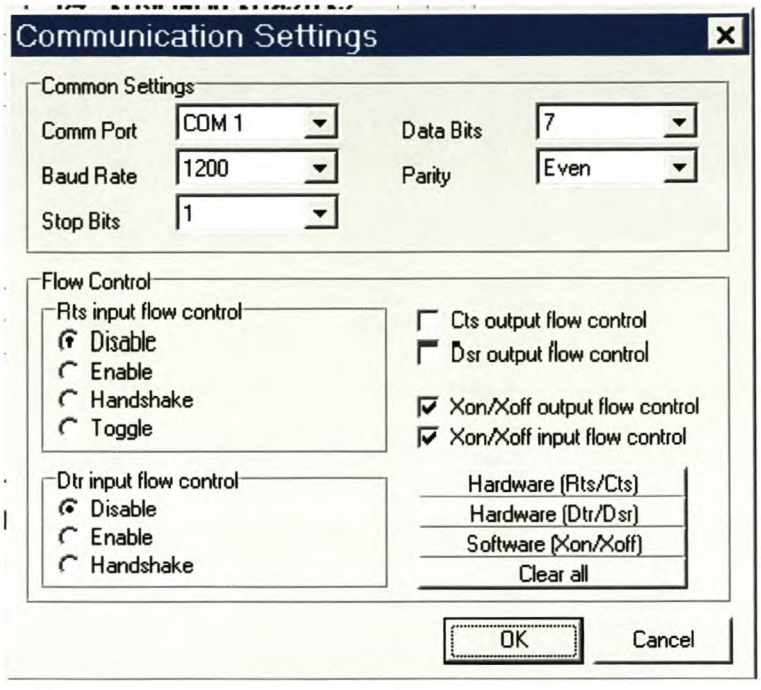


Figure 6.2: Communication Parameters Dialog

Every input, output, or memory register is addressed in a specific way to link contacts, coils or function block parameters with specific parts of the PLC. Since GM7 PLC doesnt have any division between input and output areas like Modicon PLCs, when it supports Modbus protocol communication. It only uses the M area. The address for inputs and outputs has to be assigned, and for this project addresses were assigned according to function codes as in Table 6.3.

Function code format	GM7 Memory Address	Modicon
Input coils	%MX0000 - %MX0999	0xxxx
Output coils	%MX1000 - %MX1999	1xxxx
Input registers	%MW0300 - %MW0399	3xxxx
Output registers	%MW0400 - %MW0499	4xxxx

Table 6.3: Memory allocation for Input/Output area for Modbus protocol communication

The addressing in Table 6.3 works as follows:
Symbols: I - Input, Q - Output, M - Memory, X - Bit, B - Byte, W - Word.
Format: %(I/Q/M;X/B/W/...;line.module.channel)

Examples:
%IX0.0.2 //Input Bit of the first line , first module, channel 02
%QX0.0.0 //Output Bit of first line, first module, channel 00
%MW1000 //Data Word Stored at memory address 1000

6.2.1 (a) Reading Digital Inputs

The inputs receive data saved in the M area through the input contact coil. The input are read by Moving input value (Input0 %IB0.0.0) to memory adddress defined by Digi Input0 (%MB125)(See attached disk for a Complete PLC program).

6.2.1 (b) Setting Digital Outputs

Digital outputs are set by reading the value of internal register (%MW0), and writing the result (which is boolean) to the output register (%QW0.0.0).

6.2.1 (c) Reading Analogue Inputs

Analogue inputs are read by using the function block (ADHA RD) from the A/D modules library, and storing the output to the variables WANA INPUT (%MW300) for analogue input 0 and WANA INPUT0 (%MW301) for analogue input 1. This value can be read by a Modbus command, scaled and used. (0 - 10 V) equals (0-4000) in numerical input values, thus the scaling is 0.0025 for converting memory value to Voltage.

Complete PLC program attached on a CD.

6.3 Modbus ASCII Master Software

6.3.1 Serial Communication Driver

A serial communication driver (TComport)from Dejan to establish serial communication between PC and other serial devices was used. The default Comm Port parameters was COM1, 1200 baud, 7 data bits, 1 stop bits, even parity and no hardware handshaking .

The Modbus ASCII communication driver was implemented in Borland Delphi 5 with the assistance of TComport, the implementation of Modbus function codes for the driver is discussed in the next section.

6.3.2 LRC Implementation

In Modbus ASCII, the message include error-checking field that is based on LRC method, as discussed in Chapter 4. The LRC fields checks the contents of the message, exclusive of the beginning colon and ending CRLF pair, and it is applied regardless of any parity check used for the individual characters of the message.

For query message LRC value is calculated by the sending device, which is then appended to the message. Appendix A explains how LRC is appended to the message. When a response message from the PLC is received LRC of the received message is calculated and compared with the actual value received in the LRC field. If the two values are not equal, an error result.

In this project LRC is implemented using Delphi 5 as shown in the function below, the function input is a message excluding colon, CRLF, LRC Hi and LRC Lo field in ASCII format(hex). The output of the function is the LRC value (LRC Hi and LRC Lo) in hex.

```
function TReadQuery.LRC(Str: String): String;
var
CalLRC : Byte; //Calculated LRC Value;
AdByte : Byte; //Variable to sum all the byte in the message;
i : Byte; //byte position ;
begin
AdByte := 0;
i := 1;
{keep on adding byte until the message is completed}
while i <= Length(Str) do
begin
{Add bytes}
AdByte := AdByte + StrToInt($ + (Str[i]+ Str[i+1]));
{Go to the next byte}
i := i + 2;
```

```
end;
{Subtract byte sum value from 255 to produce 1s complement,
and add 1 to produce two complement}
CalLRC := (255 - AdByte) + 1 ;
Result := IntToHex(CalLRC,2); //Return the results in hex format;
end;
```

Str parameter is the Modbus frame in hexadecimal. The output of the function is the LRC value in hexadecimal format.

6.3.3 Function Codes Implementation

A Modbus Driver in Delphi was implemented in Delphi by building a class TQuery in Delphi as per TQuery in TX_QUERY.pas. The TQuery class prepare the Modbus query message for transmission. Another class TResponse in Main Form.pas, to process the response from the slave device was implemented. For full source code see the attached CD.

The following sections explains how each Modbus functions defined in Table 4.1, as supported by LG G7M DR10A PLCs. The function codes implemented, are also discussed.

6.3.3 (a) Read coils (01 hex)

This function creates and initializes a Modbus transaction to read the ON/OFF state status of the discrete outputs or coils (Address 0000 - 0999) in the PLC. For the PLC, the response is equivalent to reading the ON/OFF status of solid state relays or switches. Broadcast transmission is not supported, i.e, Slave ID of zero is not supported.

Query
Delphi function below shows the implemented read coils function.

```
function ReadCoilsStatus(SlaveAddress,Address,Quantity : Integer):String;
```

The SlaveAddress parameter specifies the Modbus device to be read, Address parameter specifies the coil address at which the reading is to be started, and Quantity specifies the

number of coils to be read. The ReadCoilsStatus function return value is a Modbus ASCII query message frame.

The Read Coil Status query specifies the starting coil and the quantity of the coils to be read. Coils correspond to the discrete solid-state relays of the PLC and are addressed starting from 0.

Response

When PLC receive a query message from the master, it reads the output status of the contacts coils, and sends the response back to the master. The Delphi function below processes the response message and displays the results in a graphical form. If the response message is in error, it retransmits the same message.

```
function ReadCoils(Start,Quantity,Bite_Count,Data:String):String;
```

The Start parameter specifies the starting coil read, Quantity parameters specifies number of coils read, Bite_count parameter specifies number of bytes read, i.e if 8 coils are read then the number of bytes is one, Data parameter is the coils status

An example of query-response messages to read coils 0 - 15 from a slave device 1, is shown in Table 6.4. This command reads the digital output status of the PLC (Figure D.2)

Query		Response	
Field Name	ASCII characters	Field Name	ASCII characters
Header	:	Header	:
Slave Address	01	Slave Address	01
Function	01	Function	01
Starting Address Hi	00	Byte Count	02
Starting Address Lo	00	Output status 07-00	95
Quantity of Outputs Hi	00	Output Status 15-08	00
Quantity of Outputs Lo	10	LRC	67
LRC	EE	Tail	CRLF
Tail	CRLF		

Table 6.4: Read Coils Query/Response Example

The data on Table 6.4 can be interpreted as follows, digital output coils status are as follows: coil 7 - 0 is 1001 0101 - 95 hex (coil 7 is ON, coil 6 is OFF, coil 5 is OFF, ..., coil 1 is OFF and coil 0 is ON), the same argument also applies for the coils 15 - 08 is 0000 0000 - 0 hex, where 1 indicate ON and 0 indicate OFF status. With reference to Figure

6.1 the response is interpreted as Coil 7 -0 :LED1 is ON, LED2 is OFF, LED3 is ON and LED4 is OFF as per Figure 6.1; Coil 15 - 8 are not mapped to any LED for this project.

Figure 6.3 shows the query/response message, and response interpretation in a graphical form.

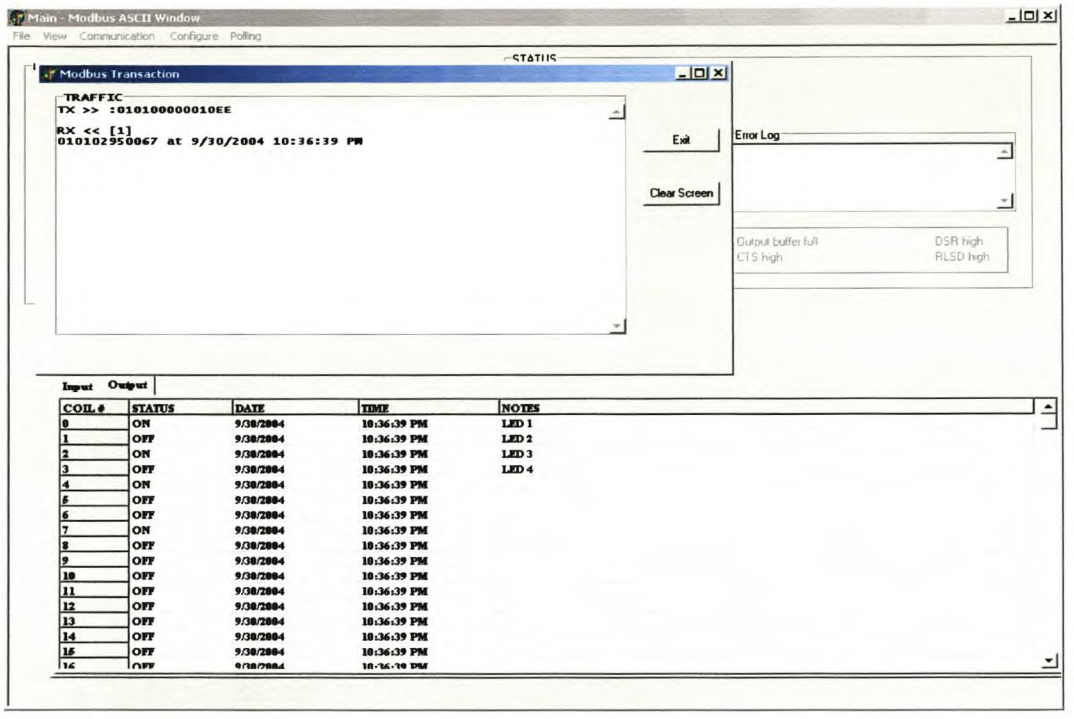


Figure 6.3: Window of Query/Response of Read Coils Function Code

6.3.3 (b) Read Input Status (02 hex)

This function code reads the ON/OFF status of discrete inputs (1X references) in the slave device. The broadcast message is not supported.

Query

The query message as shown by the Delphi function below (full code description on CD) was implemented. The query specifies the slave address, starting address to be read and quantity of the inputs to be read. The inputs are addressed starting at 0.

```
function ReadInputStatus(SlaveAddress,Address,Quantity : Integer):String;
```

The SlaveAddress parameter specifies the Modbus device to be read, Address parameter specifies the input coil address at which the reading is to be started, and Quantity specifies the number of input coils to be read. The ReadInputStatus function returns a Modbus ASCII message frame.

Response

The discrete inputs in the response message are packed as one input per bit of the data field. The status is indicated as 1 = ON, 0 = OFF. The LSB of the first data byte contains the input addressed in the query. The other follow toward the high order end of this byte, and from low order to high order in subsequent bytes.

When the response is received from the addressed slave Delphi function below process the received message after checking the LRC of the message, if frame is in error message is discarded and the next message awaited.

```
function ReadInputStatus(Start,Quantity,Bite_Count,Data:String):String;
```

The Start parameter specifies the starting input coil read, Quantity parameters specifies number of input coils read, Bite count parameter specifies number of bytes read, i.e if 8 coils are read then the number of bytes is one, Data parameter is the input coils status

Table 6.5 below shows a query and a response message as an example to read 16 digita input coils from 0 - 15, from the PLC addressed 1:

Request		Response	
Field Name	ASCII characters	Field Name	ASCII characters
Header	:	Header	:
Slave Address	01	Slave Address	01
Function	02	Function	02
Starting Address Hi	03	Byte Count	02
Starting Address Lo	E8	Input status 07-00	07
Quantity of Inputs Hi	00	Input Status 15-08	00
Quantity of Inputs Lo	10	LRC	F4
LRC	02	Tail	CRLF
Tail	CRLF		

Table 6.5: Read Input Status Query-Response Example

The response message is interpreted as follows: coils 07 - 00 (0000 0111) means switch1 is ON, switch2 is ON, switch3 is ON, switch4 is OFF with respect to Figure D.2 and the

other bits are not mapped to the the switches. Figure 6.4 shows how the query response and interpretation of the message looks like on the screen.

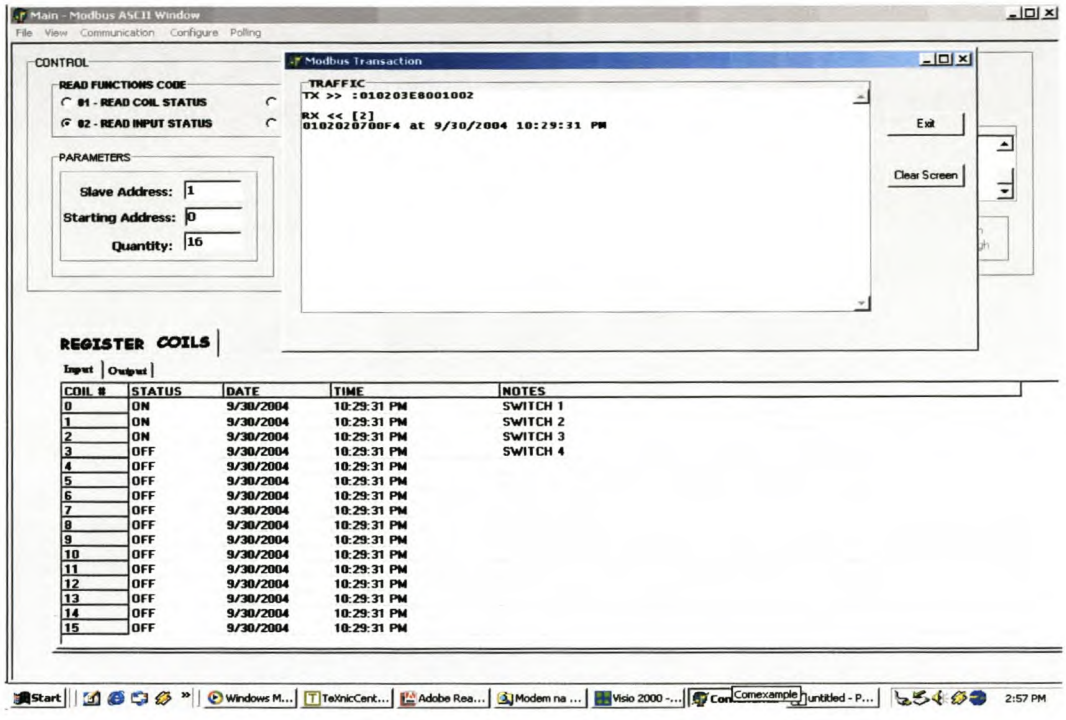


Figure 6.4: Read Input Status Query/Response Graphical Illustration

6.3.3 (c) Read Holding Register (03 hex)

This function code reads the contents of a contiguous block of holding register in a PLC. Broadcast is not supported.

Query

The Delphi function below implemented to send a query, to read holding registers(04X references) and specifies the starting register and the quantity of register to be read. Registers are addressed starting at zero.

```
function ReadHoldingRegister(SlaveAddress,Address,Quantity : Integer):String;
```

The SlaveAddress parameter specifies the Modbus device to be read, Address parameter specifies the output register address at which the reading is to be

started, and Quantity specifies the number of output register to be read. The ReadHoldingRegister function returns a Modbus ASCII query message frame.

Response

The register data in the response message is packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and second contains lower bits.

When a response message is received the master checks message for errors, if no error is detected it go to Delphi function shown below.

```
function ReadHoldingReg(Start,Quantity,Bite_Count,Data:String):String;
```

An example of a request to read output registers 400 - 402 from slave device 1, is shown in Table 6.6 below.

Request		Response	
Field Name	ASCII characters	Field Name	ASCII characters
Header	:	Header	:
Slave Address	01	Slave Address	01
Function	03	Function	03
Starting Address Hi	01	Byte Count	06
Starting Address Lo	90	Data Hi (register 400)	00
No. of points Hi	00	Data Lo (Register 400)	00
No. of Points Lo	03	Data Hi (register 401)	00
LRC	68	Data Lo (register 401)	00
Tail	CRLF	Data Hi (register 402)	00
		LRC	F6
		Tail	CRLF

Table 6.6: Read Holding Register Query-Response Example

6.3.3 (d) Read Input Registers (04 hex)

This function reads the input contents of the input registers (3X references)in the PLC. Broadcast message is not supported.

Query

The query message as in Delphi function below, specifies the slave (PLC) address, starting register and quantity of register to be read. The registers are addressed starting at zero.

```
function ReadInputRegister(SlaveAddress,Address,
Quantity :Integer):String;
```

*The SlaveAddress parameter specifies the Modbus slave device to be read, Address parameters specifies the starting input register address to be read, Quantity parameter specifies number of input register to be read.*The ReadInputRegister returns the Modbus ASCII query message to read input registers.

Response
Register data in the response message are packed as two bytes per register, with the binary contents right justified within each byte. For each register, the first byte contains the high order bits and the second byte the low order bits.

When the response is received from the PLC the master check response message for error, if no error detected the response message is processed. The Delphi function below put the data in Delphi string so that the data can be easily interpreted.

```
function ReadInputReg(Start,Quantity,Bite_Count,Data : String) : String;
```

An example of a request to read input registers 00 - 02 from slave device 1, is shown in Table 6.7: The read value of the input register of the PLC from 300 - 302 are 07E5 hex (2021), 07E6 hex (2022) and 0000 hex (0) respectively.

The response with respect to Figure PLC is interpreted as follows: register 300: 07E5 means that the input voltage across the variable resistor 1 is 5.05250 Volts, voltage of Input 2 (register 301): 07E6 is 5.05500 Volts.

Figure 6.5 shows the graphical interpretation of the query message.

6.3.3 (e) Force Single Coil (05 hex)

This function force a single output coil (0X reference) to either ON/OFF. In broadcast mode, the function forces the same coil on all the attached slave devices.

Request		Response	
Field Name	ASCII characters	Field Name	ASCII characters
Header	:	Header	:
Slave Address	01	Slave Address	01
Function	04	Function	04
Starting Address Hi	01	Byte Count	06
Starting Address Lo	2C	Data Hi (Register 300)	07
Input quantity HI	00	Data Lo (Register 300)	C7
Input quantity Lo	03	Data Hi (Register 301)	00
LRC	CB	Data Lo (Register 301)	0A
Tail	CRLF	Data Hi (Register 302)	00
		Data Lo (Register 302)	00
		LRC	1D
		Tail	CRLF

Table 6.7: Read Input Register Query-Response Example

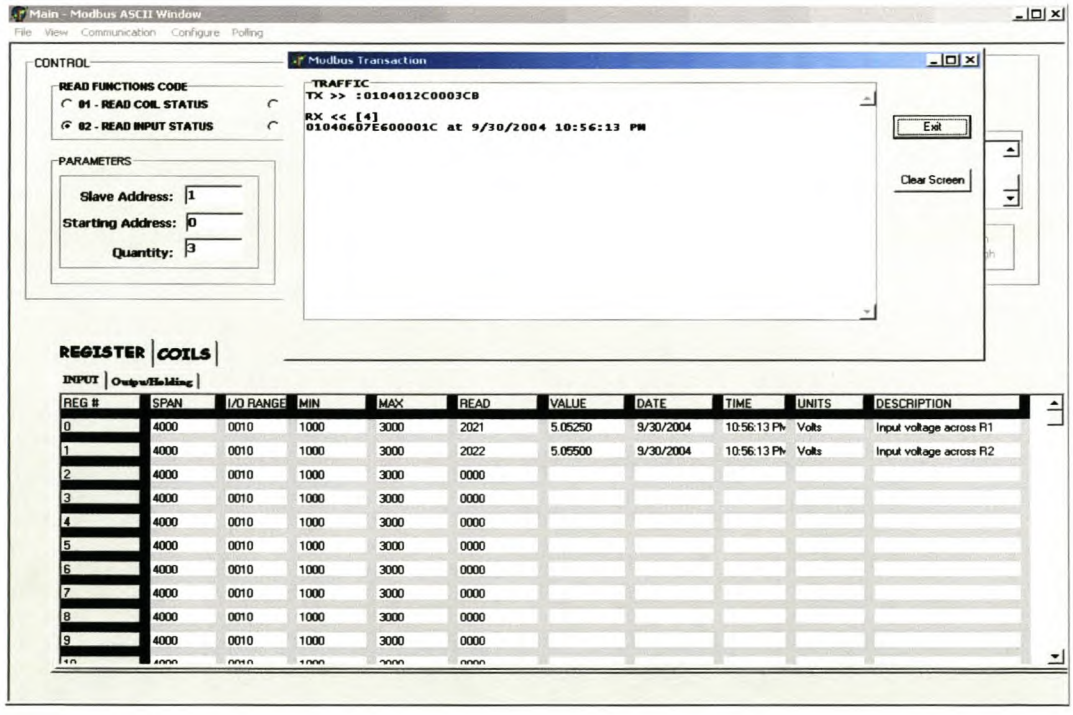


Figure 6.5: Read Input Registers Graphical Interpretation

Query

The query specifies the coil reference address to be forced. The value of FF00 hex requests the coil to be ON. A value of 0000 requests the coil to be OFF. All other values are illegal and does not change the status of the coil. Delphi function below specifies how the query parameters are set.

```
function ForceSingleCoil(SlaveAddress,Address:Integer;
OutputValue: Boolean):String;
```

The SlaveAddress parameter specifies the Modbus slave device to be read, Address parameter specifies the coil address to be forced, and the OutputValue parameter specifies whether the coil must be force to ON or OFF state, if OutputValue is true the query will force the slave device coil addressed to ON, otherwise it will set it to OFF. The ForceSingleCoil function return a Modbus ASCII query to force one output coil to either ON or OFF.

Response

A normal response is an echo of the request, returned after the coil state has been written. Table 7.6 shows an example to force output coil 0 (LED1) of the PLC to ON on a PLC addressed 1 as per Figure D.2.

Request		Response	
Field Name	Hex	Field Name	Hex
Header	:	Header	:
Slave Address	01	Slave Address	01
Function	05	Function	05
Coil Address Hi	00	Coil Address Hi	00
Coil Address Lo	00	Coil Address Lo	00
Force Data HI	00	Force Data Hi	00
Force Data Lo	FF	Force Data Lo	FF
LRC	FB	LRC	FB
Tail	CRLF	Tail	CRLF

Table 6.8: Force Single Coil Query-Response Example

6.3.3 (f) Preset Single Register (06 hex)

This function preset a single holding register (4X reference)to a specific value. In broadcast mode, the function presets the same register reference in all attached slave devices.

Query

Query specifies the register reference address to be preset, and the preset value as indicated in Delphi function below. The registers are addressed starting at 0.

```
function PresetSingleRegister(SlaveAddress,Address:Integer;
RegisterValue: Word ):String;
```

The SlaveAddress parameter specifies the slave device to write, Address parameter specifies the output register to be written, RegisterValue specifies the output register value. PresetSingleRegister return the Modbus ASCII query message to set the output register to the value specified in the Register value parameter.

An example of a request to preset output registers 400 of the PLC addressed 1 to 255 (FF hex) as per Figure D.2, is shown in the Table 6.9.

Request		Response	
Field Name	ASCII characters	Field Name	ASCII characters
Header	:	Header	:
Function	06	Function	06
Register Address Hi	01	Register Address Hi	01
Register Address Lo	90	Register Address Lo	90
Preset Data HI	00	Preset Data Hi	00
Preset Data Lo	FF	Preset Data Lo	FF
LRC	69	LRC	69
Tail	CRLF	Tail	CRLF

Table 6.9: Preset Single Register Query-Response Example

6.3.3 (g) Force Multiple Coils (0F hex)

This function code force each coil (0X reference) in a sequence of coils to either ON/OFF. For broadcast message this function forces the coil reference in all attached slaves.

Query

Query message specifies the coil reference to be forced, and coils are addressed starting at zero. The requested ON/OFF states are specified by the contents of the request data

field. A logical '1' in a bit position of the field requests the corresponding output to be ON, a logical 0 request it to be OFF.

Delphi function code below initializes and create Modbus query message for transmission.

```
function ForceMultipleCoils(SlaveAddress,Address,Quantity: Integer;
OutputValue : String): String;
```

SlaveAddress parameters specifies the slave device to read (0 - 37. For broadcast message SlaveAddress is 0. Address parameters specifies the register memory address (0000 - 0999) to read, in LG PLC the coils are addressed starting at zero. Quantity parameter specifies the number of coils to be forced. OutputValue parameters is a string of 1s and 0s. A logical 1 in a bit position of the field requests the corresponding outputs to be ON. A logical 0 request it to be OFF.

Response
A normal response to the query is the slave address, function code, starting address, quantity of coils forced and LRC. If no response is received from the slave device the same query is sent to the slave device. An error response to the query like illegal data address, is an error code, exception and LRC. When the master receive a response from the PLC, response message is displayed on the screen.

Table 6.10 shows an example of a request to write a series of 8 coils starting at coil 00 in the PLC addressed 1 (Figure D.2).

The requested data contents is one byte: 0F (0000 1111 binary) this Forces LED1 - LED4 to ON as per Figure D.2.

6.3.3 (h) Preset Multiple Register (10 hex)

This function preset values into the sequence of register in a remote device. In broadcast mode it presets the same register reference in all attached slaves.

Query
Query message specifies the slave address, starting register reference address, number of registers, and the data to be written in the ascending order as shown in Delphi function below (full code on a CD).

Request		Response	
Field Name	ASCII characters	Field Name	ASCII characters
Header	:	Header	:
Slave Address	01	Slave Address	01
Function Code	0F	Function code	0F
Coil Address Hi	00	Coil Address Hi	00
Coil Address Lo	00	Coil Address Lo	00
Quantity of coils Hi	00	Quantity of Coils Hi	00
Quantity of coils Lo	08	Quantity of Coils Lo	08
Byte Count	01	LRC	E8
Forced data(coils 7-0)	0F	Tail	CRLF
LRC	D8		
Tail	CRLF		

Table 6.10: Query/Response to Force 8 Output Coils Starting at %MX00

```
function PresetMultipleRegister(SlaveAddress, Address,Quantity: Integer;
RegisterValue:String):String;
```

SlaveAddress parameters specifies the slave device to read (0 - 37). For broadcast message SlaveAddress is 0. Address parameters specifies the output register memory address (%MW400 - 0499) to be read, in LG PLC the coils are addressed starting at zero. Quantity parameter specifies the number of coils to be forced. RegisterValue parameters is a register values packed 2 bytes per register. The function above returns a Modbus ASCII query message.

Response

A normal response to this function code returns the slave address, function code, starting reference register, and the number of registers preset, after the register contents has been set on the slave device.

Example in Table 6.11 show the query-response message for writing 2 blocks of contiguous registers in the PLC output registers memory address starting at 400 to 10 (000A hex) and 258 (0102 Hex) in a PLC addressed 1.

Request		Response	
Field Name	ASCII characters	Field Name	ASCII characters
Header	:	Header	:
Slave Address	01	Slave Address	01
Function Code	10	Function code	10
Starting Address Hi	00	Starting Address Hi	01
Starting Address Lo	00	Starting Address Lo	90
No. of Register Lo	00	No. of Register Hi	00
No. of Register Hi	02	No. of Register Lo	02
Byte Count	04	LRC	5C
Data Hi	00	Tail	CRLF
Data Lo	0A		
Data Hi	01		
Data Lo	02		
LRC	4B		
Tail	CRLF		

Table 6.11: Preset Multiple Register Query-Response

6.4 System Evaluation

Hardware were integrated as in Figure 6.1 and tested for different baudrate and different throughputs. Table 6.12 shows the maximum parameters that can be read or written to slave device.

Function	Description	Maximum Paramaters
1	Read Coil Status	984 coils
2	Read Input Status	984 coils
3	Read Holding Registers	61 registers
4	Read Input Registers	61 registers
5	Force Single Coils	1 coil
6	Preset Single Register	1 register
15	Force Multiple Coils	960 coils
16	Preset Multiple Registers	60 registers

Table 6.12: Maximum Query/Response Parameters

6.5 Timing and Delay

With all above-mentioned transactions, a time out timer is initialized upon transmission and when the response is received the timeout timer is disabled. If the timeout time Function expires before the response is received, or an error is detected, the master retransmits the same command. When broadcast mode is used, a turn around delay is started. This allows all the slave devices to process the request before receiving a new one. The overall hardware configuration is as shown in Figure 6.1.

The other factors considered for the design of this wireless network is the preamble and postamble time. The preamble time accommodate the rise time characteristics of the transmitter, while the preamble time accommodate the settling time of the receiver. If the repeaters is used , the preamble and postamble time becomes more important. The post-amble time for the modem used is 2ms. The maximum transmission plus processing time for the data rate of 1200bps at 256 bytes was 2281 ms, which gives the minimum polling time to be above 3 seconds.

6.6 Summary

The Modbus driver written in Delphi is able to do all the basics Modbus functions. The maximum number of coils that can be read per scan at the baud rate from 1200 - 4800 bps are 984 coils. The maximum registers per scan at the same baud rate is 61 registers. The maximum coils that can forced per request at the baud rate from 1200 - 4800 bps are 960 coils, and the maximum number of registers that can be preset per request at the this baud rate is 60 registers.

The software was able to perform all the function codes as supported by LG GM7 PLC, over the radio data modem. This software can be used with any other PLC supporting Modbus ASCII. For Modicon PLCs all the registers address must be set to zero, Section 7.2.2(i) explains how this is configured in the user interface. The implemetnted software can run with all PLC that supports Modbus ASCII protocol.

Chapter 7

Graphic User Interface

An important aid in telemetry and other control software is a Graphic User Interface (GUI). For this project, the GUI is designed in such a way that a person unfamiliar with the telemetry software would quickly able to asses what is exactly going on.

The user interface which is designed with Borland Delphi 5 will be discussed in this chapter.

7.1 Feature of Modbus ASCII window

The Modbus ASCII window as shown in Figure 7.1, can be utilised to perform the following:

- View the status of coils and input statuses in the slave device.
- View the value of input registers, holding registers and convert them to the correct engineering values.
- Configurable slave memory addresses.
- Configurable communications parameters.
- Configurable polling parameters.

7.2 Main Window (Menu)

The first thing that have to be considered before communication is initiated is to setup all the important parametares like baudrate, parity, PLC address and polling time.

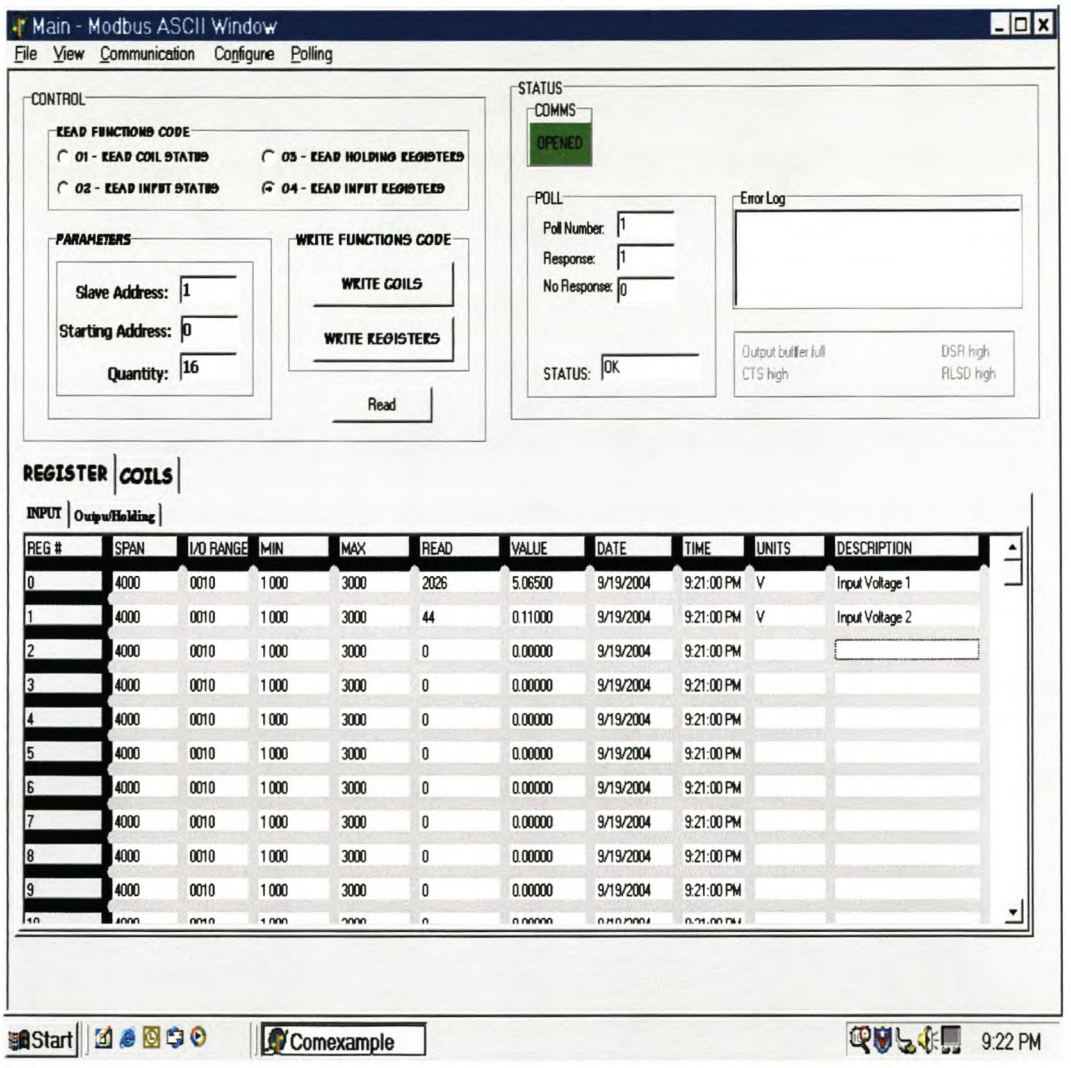


Figure 7.1: Modbus ASCII Main Window

7.2.1 Communication Menu

The communication menu is used to activate communication. By clicking on the Connect or Disconnect sub-menus, serial connection can be established. When conection is established 'COMMS' status indicator in the status panel, becomes green in color, otherwise it is red.

7.2.2 Configure Menu

7.2.2 (a) Comm. Settings

The Modbus ASCII window operates as a Master. It communicates with Modbus ASCII slaves and can be set for baudrate, data bits, parity and stop bits. These parameters can be set by clicking Configure then click Comm. Settings. The window shown as Figure 7.2, will appear.

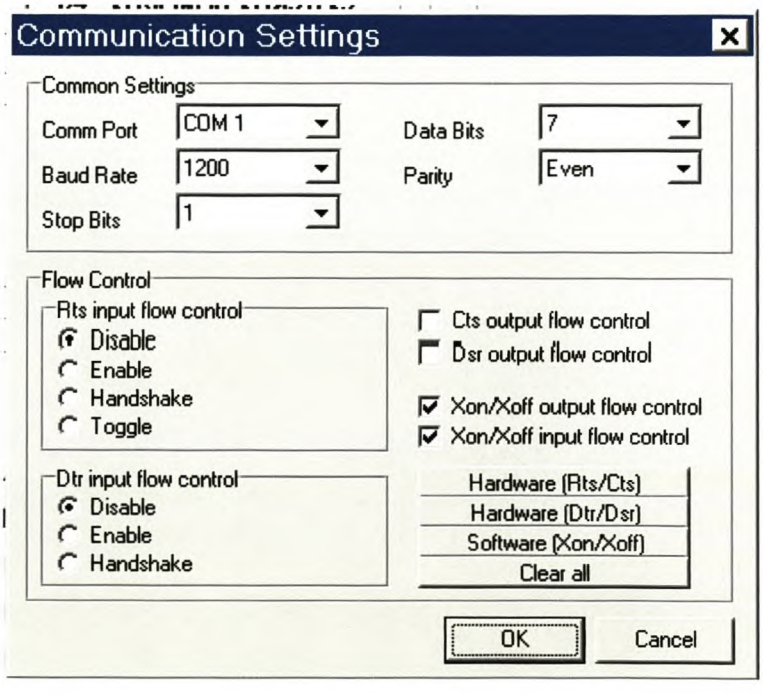


Figure 7.2: Communication Settings

7.2.2 (b) PLC Addressing

The Modbus slave (PLC) has five types of internal variables that are accessed by the Master programme:

Coils:
Coils are digital bits that can be read and written to. They are numbered from MX0000 to MX0999 [1].

Input Status:
Input status are digital bits that can be read. They are numbered from MX1000 to MX1999.

Input Registers:
Input Registers are 16 bit integers that can be read. They are numbered from MW300 to MX399.

Holding Registers:
Holding reisters are 16 bit integers that can be read and written to. They are numbered from MX400 to MX499.

The PLC memory can be configured by clicking **'Configure** and then click **'PLC Addressing**. The window in Figure 7.3 will appear:

7.2.3 Polling Menu

7.2.3 (a) Time Settings

By clicking **'Time Settings'** sub-menu the poll time and response time setup window appear as shown in Figure 7.4. Poll time intervals are in seconds and response time intervals in milliseconds.

[1]This is applicable for the memory addressing of LG G7M-DR10A as assigned for this project only. The addressing depends on the user memory assignment of the PLC

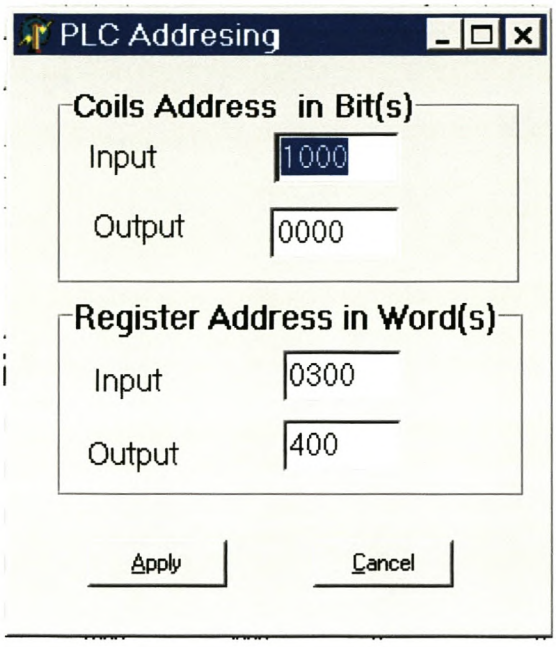


Figure 7.3: PLC Addressing Window

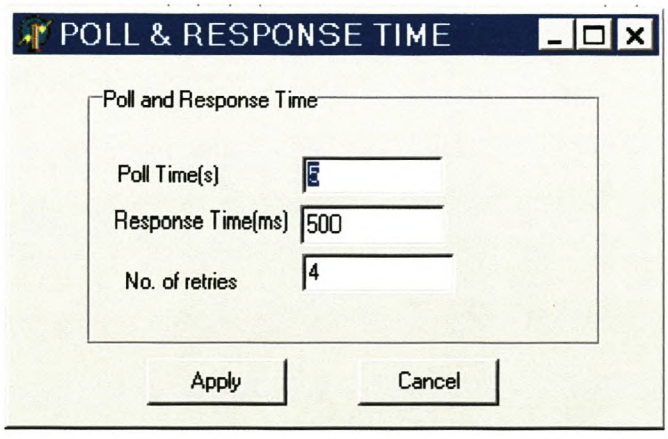


Figure 7.4: Poll Time and Response Time Settings

7.2.3 (b) Automatic

By clicking **'Automatic'** sub-menu the slave device is polled at the Poll interval specified in the Time settings.

7.2.3 (c) All Functions

By clicking **'All Functions'**, will allows the master to poll a slave station, to read function codes one at a time at the interval specified in time settings.

7.2.3 (d) Stop/Pause

This option stops the master to poll, or, pause it. Polling can be restarted by clicking the **'Automatic'** or **'All Functions'** sub-menu.

7.3 Reading PLC Coils/Registers Contents

To read PLC memory, the function to be performed is chosen on a **'read function codes'** panel. Slave address, starting register/coil address and number of registers/coils to be read are also edited on a **'Parameters'** panel. When the **'read'** button is pressed, the master reads the status/value of the coils (registers).

7.4 Writing to the Output of a PLC

To change the status of the output coils of the PLC, the **'Write Coils** or **'Write Register'** button is pressed.

7.4.1 Writing to Output Coils

7.4.1 (a) Write Multiple Coils

When the 'Write Coils' button is pressed, the following window shown in Figure 7.5, appears and the output status is set by checking or unchecking the check box, if the check box is checked, the binary state of that input is '1' (ON), otherwise it is '0' (OFF).

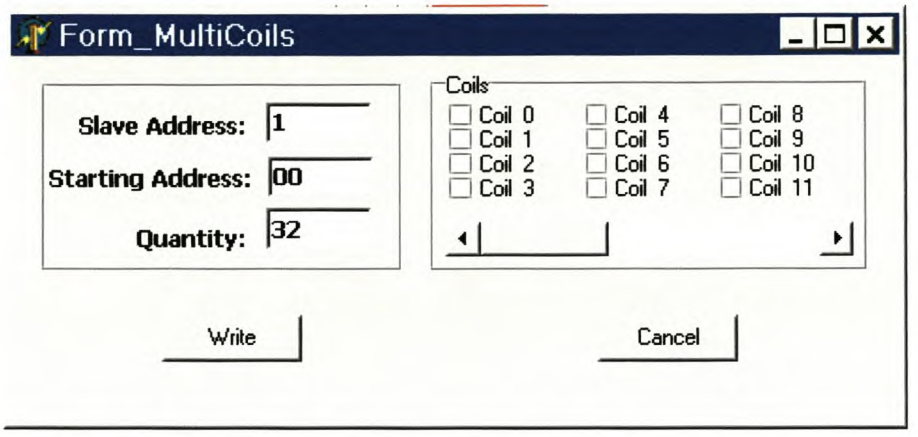


Figure 7.5: Window for Writting to the Outputt Coils of the PLC

7.4.1 (b) Write single coil

One coil can be forced at a time by double clicking the output coils grid in the main window. Figure 7.6 is used to set the coil selected on the grid to either ON/OFF.

7.4.2 Writing to the Holding/Output Registers

7.4.2 (a) Preset multiple registers

When the Write Registers button is pressed, the window in Figure 7.7 appears, where the value of the registers can be entered into the grid.

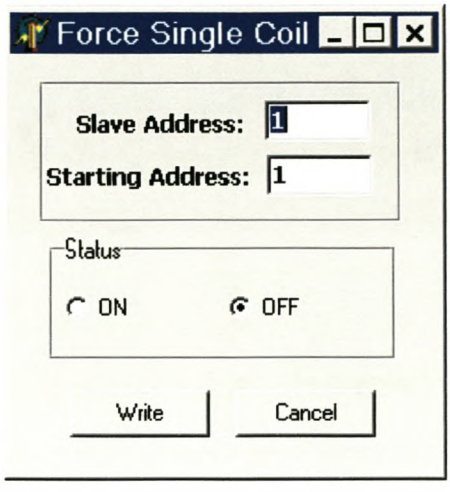


Figure 7.6: Force Coil 1 of Slave Device 1 to OFF

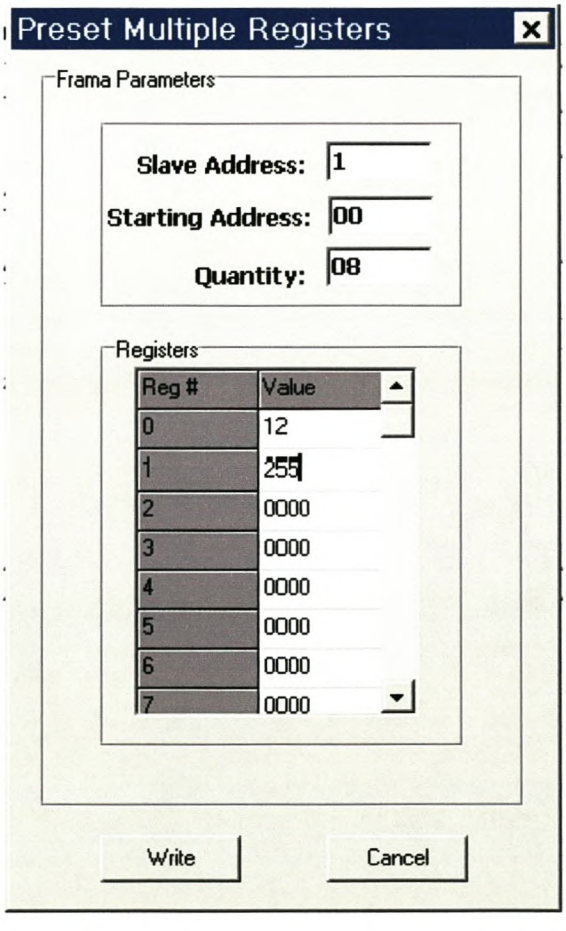


Figure 7.7: Writing to the Multiple Holding Registers of the PLC

7.4.2 (b) Preset single register

A single output register value is preset by clicking on the output/holding register grid and entering the value to written. Figure 7.8 shows the preset single register window.

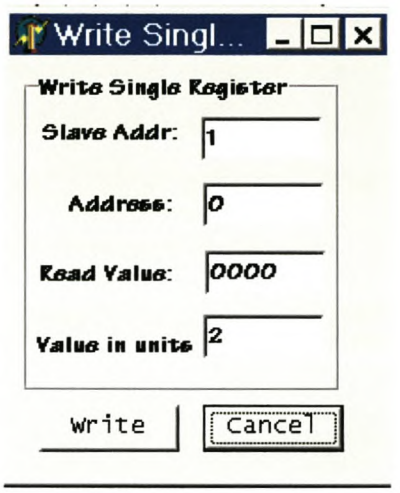


Figure 7.8: Preset Single Register Window

7.5 Other Parameters

The Modbus ASCII has a string grid in which all the read commands update the data.

7.5.1 Registers Grid

The register address are represented by rows. The first column Span of the grid is the maximum level that the slave device has. For the LG G7M DR10A the maximum is 4000, with a maximum voltage of 10 V (column 2 - I/O range). To convert the read value in column 4 Read the reading are converted to a engineering values as follows:

$$Value\ in\ Engineering\ units = \frac{Read}{Span(4000)} * I/O\ Range\ (10V)$$

7.5.2 Coils Grid

Coils grid display the output/input status of the coils in a slave device or PLC.

Since Modbus does not support date and time tagging the master program time stamps the data when a response is received. This is necessary to do proper analysis of data at a later stage, should that be required. This is also useful to see which stations are active or when last did the master station receive the response from the master. The status of the inputs/outputs are also displayed.

7.5.3 Monitoring Transactions

The Modbus transactions of the incoming and outgoing traffic can be viewed by clicking 'View' on the main menu, and then click 'Traffic' sub-menu. The window in Figure 7.9 will appear showing all the last transactions. This window is useful when debugging the system.

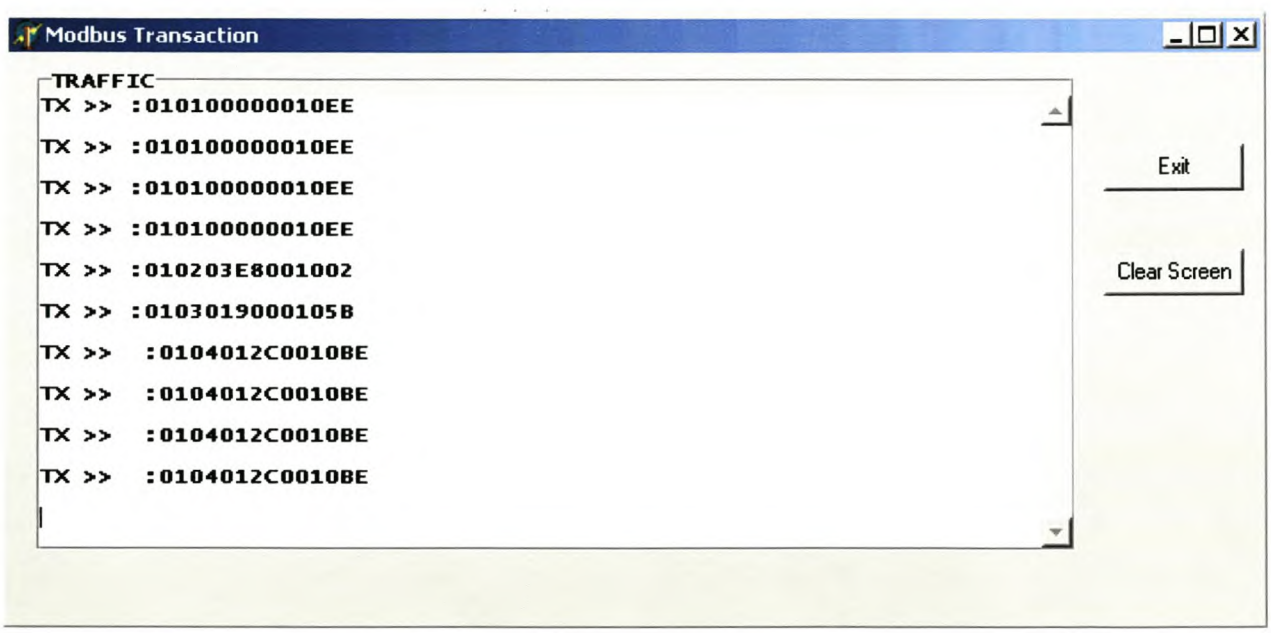


Figure 7.9: Moniroring Transactions Window

Chapter 8

Conclussion and Recomendations

8.1 Conclusion

The flexibility of the Modbus protocol allows it to handle all kinds of processes and operations in many industries. Modbus is also supported, to some extent, in many of the building control disciplines such as electricity meters, process control, etc.

Expanding Modbus to an air based version opens up a major new field of applications. This enables control and monitoring to be exercised over large distances, i.e up to 100 km where radio repeaters are utilised. It also enables peripheral control equipment to be placed where convenient and required and not as dictated by wiring constraints.

Mobile applications like cranes are a case in point. The implementation of Modbus for radio use , required the recognition and incoperation of specific restrictions into the protocol development. The final product is eminently successful and suitable for immediate practical application.

The master computer was able to perform all the basic commands as supported by the LG G7M DR10A PLC, with maximum response time of 2.3 seconds, by utilising Modbus ASCII protocol over MDS EL705 Data Transceivers. This setup is very flexible compared to the use of dial-up modems and conventional RS232 cables. Some research has been done for use of Modbus protocol over TCP/IP. However, the implementation of this over , radio data modems, has been lacking and presents a significant improvement.

The system performance at baud rates of 1200 bps to 4800 bps good, i.e. It was able to

read a mximumum message length of 256 characters. At 9600 bps the system throughput decrease by a factor of 1/2, this is due to a bandwidth limitation, in the SA licensing environment where the bandwidth is limited to 12.5 Khz. The bandwidth required by the radio data modems at 9600 bps band is about 4 times the bandwidth of conventional analogue modems.

Since Modbus protocol does not support time and date tagging, the time tagging was achieved by time stamping the response data when received from the PLC. This is a further enhancement not commonly available versions and is very valuable to sensibly track process history.

Although the HMI as implemented on the application layer, was not initially intended, it flowed naturally from the development. It greatly improves the immediate usability of the product and provides a rounded off, neat package to potential users and applications.

8.2 Recommendations

Although the end result of the project provided a ready to use, significantly expanded and enhanced version of a widely used scheme, some further development might be of benefit. The following should be considered:

- Other common protocols, such as Profibus and Fieldbus, providing for point-tomultipoint use, would be considered for similar development.
- It has already mentioned that time and date stamping has been piggy-backed on the standard Modbus structure. Some other aspects such as routing information for digi-peating applications, could be considered for the same treatment.
- The existing HMI could be much expanded to present a host of further useful features, normally duly found in proper SCADA applications.

While it is not suggested that a competative SCADA package be written some popular features could be added to provide micro-mini SCADA type environment, packaged with the wireless protocol.

Bibliography

[1] Modicon Modbus Protocol Reference Guide (PI-MBU-300 REV. J), MODICON, Inc, Industrial Automation Systems, June 1996.

[2] Gruenberg, Eliot Lewis, Handbook of Telemetry and Remote Control, New York, MCGraw-Hill,1967.

[3] Stenerson, Jon, Fundamentals of Programmable Logic Controllers, Sensors, and Communications, Upper Saddle River, Prentice Hall, 1999.

[4] Doll Dixon R, Data Communications: facilities, network, and system design, New York, Wiley, 1978.

[5] John E. Bingham and Garth W. P Davies, Planning For Data Communications, The Macmillan Press LTD, 1997.

[6] James Martin(IBM Systems Research Institute), Introduction To Teleprocessing, Prentice-Hall, inc Englewood Cliffs, New Jersey, 1972

[7] MODBUS over Serial Line Specification and Implementation Guide V1.0, modbus.org, (http://www.modbus.org/), 06 December 2002

[8] Andrew S. Tanenbaun, Computer Networks, Third Edition, Prentice-Hall International, Inc, 1996

[9] Andrew S. Tanenbaun, Computer Networks, Fourth Edition, Pearson Education, Inc, 2003

[10] Modbus Application Protocol Specification, Modbus.org, http://www.modbus.org/, 06 December 2002

[11] William Stallings, Data and Computer Communications, Fourth Edition, Macmillan Publishing Company, 1994

[12] William Stallings, Data and Computer Communications, Seventh Edition, Prentice Hall, Pearson Edducation International, 2004

[13] Stephen Morris, Delphi 5 Made Simple, Made simple books; oxford,Auckland, Boston, Johannesburg, Melbourne, New Delphi; 2000

[14] Marco Cantu, Mastering Delphi 6, Sybex Inc, 2001.

[15] EL705 OEM Series : Installation and Operation Guide, Microwave Data Systems Inc.(http:www.microwavedata.com), MDS 05.3624A01,Rev B,January 2001

[16] A Modbbus Deveopers Community, http://modbus.control.com

[17] An Introduction to Telemetry, http://www.dataradio.com/article_telemetry.html

[18] Wireless SCADA system Design Considerations, http://www.dataradio.com/article_systemdesign.html

[19] Marc's Technical pages: Building Radio Telemetry Systems that Work, http://www.marcspages.co.uk

[20] Sena, Technical Tutorial: Introduction To Modbus, http://www.sena.com/download/tutorial/tech_Modbus_v1r0c0.pdf, 2002-12-06

[21] Bob Hochreiter, Wireless Communications for Industrial Automation, Grayhill inc, http://grayhill.com

[22] http://www.modbus.org

[23] http://www.topworx.com

Appendix A

LRC Implementation

The LRC field consist of one byte, containing an 8-bit binary value. The LRC value is calculated by the transmitting device, which appends the LRC to the message. The receiving device recalculates an LRC during receipt of the message, and compares the calculated value to the actual value it received in the LRC field. If the two values are not equal, an error results.

The LRC is calculated by adding together sucessive 8-bit bytes in the message, discarding any carry, and then twos complementing the result. The LRC is an 8-bit field, therefore each new addition of a character that would result in a value higher than 255 simply rolls over the fields value through zero. Because there is no ninth bit, the carry is discarded automatically. A procedure for generating an LRC is:

1. Add all bytes in the message, excluding the starting colon and ending CRLF. Add them into an 8-bit field, so that caries will be discarded.
2. Subtract the final field value from FF (all 1s), to produce the ones complement.
3. Add 1 to produce the twos-complement.

A.1 Appending the LRC Value into the Message

When the 8-bit LRC (2 ASCII characters) is transmitted in the message, the high-order will be transmitted first, followed by the low-order character. For Example if the value is 61 hex (0110 0001), the LRC is appended as in Table A.1.

colon	Addr	Func	Data count	Data	Data	Data	Data	LRC Hi 6	LRC Lo 1	CR	LF

Table A.1: Placing LRC into the Message

Appendix B

Delphi Code

Delphi function for creating and initializing Modbus ASCII transactions. See attached CD for complete code:

```
TQuery = class
function ReadHoldingRegister(SlaveAddress,Address,Quantity : Integer):String;
function ReadInputRegister(SlaveAddress,Address,Quantity :Integer):String;
function ReadCoilsStatus(SlaveAddress,Address,Quantity : Integer):String;
function ReadInputStatus(SlaveAddress,Address,Quantity : Integer):String;
function ForceSingleCoil(SlaveAddress,Address:Integer;OutputValue: Boolean):String;
function PresetSingleRegister(SlaveAddress,Address:Integer; RegisterValue
        : Word ):String;
function ForceMultipleCoils(SlaveAddress,Address,Quantity: Integer;
        OutputValue : String): String;
function PresetMultipleRegister(SlaveAddress, Address,Quantity: Integer;
        RegisterValue:String):String;
function LRC(Str : String) : String;
function BinToDec(Str : String):Word;
function SwapBytes(Data : String):String;
end;
```

This is Delphi class for processing the response from the Slave device. See attached CD for complete source code:

```
TResponse = class
function ReadCoils(Start,Quantity,Bite_Count,Data:String):String;
function ReadInputStatus(Start,Quantity,Bite_Count,Data:String):String;
function ReadHoldingReg(Start,Quantity,Bite_Count,Data:String):String;
function ReadInputReg(Start,Quantity,Bite_Count,Data : String) : String;
function DecToBin (Num : DWord;Size : Byte):String;
function SortBits(HexValue:String):String;
function ExceptionResponse(FunctionCode,ExceptionCode:String):String;
end;
```

Appendix C

Radio Modem and Its Specifications

General	
Type	: 450 MHz Licensed OEM Board Level EL7054 - 330-512 Radio Type Synthesized, Half Duplex, Channel 12.5 kHz Channel Spacing, Split Freq or Simplex
Transmitter	
Frequency Ranges (450 MHz)	: 330 to 355 MHz 355 to 380 MHz 380 to 400 MHz 400 to 420 MHz 420 to 450 MHz 450 to 480 MHz 480 to 512 MHz 406 to 430 MHz (Canadian Plan)
Frequency Increments	: 6.25 kHz or 5 kHz (Factory Configurable)
Modulation Type	: 4 Level CPFSK
Carrier Power	: 100 mw, 1 Watt, 2 Watt Programmable (+20dBm, + 30dBm, +33 dBm)
Duty Cycle	: 50% (100% with additional heatsinking)
Output Impedance	: 50 Ohms
Frequency Stability	: 1.5 ppm, -30 to + 60 C
Channel Spacing	: 12.5 kHz
Spurious and Harmonics	: -65 dBc
Transmitter Keying	: On Data
Time-out Timer	: 1 to 255 Seconds
Key-up time	: 2 ms

Receiver	
Bandwidth	: 12.5 kHz
Data Performance	: 1x10-6 @ -108 dBm
Frequency Ranges (450 MHz)	: 330 to 355 MHz 355 to 380 MHz 380 to 400 MHz 400 to 420 MHz 420 to 450 MHz 450 to 480 MHz 480 to 512 MHz 406 to 430 MHz (Canadian Plan)
Intermodulation Rejection	: -70 dB Minimum
Selectivity	: 55 dB typical at Adjacent Channel (EIA)
Spurious and Image Rejection	: -70 dB
Type	: Double Conversion Superhetrodyne (84 MHz and 450 kHz IF)
Sensitivity	: 12 dB Sinad @ -116 dBm
Frequency Stability	: 1.5 ppm, -30 to +60 C
Interfaces	
Baud Rates Supported at Interface Port	: 1200, 2400, 4800, 9600, 19200, 38400 bps
Data Latency	: < 15 ms typical
Interface	: RS-232 through DB-25 Connector
Over-the-Air Data Rate	: 9600 bps
Power	
Circuit Protector	: 2 Amp board mounted Fuse, Internal Reverse Polarity
Protection	: Diode across primary input
Voltage	: 10 to 30 Vdc through 5.5 mm pin plug or DB-25
TX Supply Current	: 480 ma typical @ 13.8 Vdc with output power set to 2 watts
RX Supply Current	: 75 ma typical @ 13.8 Vdc
Environmental	
Humidity	: 0 to 95% @ 40 C
Temperature Range	: -30 to + 60 C

Figure C.1: MDS Data Transceiver

Appendix D

Diagrams

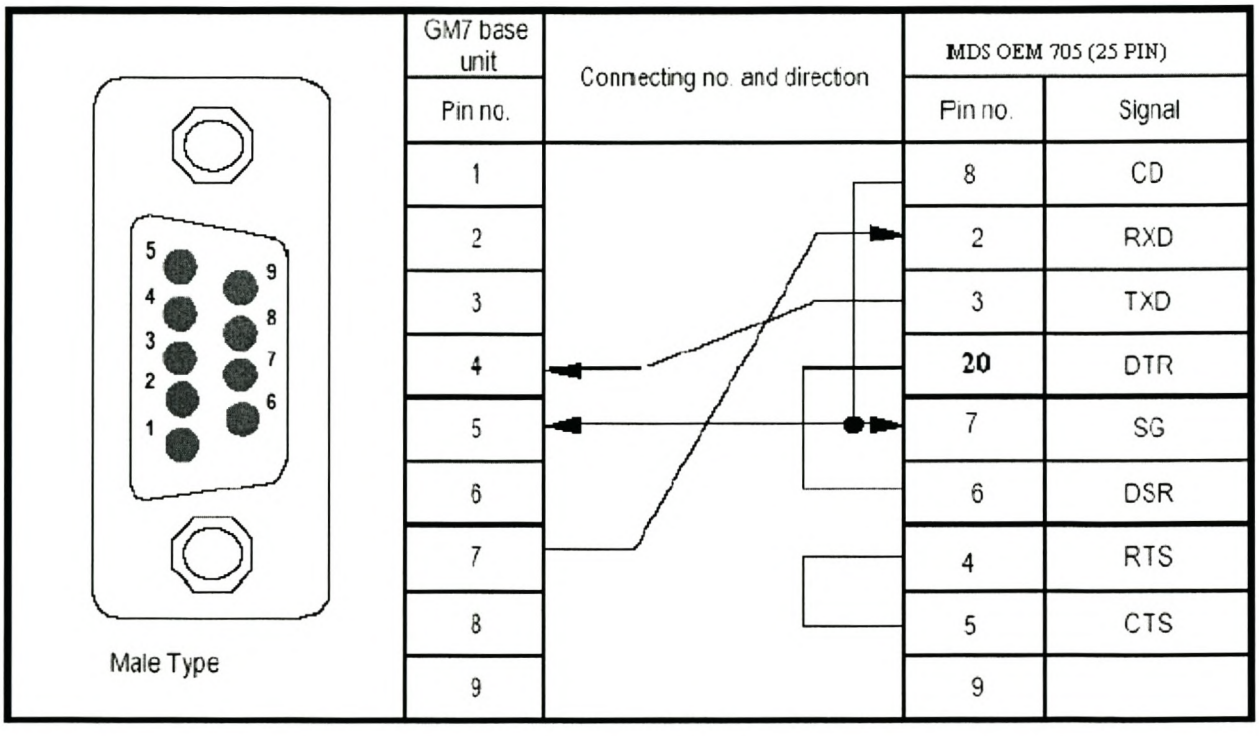


Figure D.1: MDS OEM 705 Data Transceiver to LG GM7 PLC RS232 Cable

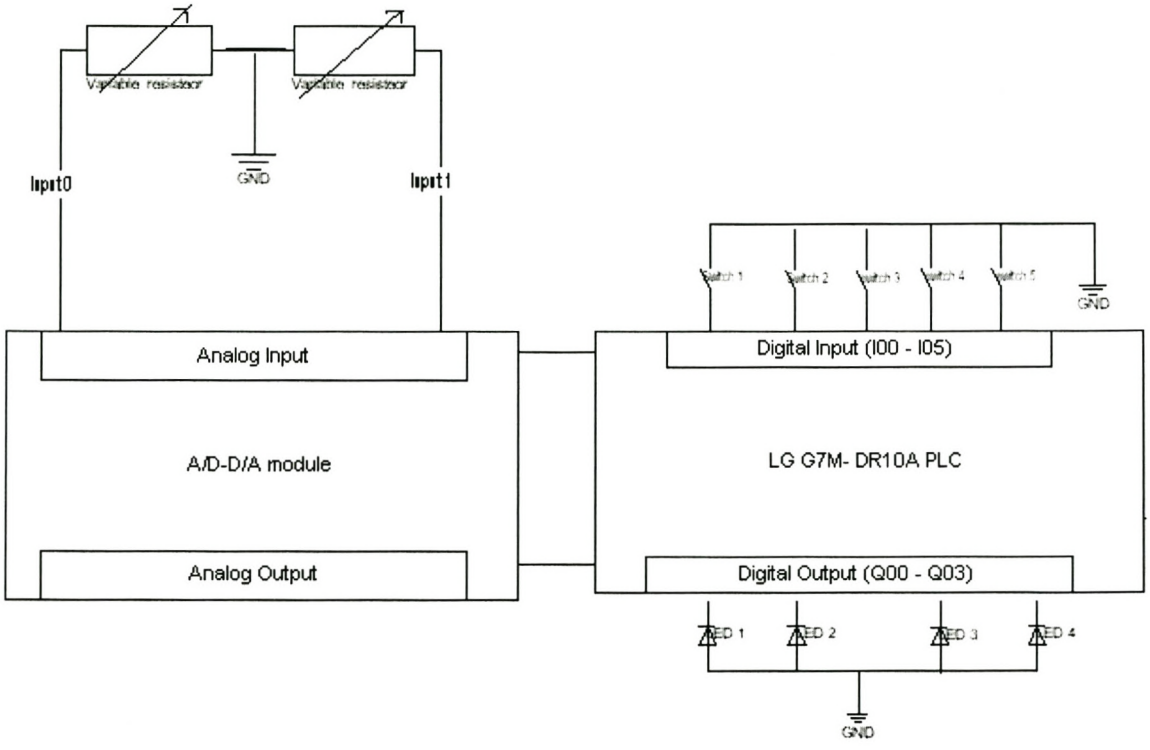


Figure D.2: LG PLC with A/D Inputs and Outputs Configuration

Appendix E

Fieldbus Comparison Chart

THE SYNERGETIC FIELDBUS COMPARISON CHART

BACKGROUND INFORMATION				
Fieldbus Name	**Technology Developer**	**Year Introduced**	**Governing Standard**	**Openness**
PROFIBUS DP/PA	Siemens	DP-1994, PA-1995	EN 50170 / DIN 19245 part 3 (DP) /4 (PA), IEC 1158-2 (PA)	ASICs from Siemens and Profichip, Products from over 300 vendors
INTERBUS-S	Phoenix Contact, Interbus Club	1984	DIN 19258 EN 50.254	Products from over 400 manufacturers
DeviceNet	Allen-Bradley	March 1994	ISO 11898 &11519	17 chip vendors, 300+ product vendors, Open specification
ARCNET	Datapoint	1977	ANSI/ATA 878.1	Chips, boards, ANSI docs
AS-I	AS-I Consortium	Fall 1993	Submitted to IEC	AS-II.C. Market item
Foundation Fieldbus H1	Fieldbus Foundation	1995	ISA SP50/IEC 61158	Chips/software/products from multiple vendors
Foundation Fieldbus High Speed Ethernet (HSE)	Fieldbus Foundation	In development - lab test phase, Prelim spec available to members	IEEE 802.3u RFC for IP, TCP & UDP	Multitude of suppliers for Ethernet components, Extremely low cost
IEC/ISA SP50 Fieldbus	ISA & Fieldbus F.	1992 - 1996	IEC 1158/ANSI 850	Multiple chip vendors
Seriplex	APC, Inc.	1990	Seriplex spec	Chips available multiple interfaces
WorldFIP	WorldFIP	1988	IEC 1158-2	Multiple chip vendors
LonWorks	Echelon Corp.	March 1991		Public documentation on protocol
SDS	Honeywell	Jan., 1994	Honeywell Specification, Submitted to IEC, ISO11989	17 chip vendors, 100+ products
ControlNet	Allen-Bradley	1996	ControlNet International	Open Specification, 2 Chip Vendors

CANopen	CAN In Automation	1995	CiA	17 chip vendors, 300 product vendors, Open specification
Ethernet	DEC, Intel, Xerox	1976	IEEE 802.3, DIX v. 2.0	Multitudes of Chips and Products
Modbus Plus	Modicon			Proprietary, requires license/ASICs
Modbus RTU/ASCII	Modicon		EN 1434-3 (layer 7) IEC 870-5 (layer 2)	Open specification, no special hardware required
Remote I/O	Allen-Bradley	1980		Proprietary
Data Highway Plus (DH+)	Allen-Bradley			Proprietary

PHYSICAL CHARACTERISTICS

Fieldbus Name	Network Topology	Physical Media	Max. Devices (nodes)	Max. Distance
PROFIBUS DP/PA	Line, star & ring	Twisted-pair or fiber	127 nodes (124 slaves - 4 seg, 3 rptrs) + 3 masters	100m between segments @ 12Mbaud; 24 Km (fiber) (baudrate and media dependent)
INTERBUS-S	Segmented with "T" drops	Twisted-pair, fiber, and slip-ring	256 nodes	400 m/segment, 12.8 Km total
DeviceNet	Trunkline/dropline with branching	Twisted-pair for signal & power	64 nodes	500m (baudrate dependent) 6Km w/ repeaters
ARCNET	Star, bus, distributed star	Coax, Twisted-pair, Fiber	255 nodes	Coax 2000 feet; Twisted pair 400 feet; Fiber 6000 Feet
AS-I	Bus, ring, tree star, of al	Two wire cable	31 slaves	100 meters, 300 with repeater
Foundation	Star or bus	Twisted-pair,	240/segment,	1900m @ 31.25K wire

Fieldbus H1		**fiber**	**65,000 segments**	
Foundation Fieldbus HSE	**Star**	**Twisted-pair, fiber**	**IP addressing - essentially unlimited**	**100m @ 100Mbaud twisted-pair 2000m @ 100Mbaud fiber full duplex**
IEC/ISA SP50 Fieldbus	**Star or bus**	**Twisted-pair fiber, and radio**	**IS 3-7 non IS 128**	**1700m @ 31.25K 500M @ 5Mbps**
Seriplex	**Tree, loop, ring, multi-drop, star**	**4-wire shielded cable**	**500+ devices**	**500+ ft**
WorldFIP	**Bus**	**Twisted-pair, fiber**	**256 nodes**	**up to 40 Km**
LonWorks	**Bus, ring, loop, star**	**Twisted-pair, fiber, power line**	**32,000/domain**	**2000m @ 78 kbps**
SDS	**Trunkline/Dropline**	**Twisted-pair for signal & power**	**64 nodes, 126 addresses**	**500m (baudrate dependent)**
ControlNet	**Linear, Tree, Star, or Combination Thereof**	**Coax, fiber**	**99 nodes**	**1000m (coax) 2 nodes 250m with 48 nodes 3km fiber; 30km fiber w/ repeaters**
CANopen	**Trunkline/Dropline**	**Twisted Pair + optional Signal & Power**	**127 Nodes**	**25-1000m (baudrate dependent)**
Industrial Ethernet	**Bus, Star, Daisy-Chain**	**Thin Coax, Twisted Pair, Fiber; Thick Coax (rare)**	**1024 nodes, expandable to more via Routers**	**Thin: 185m 10 Base T (Twisted Pair): Max 100m long (90 metres horizontal cable, 5m drops, 1m patch) Max 4 hubs/repeaters between nodes 4Km distancs w/o routers Fiber: 100 Base FX 400m 2.5 Km multi mode w/o Switches; 50 Km mono mode w/ Switches**
Modbus Plus	**Linear**	**Twisted Pair**	**32 nodes per segment, 64 max**	**500m per segment**
Modbus	**Line, star, tree**	**Twisted Pair**	**250 nodes per**	**350m**

RTU/ASCII	Network w/ segments		segment	
Remote I/O	Linear Trunk	Twinaxial	32 nodes/segment	6 km
DH+	Linear Trunk	Twinaxial	64 nodes/segment	3 km

TRANSPORT MECHANISM

Fieldbus Name	Communication Methods	Transmission Properties	Data Transfer Size	Arbitration Method	Error Checking	Dia
PROFIBUS DP/PA	Master/slave peer to peer	DP: 9.6, 19.2, 93.75, 187.5, 500 Kbps, 1.5, 3, 6, 12 Mbps PA: 31.25 kbps	0-244 bytes	Token passing	HD4 CRC	Sta cha dia
INTERBUS-S	Master/slave with total frame transfer	500kBits/s, full duplex	1-64 Bytes data 246 Bytes Parameter 512 bytes h.s., unlimited block	None	16-bit CRC	Seg of (cab
DeviceNet	Master/slave, multi-master, peer to peer	500 kbps, 250 kbps, 125 kbps	8-byte variable message with fragmentation for larger packets	Carrier-Sense Multiple Access w/ Non-Destructive Bitwise Arbitration	CRC check	Bu
ARCNET	Peer to peer	19.53K to 10M	0 to 507 bytes	Token passing	16-bit CRC	Bu Ac at
AS-I	Master/slave with cyclic polling	Data and power, EMI resistant	31 slaves with 4 in and 4 out	Master/slave with cyclic polling	Manchester Code, hamming-2	Sla fau
Foundation Fieldbus H1	Client/server publisher/ subscriber, Event notification	31.25 kbps	128 octets	Scheduler, multiple backup	16-bit CRC	Re dia net par
Foundation	Client/Server,		Varies, Uses			

Fieldbus HSE	Publisher/Subscriber, Event Notification	100Mbps	Standard TCP/IP	CSMA/CD	CRC	
IEC/ISA SP50 Fieldbus	Client/server Publisher/ subscriber	31.25 kbps IS+1, 2.6, 5 Mbps	64 octets high & 256 low priority	Scheduler, tokens, or master	16-bit CRC	Co net ma
Seriplex	Master/slave peer to peer	200 Mbps	7680/transfer	Sonal multiplexing	End of frame & echo check	Cal
WorldFIP	Peer to peer	31.25 kbps, 1 & 2.5 Mbps, 6 Mbps fiber	No limit, variables 128 bytes	Central arbitration	16-bit CRC, data "freshness" indicator	Dev tim red
LonWorks	Master/slave peer to peer	1.25 Mbs full duplex	228 bytes	Carrier Sense, Multiple Access	16-bit CRC	Dat err err
SDS	Master/slave, peer to peer, multi-cast, multi-master	1Mbps, 500 kbps, 250 kbps, 125 kbps	8-byte variable message	Carrier-Sense Multiple Access w/ Non-Destructive Bitwise Arbitration	CRC check	Bus
ControlNet	Producer/Consumer, Device Object Model	5 Mbps	0-510 bytes variable	CTDMA Time Slice Multiple Access	Modified CCITT with 16-bit Polynomial	Du Dev Fau
CANopen	Master/slave, peer to peer, multi-cast, multi-master	10K, 20K, 50K, 125K, 250K, 500K, 800K, 1Mbps	8-byte variable message	Carrier-Sense Multiple Access w/ Non-Destructive Bitwise Arbitration	15 Bit CRC	Err Em Me
Industrial Ethernet	Peer to Peer	10, 100Mbps	46-1500 Bytes	CSMA/CD	CRC 32	
Modbus Plus	Peer to Peer	1Mbps	variable			
Modbus RTU/ASCII	Master/Slave	300 bps - 38.4Kbps	0-254 Bytes			
Remote I/O	Master/Slave	57.6 - 230 kbps	128 Bytes		CRC 16	nor
	Multi-Master,					

DH+	Peer<>Peer	57.6 kbps	180 Bytes	nor

PERFORMANCE

Fieldbus Name	Cycle Time: 256 Discrete 16 nodes with 16 I/Os	Cycle Time: 128 Analog 16 nodes with 8 I/Os	Block transfer of 128 bytes 1 node
PROFIBUS DP/PA	Configuration dependent typ <2ms	Configuration dependent typ <2ms	not available
INTERBUS-S	1.8 ms	7.4 ms	140 ms
DeviceNet	2.0 ms Master-slave polling	10 ms Master-slave polling	4.2 ms
ARCNET	Application Layer Dependent	Application Layer Dependent	Application Layer Dependent
AS-I	4.7 ms	not possible	not possible
Foundation Fieldbus H1	<100 ms typical	<600 ms typical	36 ms @ 31.25k
Foundation Fieldbus HSE	Not Applicable; Latency <5ms	Not Applicable; Latency <5ms	<1ms
IEC/ISA SP50	Configuration dependent	Configuration dependent	0.2 ms @ 5 Mbps 1.0 ms @ 1 Mbps
Seriplex	1.32 ms @ 200 kbps, m/s	10.4 ms	10.4 ms
WorldFIP	2 ms @ 1 Mbps	5 ms @ 1 Mbps	5 ms @ 1 Mbps
LonWorks	20 ms	5 ms @ 1 Mbps	5 ms @ 1 Mbps
SDS	<1 ms, event driven	5 ms polling @ 1 Mbps	2 ms @ 1 Mbps
ControlNet	<0.5 ms	<0.5 ms	<0.5 ms
CANopen	<1 ms	5 ms polling @ 1 Mbps	<2.5 ms
Industrial Ethernet	Application Layer Dependent	Application Layer Dependent	Application Layer Dependent
Modbus Plus			
Modbus RTU/ASCII			
Remote I/O	12msec @230, 40 msec @57.6 bus cycle time		
DH+			

Appendix F

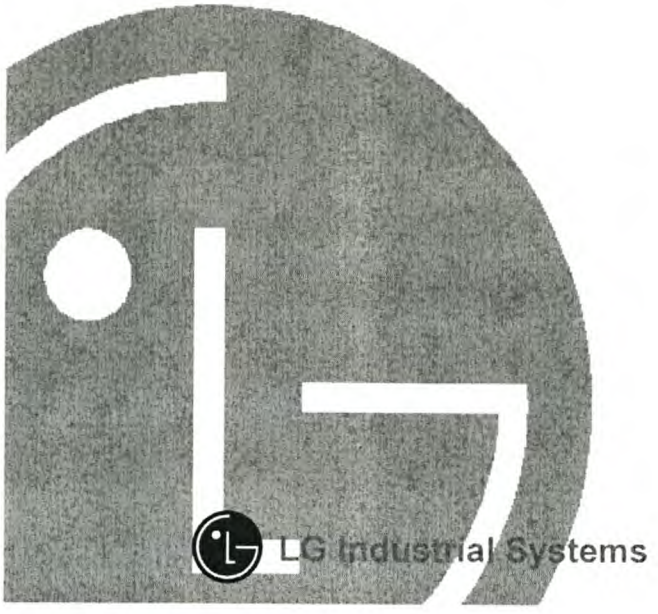
PLC Data Sheet

DATA

702008035

LG Programmable Logic Controller

GLOFA-GM7 Series



Before handling the product

Read this data sheet carefully prior to any operation, mounting, installation or start-up of the product.

Materials for GLOFA-GM

Name	Code
GMWIN(Programming software)	702005047
GLOFA-GM (Instruction & programming)	702005058
GLOFA-GM7 User's manual	702008240

o Safety Precautions

Be sure to read carefully the safety precautions given in data sheet and user's manual before operating the module and follow them.

The precautions explained here only apply to the GM7 Base unit. For safety precautions on the PLC system, see the GLOFA-GM7 User's manual.

A precaution is given with a hazard alert triangular symbol to call your attention, and precautions are represented as follows according to the degree of hazard..

⚠ Warning	If not provided with proper prevention, it can cause death, fatal injury or considerable loss of property
⚠ Caution	If not properly observed, it can cause a hazard situation to result in severe or slight injury or a loss of property.

⚠ However, a precaution followed with 'Caution' can also result in serious conditions. Both of two symbols indicate that an important content is mentioned, therefore, be sure to observe it. Keep this manual handy for your quick reference in necessary.

o Design Precautions

⚠ Caution

- Don not run I/O signal lines near to high voltage line or power line.
  Separate them as 100mm or more as possible. Otherwise, noise can cause module malfunction.

o Installation Precautions

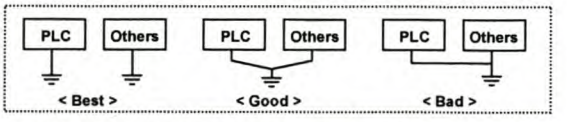
⚠ Caution

- Operate the PLC in the environment conditions given in the general specifications..
- If operated in other environment not specified in the general specifications, it can cause an electric shock, a fire, malfunction or damage or degradation of the module.
- Make sure the module fixing projections is inserted into the module fixing hole and fixed.
- Improper installation of the module can cause malfunction, disorder or falling.

o Wiring Precautions

⚠ Caution

- Make sure that FG the terminal is grounded with class 3 grounding which is dedicated to the PLC. Otherwise, it can cause disorder or malfunction of PLC.



- Before the PLC wiring, be sure to check the rated voltage and terminal arrangement for the module and observe them correctly. If a different power, not of the rated voltage, is applied or wrong wiring is provided, it can cause a fire or disorder of the

- **Beijing Branch**
  LG Industrial Systems (Thailand)
  Elevator Co., Ltd.
  T : +86-10-6462-3256
  F : +86-10-6462-3255
- **Bogota Branch**
  LG Industrial Systems de Colombia S.A.
  T : +57-1-310-6077
  F : +57-1-310-5831
- **Dalian Branch**
  Dalian LG Industrial Systems Co., Ltd.
  T : +86-411-281-2579
  F : +86-411-281-2578
- **Hong Kong Branch**
  LG Industrial Systems (HK) Ltd.
  T : +852-2598-6615
  F : +852-2598-7105
- **Singapore Branch**
  LG Industrial Systems Co., Ltd.
  T : +65-323-7361
  F : +65-323-7362
- **Bangkok Branch**
  LG Industrial Systems Co., Ltd
  T : +66-2-381-8443
  F : +66-2-381-8445
- **Chicago Branch**
  LG Industrial Systems Co., Ltd. Chicago Office
  T : +1-708-692-4500
  F : +1-708-692-4501
- **Hanoi Branch**
  LG Industrial Systems Co., Ltd. Hanoi Office
  T : +64-4-821-0388
  F : +64-4-821-0399
- **Shanghai Branch**
  Shanghai LG Industrial Systems Co., Ltd.
  T : +86-21-6248-2710
  F : +86-216248-3236
- **Taipei Branch**
  LG Industrial Systems (Taiwan) Co. Ltd.
  T : +886-2-516-5010
  F : +886-2-516-5035
- **Tokyo Branch**
  LG Industrial Systems Co., Ltd. Tokyo Office
  T : +81-3-3589-6362
  F : +81-3-3588-1810

LG Industrial Systems Co., Ltd.

- **Head Office**
  LG Twin Tower 26th F, Youido-dong, Yongdungpo-gu, Seoul, KOREA
  Tel : +82-2-3777-4641~7 Fax : +82-2-3777-4648
  Home page : http://www.lgis.lg.co.kr/fa

module.

- Drive the terminal screws firmly to the defined torque. If loosely driven, it can cause short circuit, a fire or malfunction.
- Be careful that any foreign matter like wire scraps should not enter into the module. It can cause a fire, disorder or malfunction.

o Test RUN and maintenance precautions

⚠ Warning

- Do not contact the terminals while the power is applied. It can cause malfunction.
- When cleaning or driving a terminal screw, perform them after the power has been turned off.

⚠ Caution

- Do not separate the module from the printed circuit board, or do not remodel the module. They can cause disorder, malfunction, damage of the module or a fire. When mounting or dismounting the module, perform them after the power has been turned off.
- Do not change battery while power is off. It can cause loss of data.

o Waste Disposal Precaution

⚠ Caution

- When disposing the module, do it as an industrial waste.

1. Introduction

This data sheet provides brief information about characteristics, configuration, and usage of GLOFA-GM7 Series.

2. General Specifications

No	Item	Specifications				Standard
1	Operating temperature	0 ~ 55□ (32 ~ 131°F)				
2	Storage temperature	-25 ~ 70□ (-13 ~ 158°F)				
3	Operating Humidity	5 ~ 95%RH, non-condensing				
4	Storage humidity	5 ~ 95%RH, non-condensing				
5	Vibration	Occasional vibration				IEC61131-2
		Frequency	Acceleration	Amplitude	Sweep count	
		10≤f□57 Hz	-	0.075 mm	10 times in each direction for X, Y, Z	
		57 ≤f≤150 Hz	9.8 □ {1G}	-		
		Continuos vibration				
		Frequency	Acceleration	Amplitude		
		10≤f□57 Hz	-	0.035 mm		
		57≤f≤150 Hz	4.9 □{0.5G}	-		
6	Shocks	*Maximum shock acceleration: 147 □ {15G} *Duration time :11 ms *Pulse wave: half sine wave pulse (3 times in each of X, Y and Z directions)				IEC61131-2
7	Noise immunity	Square wave impulse noise	±1,500 V			
		Electrostatic discharge	Voltage :4kV(contact discharge)			IEC61131-2 IEC1000-4-2
		Radiated electromagnetic field	27 ~ 500 MHz, 10 V/m			IEC61131-2 IEC1000-4-3
		Fast transient & Burst noise	Severity Level	All power modules	Digital I/Os (Ue ≥ 24 V)	Digital I/Os (Ue < 24 V) Analog I/Os Communication I/Os	IEC61131-2 IEC1000-4-4
			Voltage	2 kV	1 kV	0.25 kV	
8	Atmosphere	Free from corrosive gases and excessive dust					
9	Altitude for use	Up to 2,000m					
10	Pollution degree	2 or lower					
11	Cooling method	Self-cooling					

3. Performance Specifications

The following table shows the general specifications of the GLOFA-GM7 series

Items			Specifications	Remarks
Operation method			Cyclic operation of stored program, Interrupt task operation	
I/O control method			Scan synchronized batch processing method (Refresh method)	Immediate input/output is available by 'direct I/O ' function
Programming language			IL(Instruction List) LD(Ladder Diagram) SFC(Sequential Function Chart)	
Number of instructions	Operator		LD: 13, IL: 21	
	Basic function		138	
	Basic function block		11	
	Special function block		Each special module have their own special function blocks	
Processing speed	Operator		0.5 □/step	
Programming memory capacity			68K bytes	Including parameter (Approx.4k byte)
I/O points			10 points base unit 20 points base unit 30 points base unit 40 points base unit 60 points base unit	Max 2 expansion units can be attached to a base unit
Data memory	Direct variable area		2k to 8k bytes	Adjustable with parameter setting
	Symbolic variable area		32 k bytes-Direct variable area	
Timer			No limitations in points Time range: 0.001 to 4294967.295 sec (1193 hours)	1point occupies 20 bytes of Symbolic variable area
Counter			No Limitations in points Counting range: -32768 to +32767	1point occupies 8bytes of symbolic variable area
Operation modes			RUN, STOP, PAUSE and DEBUG	
Data protection method at power failure			Set to 'Retain' variables at data declaration	
Number of program blocks			128	
Program type	Scan		100	
	Time-driven interrupt task		8	
	External interrupt task		8 Total 8 points are available.	
	High speed counter task		1	
	Inside interrupt task		8	
	Initialization task		1 (_INIT)	
Internal Function	PID control function		Function block control, auto tuning, forced output, adjustable operation scan-time, forward/reverse operation control	
	Cnet I/F Function		GLOFA exclusive protocol support MODBUS protocol support User's protocol support	Common use with GMWIN port
	High-speed counter	Capacity	1 phase : 16 kHz, 1 channel 2 phase : 8 kHz,1 channel	
		Counter function	It has 3 different counter function as following; 1 phase, up/down by program 1 phase, up/down by B phase input 2 phase, up/down by phase difference	
		Multiplication function	Multiplication : 1, 2, or 4 (adjustable)	
		Data comparison function	Execute a task program when the elapsed counter value reaches to the preset value	
	Pulse catch		Minimum pulse width : 0.2msec, 8 points	
	Pulse output		2khz, 1point	Transistor output only
	External interrupt		8points, 0.4ms	
	Input filter		0~15ms	
Weight	G7M-DR10A		360	

(g)	G7M-DR20A	480	
	G7M-DR30A	551	
	G7M-DR40A	670	
	G7M-DR60A	844	
	G7E-DR10A	228	

4. Operation Processing Method

1) Cyclic operation

A PLC program is sequentially executed from the first step to the last step, which is called scan. This sequential processing is called cyclic operation. Cyclic operation of the PLC continues as long as conditions do not change for interrupt processing during program execution.

2) Time driven interrupt operation method

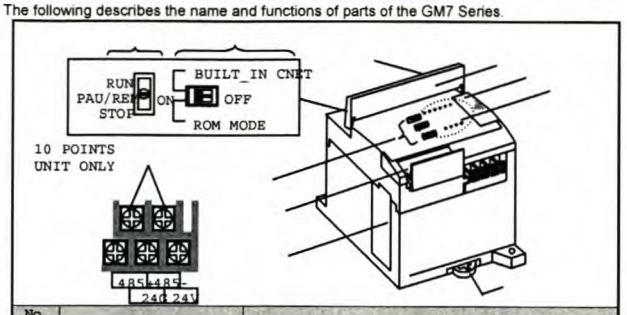
In time driven interrupt operation method, operations are processed not repeatedly but at every preset interval. In the GM7 Base unit, interval can be set to between 0.001 ~ 4294967.29 second. This operation is used to process operation with a constant cycle

3) Event driven interrupt operation method

If a situation occurs which is requested to be urgently processed during execution of a PLC program, this operation method processes immediately the operation which corresponds to interrupt program. The signal, which informs the CPU of those urgent conditions, is called interrupt signal. The GM7 CPU has two kinds of interrupt operation methods, which are internal and external interrupt signal methods.

5. Parts Name and Descriptions

The following describes the name and functions of parts of the GM7 Series.

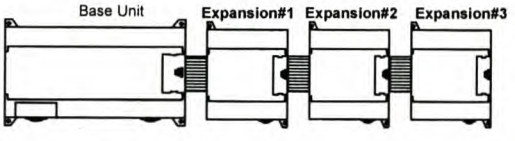


No.	Name		Function
□	CPU Status LED	PWR LED	Indicates of the power status. • On : normal state • Off : abnormal state or off
		RUN LED	Indicates the operation status of the Base unit. • On : When the Base unit operates with the mode setting switch in the local or remote RUN • Off : When the followings occur □ the voltage is not normally supplied to the Base unit □ the mode setting switch is in the stop or pause □ An error which makes operation stop is detected
		ERR LED	Indicates the operation status of the Base unit. • Flicker : An error is detected by self diagnostic during operation • Off : When the Base unit is normal state
□	Input/ Output LED		Indicates the input/output status
□	Battery holder		Battery holder
□	The mode setting switch		Sets the operation mode of the Base unit. • RUN : Program operation is executed • STOP : Program operation is stopped • PAU / REM □ PAUSE : Program operation is temporarily stopped □ REMOTE Used for the remote operation
□	Memory setting dip switch		For detailed direction for use, refer to the GM7 user's manual chapter 5
□	RS-232C Connector		• 9 pin connector for GMWIN and Cnet
□	For Expansion connector cover		• the connector which mounted expansion module cover
□	Terminal block cover		• the protection cover for external wiring
□	RS-485 COMM.Terminal		• the Terminal for RS-485 Comm. Use (10 Points Uint)
□	The hook for DIN rail		• the hook for DIN rail

6. I/O No. Allocation Method

1) I/O No. allocation means to give an address to each module in order to read data from input modules and output data to output modules.

2) Fixed 64 points are allocated to each module for I/O points. The following shows an example of I/O No. allocation method.



Mounting Module	Maximum No. of module can be mounted	Remark
Expansion I/O module	2	
A/D conversion module	2	
Analog timer module	3	
Communication module	1	Except 10Points Unit

3) The following is method I/O number allocation.

Item	Specification	Area
Base Unit (20□60 points)	Input	%IX0.0.0□%IX0.0.35
	Output	%QX0.0.0□%QX0.0.23
Expansion #1 (10 points)	Input	%IX0.1.0□%IX0.1.5
	Output	%QX0.1.0□%QX0.1.3
Expansion #2 (10 points)	Input	%IX0.2.0□%IX0.2. 5
	Output	%QX0.2.0□%QX0.2.3
Expansion #3	Special	None

7. Internal High Speed Function

1) Summary

The high-speed counting can count high speed pulse which can not be proceed with The CPU counting instructions. It can be counting pulse which occurs encoder or pulse generator.

Function	Description
Counter function	· 1 pulse operation Mode(Up/down count by program) 1 pulse operation Mode(Up/down count by phase B pulse input) 2 pulse operation Mode(Up/down count by difference of phase)
Multiplication	The multiplication factor for the input pulse may be set to 1,2 or 4(selected by special data register D4999)
Data comparison function	F170 is on when the current value reaches to the setting value
Preset operation	Sequence program or external preset input

2) Performance specifications

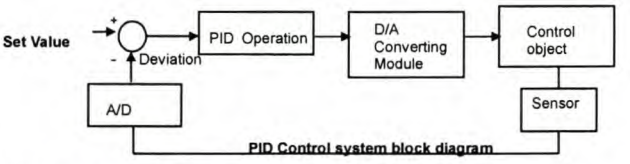
Item		Specification
Input signal	Types	A Phase , B Phase , Preset
	Types level	DC24V(15 □)
	Signal type	Voltage input
Counting range		0~16,777,215(binary 24bits)
Max. counting speed		1Phase 16 □ or 2Phase 8 □
Up/Down selection	1 Phase	Sequence Program or B-Phase input
	2 Phase	Auto-select by phase difference of A and B Phase
Multiplication		1,2 or 4
Preset input		Sequence Program or external input

8. PID Control Function

This chapter describes information about the built-in PID function of GM7 Series.

1) The characteristics of PID function of GM7 series as following

- □ The PID function is integrated into the Base unit.. Therefore, all PID control action can be performed with instructions and parameter without any separated PID module.
- □ Forward/reverse operations are available.
- □ P operation, PI operation, PID operation and On/Off operation can be selected easily.
- □ The manual output (the user-defined forced output) is available.
- □ By proper parameter setting, It can be stable operation regardless of external disturbance.
- □ The operation scan time (the internal that PID controller gets a sampling data from actuator) is changeable for optimizing to the system characteristics.



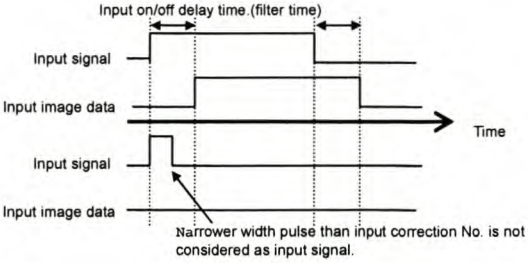
PID Control system block diagram

2) Instruction for PID control

For the PID Operation of GM7, there are two instructions, as follow.

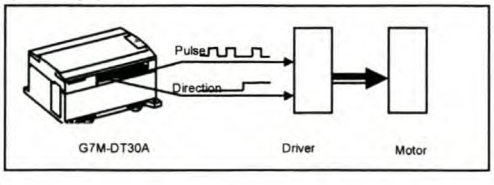
No.	Instruction	Function
1	PID8	Perform the PID operation
2	PID8AT	Perform the auto tuning operation

9. Pulse Output



1) Introduction

In the transistor output type of GM7, the pulse output function – maximum 2kpps – is internalized. By using this function with stepping motor or servo motor driver, GM7 is applicable to a positioning system.



2) Performance specification

Item	Specification
No. of output	1Point
Output type	Pulse
Output velocity	Max. 2Kpps, Min. 50pps
Output pulse	0 ~ 2147483647
Execution type of the Increasing/decreasing velocity	Designation of acceleration
Type of the direction	Forward/ Reverse direction pulse output
Load power supply	DC 12V/24V

REMARK

Several points can be used for the pulse output point if they are not output at the same time. Thus it is possible that Forward direction pulse is output as %QX0.0.0, reverse direction pulse is output as %QX0.0.1.

10.1 Dedicated communication

GM7series has built-in Cnet communication function, and it is possible that communicate with various external devices without separated Cnet I/F module. By using LGIS's dedicated protocol, user can read, write, and monitor memory devices of GM7 CPU.

Built-in Cnet of GM7 supports the following functions;

- Read single/continuous device
- Write single/continuous device
- Read the CPU status
- Register monitoring device
- Execute monitoring
- 1:1 connection between LG PLCs { GM7 Base Unit : RS-232C, RS-485(10 Points Unit) }

10.2 User defined communication

User can define an user-defined protocol to communicate with other manufacturer's devices. By supporting user-defined protocol, GM7 series can communicate with various devices have their own protocol.

10.3 Modbus protocol

GM7 series includes Modbus protocol, and it is easy to connect with Modbus devices. (You don't need to write Modbus protocol as user-defined protocol)

REMARK

Please refer the chapter 8 of GM7 user's manual for details of built-in Cnet I/F function of GM7 series.

11. Other Internal Functions

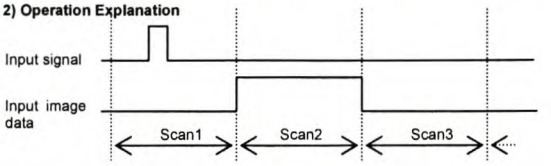
11.1 Pulse Catch Function

In the base unit , 8 points of pulse catch input contact points (%IX0.0.0 □ %IX0.0.7) are internalized. Through using this contact point short pulse signal , short as 0.2ms , can be taken which can not be executed by general digital input.

1) Usage

When narrow width of pulse signal is input , a trouble occurs which can not be executed by general digital input, so the operation does not perform as user's intention. But in this case through pulse catch function even narrow interval of pulse signal as 0.2ms min can be executed..

2) Operation Explanation



Step	Execution contents
Scan1	CPU senses input when pulse signal, min. 0.2ms, is input, then saves the status.
Scan2	Used to turn on the region of input image.
Scan3	Used to turn off the region of input image

11.2 Input Filter Function

External input of GM7 selects input on / off delay time. From the range of 0 □15ms. Credibility secured system may be established by adjustment of input correction No. through using environment.

1) Usage

Input signal status affects to the credibility of system in where noise occurs frequently or pulse width of input signal affects as a crucial factor. In this case the user sets up the proper input on / off delay time, then the trouble by miss operation of input signal may be prevented because the signal which is shorter than set up value is not adopted

2) Operation Explanation

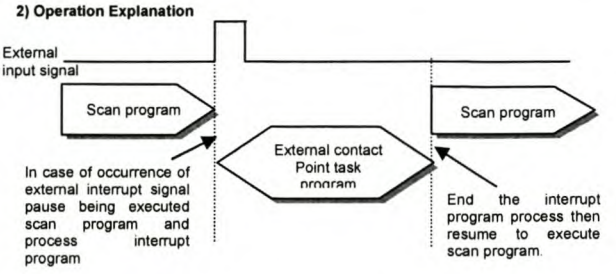
11.3 External interrupts function

In GM7 Series can perform max 8 points of external contact task by using input of base unit without special interrupt module.

1) Usage

This function is useful to execute a task program has been set to an external input signal.

2) Operation Explanation



3) Function

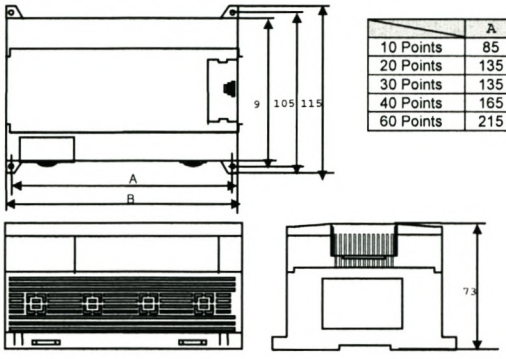
- Maximum 8 points can be used to external interrupt input within %IX0.0.0 to %IX0.0.7
- Inputting 8points of base unit are set functions like following.

	00	01	02	03	04	05	06	07
High speed counter	A Phase	B Phase	Preset	-	-	-	-	-
External interrupt task	•	•	•	•	•	•	•	•
Time driven interrupt	-	-	-	-	-	-	-	-
Internal interrupt task	-	-	-	-	-	-	-	-
8 points are available								

- Max, 8points of external contact point task are available to use. But the no. of them is decreased by using other task.

12. Dimension (mm)

1) Base Unit



	A	B
10 Points	85	95
20 Points	135	145
30 Points	135	145
40 Points	165	175
60 Points	215	225

2)Expansion Module