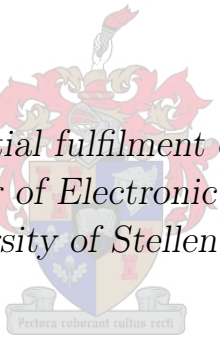
The Design of an Optimal, Dynamic, Multi-hop Telemetry Network

by

Gareth Andrew Nicholson



Thesis presented in partial fulfilment of the requirements for the degree of Master of Electronic Engineering at the University of Stellenbosch

Supervisor: Dr R. Wolhuter

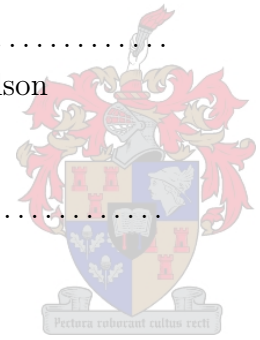
December 2006

Declaration

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

Signature: .....

G.A. Nicholson



Date: .....

Abstract

The Design of an Optimal, Dynamic, Multi-hop Telemetry Network

G.A. Nicholson

Department of Electrical and Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Thesis: MScEng (Electronic)

December 2006

The basic concepts of wide band mobile ad hoc networking are used in this thesis to extend the functionality of a typical low-speed telemetry system. Although most efforts related to wireless networking are driven by the insatiable demand for high-speed mobile data applications, telemetry applications ensure the continuous wide spread use of low-speed narrow band data networks. A multi-hop auto-routing telemetry system has the advantages of extended operational range, and portability, for nomadic monitoring, or rapid deployment applications.

All stations operate both in their telemetry capacities, providing typical functionality and I/O capabilities, as well as implicit routers, supporting a DSR based protocol to achieve dynamic, multi-hop operation.

While the emphasis of work described in the thesis is placed on the optimal design and development of a dynamic, multi-hop telemetry network, a further aspect presented herein is the extension of an existing state transition matrix queuing model to encompass the type of network designed. The model offers a prediction of latency performance of the multi-hop system using half duplex low speed radio links, and corresponds with measured results.

Uittreksel

Die Ontwerp van 'n Optimale, Dinamiese, Multi-hop Telemetrie Netwerk

("The Design of an Optimal, Dynamic, Multi-hop Telemetry Network")

G.A. Nicholson

Departement Elektries en Elektroniese Ingenieurswese
Universiteit van Stellenbosch
Privaatsak X1, 7602 Matieland, Suid Afrika

Tesis: MScIng (Elektronies)

Desember 2006

Die basiese konsepte van wyeband mobiele ad hoc netwerking word in hiedie tesis gebruik om die funksionaliteit van 'n tipiese laespoed telemetriestelsel verder uit te brei. Afgesien van die feit dat die meeste werk wat gedoen word rakende draadlose netwerke, gedryf is deur die aanhoudende vraag na vinniger, hoë-spoed, mobiele data toepassings, verseker telemetrietoepassings egter steeds die deurlopend en uitgebreide gebruik van laespoed nouband data netwerke. 'n Multi-hop outo-roetebepaling telemetriestelsel het die voordeel van verlengde reikafstand, asook draagbaarheid vir nomadiese monitering, of vir toepassings waar vinnige opstelling van die netwerk 'n vereiste is.

Alle stasies funksioneer in beide hul telemetriese kapasiteit, deur die verskaffing van tipiese funksionaliteit en intree/uittree vermoë, asook as implisiete roetebepalers wat 'n DSR gebaseerde protokol ondersteun om dinamiese, multi-hop bedryf te verseker.

Alhoewel die klem in hierdie tesis grootliks op die optimale ontwerp en ontwikkeling van 'n dinamiese, multi-hop telemetrienetwerk gelê word, is 'n verdere aspek wat bespreek word die uitbreiding op 'n reeds bestaande toestandveranderlike matriks rye model, om die omvang van die tipe netwerk wat ontwikkel is te beskryf. Die model voorspel die latente gedrag van die multi-hop stelsel vir die gebruik van half-dupleks laespoed radio verbindings en rym met gemete resultate.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations, without whom, completion of this thesis would not have been possible:

I am indebted to Allen Peterson and Clive Maasch for a invaluable learning experience which provided insight into the field of telemetry networking. Thanks to SSE Data Networks for the generous loan of hardware which was used in the testing process.

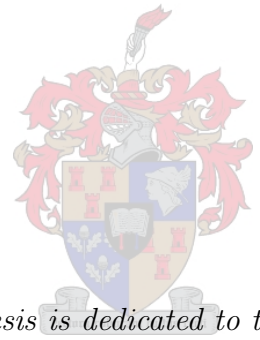
Dr Riaan Wolhuter, my supervisor, for his enthusiasm and valuable discussions, which always seemed to offer "a light at the end of the tunnel". Without his support and guidance this work would would not have been successful. It is much appreciated.

My family, for their constant encouragement, love and wise words.

To Amy, thank you for your patience, understanding, support and endless love.

Lastly, and most importantly, thanks and praise be to the Lord Jesus Christ.

Dedications



This thesis is dedicated to the memory of
Dane Maisch,
who passed away tragically during the final stages of this work.

Contents

Declaration i

Abstract ii

Uittreksel iii

Acknowledgements iv

Dedications v

Contents vi

List of Figures x

List of Tables xiv

Nomenclature xv

1 Introduction 1

1.1 General 1

1.2 Motivation 2

1.3 Primary Objective 3

1.4 Formalization of Objectives 3

1.5 Approach 5

1.6 Overview of Thesis 6

2 Communications Infrastructure 8

2.1 Introduction 8

2.2 Design Considerations 8

2.3 Selection of the Communications Hardware . . . 11
2.4 Modulation Techniques for Narrow Band Radio . . . 13
2.5 Narrow Band Hardware . . . 20
2.6 Error Detection for Flow Control . . . 21
2.7 Error Correction . . . 23
2.8 Hamming Codes Revisited . . . 31
2.9 Conclusion . . . 34

3 Dynamic Network Protocol 36
3.1 Introduction . . . 36
3.2 Mobile Ad Hoc Networking . . . 36
3.3 Ad Hoc Network Routing Protocols . . . 38
3.4 Protocol Design . . . 44
3.5 Flow Control . . . 54
3.6 Conclusion . . . 56

4 Hardware Design 57
4.1 Introduction . . . 57
4.2 PCB Design . . . 57
4.3 Embedded Microprocessor . . . 58
4.4 SRAM . . . 60
4.5 Modem . . . 62
4.6 I/O Module . . . 63
4.7 Radio . . . 67
4.8 Radio Interface . . . 67
4.9 Power Supply . . . 72
4.10 Conclusion . . . 82

5 Embedded Software 83
5.1 Introduction . . . 83
5.2 General . . . 83
5.3 End to End Data Flow . . . 84

5.4 Software Modules . . . 85
5.5 Conclusion . . . 96

6 Server Software **97**
6.1 Introduction . . . 97
6.2 Server Description . . . 97
6.3 Data Display . . . 100
6.4 Conclusion . . . 100

7 System Performance Prediction **101**
7.1 Introduction . . . 101
7.2 General . . . 101
7.3 Routing Set Up Times . . . 102
7.4 Round Robin Polling Strategy . . . 103
7.5 CSMA Strategy . . . 105
7.6 Approach to Modeling an Infinite Event Source . . . 112
7.7 State Transition Matrix . . . 114
7.8 Conclusion . . . 119

8 Measurements and Results **127**
8.1 Introduction . . . 127
8.2 Link Budget . . . 127
8.3 Measured Performance of Kenwood TK-3160 . . . 133
8.4 FX469 Modem Testing . . . 135
8.5 Testing of the Routing Protocol . . . 138
8.6 Measurement of System Latency . . . 138
8.7 Operation as a Telemetry System . . . 142
8.8 Conclusion . . . 142

9 Summary and Conclusions **153**
9.1 Motivation . . . 153
9.2 Summary of Objectives . . . 153
9.3 Summary of Thesis . . . 154

9.4 Future Work and Recommendation 156
9.5 Conclusion 156

Appendices **158**

A Additional Design Information **159**

B System Flow Diagrams **164**

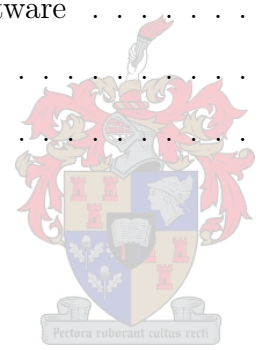
C Circuit Diagrams and PCB Layouts **181**

D Prototype Pictures **190**

E CD-ROM **193**

E.1 Embedded Program Code 193
E.2 Server Application Software 193
E.3 MATLAB Script Files 193
E.4 Relevant Datasheets 193

List of References **194**



List of Figures

1.1 Operation of a simple telemetry system. 5

2.1 BER performance comparison of various modulation schemes. 20
2.2 Shannon Limit bandwidth capacity. 24
2.3 Efficiency of various FEC schemes, for data lengths of 16, 32 and 64 bytes. . 35

3.1 Nodes in a pipelined configuration as implicit routers. 37
3.2 Example of a 802.11 mesh network. 38
3.3 Classification of a selection of routing protocols. 39
3.4 Example of the route cache of A for the given network topology. 50
3.5 Example of a unidirectional link. 52
3.6 Message header structure. 52
3.7 Example of typical header contents. 54

4.1 Hardware block diagram. 57
4.2 SRAM/PIC interface and timing diagram. 62
4.3 AIN circuitry. 63
4.4 DIN/CIN circuitry. 64
4.5 DOT circuitry. 65
4.6 I/O expansion circuitry and simulated timing diagram. 67
4.7 PTT driver circuitry. 69
4.8 TX/RX audio interface circuitry. 71
4.9 Frequency response of the TX/RX audio interface, Attenuation=20dB. 71
4.10 Squelch interface circuitry. 71
4.11 Squelch and RX audio circuitry silencing, with HSPICE simulation results. . . 72
4.12 Full wave bridge rectifier circuit. 73

4.13 Power supply block diagram. 80

5.1 Overview of layers controlled by the embedded software. 85
5.2 Software modules and sub-functions. 86
5.3 Example of how message framing is performed. 89
5.4 Example of the autocorrelation process followed to find the start/stop bytes. . 90

6.1 Screenshot of the original server application. 99
6.2 Screenshot of an example application displaying RTU I/O data. 100

7.1 Pipelined network topology. 102
7.2 Hidden terminal network topology. 106
7.3 Simple single server queue. 108
7.4 Overall event queueing system. 113
7.5 Finite source network state probability diagram, N=4. 114
7.6 RRP mean cycle time for various noise levels, mean hops=1. 121
7.7 RRP mean cycle time for various noise levels, mean hops=2. 121
7.8 RRP mean cycle time for various mean route lengths. 122
7.9 CSMA Theoretical Mean Latency for various noise levels (mean hops=1, no hidden terminals). 122
7.10 CSMA Theoretical Mean Latency for various noise levels (mean hops=1, no hidden terminals). 123
7.11 CSMA Theoretical Mean Latency for various bit cycle lengths (mean hops=1, no hidden terminals). 123
7.12 CSMA Theoretical Mean Latency for various mean backoff periods (mean hops=1, no hidden terminals). 124
7.13 CSMA Theoretical Mean Latency for various mean retry periods (mean hops=1, no hidden terminals). 124
7.14 CSMA Theoretical Mean Latency for various mean route lengths (mean hops), N=10. 125
7.15 CSMA Theoretical Mean Latency for various mean route lengths (mean hops), N=4. 126

8.1 Diagram relevant to the Path Loss Equation. 132
8.2 Extract from Agilent Application Note Agilent Technologies (2000). 135

8.3 Measured FX469 modem BER performance at 1200 baud in a white noise channel. . . . 137

8.4 Received signal power input vs distance. . . . 144

8.5 SNR at receiver modem input vs distance. . . . 144

8.6 SNR at receiver modem input vs distance for various n. . . . 145

8.7 E_b/n_0 of modem vs distance between stations for various n. . . . 145

8.8 Measured Results: RRP for various network topologies and MSG lengths. . . . 146

8.9 Measured Results: 16 data bytes without FEC. . . . 146

8.10 Measured Results: 16 data bytes with FEC. . . . 147

8.11 Measured Results: 64 data bytes without FEC. . . . 147

8.12 Measured Results: 64 data bytes with FEC. . . . 148

8.13 Measured Results: N=4, Retry Timeout=3.5s, Mean Backoff =0.8s. . . . 148

8.14 Measured Results: N=4, Retry Timeout=3.5s, Mean Backoff =2.0s. . . . 149

8.15 Measured Results: N=4, Retry Timeout=5.0s, Mean Backoff =0.8s. . . . 149

8.16 Measured Results: N=4, Retry Timeout=5.0s, Mean Backoff =1.5s. . . . 150

8.17 Measured Results: N=4, Retry Timeout=7.0s, Mean Backoff =0.8s. . . . 150

8.18 Measured Results: N=4, Retry Timeout=10.0s, Mean Backoff =0.8s. . . . 151

8.19 Measured Results: N=4, RT=3.5s, BO=0.8s MRL=1 with hidden terminals. . 151

8.20 Measured Results: N=4,RT=3.5s,BO=0.8s MRL=1.5. . . . 152

8.21 Measured Results: N=4, RT=3.5s, BO=0.8s MRL=2.5. . . . 152

A.1 Extract from UC3906 datasheet explaining the dual level charging cycle. . . . 160

A.2 Extract from UC3906 datasheet explaining the design process for the dual level charger. . . . 160

A.3 SA: Modulated carrier for 20mV$_{p-p}$, 40mV$_{p-p}$, 100mV$_{p-p}$ (1.2kHz & 1.8kHz superimposed). . . . 161

A.4 SA: Modulated carrier for 200mV$_{p-p}$, 500mV$_{p-p}$, 800mV$_{p-p}$ (1.2kHz & 1.8kHz superimposed). . . . 162

A.5 Connection points to Kenwood TK-3160 PCB. . . . 163

B.1 Main program loop. . . . 165

B.2 Server Message Handler. . . . 166

B.3 RX message from server to forward to another station. . . . 167

B.4 Initiate Message. . . . 168
B.5 Send Message Handler. . . . 169
B.6 Store message to Sent Cache. . . . 170
B.7 Retry Control 171
B.8 Receive Message. . . . 172
B.9 RX Message Handler. . . . 173
B.10 Validate Message 174
B.11 Process Received Message. . . . 175
B.12 Process Received RRQ. . . . 176
B.13 Forward Message. . . . 177
B.14 Initiate Response. . . . 178
B.15 Acknowledgement Control. . . . 179
B.16 Scan Send Queue. . . . 180

C.1 Complete schematic diagram of the Power Supply circuitry. . . . 182
C.2 Power Supply PCB Top View, Scale 1:1. . . . 183
C.3 Power Supply PCB Bottom View, Scale 1:1. . . . 184
C.4 Part A of complete schematic Diagram of the RTU circuitry. . . . 185
C.5 Part B of complete schematic diagram of the RTU circuitry. . . . 186
C.6 RTU PCB Top View, Scale 1:1. . . . 187
C.7 RTU PCB Bottom View, Scale 1:1. . . . 187
C.8 Complete schematic diagram of the I/O Module circuitry. . . . 188
C.9 I/O PCB Top View, Scale 1:1. . . . 189
C.10 I/O PCB Bottom View, Scale 1:1. . . . 189

D.1 Complete prototype. . . . 190
D.2 Power Supply board. . . . 191
D.3 RTU board. . . . 192
D.4 I/O Module board. . . . 192

List of Tables

1.1 7 Layer OSI model 6

2.1 Bandwidth efficiency of various modulation schemes. 20
2.2 A selection of Reed-Muller error correcting capabilities. 27
2.3 A selection of BCH error correcting capabilities. 28
2.4 Efficiencies of selected FEC coding schemes. 29
2.5 Maximum channel efficiency. 31

8.1 Measured SNR and SINAD performance of the Kenwood TK-3160 133
8.2 Measured rise and fall times of the Kenwood TK-3160. 134
8.3 Measured performance of the FX469 modem. 137

A.1 Voltage levels relevant to PIC18F452 I/O pins. 159
A.2 SRAM Truth Table. 159
A.3 Radio DB9 Connections 163

Nomenclature

AC	-	Alernating Current
ACK	-	Acknowledge
ADC	-	Analogue to digital convertor
AODV	-	Ad hoc On-demand Distance-Vector
ATM	-	1) Automatic Teller Machine 2) Attempt counter
BPSK	-	Binary Phase Shift Keying
CBRP	-	Cluster-based Routing Protocol
CPFSK	-	Continuous Phase Shift Keying
CPU	-	Central Processing Unit
CRC	-	Cyclic Redundancy Check
CSMA	-	Carrier Sense Multiple Access
CUR	-	Current
DAT	-	Data
DC	-	Direct Current
DIP	-	Dual-in-line Package
DP	-	Digipeater
DPC	-	Digipeater Counter
DSDV	-	Destination-Sequenced Distance-Vector
DSR	-	Dynamic Source Routing
DST	-	Destination
FEC	-	Forward Error Correction
FSK	-	Frequency Shift Keying
GMSK	-	Gaussian Minimum Shift Keying
GSR	-	Global State Routing
GUI	-	Graphical User Interface
HSR	-	Hierarchical State Routing
I/O	-	Input/Ouput
IC	-	Integrated Circuit

IP	-	Internet Protocol
ISI	-	Inter-symbol Interference
ICSP	-	In Circuit Serial Programming
LA	-	Local Address
LCD	-	Liquid Crystal Display
LED	-	Light Emitting Diode
LER	-	Link Error
LO	-	Local Oscillator
LOS	-	Line of sight
MANET	-	Mobile Ad Hoc Network
MOV	-	Metal Oxide Varistor
MSK	-	Minimum Shift Keying
OSI	-	Open Systems Interconnection
PCB	-	Printed Circuit Board
PIC	-	Programmable Integrated Circuit
POL	-	Poll
PSK	-	Phase Shift Keying
QPSK	-	Quadrature Phase Shift Keying
RRP	-	1) Round Robin Polling 2) Route Reply
RRQ	-	Route Request
RSP	-	Response
RT	-	Routing Table
RTU	-	Remote Terminal Unit
RX	-	Receive
SCADA	-	Supervisory Control and Data Acquisition
SRAM	-	Static Random Accessible Memory
SRC	-	Source
TORA	-	Temporally Ordered Routing Algorithm
TVS	-	Transient Voltage Suppressor
TX	-	Transmit
TYP	-	Type
UHF	-	Ultra High Frequency
UID	-	Unique Identifier
USART	-	Universal Synchronous Asynchronous Receiver Transmitter
VHF	-	Very High Frequency
WLAN	-	Wireless Local Area Network
WRP	-	Wireless Routing Protocol
ZRP	-	Zone Routing Protocol

Chapter 1

Introduction

1.1 General

The wireless communication market has expanded at a rapid rate over the past few decades, and shows no sign of slowing down. Research and development is fueled by the ever increasing demand for faster wireless mobile communication, as well as expanding wireless computer networks and satellite based applications. As the available spectrum is finite, the increasing demand for bandwidth forces relevant authorities to continuously revise and improve bandwidth organization, regulation and allocation. Stricter bandwidth regulations and increased network congestion impose a serious limit on data throughput.

Generally, most time and effort dedicated to the wireless data networking field focusses on wide-band high speed applications. Nevertheless, there still exists a definite demand for low speed half duplex networks. Applications where such networks are implemented have a (comparatively) low data throughput requirement; the added cost and complexity of a wide-band link is generally not justifiable. Also, a variety of competitively priced components for these networks is readily available.

As a point of departure a brief description of typical telemetry networks is provided, to indicate where the project under consideration fits into the wireless filed.

1.1.1 Telemetry Networks, a Background

Telemetry can be defined as the measurement and transmission of data by remote sources, and the remote control and management of the sources, by means of telecommunications.

Generally, a telemetry network is used for monitoring the status of various transducers or inputs at a remote outstation. It can also provide a medium to remotely control certain outputs or components at the outstation. Data provided by the telemetry network can

be logged for subsequent analysis to determine trends and facilitate overall system optimization. Telemetry applications generally have non-continuous or random data inputs, as opposed to a voice or visual sources which are more continuous.

A typical telemetry network comprises of at least one master station and multiple remote outstations. A telemetry outstation includes various transducers, a telecommunications interface and a CPU. A master station includes the same telecommunications interface and a CPU, which is usually linked to a server and database to store data. In its simplest form, the server can be considered as a mirrored display of the status of outstation I/O points. It could be a mimic panel with a number of LED and LCD indicators, or a computer based SCADA system which controls the monitoring and logging processes as well as display of alarm conditions.

Telemetry networks are implemented in practice for the monitoring and control of, amongst others, pump stations, water sanitation plants, weather stations, security systems and remote metering points. Typical parameters include flow rates, fluid levels, battery voltages, startup currents, power failures and status of control equipment.

1.2 Motivation

There are many existing types of telemetry systems available, however, most come with a high price tag, and there are few affordable solutions for low end users. Most systems utilize direct links, and often require a costly central repeater to extend the range of the network. Some systems are capable of digipeating messages from one station to another along user defined route stored in memory (the emphasis being on user defined). Also, telemetry systems are generally permanent installations, and few wireless solutions exist to aid temporary monitoring; stand alone data loggers are usually employed for such tasks. These reasons provided the initial motivation for the project:

1. A system capable of multi-hopping would extend operational range of a telemetry network without the need for costly "hilltop" repeaters.

2. Implementation of an automatic routing protocol would make a system ideal for rapid deployment, and nomadic (portable) monitoring applications, as the dynamic system would automatically reconfigure upon changes in network topology.

3. Such a system would provide an alternative to stand alone data loggers. In addition to monitoring inputs, it would provide output control at remote locations.

4. By selecting cost effective hardware, an affordable solution for low end users could be provided.

5. Lastly, no documentation could be found of a previous implementation of an auto-routing protocol for a typical telemetry system, and it was felt that this would form a unique concept for such a system.

To conclude, a niche market certainly exists for a telemetry system that provides the functionality of industry standard systems, while remaining cost effective, easy to manage and portable with the ability to digipeat and handle changes in network topology.

Some examples where such a system could be used are:

- Fault finding
- Environmental monitoring
- Pipeline monitoring
- Ad hoc telemetry systems
- Roadway and railway wayside monitoring
- Reliable replacement for stand alone data loggers

1.3 Primary Objective

Subsequent to the outlining of the motivation, a primary project objective was defined:

> -To design, develop and test a low cost optimized telemetry system, that is capable of multi-hopping by means of an auto-routing protocol, thereby making it dynamic, and suitable for nomadic applications.

Although emphasis is placed mainly on the development and implementation of an auto-routing air protocol, design and development of the entire optimized system will be covered. Optimization is strived for when designing any system. An optimized telemetry system in particular is bandwidth efficient, minimizes system latency, and maximizes effective data throughput.

1.4 Formalization of Objectives

1.4.1 Overall Objectives

1. In order to provide the prototype with all the functionality of similar industry standard systems, the operation and characteristics of such systems would require investigation. This would also provide an indication of the traffic loading of such systems.

2. It would be necessary to select communications hardware suitable for telemetry systems, which would also support the routing protocol.

3. In order to optimize the system performance, investigation of appropriate error detection and correction strategies would be required.

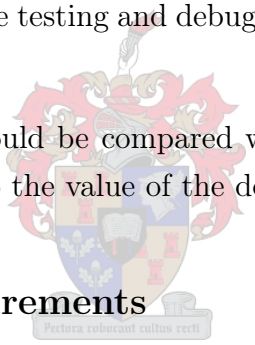
4. Development of a suitable routing protocol should be preceded by a thorough investigation of existing mobile wireless network protocols.

5. The selection of suitable hardware components to produce a fully functional system was considered vital. This would depend on the requirements of the routing protocol, and the telemetry functions of the system. It would be suitable to include a selection of industry standard I/O points in the prototype design for demonstration purposes.

6. It was felt that, although not vital, the design and development of a suitable battery backed up power supply and charger would complement the prototype.

7. A suitable user interface would be required as a point of entry to the system. Such an interface would facilitate testing and debugging of the system in the development stages.

8. Theoretical results that could be compared with measured system latency performance would contribute to the value of the design.



1.4.2 Functional Requirements

The system is not aimed at high speed networks where a high data throughput is required, as is the case when PLCs are used to marshal hundreds of I/O points at each outstation. Rather, this system is aimed at small to medium outstations, such as one would use to control small pump stations, or monitor water flow, or reservoir levels. Typically, I/O counts at such stations vary between 8 and 32 points. The system should provide the same functionality as any conventional system of similar size.

Generally, telemetry outstations can be configured to report I/O events when they occur (contention strategies), or only report I/O status upon interrogation or polling (centrally scheduled strategies). These strategies are explained in more detail later in the text. It was decided to design the system to support both modes of operation. Usually, telemetry stations provide some remote configurability. As this aids in simplifying management, it was decided to provide the system with full remote configurability. This would also aid in testing of the system.

Generally, telemetry systems operate by mapping registers in a database at the server to corresponding registers at outstations. The contents of the registers represent the status

of I/O points at each outstation. Embedded software at each outstation is responsible for linking the local registers and the relevant I/O points. The network is responsible for linking the outstation registers to the database. Application software can run on the server database, displaying the contents of registers graphically, and allowing users to control outputs by changing certain register contents. Thus the end points of the system are the registers at each outstation, and the server database. An example of a simple system is shown in Figure 1.1. To summarize, the functional requirements are:

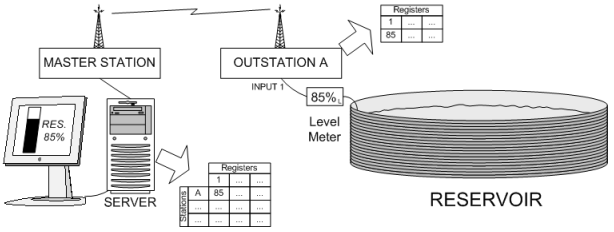


Figure 1.1: Operation of a simple telemetry system.

- Support 8-32 I/O points per station.
- Support industry standard I/O points.
- Support Contention and Centrally Scheduled operating modes.
- Provide remote configurability.
- Support register mapping.

1.5 Approach

It was felt the best way to carry out the design process would be to follow the layered OSI model. Although the model is intended for computer networks, the same basic principles are applicable. The OSI model is based on a layered approach, with each layer providing functions and services to the layer above, while calling on the layer below for more primitive functions. Although the layers are linked, changes can be made within a specific layer without affecting other layers. A description if each layer is provided in Table 1.1 (Heap, 1993).

A bottom up approach is followed, staring with the Physical and Data-Link layers, then moving on to the Network layer. The Transport layer is then developed to provide the upper layers with data form the lower layers. Finally, the upper layers are developed to provide a user friendly interface.

Table 1.1: 7 Layer OSI model

Application Layer:	provides user friendly access to the OSI environment
Presentation Layer:	provides independence to the application process from differences in data representation
Session Layer:	handles communication between co-operating applications
Transport Layer:	handles transfer of data between end points
Network Layer:	provides the upper layers with independence from data transmission technologies; responsible for managing connections
Data-Link Layer:	handles reliable transmission of data across the physical link; manages flow control, error detection and correction
Physical Layer:	concerned with transmission of raw bit streams over a physical medium, and characteristics to access the medium

With reference to the project under consideration, the Physical layer represents the communication hardware, or radio/modem combination. The Data-Link layer entails error detection and correction techniques, message framing and flow control. The Network layer is concerned with the auto-routing protocol. The Transport layer links the end points of the system, i.e. the server database and the I/O registers. In this case, the upper layers consist of the server itself, and any applications that are using information in the server database.

To the best of the author's knowledge, no such auto-routing strategy has been simulated nor implemented in a telemetry system, such as the one under consideration. For this reason it was felt that the best way to test the protocol would be to build several prototype stations, and write scenario specific code to physically test the performance.

1.6 Overview of Thesis

1.6.1 Chapter 2 considers the communication infrastructure, i.e. the design and development of the Physical and Data link layers. Motivation for expected data throughput requirements is provided. A review of modulation techniques and corresponding bit error performance is given. A comparison based on bandwidth efficiency and performance allows for selection of an optimal technique. This is followed by a comparison of error correction and error detection techniques, and motivation for selection of a suitable scheme. Various hardware options including available transceivers and modems are discussed. Motivation for design choices at each level is provided.

1.6.2 Chapter 3 entails the development of the air protocol or Network layer. A review of related protocols is given, and motivation for design choices is provided. The remaining part of the chapter contains an exhaustive description of the implemented protocol, complete with flow diagrams for clarity.

1.6.3 Chapter 4 is concerned with all aspects of hardware design including the power supply and backup battery circuitry, the I/O interface circuitry, communications infrastructure circuitry and the main board itself. PCB layouts and complete circuit diagrams are appended.

1.6.4 Chapter 5 provides a description of the embedded software, which is responsible for integrating all relevant layers, and I/O points, and controlling all on board operations. Complete code listings are appended.

1.6.5 Chapter 6 describes the server, and its functionality. A simple application with a GUI used for displaying data and controlling outputs, is also presented. The concept of register mapping, and the Transport layer, used to link system end points, is described. Both server and application are appended.

1.6.6 Chapter 7 begins with a prediction of routing setup times for a selection of network topologies, followed by a prediction of the mean cycle time for a centrally scheduled RRP strategy for various topologies, data message lengths, and channel noise. The main emphasis of the chapter is placed on the development of a model to predict the latency the system with an infinite event source, operating in contention mode. The model is based on queuing theory and utilizes the state matrix approach. The model development initially follows a documented approach for a finite source narrow band telemetry system. This allows for incorporation of system overhead parameters to provide a realistic result. The model is then extended to cater for an multi-hop infinite event source system. Results of the model are provided.

1.6.7 Chapter 8 describes the protocol and latency testing processes, and measurements and results for the system as a whole. Measurements pertaining to performance of the communications infrastructure (modem, radio and FEC) are used together with a link budget to offer a prediction of the operational range of the system. Measured latency results are documented, and compared to theoretical predictions.

1.6.8 Chapter 9 concludes the thesis with an overall summary, and recommendations.

1.6.9 A list of references is provided, as well as various Appendices, which include circuit diagrams, PCB layouts, code listings and a selection of information pertaining to hardware design.

Chapter 2

Communications Infrastructure

2.1 Introduction

This chapter discusses the design and development of the two lowest layers of the OSI, namely the Physical Layer and the Data-Link layer. The Physical layer is the actual hardware (communications medium) used to transmit raw data across a wireless channel. This entails a wireless transceiver coupled with appropriate modulation and demodulation techniques. The Data-Link layer ensures reliable data transfer across the wireless channel using message framing, synchronization, flow control and error detection and correction techniques. Together, these layers form the communications infrastructure which is used by higher layers to send data across a channel transparently.

Initially, factors that influenced the selection of communications hardware are discussed, including typical system characteristics and functionality like traffic density, range (propagation) and digipeating. A review of applicable modulation techniques is provided together with associated bit error performance. This is followed by a presentation of a suitable error detection/corrction strategy for the hardware and modulation strategy of choice. Flow control and message framing strategies, although touched upon, are covered in more detail in a subsequent chapter, as they are somewhat integrated with the routing protocol.

2.2 Design Considerations

2.2.1 General

Only two constraints existed at the outset with regard to what type of communications medium should be employed for this project, namely low cost, and wireless hardware. Terrestrial radio is preferred for most telemetry applications, as opposed to high cost

satellite links. Terrestrial radio includes any land station to land station wireless link, from low speed half duplex through to spread spectrum high speed full duplex links. A discussion of factors that influenced design choices follows.

2.2.2 Function of a Telemetry System

The purpose of the system is to monitor various inputs and provide output control at remote locations. I/O information must be collected from the remote monitoring stations and stored at a central point to be of any use. Typical telemetry systems employ either centrally scheduled (deterministic) or contention (non deterministic) protocols to achieve this. In some cases both protocols are used in conjunction. Both strategies were investigated, and the system was designed to incorporate both.

2.2.2.1 Centrally Scheduled Strategies

Centrally scheduled strategies are controlled by the master station, and the deterministic nature results in conflict free data transmission. A classic example is the Round Robin Polling (RRP) strategy, where a master station initiates communication by interrogating each outstation in turn according to a schedule. The target outstation responds with the requested I/O data.

2.2.2.2 Contention Strategies

Contention strategies are used when outstations are configured to report certain events to the master station as they occur. Outstations contend for channel access when they have data to transmit to the master station. This is known as Carrier Sense Multiple Access (CSMA). The frequency of events occurring at each outstation directly influences the system loading.

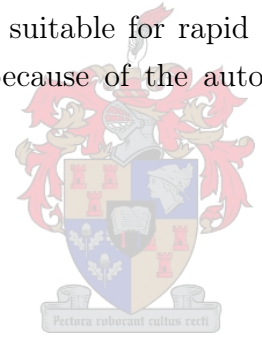
2.2.3 Traffic Density

Of critical importance in network design, is knowing the amount data which a given system will be required to handle. This system is designed with small to medium monitoring stations in mind, aimed at low end users. Typically, this correlates to between between 8 and 32 digital and analogue input and output (I/O) points per station. Some of the widely used industrial protocols required a message space of approximately 2 bytes per I/O point. Thus, the maximum data message length of 64 bytes, or 512 bits, plus the length of the header. Typical industrial protocols have header lengths of several bytes Wolhuter (2002).

It was clear from the outset that the data message lengths of the system would be relatively short. Therefore the traffic density would be directly related to the type of system being monitored, as this would determine the number of stations in the network, and the frequency at which I/O data is exchanged. A system with a very low event rate, for example remote weather monitoring stations, was considered to be of little interest. Rather, the maximum traffic loading that the system could expect was considered. (Wolhuter, 2002) provides motivation for a network wide event rate of approximately 1.4/sec in a water sanitation plant consisting of approximately seventy outstations, each monitoring several pumps. As this represents a fairly large network, it was decided that the maximum network wide event rate would be chosen somewhat lower at 1/sec for design purposes, bearing in mind the system also has to handle routing overhead.

2.2.4 Typical Applications

This system was designed as a nomadic telemetry system. It lends itself to temporary monitoring applications over relatively large geographical areas. Its range is extended by multi-hop capabilities. It is suitable for rapid deployment, and capable of handling changes in network topology because of the auto-routing functionality. Some typical applications could include:



- Pipeline monitoring
- Power line monitoring
- Railway wayside and roadside monitoring
- Environmental monitoring
- Perimeter fence monitoring

In conclusion, it was evident that stations would be spread out over a large area. Distances between stations could range up to several kilometers. It was considered highly probable that stations would not have line of sight links with each other. Therefore, a communications medium with good propagation characteristics was required.

2.2.5 Low Cost

As outlined, low cost is of primary concern. Communication hardware can vary greatly in cost. Availability and durability is also of concern. Frequency band licensing can impose a great cost, due to the limited available bandwidth. License exempt bands are a worthwhile consideration, although power constraints and interference are definite problems.

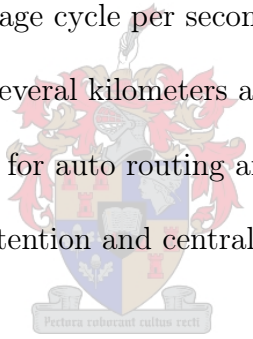
2.2.6 Dynamic Routing Protocol

A primary project objective was to eliminate the need for costly central repeaters, and rather extend the network range using multi-hop techniques. In order for the network to be adaptable, omni-directional antennas were required, that did not adversely affect the operational range to an unacceptable extent. Obviously, directional antennas are not suitable for systems that automatically adapt to changes in network topology, as they would require continual physical adjustment. The auto routing property of the network requires each node to have communication with its neighbours. This means that a single channel must be used as a communications medium if the system is to be capable of automatically discovering routes and broadcasting messages.

2.2.7 Conclusion

From the initial design considerations, the following requirements were evident:

1. Traffic density of one message cycle per second, excluding routing overhead.
2. Good propagation, up to several kilometers and possibly non LOS links.
3. Omni directional antennas for auto routing and digipeating.
4. Capable of supporting contention and centrally scheduled strategies.
5. Low cost hardware.



2.3 Selection of the Communications Hardware

There are various hardware options available for wireless data transmission relevant to telemetry applications, including wide band radio, narrow band radio or GSM and GPRS. Although often used in telemetry applications, GSM and GPRS options were considered unsuitable, as they would not allow project objectives to be met. Many such systems already exist, however, such systems cannot guarantee throughput and latency, as they are dependent on an external network backbone. Relevant to the design considerations presented in Section 2.2, the advantages and disadvantages of wide band and narrow band radio are listed below.

2.3.1 Wide Band Radio

Example: 802.11 WLAN equipment (2.4GHz).

- **Advantages of Wide Band Radio**
  - Wide variety of hardware available ranging in price
  - High data rate achievable
  - License exempt frequencies available (with power limitations)
  - Minimal hardware delays (RX/TX rise times)
  - Full duplex communication possible
- **Disadvantages of Wide Band Radio**
  - High bandwidth required
  - Poor propagation characteristics limit range
  - Costly high gain, generally directional, antennas are required for longer links

2.3.2 Narrow Band Radio

Example VHF/UHF voice grade or digital radios, 12.5kHz channel spacing

- **Advantages of Narrow Band Radio**
  - Cheap analogue or (somewhat more costly) digital data radios available
  - License exempt frequencies available (with power limitations)
  - Propagation characteristics far superior to wide band WLAN
  - Suitable omni directional antennas available
- **Disadvantages of Narrow Band Radio**
  - Half duplex communication
  - Low data rate because of bandwidth limitations
  - Hardware delays (RX/TX rise times) contribute to system overhead

It was clear that narrow band radio was the most suited to the system under consideration, primarily because of the propagation characteristics, but also because of the low cost and simplicity of such hardware. This came as no surprise, as most similar telemetry systems use half duplex narrow band radio links. Although the narrow bandwidth limits the data rate, the required system throughput is relatively low. With regard to telemetry, wide band radios are commonly used in point to point links where the data stream is much higher, for example CCTV or PLCs with many I/O points. The choice of narrow band radio also lends itself to simple frequency variation should it be required. The

carrier frequency can be adjusted by simply reprogramming or changing the radio. Thus a user could utilize license exempt bands, or frequencies for which they may have existing licenses.

Before available hardware is considered, a review of relevant modulation techniques is given. It is not meant to be a comprehensive list, but rather an overview of the most common strategies that can be used in narrow band channels.

2.4 Modulation Techniques for Narrow Band Radio

2.4.1 Modulation

Modulation can be defined as the varying of a sinusoidal carrier signal's amplitude, frequency or phase to represent a signal. Most modern wireless systems use digital modulation, where the modulation variables change in discrete steps with respect to a binary data source. This is in contrast to older methods of analogue modulation where the variables change continuously. These three modulation variables, amplitude, frequency and phase, give rise to the basic building blocks of digital modulation, namely, amplitude-shift keying (ASK), frequency shift keying (FSK) and phase shift keying (PSK).

2.4.2 Channel Noise

In any communications system, error free transmission is ideal. However, in reality, errors cannot be avoided, especially when digital wireless systems are considered. During transmission the signal is subjected to noise (interference). Noise occurs both inside and outside of a communications device. Internal noise can be generated by physical components within a communications system due, in part, to the random motion of thermally energized electrons. External (or channel) noise results from sources outside the actual hardware and includes lighting, radio energy from the sun, static, electromagnetic noise and corona discharge (Pozar, 2001).

Signal degradation in the channel due to random changes in attenuation is referred to as *fading*. Another noise source is multiple transmission paths, commonly known as the *multipath* effect. This occurs when a signal reflects or refracts off some physical medium (buildings, the earth etc.) and multiple reflections arrive at the receiver in a noise-like form. Noise is inevitable in wireless communications system, and its characteristic uncertainty requires the use of probabilistic techniques to predict and optimize performance.

In a digital system, an error occurs when a binary level is incorrectly detected. The more noise that a signal contains, the more likely it is that an error will occur. Thus, the probability of error is proportional to the signal to noise ratio (SNR). It is well known

that the PDF of random noise is Gaussian with a zero mean and that the probability of an error occurring in a noisy channel is given by

$$P_e = \frac{1}{2} erfc\left(\frac{x}{\sqrt{2}}\right) \tag{2.4.1}$$

where x is related to the SNR (Ziemer and Tranter, 2002). Certain modulation schemes perform better in noisy channels than others, but have different signal properties, thus it is not suitable to compare them using the SNR. The most common parameter used to compare communication systems is the ratio E_b/n_0. It can be thought of as the SNR normalised to the bit rate bandwidth. E_b is a measure of the bit energy. It is calculated by dividing the average signal power by the bit rate.

$$E_b = \frac{P_{avg}}{R_b} \tag{2.4.2}$$

The noise density, n_0, is a measure of noise power per hertz, and is calculated by dividing the noise power in the signal frequency bandwidth, by the bandwidth.

$$n_0 = \frac{N}{B} \tag{2.4.3}$$

By substituting x with E_b/n_o in (2.4.1), various modulation schemes can successfully be compared. Each modulation scheme is suited to particular applications, as each scheme offers a trade off between bandwidth efficiency, power efficiency and cost efficiency.

2.4.3 Bandwidth and Data Rate

The global increase in bandwidth demand has led to the reduction of allowable narrow band channel spacing from 25kHz to 12.5kHz, and in some countries to 6.25kHz. In addition to this, sufficient guard bands must be used to ensure that a signal in one channel does not interfere with the adjacent channel. This effectively reduces the available bandwidth even further. For a 12.5kHz channel, the effective usable bandwidth (i.e. that in which the signal energy must be contained) is approximately 7.5kHz (Wood, 2004*a*).

Of primary interest is the maximum achievable data transfer rate in a given channel. The rate at which an analogue signal may change in a channel is determined by the channel bandwidth B. It follows from the well known Nyquist-Shannon theorem that a signal of bandwidth B may change at a maximum rate of $2B$. If each change represents a single data bit, the maximum data rate is given by $2B$. In some cases, each change represents multiple bits. If n denotes the number of bits represented by each change, then the number of possible symbols or levels of change is given by

$$M = 2^n \tag{2.4.4}$$

For a noiseless channel, it follows that the maximum data rate in bits per second is given by

$$R_b = 2B \log_2 M \tag{2.4.5}$$

Noise places a limit on the rate given by (2.4.5). As M strives to infinity, it becomes increasingly difficult to distinguish between symbols because of the presence of noise. Generally, in schemes where multiple bits are represented by a single symbol (M-ary systems), a better signal to noise ratio, and greater amplifier linearity, are required to achieve optimal performance.

The Shannon Limit, or Shannon Capacity, is the theoretical maximum rate at which error-free information can transferred at a given SNR. It can be calculated by the Shannon-Hartley law:

$$C = B \log_2(1 + \frac{S}{N}) \tag{2.4.6}$$

For an infinite signal-to-noise ratio, i.e. noiseless case, the capacity is infinite for any non-zero bandwidth. In practice, channel capacity can only begin to approach this theoretical limit because of the presence of noise, and limited signal power.

2.4.4 Review of Digital Modulation Techniques

Most digital modulation formats are synchronous, i.e. transitions between symbols are synchronized with a reference clock. The clock is inherent to the data and, therefore, recoverable from it. This is preferred as data rates increase. Asynchronous signals do not use a reference clock, but rather rely on special bit patterns to control timing during decoding.

Digital modulation comprising of phase discontinuities between symbols, or non-coherent modulation, produces harmonics in the frequency domain which are undesirable as they increase the required bandwidth, and can cause problems when demodulating. The use of continuous phase modulation (CPM) techniques (coherent modulation) reduces bandwidth requirements, and improves performance by eliminating discontinuities between symbols.

Coherent demodulation makes use of the carrier phase and frequency, mixing the received signal with a carrier-synchronized LO, and low pass filtering to reproduce the original message. Non-coherent demodulation employs band pass filtering and envelope detection to compare the amount of energy in a signalling band to a threshold. Coherent demodulation generally outperforms non-coherent demodulation, however the need for a

synchronized LO makes coherent demodulators more complex and more expensive than non-coherent demodulators (Pozar, 2001).

2.4.4.1 Amplitude Shift Keying

ASK is also known as on-off keying (OOK). The carrier wave is turned on and off according to the binary data sequence. This makes ASK transmitters very simple, and power efficient, because power is only drawn when a binary one is transmitted. ASK performs better if demodulated coherently but it is also possible to demodulate ASK non-coherently, because the modulating signal, and therefore the relevant envelope, is never negative (Pozar, 2001). For ASK signalling, the respective signal power for a zero and one is:

$$S_0(t) = 0,\ m(t) = 0$$

$$S_1(t) = Acos(2\pi f_c t),\ m(t) = 1$$

Although ASK circuitry is simple, ASK performs poorly in a noisy or fading environment because detection depends on the received signal level. It is generally limited to short range low-cost applications, like RFID (Ziemer and Tranter, 2002), and was therefore not considered suitable for this project.

2.4.4.2 Phase Shift Keying

In the case of PSK modulation, the phase of the carrier signal is changed discretely in accordance with the input source, while the frequency and amplitude remain constant. The constant envelope of the PSK waveform makes direct envelope detection impossible, so PSK requires coherent demodulation, increasing the complexity and cost of demodulators. Even after smoothing by filtering, the abrupt phase changes result in a considerably wide spectrum.

BPSK, or binary phase shift keying, is the simplest form of PSK. A binary zero is represented by:

$$S_0(t) = Acos(2\pi f_c t),\ m(t) = 0$$

A binary one is represented a phase shifted version of the same signal:

$$S_1(t) = Acos(2\pi f_c t + \pi),\ m(t) = 1$$

M-ary PSK allows for n bits to coded as one signal. This requires M different phases, as given by (2.4.4). For example, if $n = 2$, then by (2.4.4) four phase states are required. This is commonly known as QPSK. Each phase state (symbol) represents two data bits.

Phase states are separated by $\frac{\pi}{2}$. QPSK can effectively achieve twice the data rate and the same error rate in the same bandwidth as BPSK. The likelihood of a *symbol error* increases with M, however the *bit error* probability remains the same, as it is assumed that a symbol error will usually be caused by a single bit error. PSK has the best BER performance of any digital modulation scheme. However, PSK requires a synchronized LO for demodulation and a fairly wide bandwidth, typically ranging from twice to four times the bit rate. PSK applications are generally used for high performance (and high cost) systems like satellite links and GPS modules (Pozar, 2001). PSK was therefore considered unsuitable for use in this project.

2.4.4.3 Frequency Shift Keying

FSK involves switching the instantaneous carrier frequency discretely as a function of the modulating signal. A logic one is represented by the mark frequency, and logic zero by the space frequency. The modulation signal can be written as the output of two oscillators which are switched between according to a binary source,

$$S_1(t) = Acos(2\pi f_1 t),\ m(t) = 1$$
$$S_0(t) = Acos(2\pi f_0 t),\ m(t) = 0$$

The instantaneous frequency of the carrier will be $f_{inst} = f_c - f_1$ for a binary one, and $f_{inst} = f_c + f_0$ for a binary zero. The difference between the mark and space frequencies is known as the shift. This is the amount by which the instantaneous carrier frequency will change when a transition occurs. The amount that the carrier changes from its original frequency is known as the carrier deviation. It is given by

$$\triangle f = \frac{shift}{2} = \left| \frac{f_0 - f_1}{2} \right| \tag{2.4.7}$$

The modulation index m is measure of the signal bandwidth and is defined as

$$m = \frac{\triangle f}{f_m} \tag{2.4.8}$$

where f_m is the message frequency or symbol (keying) rate. If $M = 2$ then $f_m = R_b$. For narrow band signals, the modulation index is less than one. Generally, mark and space frequencies are chosen so the shift is a harmonic (integer multiple) of the symbol rate. This is advantageous for demodulation techniques, as the original signal can be extracted without external timing information, thereby increasing immunity to multipath effects and LO drift.

Generally, due to simplicity, FSK is transmitted non-coherently. This implies that there is no special phase relationship between elements (the two oscillators mentioned above) of the modulating signal when zero-one or one-zero transitions occur. Phase discontinuities thus occur. This means that changes in instantaneous carrier frequency will not occur at zero crossings, resulting in impulses in the frequency domain, in other words, spectrum inefficiency. The sidebands created by transitions can be reduced by eliminating discontinuities. This technique is known as CPFSK (continuous phase FSK) and is normally implemented using a single frequency modulator, or VCO, which is controlled by the binary source to produce a phase continuous modulating signal.

Demodulation of FSK signals can be performed coherently or non-coherently. For coherent demodulation, two synchronized local oscillators are required, operating at the the mark and space frequencies. In the case of non-coherent demodulation, or envelope detection, the energy in each of the mark and space signalling bands is measured, and compared to determine the most likely received signal.

The effective data rate of FSK can be increased by implementing M-ary FSK techniques, where each part of the modulating wave form carries multiple bits. As M increases, the BER performance improves at the cost of increased bandwidth. Optimal choices are 4-FSK and 8-FSK (Ziemer and Tranter, 2002).

FSK circuitry is slightly more complex than ASK circuitry, but the performance is much better. The error rates for non-coherent and coherent demodulation are comparable, but both are less than that of PSK. FSK has found widespread application in many baseband and modulated data transmission systems, for example data modems and fax.

2.4.4.4 Minimum Shift Keying

A special type of CPFSK, known as MSK, or FFSK (fast FSK), is a continuous phase modulation (CPM) technique. The carrier contains no phase discontinuities and therefore frequency changes occur at the carrier zero crossings. MSK is unique, because the difference between frequencies representing a logical zero and a logical one is always equal to half the symbol rate. In other words, the modulation index is 0.5. This is the minimum separation which allows for the signals to be orthogonal. Orthogonal MSK is desirable for two reasons: Firstly, the minimum shift eliminates phase discontinuities and, secondly, orthogonality simplifies demodulation techniques (Agilent Technologies, 2000) thereby making non-coherent detection comparable to coherent detection (Lee and Messerschmitt, 1988). The result is a spectrum that has a slightly wider main lobe than QPSK, with side lobes that reduce far more rapidly, which increases bandwidth efficiency and noise immunity.

MSK is used in GSM networks, and has found wide application in low speed telemetry type networks.

2.4.4.5 Gaussian Minimum Shift Keying

Although MSK out performs FSK, data rates approaching the theoretical limit can be approached by further reducing the energy in the upper side lobes (Kostedt and Kemerling, 1998). This can be achieved by low pass filtering the data stream prior to presenting it to the modulator (pre-modulation filtering). The low pass filter must have a narrow bandwidth with a sharp cutoff frequency and very little impulse response overshoot. A Gaussian filter is the most suitable because it has an impulse response characterized by a classical Gaussian distribution (bell shaped curve). Hence the name Guassian MSK. The spectrum bandwidth is defined by the pre-modulation filter band width B, and the bit period T. When $BT = \inf$, GMSK is in effect MSK. As the BT ratio is decreased, bandwidth efficiency improves at the cost of noise performance. This occurs because the filter spreads each bit over several periods, and therefore causes inter symbol interference (ISI). GMSK has a BER performance about 3dB worse than PSK with $BT = 0.5$, and about 4dB less with $BT = 0.3$ (Ziemer and Tranter, 2002). If a good signal to noise ratio can be achieved, GMSK provides an excellent solution, and is therefore used by many wireless protocols, including Cellular Digital Packet Data and Mobitex.

2.4.5 Summary of Digital Modulation Techniques

In order to compare strategies that have been discussed, Table 2.1 was compiled and Figure 2.1 produced using information from various sources (Kostedt and Kemerling, 1998; Langton, 2002; Pozar, 2001; Wolhuter, 2005; Wood, 2004*b*; Ziemer and Tranter, 2002). For M-ary PSK and M-ary FSK, the bandwidth efficiency is calculated by dividing the bit rate by the bandwidth required to pass the main spectral lobe (null to null). For MSK and GMSK, the three bandwidth efficiencies are calculated by the dividing the bit rate by the bandwidth containing 90%, 99%, and 99.99% of the spectral energy. The E_b/n_0 ratios are the theoretical values required to achieve a $P_e = 10^{-6}$ (coherent detection). In applicable schemes, non-coherent detection performs slightly worse than the coherent case.

From Table 2.1 and Figure 2.1, bearing in mind that QPSK was in merely included for completion, MSK and M-ary FSK clearly offer the best trade offs between bandwidth efficiency and BER performance of the available options.

Table 2.1: Bandwidth efficiency of various modulation schemes.

Modulation Strategy	**BW Efficiency** $\eta = \frac{R_b}{BW}$	E_b/n_0 [dB] $(P_e = 10^{-6})$
BPSK	0.5	10.5
QPSK	1.0	10.5
FSK	0.25	13.5
CPFSK	0.4	13.5
4-FSK	0.57	10.8
8-FSK	0.55	9.3
MSK	1.28; 0.83; 0.362	10.5
GMSK BT=0.3	1.67; 1.16; 0.92;	14
GMSK BT=0.5	1.45; 0.96; 0.75;	13.5

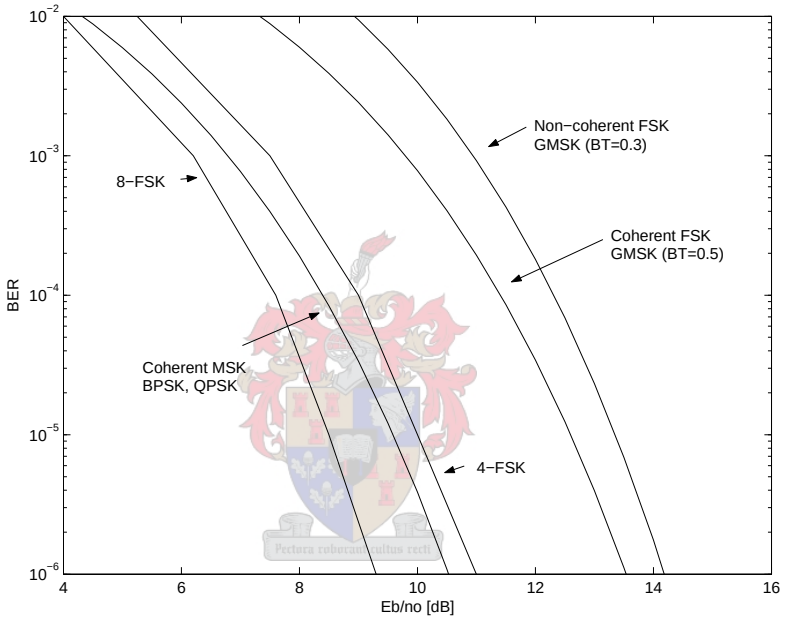


Figure 2.1: BER performance comparison of various modulation schemes.

2.5 Narrow Band Hardware

There is a wide variety of narrow band RF hardware available for telemetry applications. Bandwidth efficiency is generally optimal, and data rates of up to 9600 baud can be realized in a 12.5kHz channel using advanced modulation techniques like M-ary FSK or GMSK. The majority of the modules are intended for short range applications, and have a low RF output power, typically ranging from 1mW to 100mW. Certain modules require an additional RF front end. Good quality data radios with higher RF power outputs are expensive. The carrier rise and fall time is minimal, in the region of a few milliseconds. However, many have on board FEC and flow control, hampering ability to design this particular system in its entirety. For these reasons, it was decided to investigate the use of analogue radios for the project.

Analogue "voice grade" radios, are cheap and readily available. They provide versatility because they can be purchased off the shelf for just about any narrow band carrier frequency. Also, they provide the opportunity to have total control over the design of Data-Link and Network layers. Disadvantages are that a modem will be required to convert binary data into modulating signals, and that analogue radios have carrier rise and fall times in the order of a few hundred milliseconds. There are a selection of modem solutions available that incorporate modulation and demodulation circuitry on a single chip. Modulation types include GMSK, FSK, MSK and M-ary FSK. Modulation is performed at baseband frequencies to generate tones which are then used to modulate the RF carrier of the radio. However, a problem exists when attempting realize high baud rates with analogue radios. A standard radio is designed to modulate and demodulate speech tones, which generally range between 300Hz and 3000Hz. In effect, the audio processing circuitry of the radio is a bandpass filter. Baseband GMSK and Mary-FSK signals have a spectral response extending from DC to several kHz so the audio circuitry of the radio filters out parts of the modulating signal (Kostedt and Kemerling, 1998). Also, certain hardware in a typical radio, including the synthesizer, IF filter and power amplifier, demonstrates non-linear behaviour. Most synthesizers have a high pass characteristic and will not respond to near DC signals. More complex techniques like quadrature modulation, or two point modulation, where the signal is directly injected into the input of the VCO and the master oscillator are required to achieve GMSK and M-ary FSK modulation with voice grade radios (Hunter and Kostedt, 2000). The achieved performance after such modifications was found to be debatable. Therefore, it was decided that the best option would be to resort to tried and tested MSK modulation. CML Microciruits produces a relatively low cost MSK modem IC which is suitable for use in this project. It is capable of producing MSK tones for data rates of 1200, 2400 or 4800 baud. Data rates of 1200 or 2400 baud are achievable in the 12.5kHz channel without any hardware modifications to a standard analogue radio, as the modulation signal remains in the pass band of the radio audio circutry (CML Microcircuits).

It should be kept in mind that the main objective is to implement the routing protocol successfully. If, at a later stage, a higher data rate should be required, analogue radios could be simply replaced with narrow band digital or data radios.

2.6 Error Detection for Flow Control

Error detection is implemented to aid flow control in the Data-Link layer. Error detection is necessary, so that when a corrupted message is received, some form of structured flow control will be followed to organise retransmissions of the data. Error detection requires that additional bits be transmitted, along with the original information bits.

This redundancy is defined as

$$R = \frac{Total\ bits}{Information\ bits} \tag{2.6.1}$$

Efficiency is simply the inverse of (2.6.1). There are various schemes to implement error detection, and a short review of a selection of such schemes follows.

2.6.1 Repetition Codes

The simplest error detection code is known as *binary repetition type code*. A simple example would be that each original information bit is transmitted (sequentially) three times. The receiving station will perform a majority count on each three bit package, to decide what the original bit was. This method is capable of detecting a single error, and has an efficiency of only 33%. Note that repetition codes also lend themselves to error correction. This is discussed in Section 2.7.1.

2.6.2 Parity Check Codes

Given a stream of M bits, a parity bit is appended which represents the number of ones in the stream (odd/even). The decoder will calculate the parity of the received stream M, and compare it to the appended parity bit to determine if an error has occurred. Although the redundancy is low, performance is poor. This code is usually implemented in hard wired links, like PC serial ports, where there is little chance of an error occurring.

2.6.3 Cyclic Redundancy Checking

A far superior, and widely used, method of error detection is known as cyclic redundancy checking (CRC). With this method, a checksum is appended to the original information string. There is a specific way to compute the checksum from the information bits. On the receive side, the checksum is computed on the received information bits, and compared to the received checksum. If there is a difference between the two, it is highly probable that an error has occurred. A 16 bit checksum such as CRC-16 can detect all single and double bit errors, all burst errors less than 17 bits and 99.997% of errors more than 16 bits (Wolhuter, 2005).

CRC coding treats the binary bit string as a polynomial, $M(x)$, with coefficients of either 1 or 0. The algorithm appends a checksum of r bits, corresponding to a polynomial $C(x)$. The original bit string and the checksum together correspond to a polynomial $T(x) = x^r M(x) + C(x)$. $C(x)$ is chosen so that $T(x)$ is divisible by a predefined generator

polynomial $G(x)$ of degree r. A widely accepted polynomial for CRC-16 is

$$G(x) = x^{16} + x^{15} + x^2 + 1$$

If there is a remainder when $T(x)$ is divided by $G(x)$, a transmission error has occurred, and the transmitter must resend the message. Although relatively complex, but extremely powerful, this method is simply and efficiently implemented in software using shift registers and the XOR function. For these reasons, CRC-16 was selected for implementation in this project.

2.7 Error Correction

Various strategies exist that allow original data to be reconstructed from received data where errors have occurred. The Shannon limit, given by (2.4.6) is the theoretical upper limit for error free transmission in a channel, however this limit cannot be reached in practice. Shannon's theorem states that:

> *Given a discrete memoryless channel with capacity C and a source with rate R where R<C, error correction codes exist such that the output of the source can be transmitted over the channel with an arbitrarily small probability of error, thereby approaching the theoretical limit.*

The theorem does not say how to construct such codes, but only tells of their existence. Using (2.4.2) and (2.4.3), (2.4.6) can be written as

$$\frac{R_b}{B} = log_2\left(1 + \frac{E_b}{n_0}\frac{C}{B}\right) \tag{2.7.1}$$

Solving for E_b/n_0 yields

$$\frac{E_b}{n_0} = \frac{B}{C}(2^{C/B} - 1) \tag{2.7.2}$$

Consider the ideal case when the channel capacity equals the bit rate, $R_b = C$. A plot of (2.7.2) is depicted in Figure 2.2 and indicates an important tradeoff: If the bit rate is greater than the bandwidth, then a significantly higher E_b/n_0 ratio, or higher signal power, is required for operation. The region above the curve, where $R_b < C$, is the area where arbitrary small error probabilities are theoretically achievable, according to Shannon's theorem. As the bandwidth increases relative to the bit rate, the curve approaches the asymptote of $E_b/n_0 = -1.6dB$. This indicates the minimum signal power for operation in the $R_b < C$ region.

Error correction schemes also require that additional bits are sent along with the original

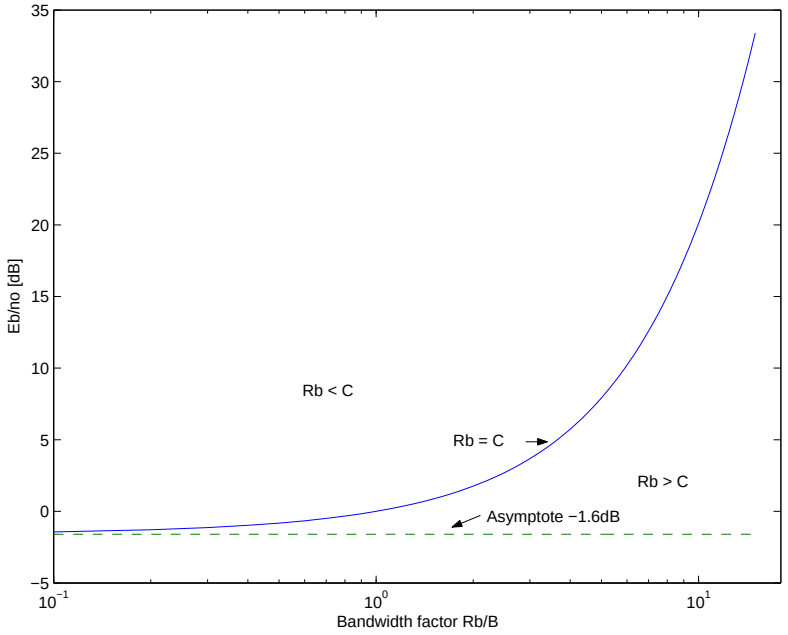


Figure 2.2: Shannon Limit bandwidth capacity.

information bits. The redundancy of (2.6.1) applies, but error correction can improve the system BER performance by adding coding gain. Coding gain occurs because the error correction results in a lower error probability, and therefore an effective SNR increase. There is a trade off between how much redundancy is added (increased message overhead results in longer transmission time) and how many errors can be corrected (coding gain). Error correction codes cannot be designed to corrected an infinite number of errors. Shannon provides an upper bound that shows that if no limit is placed on the length of the codeword (original message and coding overhead), the effective BER will approach zero. However, this may mean that the codeword length is extremely long. This limit is difficult to reach in practice, however significant coding gains can be achieved without excessive overhead increase.

In contrast to Shannon's upper limit, Hamming provides a lower limit on how many additional bits are required to correct a certain number of errors using a technique called minimum distance keying. The Hamming weight of a codeword is defined as the number of binary ones in the codeword. The Hamming distance, d, between two codewords of equal length is defined as the number of bit positions at which two code words differ. It can be calculated by XOR'ing two codewords, and calculating the Hamming weight of this result.

A codeword of length M_c, has M_c possible error positions. The Hamming decoding principle constructs a syndrome using the received codeword. The value of the constructed syndrome must indicate the position of an error. The minimum syndrome bit length, k, is the smallest integer that satisfies $M_c \leq 2^k$. One syndrome value is reserved for the error

free case, assume it is zero, so $2^k - 1$ error positions remain.

From this it can be deduced that, for error correction, the minimum codeword length M_c is related to the original message length M by the following inequality.

$$M \leq 2^{(M_c - M)} - (M_c - M) - 1 \tag{2.7.3}$$

Stated differently,

$$M_c/M = (2^m - 1)/(2^m - 1 - m) \tag{2.7.4}$$

where m is an integer value. Every binary message corresponds to a message space, the size of which is equal to its length. Each point in the message space is a possible value of the binary message. To illustrate, consider a binary message of length 3. This message has a message space of $2^3 = 8$, and each point in the space corresponds to a possible value of the binary message. The space is constructed so that the Hamming distance between adjacent points is one, i.e. the values differ by one bit.

Assume that it is necessary to transmit a single data bit, represented by a codeword of length three. A binary 0 will be represented by "000", and a binary 1 by "111". Out of the eight possible values that the codeword could have, only two are valid. This leaves six codewords unassigned, or invalid. Note that the Hamming distance between the assigned codewords is three, as they differ by three bits. Upon reception, the receiver will attempt to match the received codeword to the nearest of the two valid codewords. "000" has a Hamming distance of one from "001",010" and "100", as they differ by one bit. If any of these words are received, the decoder will assume the nearest valid codeword is the correct one. Thus if a single error occurs, it will be corrected, but if multiple errors occur, they will not be detected. It is obvious that if the Hamming distance between the assigned codewords is less than three, the system will be incapable of correcting errors. This example can be extended to messages of any length, and it follows that for a coding scheme to be capable of correcting e errors, the Hamming distance between the any of the assigned (valid) codewords cannot be less than $2e + 1$:

$$d \geq 2e + 1 \tag{2.7.5}$$

It follows that if d is odd, all received words can be assigned to a valid codeword. However, if d is even, a received codeword can lie between two valid codewords. For this case, errors cannot be corrected.

Consider a binary source that produces a bitstream at rate R_b. Assume that the bits are produced in blocks, of bit length M. A check word of length k is constructed and appended to each block to produce a codeword of length M_c. Such an encoder is said to produce an (M_c, M) block code. Block codes can generally be implemented using

simple and efficient techniques, and therefore do not require excessive processing power. Codewords are generated by the decoder by applying a mathematical function to calculate the check bits. The mathematical process is reversed at the decoder to recover the original data. Systematic codes are codes where original message bits appear unchanged as part of the codeword. Non-systematic codes have the original message bits interleaved in the codeword. A code is said to be linear if the addition of two code words results in another valid codeword. Block codes are based on the theory of finite fields, or Galois Fields. These are fields that contain a finite number of elements. A finite filed has the property that any arithmetic operations on field elements always result in an answer that is in the field (Wolhuter, 2005; Ling and Chaoping, 2004). A code is cyclic if any shift of a codeword results in another codeword. Cyclic codes make use of polynomials in a finite field. The encoder multiples a message function, $M(x)$, by a generator function, $G(x)$, to produce a codeword function, $M_c(x)$. Upon reception, the codeword function is divided by the generator function to produce a remainder, $E(x)$, which corresponds to error positions is the received word. Cyclic codes lend themselves to relatively simple implementation with shift register and XOR operations.

Other types of codes used for error correction are convolutional, trellis and turbo codes. Instead of performing coding on blocks of data, convolutional coding can be applied to continuous data streams. It is most often used in satellite links where transmission is continuous. Trellis and turbo coding methods have achieved results approaching the Shannon Limit, however the process requires significant processing power which puts a limit on the time available to decoded messages. Such schemes are suited to high data rate applications, where power optimization constraints are critical, for example, deep space programs. For these reasons, such codes were not suitable for the system under consideration, and the focus was placed on block codes. A review of a selection of block coding schemes follows.

2.7.1 Repetition Codes

As previously mentioned, binary type repetition codes can be used to correct errors. Each bit is transmitted n times consecutively, so $M_c = n$. There are $k = M_c - 1$ check bits. This technique produces an $(M_c, 1)$ code i.e it is simply a block code with $M = 1$.There are two valid codewords of length M_c: all ones, and all zeros. The decoder performs a majority rules decision on each received frame of length M_c. This is equivalent to minimum distance keying, so up to $e \leq (0.5(M_c - 1))$ errors can be corrected in a frame of length M_c. Although very simple and powerful, this method has a high redundancy, given by (2.6.1), and therefore a poor efficiency.

Table 2.2: A selection of Reed-Muller error correcting capabilities.

M_c	M	e	M_c	M	e
16	5	3	128	8	31
32	6	7	256	9	63
64	7	15	512	10	127

2.7.2 Hamming Codes

A Hamming code is a type of parity check code that is capable of correcting a single error. There is a minimum distance of three between valid codewords. A Hamming code exists for each integer value of m in (2.7.4). As m increases, the code rate (M_c/M) approaches unity, but the fraction of bits that can be corrected becomes very small. Each check bit is generated according to the parity of a predefined combination of data bits. Decoding is performed by reversing this operation on the received codeword to produce a syndrome. The syndrome is used to indicate the position of an error. These codes can generally be implemented easily in software with modulo 2 addition (XOR), or in hardware by means of shift registers.

2.7.3 Reed-Muller Codes

Reed-Muller codes are well-known and widely used (Raaphorst, 2003). Reed-Muller codes are non-systematic linear codes. Encoding and decoding is relatively simple, and involves modulo 2 multiplying the original message of bit length M, by a generator matrix with dimension $(M, 2^{M-1})$, Thus the length of the codeword, Mc, is related to the length of the original bit string M by the following equation

$$M_c = 2^{M-1} \tag{2.7.6}$$

The Hamming distance between any two valid codewords is given by 2^{M-2}. It follows from equation (2.7.5) that the error correcting capability is given by

$$e \leq 2^{M-3} - 1 \tag{2.7.7}$$

where M is the length of the original bit string. Table 2.2 indicates the error correcting capabilities and codeword lengths for various message lengths.

Reed-Muller decoding is commonly performed by majority logic. This process is followed because there are multiple check combinations for each bit in the original message.

Table 2.3: A selection of BCH error correcting capabilities.

M_c	M	e	M_c	M	e
15	11	1	63	57	1
15	5	3	63	7	15
31	26	1	255	247	1
31	6	7	255	9	63

2.7.4 BCH Codes

Bose, Chanudhuri and Hocquenghem codes are an extension of Hamming codes, with increased Hamming distance between codewords, therefore capable of correcting multiple errors (Ling and Chaoping, 2004). BCH codes employ primitive (GF) polynomials for the generator function $G(x)$. There is a relationship between codeword length M_c, message length, M and the number of bits that can be corrected, e. It is given by the following two equations:

$$e \leq \frac{M_c - M}{m} \tag{2.7.8}$$

where

$$M_c = 2^m - 1 \text{ for } m \geq 3 \tag{2.7.9}$$

and m is an integer value. A code is usually described by its codeword length, message length and error correcting capability. The processing overhead increases proportionally with the error correcting capability. Table 2.3 indicates the relationship between the codeword length and message correcting capabilities, (M_c, M, e), for BCH codes.

2.7.5 Reed-Solomon Codes

Reed-Solomon coding is a sub-class of BCH codes, and they are therefore closely related (Ling and Chaoping, 2004). However, a Reed-Solomon code is a symbol based code, not bit based like Hamming, Reed-Muller or BCH. Each symbol has a length of s bits. The encoder calculates c check symbols (each of length s) and appends them to the original symbols to create a codeword. The maximum codeword length in symbols is given by

$$M_c = 2^s - 1 \tag{2.7.10}$$

where S is the number of bits per symbol. The symbol error correcting capability of a Reed-Solomon code is given by:

$$e \leq \frac{M_c - M}{2} \tag{2.7.11}$$

A widely used Reed-Solomon code is one where each symbol length is chosen as 8 bits, hence each symbol corresponds to a byte. Then, from (2.7.10) the codeword length is

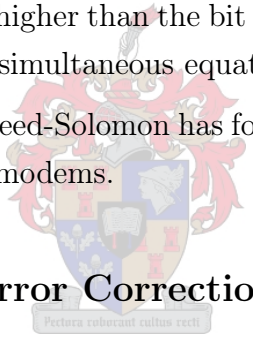
Table 2.4: Efficiencies of selected FEC coding schemes.

	η
(7,4,1)Hamming	0.57
(15,11,1)Hamming	0.73
(31,16,3)BCH	0.51
(16,5,3)Reed-Muller	0.31

calculated to be 255 bytes. If the original word length is 233 symbols, i.e. there are 32 check symbols, then from (2.7.11), up to 16 symbol errors can be corrected out of the 255. Note that any number of bit errors in a symbol can be corrected. This makes Reed-Solomon suitable for correcting burst errors. This coding scheme is widely used in areas of data storage. For example, a scratch on a CD represents a burst error to the decoder reading the CD. Reed-Solomon coding allows the original data to be reproduced, as long as the burst error length does not exceed the capability of the code.

The processing overhead for Reed-Solomon codes is proportional to the error correcting capability, but it is considerably higher than the bit wise coding schemes, as time intensive operations are required to solve simultaneous equations.

Other than data storage areas, Reed-Solomon has found application in satellite links, deep space programs and high speed modems.



2.7.6 Comparison of Error Correction Strategies

A selection of error coding schemes were selected for comparison. The selected schemes along with respective efficiencies are given in Table 2.4.

These particular schemes where selected for the following reasons:

1. The nature of the telemetry system combined with the routing protocol requires messages of variable lengths, ranging from a few bytes, two several tens of bytes. These schemes are chosen so that a message of any length can be sent, without having to excessively zero-pad the original data to make it an integer multiple of the required frame size of the relevant scheme.

2. These schemes are fairly simple to implement, and do not require excessive processing power, therefore a high performance (and expensive) microprocessor is not required.

3. These codes provide a reasonable redundancy/correcting tradeoff.

4. Although not very complex, these codes are powerful.

There are three areas to compare the various coding schemes. Firstly, the coding gain. This can be thought of as the amount by which the signal power of a coded message can be reduced in order to achieve the same BER as the un-coded case. Secondly, efficiency, or code rate (inverse of redundancy). This is a ratio of information bits over the overall codeword length. The third area to compare the coding schemes is a combination of the first two. In order to provide a thorough performance comparison of the different schemes, a real world simulation was created, taking all parameters into account, including zero-padding, message fails, retries and timeouts. The simulation compares the schemes on the average data throughput, or the time that it takes to successfully complete a cycle (data message plus ACK). Although the simulation is capable of handling system overhead parameters, (rise times, processing times and propagation delays etc) they are ignored at this stage, and used later for testing of the system.

2.7.6.1 Simulation to Compare Coding Schemes

The simulation was written in MATLAB, and the basic process was as follows:

A master station would attempt to poll a slave station a predefined number of times for various channel noise levels, and record the mean time that it took to get data back from the outstation at each noise level. The polling cycle consisted of a poll message, and response message. Both messages were subject to corruption, and corruption of either would constitute a polling cycle failure. If the master station had not received a valid response after a predefined time, t_{retry}, the master station would begin the poll cycle again. The simulation was run over many iterations, and at each iteration, the total time (including overhead) to complete a count of successful cycles was logged. Upon completion of all iterations, the average wait time per cycle was calculated. The time it would take to transmit only the raw information bits (t_{data}) was then divided by the average wait time per cycle. The unit seconds canceled out, resulting in a normalised parameter, independent of bit rate.

$$t_{pol} = (b_{HEADER} + b_{CRC}) \times t_b \tag{2.7.12}$$

$$t_{rsp} = (b_{HEADER} + b_{DATA} + b_{CRC}) \times t_b \tag{2.7.13}$$

$$t_{cycle-min} = t_{pol} + t_{rsp} \tag{2.7.14}$$

$$t_{retry} = t_{rsp} \tag{2.7.15}$$

- The outstation (slave) would only respond to a poll, and never initiate a cycle. The master station controlled message flow.
- The maximum efficiency was given by $t_{data}/t_{cycle-min}$, i.e. ratio of data bits time to total bits time. A fixed cycle overhead of 128 bits existed, irrespective of the

data length. It consisted of the two 48 bit headers and two 16 bit checksums. The total bits in a cycle was the sum the fixed overhead, and the data bits. Table 2.5 indicates the maximum efficiency for various data sizes.

- If the master station failed to receive a valid reply from the slave station within time t_{retry} after sending a poll request, it assumed that the cycle had failed, and would re-poll the outstation. If a valid response was received after the second poll, the overall time to get the data from the outstation had increased to $2t_{cycle-min}$.
- Errors followed a random Gaussian distribution, and occurred according to the BER at each E_b/n_0 rate for Coherent MSK.

Figure 2.3 depicts simulation results for various data lengths. The simulation results clearly indicated the trade off between coding gain and efficiency. It was decided that the (15,11,1)Hamming code would be the most suitable for implementation, as it offered a fair trade off between efficiency and coding gain. It was felt that operation in areas where E_b/n_0 is low enough to justify use of a scheme like (16,5,3)RM, would not be encountered, as it is unlikely that the squelch of the radios would open at such levels. The minimum receivable signal level is discussed and measured in a subsequent chapter.

Table 2.5: Maximum channel efficiency.

Data Bits b_{DATA}	Cycle Bits b_{CYCLE}	$\eta_{max} = \frac{b_{DATA}}{b_{CYCLE}}$
128	256	0.50
256	384	0.67
512	640	0.80

2.8 Hamming Codes Revisited

Since (15,11,1)Hamming was selected as the code of choice to implement the system under consideration, a more detailed description of Hamming codes is provided here, while the actual implementation process is detailed in Chapter 5.

A general systematic codeword of length M_c consisting of k data bits and r parity check bits can be written as

$$m_1, m_2, ..., m_k, c_1, c_2, ..., c_r$$

so that $M_c = k + r$. The r parity check bits are chosen to satisfy r linear equations. Written in vector matrix format (Blahut, 2003), where $[M_c]$ is the codeword vector with

dimensions $(1, M_c)$, and $[P]$ is the parity-check matrix with dimensions (M_c, k), the parity bits are chosen so that:

$$[P][M_c] = 0 \tag{2.8.1}$$

If the received word is M_{c-RX}, the syndrome matrix $[S]$ is:

$$[S] = [P][M_{c-RX}] \tag{2.8.2}$$

If $[S] \neq 0$, then the received word is not equal to the original codeword M_c, and at least one error has been made. The received word can be written as a the original codeword modified by the error matrix $[E]$:

$$[M_{c-RX}] = [M_c] \oplus [E] \tag{2.8.3}$$

It follows that

$$[S] = [P][E]. \tag{2.8.4}$$

The inverse of $[P]$ does not exist because it is not square, so [E] cannot be solved directly. However it can be assumed that if a single error occurs, $[E]$ will only contain one non-zero element. The syndrome indicates the position of the error (non-zero element in $[E]$), corresponding to the position of the column in the parity check matrix that the syndrome is equal to, i.e. if the syndrome is equal to the ith column in $[P]$, the the non-zero element in $[E]$ will be at position i, in order to satisfy (2.8.3).

It follows that the parity-check bits matrix, $[C]$, can be written as the product of the parity-check matrix, $[P]$, and the original message matrix, $[M]$.

$$[C] = [P][M] \tag{2.8.5}$$

Then $[Mc]$ is related to $[M]$ by a generator matrix, $[G]$, where the generator matrix is given by

$$[G] = \begin{bmatrix} I_k \\ \cdots \\ P \end{bmatrix} \tag{2.8.6}$$

The resulting form will produce a linear code (Ziemer and Tranter, 2002).

$$[M_c] = [G][M] \tag{2.8.7}$$

To illustrate how Hamming codes work, consider a simple (7,4) Hamming code.

Given four data bits, $M = (a_1, a_2, a_3, a_4)$. Three parity bits are appended to create a codeword M_c of length 7. The parity bits are calculated by modulo 2 addition as

$$c_1 = m_1 \oplus m_2 \oplus m_3$$

$$c_2 = m_2 \oplus m_3 \oplus m_4$$

$$c_3 = m_1 \oplus m_2 \oplus m_4$$

From this, codeword generation can be written in matrix the form of (2.8.7):

$$\begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{bmatrix} \tag{2.8.8}$$

The receiver will calculate the syndrome from the received 7 bit word in the following manner:

$$s_1 = \acute{c_1} \oplus \acute{m_1} \oplus \acute{m_2} \oplus \acute{m_3}$$

$$s_2 = \acute{c_2} \oplus \acute{m_2} \oplus \acute{m_3} \oplus \acute{m_4}$$

$$s_3 = \acute{c_3} \oplus \acute{m_1} \oplus \acute{m_2} \oplus \acute{m_4}$$

It is clear that if there is no error, the syndrome will be zero. From this, syndrome generation can be written in matrix form:

$$\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \acute{m_1} \\ \acute{m_2} \\ \acute{m_3} \\ \acute{m_4} \\ \acute{c_1} \\ \acute{c_2} \\ \acute{c_3} \end{bmatrix} \tag{2.8.9}$$

If a single error occurs, the decoder must calculate the position of the column of the parity check-matrix which has a zero Hamming distance from the calculated syndrome,

and invert the bit in this position. If two or more errors occur, the capability of the code will fail and it will not correct the received data properly.

Assume a bit stream M=1010 must be transmitted. From (2.8.8), the codeword M_c=1010011 is generated. If an error occurs so that the received word is RM_c=1011011, (2.8.9) will generate a syndrome of 011. This corresponds to column 4 of the parity-check matrix, which is in fact the error position in the received word.

2.9 Conclusion

This chapter has described, in detail, the selection process for communications hardware, a modulation scheme and error detection/correction strategies, by presenting relevant available options and selecting the most viable option in each case.

Initially, design considerations were stated regarding the range, functionality, cost, and traffic density expectation of the system. Due to the low data throughput requirement, and the possibility of links of up to several kilometers, narrow band radio was selected over wide band radio because of the superior propagation characteristics available. Narrow band radios were considered suitable as omni-directional antennas could be used to support the auto-routing capability over substantial distance. MSK and M-ary FSK were found to provide the best trade off between bandwidth efficiency and noise immunity. Voice grade radios were selected over digital radios because of their lower cost, and "off the shelf" availability. This ruled out data rates of more than 2400 baud without physical modifications to the radio, because of the band pass characteristics of the audio circuitry of the radio. It was felt that this was acceptable, because of the low data throughput requirement of the system. An MSK modem IC (FX469) was selected for use in conjunction with the voice grade radios. After the hardware and modulation had been selected, an error detection scheme, CRC-16, was selected to aid flow control. Lastly, a (15,11,1)Hamming Code error correction strategy was selected for use, as it provided a fair trade off between coding gain and overhead. It was decided against using schemes with more coding gain, as it would be unlikely that the radio itself would be able to receive signals with a very low SNR.

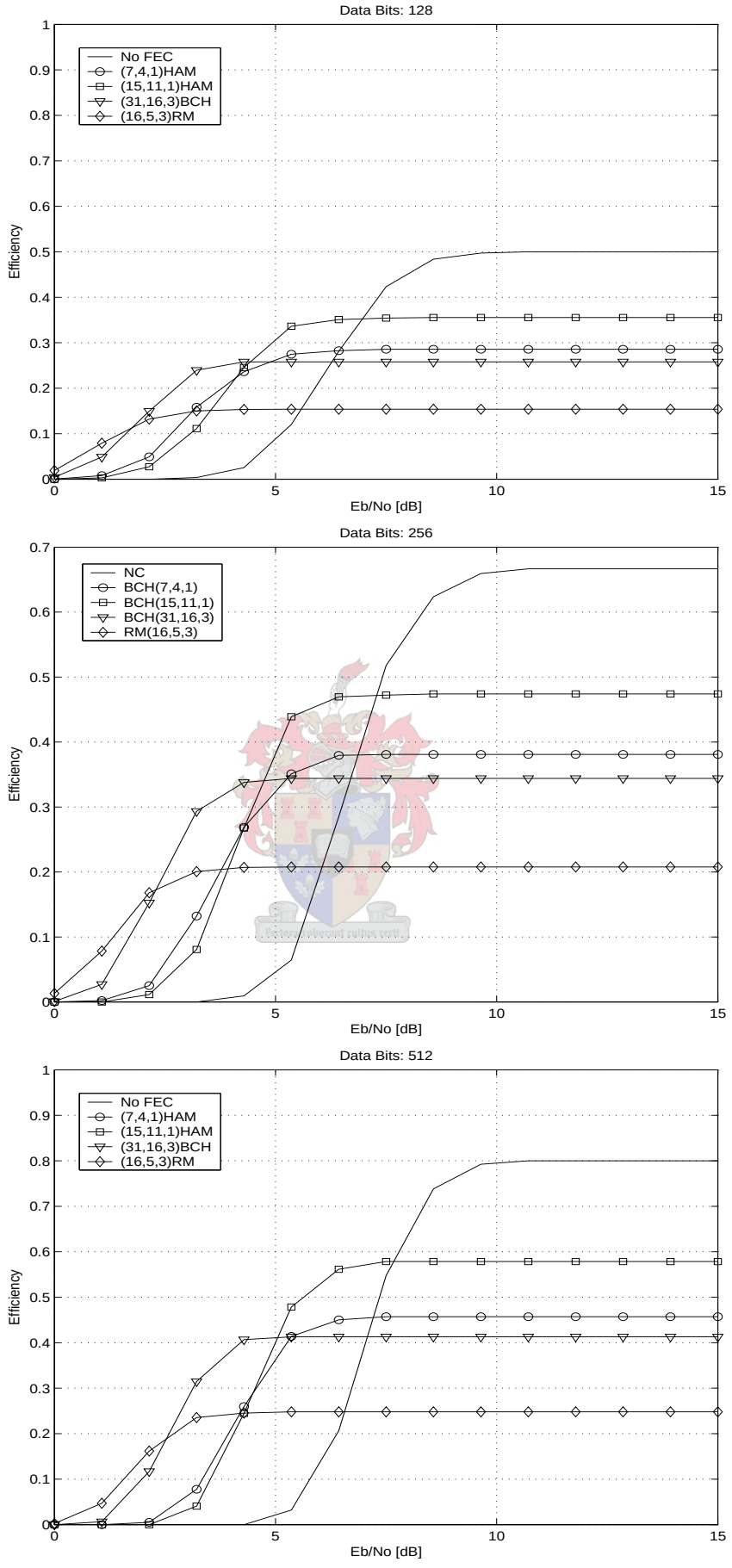


Figure 2.3: Efficiency of various FEC schemes, for data lengths of 16, 32 and 64 bytes.

Chapter 3

Dynamic Network Protocol

3.1 Introduction

A primary objective was to design a system capable of multi-hopping, and handling changes in the network topology. Multi-hop strategies are used to extend the range of a network which is limited by transmission power. The dynamic multi-hop capability of the system demanded a routing protocol enabling stations in the field to automatically find routes to target stations. This protocol is controlled by the Network layer of the OSI model, which is responsible for reliably routing data between stations. This chapter provides a brief description of a selection of routing protocols that are commonly used for mobile wireless data communications.

Generally, the focus of mobile ad hoc network routing protocols is on high speed, highly mobile short range wireless devices, for example, cellphones, laptops, PDAs etc. However, this system is typically intended for temporary telemetry implementation, using low speed half duplex links. Changes in the network topology would be fairly infrequent, as they would only occur if stations were physically moved, or became faulty. Aside from high speed mobile uses, some protocols have found application with Amateur Radio users for transferring data packets, but to the best of the authors' knowledge, there are no protocols used for implementation in a network such as the one in this project. A full description of the implemented protocol, based on Dynamic Source Routing, is included in the chapter.

3.2 Mobile Ad Hoc Networking

Mobile Ad Hoc Networking (MANET) stems from Mobile Packet Radio Networking which was first developed during military research a few decades ago. The emphasis of mobile computing is on mobile IP operation, which allows mobile devices to connect to the Internet. Users demand fast and efficient wireless connections that support data applications.

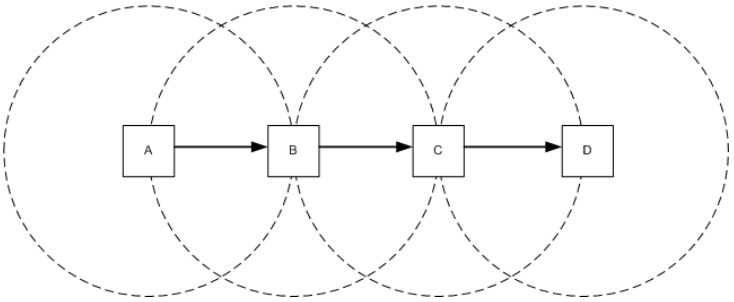


Figure 3.1: Nodes in a pipelined configuration as implicit routers.

Wideband wireless local access technologies (e.g. Bluetooth, IEEE 802.11 WLAN) are used to achieve this, hence most of the wireless protocols discussed in this chapter are aimed at the high speed applications, and this should be kept in mind throughout. Nevertheless, the routing protocol makes the system under consideration suitable for rapid deployment, and capable of handling changes in the network automatically, therefore no user intervention is required to configure digipeating routes.

3.2.1 Ad Hoc Networks

A mobile ad hoc network can be defined as a collection of mobile nodes (devices) that can form an arbitrary network topology without the aid of a fixed networking infrastructure or centralised administration. Nodes in such a network can move around and might leave or enter the network, as and when they please. The network is considered dynamic because it must realign upon any change in the topology. A change in topology could occur when a node enters or leaves the network, moves in and out of range of other nodes in the network, or becomes faulty. The absence of an infrastructure implies that there are no preselected routers in the network, and every node is an implicit router. To illustrate the physical links of a wireless multihop network, consider four nodes which are "pipelined", i.e. each node is only in direct transmission range of nodes on either side of it due to limited transmission power. This is depicted in Figure 3.1. Node A is in range of B, but not C or D. B is in range of A and C, but not D and so on. Assume the network is in a steady state, i.e. the nodes are not moving around.

If A wishes to send a message to D, the message has to traverse the pipeline via B then C. At the embedded software level, each node keeps some sort of routing information, either a table of the entire network topology, or some information about recent routes it has used, or its neighbouring nodes. The information contained at each node defines the type of routing protocol. In the above example, the routing information held at node A would instruct A that any messages to C or D must first be forwarded to B. The routing protocol should be able to adapt quickly to changes in the network, and update relevant

information in the routing modules at each node. This allows the network layer to present the upper layers with a constant, seamless data path between all nodes.

3.2.2 Mesh Networks

A mesh network can be considered as a network that links collections of isolated ad hoc networks. A single node in each ad hoc network is selected as the gateway node. The gateway node links the ad hoc network to the mesh network. Each ad hoc network in effect becomes a stub network, i.e. one that can send and receive traffic directed to and from it via the gateway. However, it does not, or cannot, allow or itself to be an intermediate network for traffic flow. The nodes within each stub can still move about within, or leave the present ad hoc network, or travel to another stub network. Mesh networks are often used with 802.11 WLAN devices, where the gateway is a 802.11 wireless base station (wireless router). Devices (e.g laptops) that are out of reach of the router can use other devices nearer to the router as intermediate hops to the router, thereby creating an ad hoc stub network.

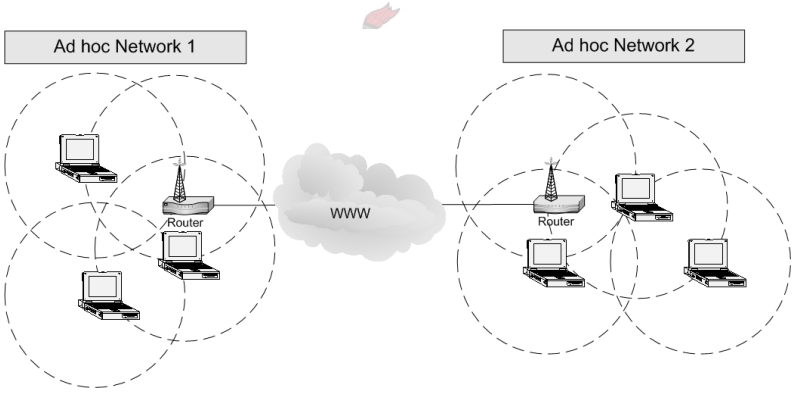


Figure 3.2: Example of a 802.11 mesh network.

3.3 Ad Hoc Network Routing Protocols

Generally, routing protocols can be classified in one of three groups (Hac, 2003):

1. **Table Driven: Proactive**

   Table driven protocols maintain up to date routing information by some proactive strategy. Each node maintains a table containing routes to all nodes in the network. Nodes exchange routing information periodically. For example, periodic advertisements, or "Hello" messages could be sent to make neighbouring nodes aware of each other's presence. The routing overhead is dependant

only on the "Hello" messages, and is therefore independent of which routes are actually being used. The adaptability of the network depends on the frequency of the "Hello" messages. Protocols differ in the manner in which the routing information is propagated across the network and in the number of required information tables.

2. **Source Initiated: Reactive**

   Source initiated protocols are demand driven, or reactive protocols, because a source will attempt to find a route to a destination only when one is required. Generally, only "in use" routes are maintained, i.e. checked if still operational. The source will initiate an explicit route discovery strategy when a route to a destination is required.

3. **Hybrid**

   Hybrid protocols are a combination of both proactive and reactive protocols. The routing information that requires maintenance is minimized by using the advantages of both proactive and reactive protocols; route information for the most frequently used paths is maintained, and any other required routes are found on demand.

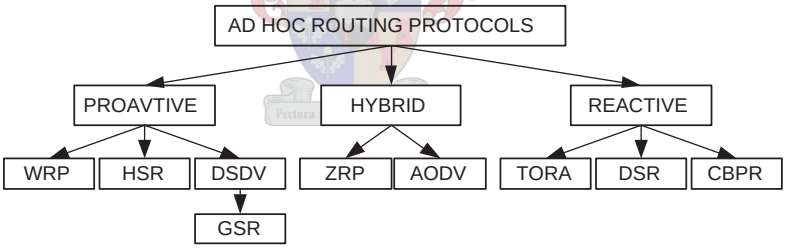


Figure 3.3: Classification of a selection of routing protocols.

3.3.1 Destination-Sequenced Distance-Vector Routing

DSDV stems from the Routing Information Protocol (RIP) which is used in certain parts of the internet. The routing table (RT) at each node contains addresses of all nodes in the network, the address of the next hop in the route to any node, the route metric (total number of hops) to reach each node, and a sequence number indicating the age of each route. Periodic route information update messages are sent by each node. Initially, a hop count in the update message is set to one. This indicates to receiving nodes that the initiator of the message is one hop away. Each receiving node will increment the hop

counter, and forward the update message. This process is repeated until each node in the network has received the update message. Each node must periodically transmit its entire routing table to its neighbours using update type messages. A node also broadcasts its routing table if a significant change has occurred in its table after the last update was sent. Thus, the update is both time and event driven. Receiving nodes will update their tables on the basis of this information. All nodes operate in a promiscuous mode, so they learn from each message they overhear. The node initiating the update message will tag the update message with a incremental sequence number. The purpose of the sequence number is to prevent the occurrence of loops, and prevent the propagation of old update messages. Should multiple messages with the same sequence number and the same source be received, the shortest route will determine which route is stored. Transmission of an entire routing table may require transmission of several messages, which is significant overhead.

3.3.2 The Wireless Routing Protocol

The Wireless Routing Protocol (WRP) is a table-based distance-vector routing protocol. Each node maintains four tables: a Distance Table, a RT, a Link-Cost table, and a Message Retransmission List (MRL). The Distance Table contains the distances to each destination node via each immediate neighbour, as well as the second hop in the route to the destination. The RT contains the distance to each destination node, as well as the nodes before (predecessor) and after (successor) in each route. Predecessor and successor nodes are stored to avoid possible loop problems. The Link-Cost table contains the cost of the link to each neighbouring node and the time since a valid (error free) message was received from each neighbour. The MRL keeps track of which neighbouring nodes failed to acknowledge an update message. Update messages are retransmitted to nodes appearing in the MRL. RT update messages are exchanged between neighbours periodically, and upon detection of topology changes. Nodes exchange idle "Hello" message to ensure connectivity if there is no change in the RT since the previous update. Each time a change is detected, a node will check the validity of links to each of its neighbours.

3.3.3 Global State Routing

Like the WRP, each node in GSR maintains four tables, however, the contents of the tables differ, and the protocol is more similar to DSDV. Firstly, a Neighbour Table contains the address of each neighbouring node. Secondly, a Topology Table contains timestamped information about the state of a link to every node. Thirdly, a Next Hop Table indicates the next hop in the route to every destination node. Lastly, a Distance Table stores the shortest distance to every node.

Route update messages are sent only when a change in the state of any link is detected. If the timestamp of the received message is more recent than the current information in the Topology Table, the Topology Table will be updated with the new information. Upon update of the Topology Table, the node will broadcast the updated Topology Table to its neighbours. The improvement over DSDV is the avoidance of network flooding with periodic route update messages as described in Section 3.3.1. GSR can be further improved by reducing the size of the (possibly) large update messages. These messages can be large, as they contain information about all nodes. They can be reduced to contain information only of nodes that are closer to the initiator. This is known as Fisheye-State Routing (FSR).

3.3.4 Hierarchical State Routing

HSR is a cluster-based algorithm where nodes are divided into groups, or clusters, at different levels. Starting at the lowest level, nodes are partitioned into a series of clusters depending on their proximity to one another. A single node in each cluster is selected as a gateway. In a large network, the gateways can also be grouped into second level clusters, which in turn have their own elected gateways and so on. Each gateway node monitors the topology of its cluster via local link information broadcasts between nodes in its cluster. The gateway node then sends its cluster link information as well as its own link information to nearby gateway nodes. Also, each gateway node will broadcast its link information to its cluster. Thus, each node has information of the network hierarchy. Hierarchial addresses are used to locate nodes anywhere in the network.

3.3.5 Temporally Ordered Routing Algorithm

TORA was designed to reduce routing overhead by discovering routes as required and keeping topology change information local. It is based on a link reversal algorithm to adapt to changes; a message bound for a faulty node will be sent back the way it came by the predecessor of the faulty node. TORA has three operational functions, route discovery, route maintenance and route erasure. Multiple routes to a destination are supported. Consider a network of pipes on a gradient, which can carry water (data packets) from a reservoir (source) at the top of the gradient, to the destination at the base of the gradient. Each pipe represents a wireless link between nodes (pipe junctions). Each node has a certain height on the gradient with respect to the destination. The source has the maximum height, and the destination, the lowest (0). When the source requires a route to the destination, it will broadcast a route query message containing the address of the destination, i.e. lets some water into the pipe network. The water will travel through the network until it reaches the destination, or some node that knows a route to the

destination. Such a node then initiates a reply (update) to the query that contains the its own height with respect to the destination. If the responding node happens to be the destination, the height will be zero. The update is sent to the source along the reverse of the route by which it was received, i.e. back up the gradient. Each node along the way sets its own height to greater the received height of the previous node. Upon reception of an update message each node can construct a Directed Acyclic Graph (DAG). The DAG is a route from itself to the destination, using the heights and addresses of previous nodes. If a node fails, the DAG will become invalid, and route maintenance is required to find a new route to the destination. If a node detects that a route has failed, i.e. the pipe between two nodes becomes blocked, it will set its height to a local maximum, and the water will flow back out the blocked pipe and attempt to find an alternative route to the destination. Invalid routes are removed from the network by flooding the network with an erasure message, instructing nodes to delete the failed route from their routing tables. Each node must maintain information about its neighbour nodes, and its height with respect to these nodes.

3.3.6 Dynamic Source Routing

DSR is a entirely reactive protocol specifically designed for use in MANETs. The source, or initiator, determines the entire route to the destination, and includes the addresses of all intermediate hops in the message header. Although this increases the length of the message, it has the advantage that intermediate nodes do not have to maintain routing information to route received messages. Each node does, however, keep a record of recent routes that the node has learned from promiscuous operation. This is known as the route cache, the size of which is limited by available memory. There are two operational functions of DSR, route discovery and route maintenance. When a source node requires a route to a destination, it will first search its route cache to see if a valid route to the destination can be found. If not, it will broadcast a route request message which includes the address of the desired destination. The message will be forwarded until the destination, or a node that has a route to the destination, is reached. Such a node will respond to the source with a route reply message, which includes the entire route to the destination. Each node that forwards the route request message will first append its own address to the message. Nodes will forward the route request message if (1) they are not the destination, (2) they do not know a route to the destination, i.e. there is no route to the destination in the local route cache and (3) their address is not already in the message. DSR does not make use of periodic update messages. Instead, each node monitors the operation of current links, using hop by hop or end to end acknowledgements. If an error is detected, i.e. a target node cannot be reached, the network is flooded with an update message, and a new route discovery process is initiated.

3.3.7 Cluster-based Routing Protocol

CBRP is based on DSR. Nodes are divided into clusters. If a node is not a member of a cluster, it will continue to periodically advertise its presence until a valid response is received from the nearest cluster head (gateway), so becoming a member of that cluster. Each node has a table containing information about the nodes in its cluster. Each cluster head contains this information, as well as routes to other cluster-heads. When a source requires a route, it sends a request message via its cluster head to other cluster heads. Each cluster head will check if the destination is in their cluster. If so, the cluster head will respond will a route reply message. If not, the request will be forwarded to other cluster heads. A primary advantage of CBRP over DSR is the overhead reduction of the route discovery operation. Only the gateways are flooded with request messages, and not the entire network as with DSR. However, this is countered with the disadvantage of the periodic advertisements required when nodes are searching for clusters.

3.3.8 Ad hoc On-demand Distance-Vector Routing

AODV is a combination of DSR and DSDV. Like DSDV, neighbouring nodes share routing information periodically. However, if a route is required that a source does not have a path to, it is determined the by the broadcast of a route request message by the source. The message is forwarded either until it reaches the destination node, or until it reaches a node that has recent route information about the destination. The route request messages use sequence numbers to ensure that route replies contain the most recent information. Upon reception of a route request, the node will record from which node it received the request, in order to construct a return path to the source. Like DSR, if a link error is detected, the source is alerted by means of an update message, and route discovery can be re-initiated accordingly. An advantage of AODV over DSR is a reduction in overhead, as the source route (addresses of all intermediate nodes) does not need to be included with each message. The next hop is determined by each intermediate node in turn. Also, periodic advertisements can detect changes in the topology more rapidly. However, in situations where nodes have low mobility, or the network topology changes infrequently, the periodic messages are unnecessary. In fact, simulations indicate that the periodic messages of AODV use more bandwidth than is saved from not including source routes in the message (Broch *et al.*, 1998). A variation of AODV, known as Signal Stability-based Adaptive routing (SSA) takes signal quality into account when determining routes. Links with stronger signals are considered better. The signal strength is measured during the periodic advertisements.

3.3.9 Zone Routing Protocol

ZRP is a hybrid of DSR and DSDV. The zone of a node has a radius of n hops, and therefore contains all nodes up to n hops away. DSDV is used to maintain the zone of each node. Thus a node has routing information to all nodes in its zone. The nodes on the edge of the zone are called border nodes. If a source requires a route to a destination outside of its zone, it will transmit a DSR route request to the border nodes. The border nodes will in turn search for the destination in their own zones, or forward the request to their own border nodes. Although periodic advertisements are employed in zones, on demand routing over head is reduced as it only involves the border nodes. ZRP would have identical characteristics of DSDV if the zone radius was infinite, and characteristics identical to DSR if the zone radius was equal to unity.

3.3.10 Concluding Remarks

Bearing in mind that the telemetry system under consideration is nomadic rather than mobile, node mobility can be considered to be very low. Table driven protocols are more suited to networks where nodes have high mobility because the link status is continuously updated and the routing table can quickly adapt to the new topology. However, for networks with low mobility nodes, periodic advertisements messages become unnecessary overhead.

DSR is the most suited to networks with low mobility nodes, and particularly this system, for several reasons. Firstly, the overhead caused by unnecessary periodic routing information messages is avoided. Secondly, the simplicity of the route cache method minimizes required processing power, as there is only a single table that is maintained when required, and not continuously. Thirdly, although the inclusion of the full route in the data packet increases overhead, the routing algorithm does not have to run at every node each time a packet is received; another reduction in required processing power. Lastly, the additional overhead caused by including the source route in each message header is minimal when compared with the hardware rise times of the analogue radios. Once network topology has reached steady state, the routing overhead is minimal.

3.4 Protocol Design

The Dynamic Source Routing as described in Section 3.3.6 is a simple and efficient reactive routing protocol designed specifically for use in multi-hop MANETs (Johnson *et al.*, 2000). DSR allows the network to be completely self-organising and self-configuring, requiring no central network administration. The basis of this protocol was used for implementation in this project. As a starting point, some important considerations are highlighted.

3.4.1 Initial Considerations

- All stations are willing to participate fully in the protocol.
- Only messages that do not contain errors will be passed from the Data-Link layer to the Network layer.
- Mobility is assumed to be low, because of the nature of the network. However, stations may be moved at any time. The speed at which the station is moved is considered to be slow when compared with the transmission latency.
- Although DSR can support uni-directional links, this feature is not supported here. Uni-directional links occur when a certain station, A, can receive transmissions of another station, B, but B cannot receive transmissions from A, due to channel effects like multipath. Some MAC protocols like IEEE 802.11 only support bi-directional links because of the requirement for RTS/CTS handshaking, and link level acknowledgement (IEEE Computer Society LAN MAN Standards Committee). The convention of only allowing bi-directional links was followed here for the following reasons:

  1. The inclusion of the source route in each message header provides a simple route reversal method whereby the route back to the origin of the message can be obtained, reducing required processing overhead.
  2. It was assumed that uni-directional links would not be commonly encountered in the type of network under consideration, because of the propagation characteristics of narrow band radio.

  Although uni-directional links are not supported, care has been taken to handle the occurrence of such a case, and this is discussed later in this chapter.

- Stations can learn routes from messages that they receive as an intermediate node, or simply by overhearing messages. Thus, stations operate in a promiscuous mode, meaning that all received messages will be handed to the Network (protocol) layer, without being blocked or filtered by the Data-Link layer (unless, of course, the received packet contains uncorrectable errors).
- Each intermediate node involved in a route is responsible for ensuring the message is correctly received by the next node in the route. Note that this was chosen only to apply to defined routes, and not to the RRQ case. With a RRQ, only the initiator is responsible for flow control, i.e. waiting for a RRP, and retransmitting the RRQ if no such reply is received with a ceratin period. Routers do not wait for acknowledgement once they forward a RRQ message, as they could wait or retransmit in vain should they be on the network edge. Thus, the Network layer

has been designed to inform the Data-Link layer whether it is necessary to expect an acknowledgement or not.

3.4.2 Route Discovery

The source (SRC) will initiate the route discovery sequence by broadcasting a single route request (RRQ) message, containing the address the destination (DST). Ideally, all neighbours of the SRC will receive the request, and each receiving neighbour takes some action to help the SRC locate the DST. However, if all the neighbours were to respond simultaneously, the messages would collide, and the SRC will not receive any valid messages, so some rules must be followed to prevent such an occurrence.

- If the station is the DST, it will respond immediately with a route reply (RRP) message.
- If the station is not the DST, but it has a route to the DST stored in its route cache, it will delay its RRP message for a random time, the mean of which, is proportional to the number of hops away it is from the desired DST. In this manner, the DST will have the first chance to access the channel, followed by stations with shorter routes to the DST. If a station hears another station's RRP before it has sent its own, it will cancel the sending of its RRP message. The random time will minimize the possibility of collisions between stations having a similar length route to the DST.
- If the station is not the DST, and it does not have a route to the DST, it will append its own address to the received RRQ, and prepare to forward the message to its neighbours. The station will delay transmission for a random time (with a greater mean than the RRP case). In this manner, stations that have a RRP are given a chance to respond first. If it overhears any stations RRP, it will cancel forwarding the RRQ, thereby preventing unnecessary and wasteful propagation through the network. The random time will minimize the possibility of collisions between multiple stations which wish to forward the RRQ.

In order to avoid loops in routes, and unnecessary message propagation, some rules were developed which are highlighted here.

- Each message cycle (RRQ + RRP) is tagged with a unique identifier (UID) byte, which is valid for the entire message cycle. The UID is included in the RRQ and RRP message headers.
- A station will disregard a RRQ message which its own address is already in the message header as a digipeater to prevent loops occurring.

- Each station keeps a record of the recent RRQs that it has forwarded in a sent messages cache. Upon reception of a RRQ, the station will first check if it has not already (recently) handled a RRQ from the same SRC with a matching UID. Consider an example to illustrate the undesired effect that this UID check would prevent in network of four nodes, A, B, C and D.

  1. A initiates a RRQ bound for DST D, which can only be reached via stations B or C $[A \rightarrow ?]$.
  2. Stations B and C receive the broadcast RRQ simultaneously, but neither have a route to D. $[A \rightarrow B]$, $[A \rightarrow C]$
  3. Both B and C delay forwarding their RRQs for unique random periods, of which B's expires first.
  4. B proceeds to forward the RRQ with its address appended. $[A \rightarrow B \rightarrow ?]$
  5. Without a UID check, C will receive the broadcast message from B, and queue-to-forward this RRQ too. Similarly, B will rehandled C's first message.
  6. If D does not respond immediately with a RRP, or the response collides, it is possible that a total of four messages will arrive at D:
     a) $[A \rightarrow B \rightarrow D]$
     b) $[A \rightarrow C \rightarrow D]$
     c) $[A \rightarrow B \rightarrow C \rightarrow D]$
     d) $[A \rightarrow C \rightarrow B \rightarrow D]$

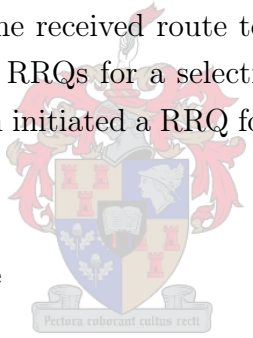
  The incorporation of a UID check will prevent unnecessary rehandling of RRQ messages which result in undesirable cases of c) and d) above. The only case where rehandling a RRQ message with a matching UID is necessary would occur when a SRC has not received a valid RRP to a RRQ after a predefined period, and another RRQ is sent. The RRQ retry will be received via exactly the same path as before, so allowing it to be forwarded again.

- Each RRQ message contains a digipeater (intermediate node) counter (DPC) which defines the maximum allowable digipeaters in the route. It is similar to a time-to-live (TTL) indicator (Johnson *et al.*, 2000). Each receiver will consider how many digipeater (DP) spots are open when it receives a RRQ. If the counter is set to 3, this indicates that the SRC requires a route to the DST which contains a maximum of 3 digipeaters, or 5 hops. Thus, if a station receives a RRQ directly from the SRC, there will be place available for three DPs. Assuming the station has no route to the DST, it will append its address to the message header, taking the first of the available DP positions, and forward the message to its neighbours. The receiving nodes will see that there are still two DP positions available, as the

counter is set to three, and one DP address has been appended. This prevents RRQ messages propagating infinitely through a network, as once the allocated spaces for digipeaters have been filled, the message will not be propagated any further.

When the target DST of a RRQ creates a RRP, it simply reverses the route listed in the header of the RRQ and lists this reversed route in the header of the RRP message. For example if A required a route to D in the case of Figure 3.1 the header of the RRQ arriving at D would contain the route: $A(SRC) \rightarrow B(DP1) \rightarrow C(DP2) \rightarrow D(DST)$. In response, D would create a RRP with a header containing the same UID and the reversed route $D(SRC) \rightarrow C(DP1) \rightarrow B(DP2) \rightarrow A(DST)$ The DPC is set according to how many DP slots are filled, and is used as a reference for the header length when decoding.

When a station initiates a RRQ, it does so because it requires a route to a DST, to send certain data to the DST. The RRQ does not contain any of this data, as this would cause unnecessary overhead. Rather the data is placed in a holding buffer at the initiator, and tagged with the RRQ UID and DST address. Thus, when a valid RRP is received with a matching UID from the correct SRC (intended DST), the data is removed from the holding buffer, and sent along the received route to the destination. The holding buffer can hold many entries, allowing RRQs for a selection of nodes to be sent by the source asynchronously, i.e. the SRC can initiated a RRQ for a second DST before it has received a RRP for the first DST.



3.4.2.1 Route Maintenance

Once a route has been discovered, it must be monitored each time it is used to ensure it is still valid. In order to achieve this, link acknowledgement is required. In other words, each node along a route must ensure that the message it transmits is correctly received by the node it is targeting. Receipt confirmation is easily performed, with no additional overhead, by means of passive acknowledgement. In the case of Figure 3.1, B would receive a passive acknowledgement from C upon hearing C forwarding the message to D. If C fails to forward the message to D, B will retransmit the message to C. If, after a maximum number of retries there is still no passive acknowledgement from C, B will assume the link to from itself to C has failed, and will return a link error message to the SRC (A), listing the link $B - C$ as faulty. A will proceed to erase this link from its route cache, and await instruction from the upper layers regarding whether to initialize another RRQ to D or not. Any other stations that hear the link error message will also remove the link from their route caches.

3.4.2.2 Route Cache

A station will always store any routes that it receives by means of RRP messages where the said station was the initiator of the RRQ. If the station operates in promiscuous mode, the Network layer can extract routes from all messages that the station receives, irrespective of whether the station is acting as a router, or if it simply overhears transmissions of neighbouring stations. There is no fixed definition of how the route cache should be structured or maintained. It was decided to follow a simple approach to implement the route cache in matrix form, and store it as such in memory. The method followed is based on path caching, a technique that has been studied an simulated extensively (Broch *et al.*, 1998; Johnson and Maltz, 1996). Basically, each route received in a header is stored separately, if, of course, such a route does not already exist in the cache. The method is simple to implement, and ensures routes are loop free, as received routes are loop free. It was decided that the route cache algorithm would prefer the shortest route to a node should multiple routes exist to the same node. The basic method was expanded slightly to allow the algorithm to merge routes together, where possible, to create new paths, which would then be stored as separate entries in the cache. For example, assume A has no paths in its route cache, and proceeds to learn a route to E, $[A \rightarrow B \rightarrow C \rightarrow D \rightarrow E]$, by means of Route Discovery. Assume A then overhears F forwarding a message in a route $[G \rightarrow D \rightarrow F \rightarrow x \rightarrow y]$. The algorithm extracts the route $[A \rightarrow F \rightarrow D \rightarrow G]$, and also creates the merged route $[A \rightarrow F \rightarrow D \rightarrow E]$, which provides A with a shorter route to E. Note that x and y are ignored by the algorithm. This is because it was felt the system would be more robust if the route cache algorithm only stored the part of the route which has already been used when it intercepts a transmission, as the validity of the used section of routes are certain. In this case, the used (valid route) is $[G \rightarrow D \rightarrow F]$, while the future (uncertain route) is ignored $[F \rightarrow x \rightarrow y]$.

Another type of route cache organisation is known as link caching (Johnson *et al.*, 2000). Each link that is learned is added to a graph data structure of this station's current view of the network topology. A more complex algorithm (e.g. Dijkstra's shortest-path algorithm) is required to find the best route to the DST from the graph. Significantly more processing power is required for link caching, as opposed to path caching, and it was felt somewhat unnecessary for the system at hand.

The implemented route cache consists of n rows, an m columns, where n is the number of routes that can be stored, and m is the maximum permissible route length of the network (i.e number of hops to the DST). The values of m and n are dependant on available memory and the size of the network. For the purposes of this project, $m = n = 10$, however this can easily be adjusted, and the parameters can be changed from the server. Each row represents a route, and each column in the row represents how many hops away the destination is. In other words, neighbour addresses would be stored in row 1, and a

node reachable via a particular neighbour would be stored in the same row, but in the second column.

The concept is simply illustrated by a diagram and a corresponding route cache, shown in Figure 3.4. Assuming the network has stabilized, i.e. all routes have been discovered. The route cache of A is depicted as it is stored in memory. A has direct links to B, D and E and these addresses are stored in the first column of the route cache; they are one hop away. A has links to C and F via B, therefore C and F are stored in the second column preceded by B in the first column. The rows are not sorted in any particular order, however the route cache maintenance algorithm keeps track of the age of the routes, and will replace the older ones as required.

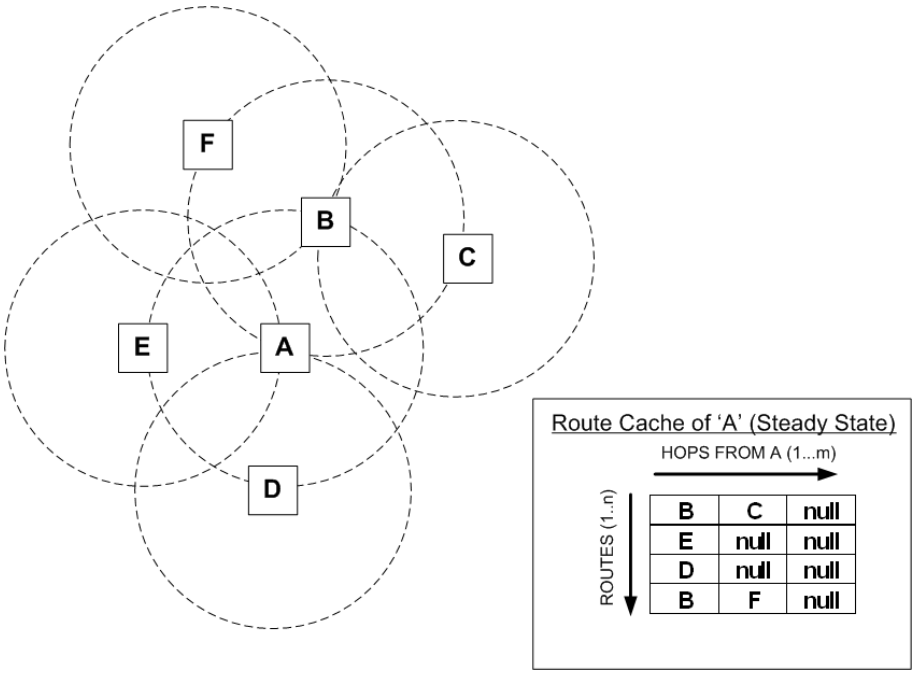


Figure 3.4: Example of the route cache of A for the given network topology.

The route cache algorithm performs the following tasks:

1. Keeps track of the age of the routes, and replaces the oldest route if a new route is discovered, and all rows are occupied.

2. Searches the route cache for redundancy, duplication, erasures and loops and erases as necessary.

3. When a route to a station is received, the route cache is searched for an existing route to the station. If the received route is found to be shorter than the existing route, the existing route is replaced. All other dependant links are adjusted accordingly so that no routing information is lost during replacement.

4. When a RRQ is received, the route cache is searched for a route to the destination that is compliant with the received DPC constraint.

5. When a LER message is received, the faulty link is removed from the route cache. All links to nodes dependant on the removed link are also removed.

As previously mentioned, uni-directional links can cause a problem if the station operates promiscuously, learning routes from overhearing. Assume there is a simple network of three nodes as depicted in Figure 3.5. A shares a bi-directional link with B, and B shares a bi-directional link with C. However, for some reason (e.g. multipath effects), C can receive A's transmissions, but A cannot receive C's transmissions, so a uni-directional link exists between A and C.

Assume C wishes to send data to A, and requires a route to A. C will initiate a RRQ, which A will not receive directly, but instead, via B. When A responds with a RRP targeted at B, C will receive this transmission directly from A and assume (because only bi-directional links are supported here) that it has a direct link with A. Subsequently C will receive the correct RRP ($[C \rightarrow B \rightarrow A]$) as digipeated by B. The route maintenance algorithm will disregard the new route, as a shorter "valid" route already exists in the route cache $[C \rightarrow A]$. When C sends its data directly to A, without using B as a router, it will not receive any response from A, and the link will be deemed invalid. The entry A will be erased from the route cache. A new RRQ will be initiated, and the process will be repeated.

In order to overcome this the following was done:

Two modes of station operation were defined. The first allows the station to learn routes from all received messages, while the second only allows the station to acquire routes by means of route discovery (RRQ/RRP) initiated by the station.

Initially each station is in the first mode. If a link failure occurs to a neighbouring DST (i.e. the SRC believes it has a direct link to the DST) the station will remove the DST entry from the route cache, instate the second mode of operation, and initiate a RRQ to the DST. The SRC will remain in this mode of operation for a fixed (lengthy) period, after which it will revert back to the first (promiscuous) mode of operation. Note that this action is only taken by a SRC node. Intermediate nodes will return LER messages to the SRC if a link is found to be invalid as discussed previously.

3.4.3 Header Construction

There are no rigid data structures that must be adhered to when implementing the DSR protocol. Thus the header header is designed specifically for the system under consideration; to provide for normal operation of the telemetry system, and, support the routing

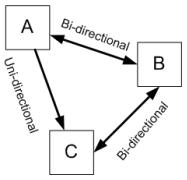


Figure 3.5: Example of a unidirectional link.

protocol. As a point of departure, necessary header contents for relevant layers were considered. The Data-Link layer is concerned with flow control while the Network layer is concerned with the entire route.

The Data-Link layer would require:

1. Unique message identifier to allow for asynchronous transmissions.
2. A retry attempt counter for flow control purposes.
3. A message type identifier.

The Network layer would require:

1. Unique message identifier to allow for asynchronous transmissions, and RRQ/RRP matching.
2. SRC address
3. DST address
4. Variable header length in accordance with the route length.
5. Intermediate node/s address/es
6. Indication of header length as the inclusion of routers would vary the header length.
7. Indication of the current transmitter.

The following variable length header was proposed. It has a base length of 48 bits, which can is extended by 8 bits for each additional intermediate node in a route.

UID	TYP	ATM	DPC	CUR	DST	SRC	DPS	MSG
8 bits	8 bits	4 bits	4 bits	8 bits	8 bits	8 bits	0-120 bits	0-1024 bits

Figure 3.6: Message header structure.

- **UID:** The Unique ID is a initiator (SRC) generated pseudo random 8 bit number between 0-255 used to tag messages. The UID remains unchanged for the life of the

message cycle to which it was assigned, i.e. a RRP will carry the same UID as the RRQ it serves. In a small network, an 8 bit number is sufficient to differentiate all current messages. This could be scaled to a 16 bit UID to cater for larger networks without much difficulty.

- **TYP:** The TYP is an 8 bit message type indicator. The four upper bits indicate the type group, while the 4 lower bits indicate the message sub-type. A message can be of type:
  - RRQ: Route Request
  - RRP: Route Reply
  - LER: Link Error
  - POL: Poll station
  - RSP: Response to POL
  - DAT: Data message
  - ACK: Acknowledgement for DAT message

  Message sub-types indicate read or write functions of the Transport layer protocol which is described in a later chapter. Thus, all functions of the system, both routing and telemetry, are served with a single message type byte.

- **ATM:** The ATM is a 4 bit counter of how many times (attempts) the message has been transmitted in the current link. The transmitting node (either initiator or digipeater) will set the count to one upon first transmission, and increment it each time the message is retransmitted. On the transmitting side the count is stored along with the rest of the message in the sent messages cache, allowing the sender to determine when the maximum number of retransmission attempts has been reached. On the receive side it proved useful for link monitoring purposes, especially during system testing, in that the receiving node could see how many times the sender had to resend the message due to collisions etc. It was chosen as 4 bits, as it was felt the system should never have a maximum retry count per link of more than 15, as this would result in excessive latency. The maximum retry parameter is configurable.

- **DPC:** For all messages except RRQ, the DPC indicates the number of digipeaters (0-15) included in the header. From this, the length of the header can be calculated, i.e. base length in bytes (6) + DPC (0-15). This saves the need for a either a header end/data start byte or a fixed length header. In the case of a RRQ, the DPC indicates the maximum allowable number of digipeaters, thereby limiting unnecessary propagation of the RRQ. Note that the length of the RRQ will only equal the maximum when a total of DPC digipeaters have appended their addresses.

In other words, null addresses are not transmitted, as this would be wasteful of bandwidth. Once again, 4 bits was deemed sufficient, as it was felt unnecessary to digipeat more than 15 times.

- **CUR:** CUR is an 8 bit address slot in which a station about to transmit (CURrent transmitter) a message inserts their own address. This will indicate to receiving nodes (neighbours) which node they have just received the message from. From this it can be deduced how far along a message is in its route, and which the next target node is.
- **DST:** DST is an 8 bit address slot in which the SRC inserts the address of the intended destination of a message.
- **SRC:** SRC is an 8 bit address slot in which the initiator (source) of a message inserts its own address. Upon SRC transmission, SRC=CUR.
- **DPS:** DPS is an array of 8 bit address slots in which the addresses of intermediate nodes are inserted sequentially. The length of the DPS array can vary between 0 and 15 and is specified by DPC.

In a small network, 8 bit addresses are considered sufficient (0-255 addresses available). However this could easily be extended to scale to larger networks, by increasing the address length, or adding a system address byte to the header.

Below is an example of typical header contents, where C has just received (first attempt from B), a data message that A is sending via three digipeaters B,C and D to E:

UID	TYP	ATM	DPC	CUR	DST	SRC	DPS	MSG
123	DAT	1	3	B	E	A	B; C; D	0-1024 bits

Figure 3.7: Example of typical header contents.

3.5 Flow Control

As previously mentioned, each node in a route is responsible for ensuring that the next target node receives its message. This can easily be done by means of passive acknowledgement. The flow control of the Data-Link layer administrates the timeouts, acknowledgements and retry attempts.

Upon transmission, a node will set a timer of period, t_{retry}, within which it expects confirmation of receipt from the target node. If no such confirmation is received, the message will be resent. The process will be repeated persistently until confirmation is

received, or the maximum number of retries is reached. If the maximum number of retries is reached, and t_{retry} of the final transmission attempt expires, the station will decide that the link has failed. If the station is an intermediate node, it will return a LER message to the SRC listing the faulty link. If the station is the SRC, it will initiate a new route discovery for the target neighbour. t_{retry} is chosen as a Poisson random number, with a mean that includes a slightly inflated system overhead time (rise times, processing time) and transmission time. It is chosen randomly to prevent synchronous collision problems. To illustrate, consider the nodes A, B, and C in Figure 3.1. A and C are out of transmission range of one another, however B is in range of them both. Should A and C happen to transmit to B at the same time, B will not receive either of the messages, and will provide no confirmation of receipt to either A or C, and both A and C will prepare to retransmit. If A and C have identical t_{retry} periods, a collision will occur at B each time A and C retransmit. This is known as synchronous collision, and is avoided by choosing t_{retry} randomly.

Care must be taken when deciding whether a passive acknowledgement should be expected or not, depending on where the node is in the route. For example, in Figure 3.1, assume A knows the route $[A \rightarrow B \rightarrow C \rightarrow D]$ and, wishes to poll D. After C forwards the message to D, it must await D's, response to A's poll, as C is the first node in the return path from D to A. This must be considered as an acknowledgement of receipt of the poll. C is a turn around node, i.e. the last digipeater before the DST on the forward path from the initiator, but the first digipeater on the return path to the initiator. When B's turn comes to forward the returned response message to A, it must not expect any confirmation from A, as the polling cycle is completed when the response reaches A.

End-to-end acknowledgment is employed in conjunction with passive acknowledgement, by the Data-Link layer, to prevent messages from "getting lost" in the network. Consider the case where A initiates a RRQ for B, and none of its neighbours have a route to B. With only passive acknowledgement, A will be satisfied once it hears its neighbours forwarding the RRQ on to their neighbours. However, if for some reason, this RRQ never reaches B, or a node that has a route to B, A will wait indefinitely for a RRP. End-to-end acknowledgement is required to prevent such a case. After a ceratin period of waiting for a RRP, A should decided that the first RRQ has failed, and initiate another RRQ. Obviously, the timeout for end-to-end acknowledgement should be longer than it would take for the message to traverse the entire return route.

Where t_{retry} is used for link acknowledgement, t_{route} is used for end-to-end acknowledgement. In the case of a direct link form SRC to DST (no digipeaters), end-to-end acknowledgement is effectively identical to passive acknowledgement, so t_{retry}=t_{route}. In cases where digipeaters are involved, t_{route} must long enough to cover bi-directional propagation of a message, with the possibility of retries occurring at each node.

$$t_{route} = t_{retry} \times (maximum\ attempts) \times (\text{DPC}) \times 2$$

Note that it is unlikely that t_{route} will expire, and it is merely a safeguard measure used where the SRC requires a response from the DST. Generally, either response from the DST or LER from a digipeater will be received before t_{route} expires.

RRQ messages are an exception to the above as digipeating nodes do not use passive acknowledgement for reasons explained earlier, and therefore do not implement retries at link level. Thus, in the case of RRQ messages, the SRC calculates t_{route} as follows:

$$t_{route} = t_{retry} \times (\text{DPC}) \times 2 \times 1.25$$

The inflation factor of 1.25 is to allow for the case where intermediate nodes enter a random wait period before forwarding the RRQ, or sending a RRP. This allows the DST to respond first, followed by stations with shorter routes to the DST, thereby preventing a storm of RRQ messages which would result in collisions.

3.6 Conclusion

This chapter presented some common protocols used for wireless mobile ad hoc networking. Although these protocols are aimed at high speed, highly mobile devices, the principle behind the protocols is nevertheless applicable to the low speed network under consideration. Dynamic Source Routing was found to be the most suitable protocol, as it is reactive, and therefore requires no periodic updates, which would be unnecessary due to the low mobility rate of the system. Although the entire source route is contained in each message header, the additional message overhead is marginal compared with the hardware rise times of the radio, which is in the region of 200 bit periods per transmission cycle. Inclusion of the source route also requires minimal processing power at each station.

The basics of the DSR protocol as described in Johnson *et al.* (2000) have been adapted to suit the particular system. The implemented protocol, and ensuing message header structure, was described in this chapter. Flow diagrams depicting the complete operation of the protocol are appended.

Chapter 4

Hardware Design

4.1 Introduction

Each station consists of various hardware components which have been integrated to provide a working medium to transport data across the multi-hop network. At the outset, there were many options available regarding the choice of the hardware components. This chapter discusses the selection of, and motivation for, the use of the chosen components, and provides detail design calculations where applicable.

4.2 PCB Design

Three circuit boards were designed using Protel software. The first supported the power supply circuitry. The second was designed as the RTU board, and supported the microprocessor, modem, radio interface, RS232 driver and external memory. The last board was designed and built for demonstration purposes, and supports various I/O circuits.

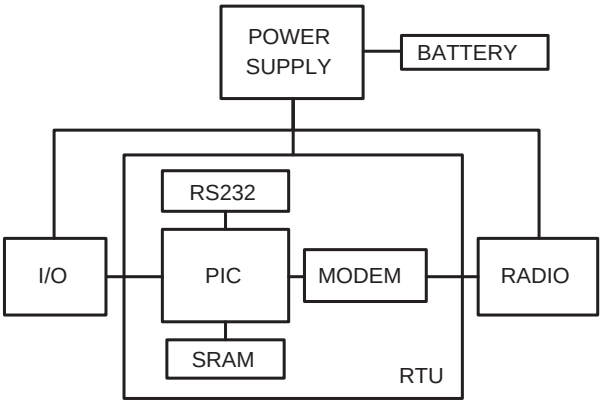


Figure 4.1: Hardware block diagram.

The power supply provides the RTU board, radio and I/O module with 12VDC. The I/O module connects to the RTU board via 10×2 headers and ribbon cable.

Circuit board layouts, schematic diagrams and pictures are appended.

4.3 Embedded Microprocessor

The embedded microprocessor controls all operations of entire RTU. The basic criteria for selection was as follows: The device should be low cost, and have sufficient I/O pins to control all external operations. A substantial number of pins are required to interface to the radio and modem alone. On board A/D convertors and serial communications interface (SCI) support would be preferable, as this reduces overall design cost. The device should be readily available, in a package suitable for low cost prototype design with regard to PCB layout and debugging. For debugging reasons, in circuit reprogramming of the device should be supported. In circuit programming also allows simple embedded software upgrades should the prototypes be implemented in practice. Although there are few operations in the code demanding excessive processing power, the network protocol will require substantial program memory.

The Microchip PIC family of microprocessors is widely available, and well supported locally with regard to programming hardware and development environments. Microchip, as well as several other vendors, provide C code compilers which allow one to write embedded program code in C. This simplifies the design process greatly when compared with low level programming. The Microchip PIC18F452 was selected for use in the prototype for the following reasons:

- Low cost
- 5VDC operation (4.5-6VDC) with a typical current drain of less than 10mA.
- Up to 10MIPs
- 8 and 16 bit timers
- 32KB FLASH program memory (16K instructions)
- 1.5KB RAM, 256B EEPROM
- 8× 10-bit A/D input channels with fast sampling rate (100kHz)
- SCI and I^2C supported
- 40 pin DIP package

- Several external interrupt input pins
- ICSP (in circuit serial programming) supported
- 33 software controllable I/O pins
- I/O pins sink/source up to 25mA
- Local development environment support in the form of Microchip MPLAB (hardware and software)

The PIC18F452 is based on a high performance RISC CPU which supports 75 different low level instructions. The C compiler optimizes, then converts, the C-code into a sequence of the low level instructions (.hex file). The .hex file is loaded into the program memory of the PIC via the in circuit programming hardware, in this case, the MPLAB ICD2. This particular PIC has a software selectable 4×phase lock loop (frequency multiplier of factor four). To achieve an operating frequency of 40MHz, the 4×PLL must be enabled, in conjunction with an external 10MHz crystal oscillator. The effective 40MHz clock translates to a period of $25ns$. Four clock cycles are required for execution of each instruction. Thus, with a 40MHz clock, ten million instructions are executed every second (10 MIPs).

The serial communications interface (SCI), or USART module can be configured in full duplex asynchronous mode, making it suitable for communication with a computer serial port. Pin C6 is the allocated transmit pin, and C7 the allocated receive pin. A software interrupt is triggered upon reception of a byte through pin C7. A standard RS-232 line driver is required to convert the logic levels of the PIC to the RS-232 levels of a computer. Note that a second USART can be defined in software on other pins, with the exclusion of a hardware interrupt.

The 18F452 has 33 I/O pins which are divided into five ports (A-E). Pins operate at various voltage levels depending on the configuration and internal hardware of the PIC. A description of each pin is provided in the device datasheet, however important relevant points are highlighted here. The voltage levels referred to here (CMOS, TTL, ST) are tabulated in the appendix.

- **I/O Port A: 6 pins, A0-A5**
  All pins except A4 have TTL input levels, full CMOS output drivers, and can be configured as AINs (analogue inputs). A4 has a ST input level, and an open drain output (half CMOS). This means that a logic one output on A4 is high impedance. Therefore an external pull up resistor is required to provide a CMOS high level to other circuitry. A low level is generated as normal. A4 cannot be configured as an AIN.

- **I/O Port B: 8 pins, B0-B7**
  All pins have TTL input levels, and full CMOS output levels. B0 through B2 can be used as external interrupt triggers, and require ST input levels when configured as such. B6 and B7 are used for both ICSP and normal I/Os.

- **I/O Port C: 8 pins, C0-C7**
  All pins have ST input levels, and full CMOS output levels. C6 and C7 are used for USART.

- **I/O Port D: 8 pins, D0-D7**
  All pins have ST input levels, and full CMOS output levels.

- **I/O Port E: 3 pins, E0-E2**
  All pins have ST input levels, and full CMOS output levels. All pins can be configured as AINs.

The onboard program memory of sixteen thousand instructions was deemed adequate to house the program code. The onboard EEPROM was considered more than sufficient to store required configuration settings and I/O information. Although the 18F452 has more RAM than many of its counterparts, it was decided that external memory would be demanded by the network protocol, which requires several large caches (Sent Cache, Route Cache, Send Queue etc.). For example, each station must keep track of the most recent messages that it has sent or forwarded for reasons of flow control. If this record is limited to ten messages, and the average message length is fifty bytes, five hundred bytes of RAM will be required, a third of what is available on the PIC itself. This justifies the need for external RAM.

4.4 SRAM

Read and write cycle times were considered crucial when deciding what type of memory should be used. Ideally, the memory access should be as fast as if the PIC were accessing its own (internal) RAM, for functions like searching the route cache. External RAM types that have serial or I^2C interfaces have much longer read/write cycle times (in the order of a hundred microseconds per byte) when compared with devices that employ parallel access. This is unacceptably long for the application under consideration, especially since each data byte must be preceded by relevant address bytes and delays for data setup times. On the basis of cost, read/write cycle time and ease of implementation onto a prototype PCB, a suitable parallel access device was selected. The CYPRESS CY7C109B is a low cost, high speed, low power 128KB SRAM IC. It requires a 5VDC supply, and supports CMOS output and TTL input levels. The device has read and write cycle times in the order of

$20ns$. The 32 pin SOJ packaging has a conveniently small footprint, but nevertheless lends itself to simple surface mount implementation on a prototype PCB. Although 128KB may seem to be too much RAM, it should be noted that this particular IC is far cheaper than many other devices with much less memory space. Also, it was felt rather to have too much RAM than too little, especially since there is no additional cost in the former. The amount of RAM used simply depends on the number of address lines one uses. 16 lines were used here, so 64KB of RAM was available. The disadvantage of the chip is the number of pins that it requires for operation, remembering that there are only 33 I/O pins on the PIC. To access to each of the 64KB locations, a 16 bit address bus, 8 bit data bus, write enable and output enable lines were required. When the output is disabled the data bus is placed in a high impedance state. Note that the chip has dual chip select pins for cases where further memory expansion is required, as is not the case here, so these pins did not require driving from the PIC. The chip is permanently selected by connecting the pins to relevant voltage levels. In order to conserve the number of pins required from the PIC, it was decided to implement an 8 bit bi-directional data bus on port D of the PIC. The data bus would provide high and low address bytes to the SRAM chip by means of logic circuitry controlled by the PIC. Two 74HC374 octal D-type flip flops were used to achieve this. Each octal flip flop has a gate input, which, when pulsed, places the data present on the eight input pins at the eight output pins. Connecting the active low output enable pin to ground permanently enabled the devices. To further conserve the use of PIC pins, it was decided to tie the gate inputs of the two address flip flops together, i.e. drive both gates from a single PIC pin. The data bus from the PIC was connected to the input of the first flip flop. The two flip flops were cascaded so that the outputs of the first are connected to both the lower 8 of the 16 address inputs of the SRAM, and to the inputs of the second. The outputs of the second were connected to the higher 8 of the 16 address inputs of the SRAM. It was found that this configuration simplified PCB layout considerably.

By placing the SRAM in high impedance state, the PIC sets up the 16 bit address in two eight bit steps, each accompanied by a pulse of the flip flop gate pins. The circuit configuration, along with a simulated timing diagram using Altera software, is depicted in Figure 4.2. The time taken to execute such a cycle depends entirely on the PIC, as the maximum propagation delay through the logic circuitry ($\approx 6ns$ from simulation), and the read/write cycle time of the SRAM ($\approx 20ns$) is much less than the time it takes the PIC to execute a single instruction ($100ns$). Thus, no software delays were necessary to allow for data setup times. Cycle time is in the order of $600ns$.

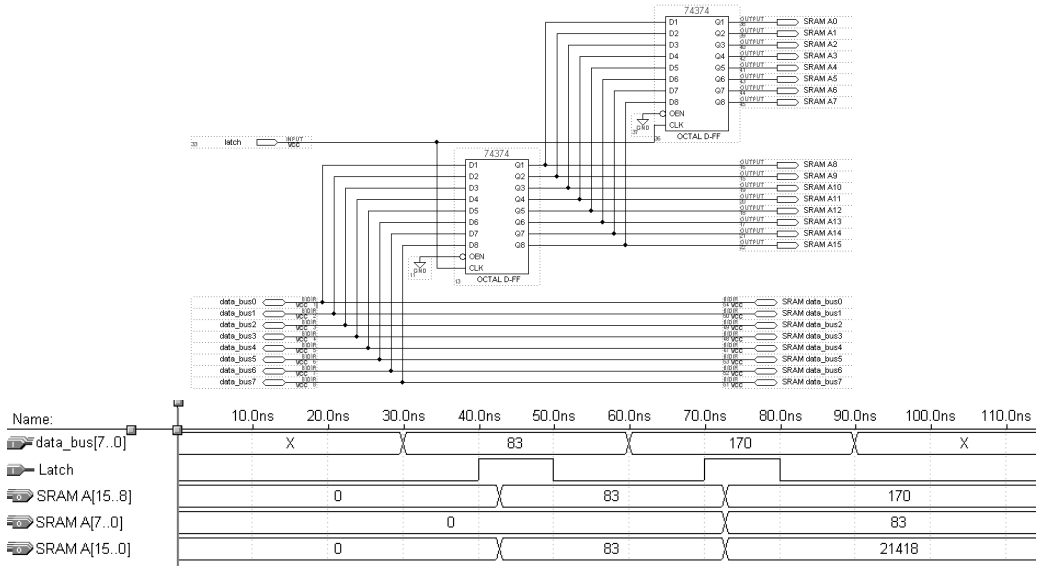


Figure 4.2: SRAM/PIC interface and timing diagram.

4.5 Modem

The primary function of the modem is to modulate binary data into audio tones that the radio can transmit, and to demodulate audio tones received by the radio to produce binary data. CML Semiconductor produces a wide range of modem solutions suited to various applications. The CML FX469 is a MSK modem IC capable of 1200/2400/4800 baud operation, and is particularly suited to this project. It requires a 5VDC supply, supports TTL/CMOS operation, and has a good BER performance. Testing showed that the performance is comparable to the theoretical performance of MSK. Although the FX469 is capable of full duplex operation, analogue radios such as the ones used in this project are only capable of half duplex operation.

Due to the limited channel bandwidth of 12.5kHz, and the band pass characteristics of the radio audio circuitry (discussed in Chapter 2), the modem is ideally suited to operate at 1200 baud, while 2400 baud is still achievable, but 4800 baud is not.

The FX469 The modem can operate both synchronously and asynchronously, with the former making use of a clock signal to reference bit periods. It was decided to use the synchronous mode of operation to ensure no bits are lost due to synchronization errors between modem and PIC.

For 1200 and 2400 baud operation the modem has carrier detect indicator which can be used to alert the PIC of an incoming message. When the demodulating circuitry detects a MSK signal, an output pin is driven high. The response time of the carrier detect indicator is user controllable via an external capacitor, the value of which is a trade off between sensitivity and the response time. In other words, a longer response time (large

capacitor value) will result in improved noise immunity. The data sheet of the FX469 indicates that there is a 0.995 probability that the carrier detect pin will be high after a 16 bit 0/1 alternating preamble, at a E_b/N_0 of 12dB and a capacitor value of $100nF$.

Testing at 1200 baud indicated that the response time of a $100nF$ cap was in the order of $10ms$ (12 bit periods) while the response time of a $470nF$ cap was closer to $30ms$ (36 bit periods). However, when a $100nF$ cap was used, the carrier detect pin would trigger an interrupt when radio discriminator noise alone present at the modem input. In the $470nF$ case, the false triggering was greatly reduced. It was decided that the system would be more robust using the $470nF$ cap, as less time would be spent waiting an extra $20ms$ on each transmission than would be spent trying to determine the validity of false messages each time the carrier detect indicator goes high. Apart from the carrier detect response capacitor, the FX469 requires decoupling capacitors on the supply rails, and DC block caps on transmit and receive lines on the radio side. A 4.032MHz crystal oscillator is required to drive the internal circuitry. It is available locally in a 22 pin DIP package, making it ideal for prototyping.

A jumper was included in the PCB design allowing for selection between 1200 and 2400 baud operation.

4.6 I/O Module

In keeping with typical devices, three types of I/O points are considered here, namely analogue inputs, digital inputs, and digital outputs. The digital inputs can be configured as counter inputs in software. As this project is a prototype design, only two of each type of I/O were implemented, as this was felt sufficient to demonstrate full functionality.

4.6.1 AIN Circuitry

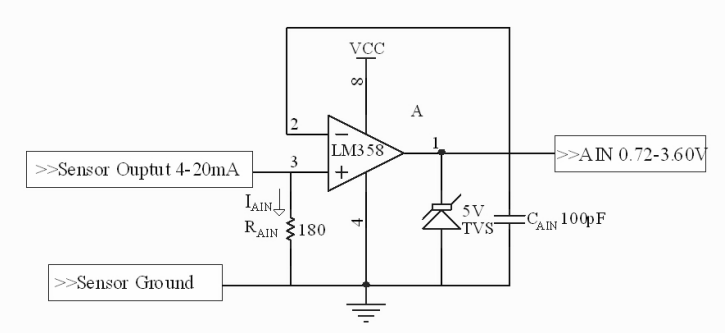


Figure 4.3: AIN circuitry.

Typical measurement devices have a current output ranging between 4 and 20mA. The PIC ADC requires a voltage signal of between 0 and 5V. The current signal can be converted to a voltage signal by passing it through a resistor, R_{AIN}, to ground. The circuit is shown in Figure 4.3. This voltage level is buffered by the LM358 op-amp in a voltage follower configuration for two reasons: Firstly, the high input/low output impedance of the op-amp provides isolation between the PIC and the sensor output. Secondly, a low output impedance is desirable, as the datasheet of the PIC recommends that a low impedance input at its ADC circuitry, as this reduces the time constant of the sample and hold capacitor. Choosing R_{AIN}=180Ω produces a voltage over ranging from 0.72-3.60V for a current of 4-20mA. This is both in the operational range of the PIC, and the op-amp with a 5V rail. The input to the PIC ADC is protected by a transient voltage suppressor (TVS) diode with a clamping voltage of 5V. A TVS diode is seen as an open circuit (infinite impedance) when a voltage below its breakdown threshold is applied to it. However if a voltage above its threshold is applied, it immediately begins to reduce its impedance (PN junction breaks down), thus dropping, or clamping, the voltage across its terminals, and diverting current away from the protected device. They have an extremely fast response time, typically less than a nanosecond. A small capacitor is included to remove high frequency noise from the voltage signal. The PIC ADC samples the voltage input at up to 100kHz, and has a software selectable resolution of 8 or 10 bits.

4.6.2 DIN/CIN Circuitry

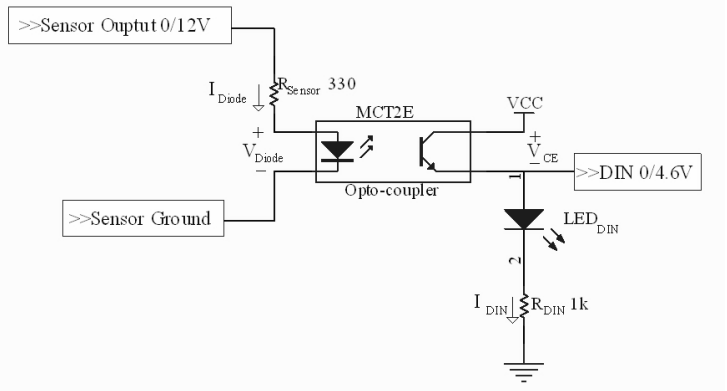


Figure 4.4: DIN/CIN circuitry.

Although the low level of a digital signal is ground, the high level output is device dependant, and can vary from logic level (5V) to 12V or even 15V. Once again, the level must be converted to a logic level usable by the PIC. An opto-coupler provides an ideal solution, as it can handle a wide voltage range and offers protection to sensitive circuitry. Consider the circuit depicted in Figure 4.4. The opto-coupler requires a minimum diode

current, I_{Diode}, of 5mA to bias the detector transistor. The maximum current the diode can handle is 60mA. The voltage drop over the diode is approximately 1.1V, as specified in the MCT2E datasheet.

$$I_{Diode} = \frac{V_{Sensor} - V_{Diode}}{R_{Sensor}} \tag{4.6.1}$$

Choosing R_{Sensor} as 330Ω ensures the opto-coupler will turn on for sensor output voltages ranging from 3-18V.

The PIC input follows the emitter of the detector transistor so that

$$V_{PIC} = V_{CC} - V_{CE} \tag{4.6.2}$$

When the transistor is off, the PIC will observe a low level, as the emitter is pulled to ground by the LED and R_{DIN} and $V_{CE} = V_{CC}$. R_{DIN} is chosen as 1kΩ, which is small enough to drive the indicator LED, but large enough so that as the transistor begins to turn on, the PIC will observe a high voltage for collector currents of $I_{DIN} > 3mA$.

$$V_{CE} = V_{CC} - V_{LED}(on) - I_{DIN}R_{DIN} \tag{4.6.3}$$

$$V_{CE} = 5 - 0.7 - 3 \times 10^{-3} \times 1000 = 1.3\text{V}$$

When the transistor becomes saturated, $V_{CE} = 0.4$V, so the PIC will observe a high level of 4.6V.

Note that all DINs can be configured as CINs in software.

4.6.3 DOT Circuitry

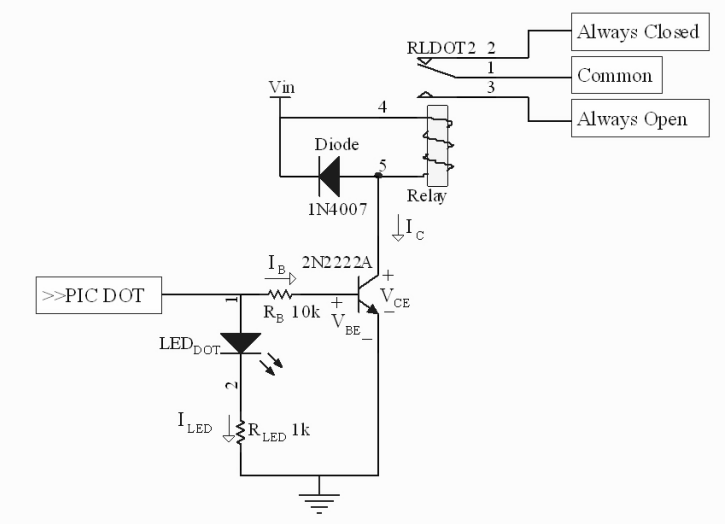


Figure 4.5: DOT circuitry.

Output control on typical devices is in the form of relays providing "usually open" and "usually closed" contacts, which alter state when the relay switches on. In this case, the selected relays are suitable for PCB mounting and can handle 12A. In order to switch the relay, a current of 89.5mA (5V coil voltage) or 37.5mA (12V coil voltage) is required. Thus, a transistor is required to drive the coil. The 2N2222A was chosen for use, as it can easily handle the required collector current, and has sufficient DC current gain ($\beta = 75 \rightarrow 300$) to demand minimal base current. A 12V level is available from the power supply, but I_C was chosen as 65mA for design calculations, to ensure the relay will switch. Choosing $\beta = 100$, and referring to Figure 4.5,

$$R_B = \frac{V_{PIC} - V_{BE}(on)}{I_C/\beta} \tag{4.6.4}$$

$$R_B = \frac{5 - 0.7}{65 \times 10 - 3}/100 = 6615\Omega$$

R_B was chosen as 6.2kΩ. An indicator LED in series with a current limiting resistor is included to show the state of the output.

4.6.4 I/O Expansion Circuitry

The limited number of PIC pins is a problem when more DINs or DOTs are required. It was decided to provide I/O expansion by making use of the same data bus used for SRAM. The 74HCT374 octal D-type flip flops support TTL levels, and can source 35mA, making them suitable to drive the digital I/O circuitry. When a rising edge signal is applied to the clock pin, the present inputs will be transferred to the outputs. The output state will be maintained until the next rising edge is detected or the output is disabled. A 74HCT259 8 bit addressable latch was configured as a 3-8 demultiplexer and used to pulse the clock pins of the flip flops. The output corresponding to the value on the three address lines is be high, while all the others are low. Thus eight octal flip flops are effectively controlled by three address lines.

The flip flop output can be forced into high impedance state making it ideal for bus type applications. This is necessary to prevent logic contention on the data bus when the output of one flip flop is common to the input of another, on the same bus. Consider the configuration and simulated timing diagram shown in Figure 4.6. When one wishes to write to the DOT flip flop, the DIN flip flop output must be forced into high impedance, otherwise the data that is placed on the bus, will be in contention with the DIN flip flop output. This is achieved by pulling the $\overline{\text{RE}}$ pin high.

One DIN octal flip flop and one DOT octal flip flop have been included for demonstration purposes. Additional DOTs could be added by simply by connecting them to the data bus, and assigning their clock pins to the demultiplexer. Additional DINs each require

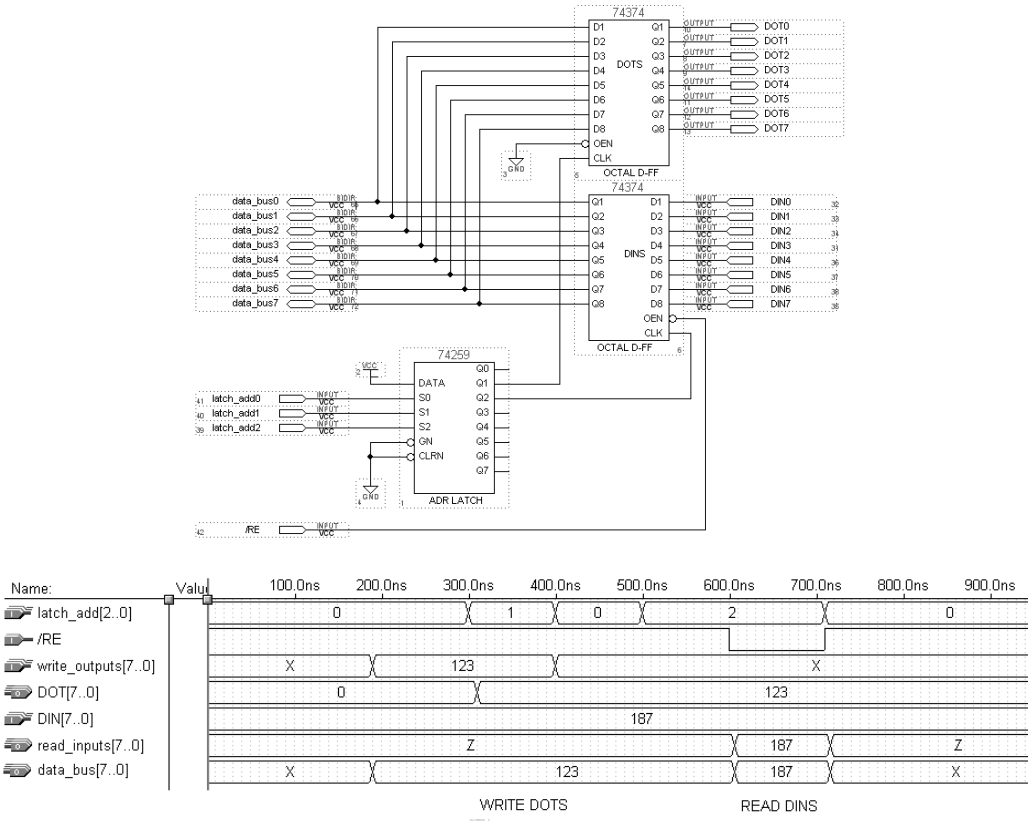


Figure 4.6: I/O expansion circuitry and simulated timing diagram.

their own $\overline{\text{RE}}$ driver, as their outputs are all common to the same bus, so only one can be enabled at a time. This could be achieved without much complication using a second demultiplexer.

4.7 Radio

Kenwood TK-3160 UHF portable radios were used for the prototype stations. The datasheet is appended. They have a selectable RF power output of 1W or 4W, and operate at a frequency of 430-450MHz. These radios demonstrate typical analogue radio functionality and characteristics. Antennas used were RFI PR-400 end fed half wave dipoles. Control over the radios was provided via various connections to the radio circuit board. This is discussed in the next section.

4.8 Radio Interface

There were various connections to the radio that required interfacing to the modem and PIC. Connections to the radio PCB were MIC_ I (microphone input), DEO (discriminator

output) PTT (push to talk) and AUX3 (SQL indicator). The radios required an 8V supply. Connection points are indicated on the radio PCB diagram included in the appendix.

4.8.1 Push To Talk

The radio PTT is controlled by the PIC. The PTT is pulled up internally by the radio, and requires grounding in order to transmit. It was decided to isolate the radio from the PIC circuitry for protection purposes. This is achieved by using an opto-coupler to interface the PIC to the radio PTT connection. The circuit configuration is depicted in Figure 4.7. When the PIC output pin is high, the transistor will be on, and its collector current will flow through the opto-coupler diode, which turns the detector transistor on, thereby pulling the radio PTT to ground, and engaging transmission. The datasheet of the MCT2E opto-coupler has a diode current maximum of 60mA. Actual testing showed that a diode current of approximately 30mA was required to drive the detector transistor sufficiently into saturation to ensure the radio PTT is pulled to ground. The PIC can only source a maximum of 25mA, therefore a transistor was used to source the extra current. A 2N3904 NPN transistor was suitable for application, as it can easily handle the require current load, and has sufficient DC current gain. The 2N3904 datasheet specifies a DC current gain (β) of approximately 100 at the desired collector current, when the transistor is operating in the active region, i.e $V_{CE} > V_{CE}(\text{sat})$. With reference to Figure 4.7, assuming a forward current I_C of 30mA, the diode voltage drop (V_{Diode}) will be approximately 1.1V, as indicated in the MCT2E datasheet. To calculate the value maximum of R_C, V_{CE} was chosen as 0.3V, which is the saturation voltage given in the datasheet. Choosing R_C smaller than this value ensured that the transistor would always be in the biased in the active region at the desired collector current, and the gain linearity of (4.8.2) would apply.

$$R_C = \frac{V_{CC} - V_{CE} - V_{Diode}}{I_C} \tag{4.8.1}$$

Thus,

$$R_C = \frac{5 - 0.3 - 1.1}{30 \times 10^{-3}} = 120\Omega$$

$$I_B = \frac{I_C}{\beta} \tag{4.8.2}$$

$$I_B = \frac{30 \times 10^{-3}}{100} = 300\mu\text{A}$$

Lastly,

$$R_B = \frac{(V_{PIC} - V_{BE}(on) - V_{Diode})}{I_B} \tag{4.8.3}$$

$$R_B = \frac{(5 - 0.7 - 1.1)}{I_B} = 10.667\text{k}\Omega$$

Selecting R_B and R_C as 100Ω and 10kΩ respectively ensured the desired operation. The choice of values was verified by an Hspice simulation which showed that for an input of 5V, $I_B = 312.28\mu$A, $I_C = 30.21$mA, β=(96.7)and V_{CE}=0.88V. The base current is slightly higher as R_B is chosen slightly less than the calculated value.

Note that the open base of the opto-coupler results in a slow response time, in the order of a few micro seconds. This delay is negligible here as it is much shorter than transmission times for which the opto-coupler is used.

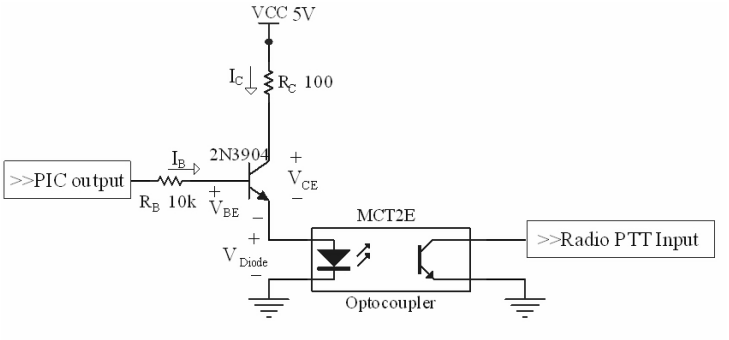


Figure 4.7: PTT driver circuitry.

4.8.2 TX/RX Audio Circuitry

In the transmit path, the audio tones from the modem are inserted at the microphone input (MIC_ I) of the radio. The output of the modem is specified as 775mV$_{rms}$, however a much lower value is required by the radio to achieve the required deviation. This level was found to be around 20mV$_{p-p}$ by measuring the deviation with a spectrum analyzer for various input levels. This level is consistent with the radio datasheet. On the receive path, the audio tones from the discriminator output of the radio are passed to the modem for demodulation. The discriminator output was measured to be 1.5V peak to peak, or 530V$_{rms}$, while the modem specifies an optimal input level of 230mV$_{rms}$. Therefore adjustment circuitry was required between the radio and modem on both transmit and receive paths. The attenuation in the audio signal on each path was achieved with an inverting op-amp configuration with a gain less than unity, i.e. $R_2 < R_1$. The gain (attenuation) is adjustable via a variable resistor, R_2. As only a single voltage rail was available (+5V), a bias voltage of 2.5V was applied to the non-inverting input of the op-amp to keep the sinusoidal signal in the operational range of the op-amp. The bias voltage is sourced from PIN10 of the FX469 modem, which is a fixed 2.5V reference output. DC blocking caps (10nF) were placed on either side of the circuit to shield the modem and radio from the DC bias voltage. Multi-turn resistors were chosen to provide greater level adjustment accuracy. The center tap of each multi-turn was made common to the base

pin (as shown in the circuit diagram) effectively halving the rated resistance. The gain of a inverting amplifier is given by:

$$\text{Gain(dB)} = 20\log_{10}(|-\frac{R_2}{R_1}|) \tag{4.8.4}$$

The chosen resistor values allow gain in the following range:

	R_1 (Ω)	R_2 (Ω)	Gain (dB)
TX	33k	$0.1k \rightarrow 12.5k$	$-50 \rightarrow -8.43$
RX	10k	$0.1k \rightarrow 25k$	$-40 \rightarrow +7.96$

A frequency response of the circuit was computed using Hspice, to ensure that the circuit did not introduce any adverse filtering effects on the audio tones of 1.2kHz and 1.8kHz. The circuit is depicted, along with the typical frequency response in Figures 4.8 and 4.9. It should be noted that although the received the audio level output requires attenuation in this case, the level can vary between types and makes of radios, in some cases, the output could be lower than the optimal modem level, and the signal will then require amplification, not attenuation. For this reason, R_2 on the receive path is variable to allow both gain and attenuation. The LM358 dual op-amp was selected because of the dual packages available, i.e. only one package was required to service both TX and RX audio paths.

4.8.3 Squelch Circuitry

The squelch output of the radio is used as an indication of when the radio is receiving, i.e. channel busy. The indicator connection point is connected to the radio microprocessor by a series resistor. The radio manual indicates that the output of the microprocessor pin will be high (5V) when the radio is receiving and at ground otherwise. In order to protect the radio microprocessor from providing excessive current, the same approach was used as for the PTT circuitry, by using a transistor to drive an opto-coupler. Referring to Figure 4.10, on the PIC side of the opto-coupler, a resistor is required to pull up the open collector to 5V when the detector transistor is off. The PIC observes the level of the collector (collector follower). A low value ($\approx$0.4V) indicates the radio is receiving, and a high value (5V) indicates the channel if free. The value of the pull up resistor was chosen to limit the collector current of the detector transistor which has a maximum rating of 50mA. By choosing $I_{Detector}$ conservatively as 15mA, and assuming $V_{Detector}$=0.4V (saturation), the value of the pull up resistor was calculated as

$$R_{PU} = \frac{5 - V_{Detector}}{I_{Detector}} \frac{5 - 0.4}{15 \times 10^{-3}} = 306\Omega \tag{4.8.5}$$

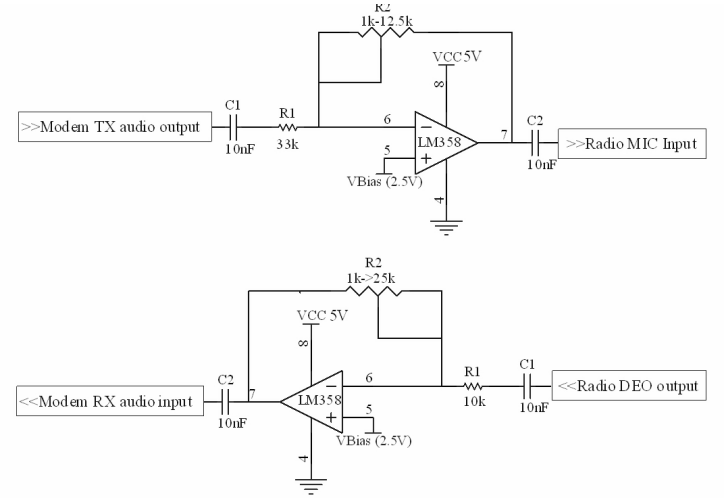


Figure 4.8: TX/RX audio interface circuitry.

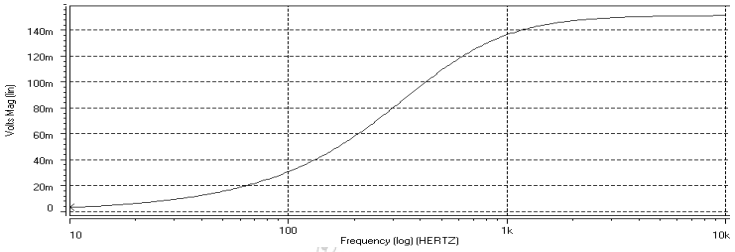


Figure 4.9: Frequency response of the TX/RX audio interface, Attenuation=20dB.

The nearest available value of 330Ω was chosen.

On the radio side, using the same resistor values (R_B and R_C), transistor and opto-coupler types as the PTT case requires the radio microprocessor to source approximately 300μA (as calculated in Equation 4.8.2). This circuit was built, tested and found to work perfectly. A further addition to the circuit allows the receiving audio line to the modem

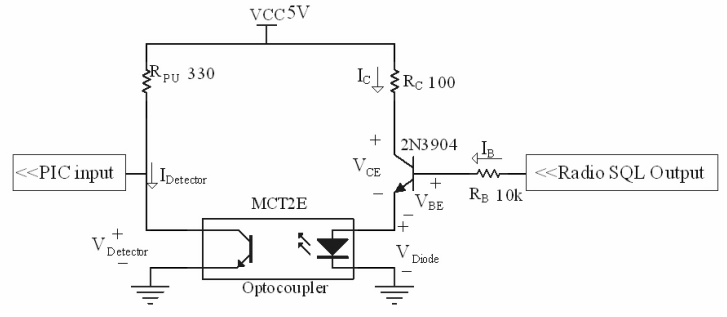


Figure 4.10: Squelch interface circuitry.

to be pulled to ground when the squelch is closed (i.e. when the radio is not receiving). This is desirable because the noise that the radio discriminator produces when the radio is not receiving can cause the modem to falsely detect a carrier signal, causing unnecessary

processing overhead at the PIC. This is done simply by connecting a 2N3904 with the emitter common to ground, collector common to the audio input of the modem and base driven via a series resistor by the collector of the squelch opto-coupler, as illustrated in Figure 6.2. When the radio is not receiving, the collector of the squelch opto-coupler will be high, thus biasing V_{BE2} of the 2N3904, effectively pulling the RX audio line to ground. When the radio receives, the collector of the squelch opto-coupler will be low, and the transistor will no longer be biased as $V_{Detector} < V_{BE2}$, thus the collector will be floating, and the audio signal will pass to the modem. The audio signal has a slight DC offset, as shown in the HSPICE simulation (Figure 6.2, resulting from the non zero $V_{Detector}$. However, the offset is so slight that it does not adversely effect modem operation at all. The resistor R_{B2} is chosen to limit the base current of the transistor, and also to ensure the PIC will observe a high level at the opto-coupler-collector when the opto-coupler is off, and not a low level of V_{BE2}.

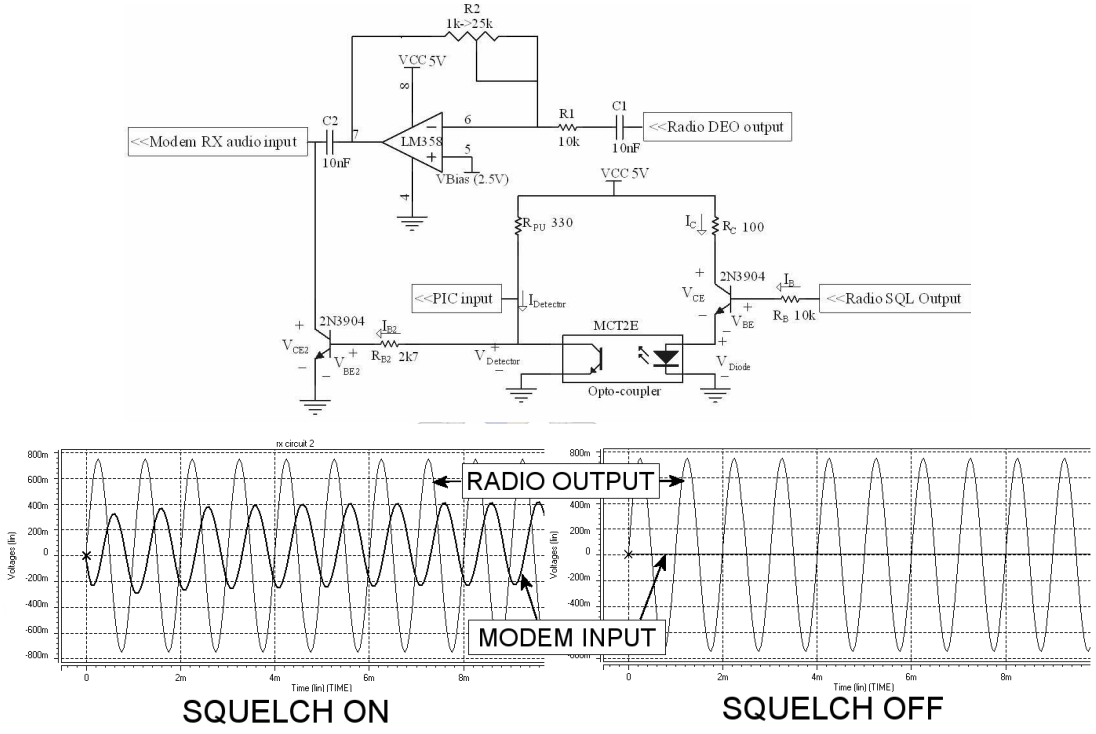


Figure 4.11: Squelch and RX audio circuitry silencing, with HSPICE simulation results.

4.9 Power Supply

Mains power (220VAC) was selected as the primary power source, however, it was decided to provide the RTU with full backup battery supply capable of sustaining system operation for lengthly periods, for cases where the primary power source fails. The power supply is

designed to double as a optimal charger for the back up battery. The AC/DC conversion is discussed initially, followed by the design of the battery charger, whereafter the entire power supply model is presented.

4.9.1 AC/DC Conversion Circuitry

A standard full wave bridge rectifier configuration was used for the conversion from AC mains to useable DC. The required output DC voltage and current were used to calculate the required transformer rating.

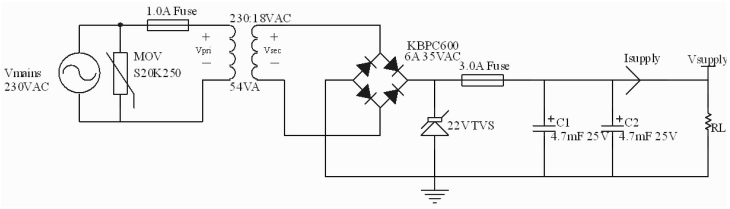


Figure 4.12: Full wave bridge rectifier circuit.

4.9.1.1 Voltage and Current Levels

When a battery is drawing full charging current, and the RTU transmits, the power supply must deliver up to two amperes at rated voltage. If a higher powered radio is used, more current will be required. This justifies a choice of a 3.0A power supply. The voltage selection was a little more complicated. Although the radio used in this project only requires 8VDC, other radios may require a supply of up to 12VDC. A 12VDC supply also lends itself to other applications, such as driving sensor or output control circuitry. Secondly, although the back up battery is rated as 12VDC, the terminal voltage is closer to 14VDC, hence the required charging voltage will be higher still. Lastly, there will be a considerable voltage drop over the regulator and protection diodes in the circuit. Basically, all this means is that the rectified voltage at rated current (3.0A) must be somewhat higher than what the output requirement is. The trade off of choosing a higher rectified voltage is that more power will have to be dissipated in pass devices. However, if the power dissipation is not excessive, it can easily be countered with a suitable low cost heatsink. A value of 18VDC was decided to be suitable.

4.9.1.2 Transformer and Rectifier Circuit

V_{sec}, as depicted in Figure 4.12 is an rms value that can be calculated in terms the rectified voltage V_{supply}, and the voltage drop over the bridge diodes. The datasheet of

the full wave bridge rectifier specifies a drop of 1.1VDC over each diode. The VA rating of the transformer can then be determined by multiplying the secondary voltage by the secondary current, which is simply I_{supply} (Mohan, 2003).

$$V_{sec} = \frac{1}{\sqrt{2}} (V_{supply} + 2V_{Diode}) \tag{4.9.1}$$

$$V_{sec} = \frac{1}{\sqrt{2}}(18 + 2 \times (1.1)) = 14.28\text{V}_{rms}$$

$$T_{VA} = V_{sec} \times I_{supply} \tag{4.9.2}$$

$$T_{VA} = 14.28 \times 3.0 = 42.85\text{VA}$$

The nearest available transformer was found to be a 230:18VAC 54VA, i.e. it is rated to deliver 3.0A with V_{sec}=18VAC or V_{supply}=23.25V. Although somewhat higher than the required value, it must be kept in mind that the VA rating is regarded as a maximum, and heat dissipation in the transformer windings will increase as this maximum is approached. Secondly, the ripple on V_{supply}, although limited by the filter capacitors, can cause voltage dips. Lastly, transformers are not ideal, in fact the measured value of V_{supply} at 3.0A was 20.0V, and 23V at no load. Therefore is was decided to use the transformer in conjunction with heatsinks on pass devices. Selecting it also ensured that the supply voltage would never drop below the minimum threshold, so the output voltage would always be constant, at rated current.

4.9.1.3 Filter Capacitors

The filter capacitors were chosen with high values for two reasons. Firstly, to minimize ripple on the supply voltage, and secondly to prevent dips in supply voltage when a large amount of current is suddenly drawn, as occurs when a radio begins to transmit. The maximum peak to peak ripple on V_{supply} was chosen as $V_{p-p} = 3.5$V. This translates to an rms value of 1.24V, which is equivalent to 6.9% of V_{sec} at rated current.

$$C = T\frac{dV}{dt} = \frac{T \times I_{supply}}{V_{p-p}} \tag{4.9.3}$$

Note that in the case of a full wave rectifier, the period between peaks is 10ms, half of the 20ms in the case of the non rectified 50Hz sinusoid.

$$C = \frac{0.01 \times 3.0}{3.5} = 8.571\text{mF}$$

Allowing for tolerances, the nearest higher available value is 9.4mF, achieved by placing two 4.7mF capacitors in parallel.

4.9.1.4 Protection Circuitry

On the primary side of the transformer, a metal oxide varistor (MOV) was placed across the input voltage supply. MOVs are used to protect components against transients (spikes) caused by electrostatic discharge and lightning impulses on the supply line. A MOV has a typical response time of less than $25ns$. An Epcos SK20K250 was selected for use, as it has a maximum operating voltage of 250VAC, and can handle a maximum of 100A. The maximum clamping voltage is 650V.

Following the MOV, a fuse is inserted. Given that the required secondary current is 3.0A, with the transformer ratio of 230:18VAC, the ideal primary current is in the region of 0.23A. ($V_{pri}/V_{sec} = I_{sec}/I_{pri}$) Keeping in mind that the filter capacitors will draw substantial current when power is initially applied, this fuse rating was chosen as 1.0A.

On the DC side, after the bridge rectifier, a TVS diode is placed, together with a 3.0A fuse.

4.9.2 Battery Charging Circuitry

4.9.2.1 Sealed Lead Acid Batteries

There are many types of rechargeable batteries available, however sealed lead-acid batteries remain in high demand in industry because of their low cost, low self discharge rates, long cycle life and high capacity. For these reasons, sealed lead-acid batteries were selected for use in this project.

The amp-hour (AH) rating of a battery is a ratio of what current the battery can deliver over time, e.g. a 7.0AH battery can deliver 7.0A for 1 hour, or 700mA for 10 hours. The capacity (C) of a battery is is generally used to express charge or discharge rates, e.g. a 7.0 amp-hour battery charging at 700mA is charing at rate C/10. The cycle life refers to the number of charge and discharge cycles a battery can go through before its capacity is reduced, and the standby or float life refers to the time that a battery can be maintained in fully charged state and still perform optimally when required. Manufacturers claim that sealed lead-acid batteries have a life of up to 1000 cycles, and a float life of up to ten years (Horowitz and Hill, 2002).

The method by which a battery is re-charged has a significant effect when it comes to battery capacity and life cycle, especially in the case of sealed lead acid batteries. There are many variations on how exactly to recharge a sealed-lead acid battery. The simplest, but most inefficient, would be to connect the battery terminals to a current limited regulated voltage supply. The battery would initially draw a high (damaging) current, up to 2C, and eventually taper off to a trickle current as the terminal voltage rises (Horowitz and Hill, 2002). A sealed lead-acid battery under charge goes through various

states of chemical reaction, each requiring different levels of charge current. To maximize the life expectancy of the batteries, precision monitoring is required to determine what state a battery is in, and deliver the correct current.

4.9.2.2 UC3906 Charging Controller

Unitrode (Texas Instruments) manufactures an IC specifically for controlling the charging cycle of sealed lead acid batteries. With a few external components, the UC3906 provides a dual level float charger which controls both the current, and voltage at the charger output. The UC3906 also tracks the temperature characteristics of the lead acid battery by using an internal voltage reference. A dual level float charger has three states: bulk charge, over charge, and float charge. In bulk charge state, the charger provides a high constant current, I_{MAX}, to the battery. When the battery reaches a certain threshold voltage, V_{12}, over charge state begins. In over charge, the charger holds the battery at a higher voltage than the battery rating. The charge current is monitored until it drops below threshold I_{OCT}, indicating the battery is almost fully charged. At this stage, the charger enters float charge state, and provides the battery with an regulated voltage V_F. If a load is applied to the battery, the charger will enter bulk charge state should the battery voltage fall below 90% of V_F.

A further feature that the UC3906 provides is low current turn on mode. If a battery voltage is below a minimum threshold, V_T, when power is applied to the charger, it would be damaging to immediately apply bulk charge current to the battery. Therefore, the UC3906 limits the charge rate (trickle charge) until the battery voltage reaches the minimum threshold voltage which indicates it is safe to begin bulk charging. A typical case of this occurring would be if the mains power fails, and the battery is drained to below V_T, whereafter mains power is restored.

Note that the circuitry can only enter the trickle charge mode when power is initially applied to the charger, and a battery is connected to the charger output. If a battery is connected to the charger after the charger has power applied to it, the charger must be manually reset by cutting and restoring its power, to ensure the correct mode is entered into. This is provided for by a switch on the power supply PCB.

4.9.2.3 Design of Charger Circuitry

The UC3906 datasheet, application notes, and battery ratings were used to determine voltage and current levels and external component values. An extract from the device datasheet with relevant design information has been appended.

The battery ratings specify voltage between 13.5-14.7V for stand by and cycle use. The float voltage, at which the battery will be held when fully charged is chosen in this range

as $V_F = 14.0$V. The over charge voltage is chosen slightly higher than the float charge voltage, at $V_{oc} = 14.5$V. The bulk charging I_{MAX} current is chosen as C/7, which is in the conventional range of C/5 to C/20 (Horowitz and Hill, 2002). For the 7.0AH battery, this correlates to a value of $I_{MAX} = 1.0A$. From the appended circuit diagram, it is clear the the minimum input voltage must be greater than the float battery voltage by the combined voltage drop over R_{SENSE}, the protection diode, and the on-voltage of the pass device. The datasheet indicates a difference between supply voltage V_{in} and float V_F battery of no less than 2V during bulk charge, without a protection diode. The measured voltage delivered to the charger by the power supply is 18VDC during bulk charge, which is sufficiently above the minimum threshold.

The threshold voltage, V_T, is the minimum battery voltage at which bulk charging can begin. Below V_T the battery will be trickled charged at I_T. V_T was chosen as approximately 70% of V_F, while I_T was chosen as C/500, which is a typical value (Horowitz and Hill, 2002).

The UC3906 datasheet recommends a divider current ranging from 50μA 100μA. Varying this value allows one to achieve resistor values closer to those commonly available. It was chosen 80μA.

$$\begin{array}{ll} V_{in} = 18\text{V} & I_{MAX} = 1.0\text{A} \\ V_T = 10\text{V} & I_T = 14\text{mA} \\ V_F = 14\text{V} & I_D = 80\mu\text{A} \\ V_{oc} = 14.5\text{V} & \end{array}$$

From equations provided in the device datasheet, and using the above values the voltage and current values, resistor values were calculated to be

Resistor	Calculated Value	Nearest Value
R_{SENSE}	0.25Ω	$1\Omega\|\|1\Omega\|\|1\Omega\|\|1\Omega = 0.25\Omega$
R_T	392.86Ω	420Ω
R_A	133.84kΩ	$100\text{k}\Omega + 33\text{k}\Omega = 133\text{k}\Omega$
R_B	12.41kΩ	$10\text{k} + 2.7\text{k}\Omega = 12.7\text{k}\Omega$
R_C	$28.75k\Omega$	$27\text{k}\Omega + 1.8\text{k}\Omega = 28.8\text{k}\Omega$
R_D	672.75kΩ	$620\text{k}\Omega + 56\text{k}\Omega = 676\text{k}\Omega$

Decoupling capacitors were placed at all voltage reference points in the circuit to minimize high frequency transients.

The UC3906 requires an external pass device between the input supply and the battery. The device should be capable of handling a collector current of I_{MAX}, as well as the associated power dissipation. It should have sufficient DC gain to ensure the UC3906

drive current limit of 40mA is not exceeded during bulk charge. The power dissipated by the transistor is calculated by the current through it multiplied by its voltage differential.

$$P = I_{EC}V_{EC} \tag{4.9.4}$$

Maximum power dissipation will occur during the initial stages of bulk charge, i.e. when $I_{EC} = I_{MAX}$, and the battery voltage is at a minimum of V_T (V_{EC} is at a maximum).

$$V_{EC-MAX} = V_{in} - I_{MAX}R_{SENSE} + V_{Diode} - V_T.$$

$$V_{EC-MAX} = 18 - 1 \times 0.25 - 0.7 - 10 = 7.05\text{V}$$

$$P_{MAX} = I_{MAX}V_{EC-MAX} = 7.05\text{W}$$

The TIP32C PNP transistor, together with a suitable heatsink, provides an ideal solution as a pass device, as the datasheet specifies it to have a minimum DC gain of 25 for a collector current of 1A and a V_{EC} of 4V. With a suitable heat sink, the device can dissipate up to 40W, and without a heatsink, only 2W. Therefore it is necessary to determine the optimal heatsink value for the worst case scenario.

Heatsinks are rated in terms of thermal resistance θ in °C/W. The difference between the junction and ambient temperatures of a device is equal to the power, P, dissipated in a device, multiplied by the total thermal resistance of the device, θ_{total}.

$$T_{junc} - T_{amb} = P\theta_{total} \tag{4.9.5}$$

This can be rewritten in terms of individual thermal resistances:

$$T_{junc} - T_{amb} = P\left(\theta_{junc-case} + \theta_{case-sink} + \theta_{sink-amb}\right) \tag{4.9.6}$$

The TIP32C datasheet specifies that $\theta_{junc-case} = 3.125$°C/W and $T_{junc-MAX} = 150$°C A 10% safety margin is imposed on the maximum junction temperature, so that $T_{junc} = 135$°C for calculation. Although variable, a generally accepted value for $\theta_{case-sink}$ is 1.0°C/W (Neamen, 2001). T_{amb} is chosen as 50°C for the worst case. Keeping in mind that the RTU could be inside a steel cabinet in direct sunlight, this value is easily justified. The power has already been calculated as 7.05W from (4.9.4). From (4.9.6), the thermal resistance of the heatsink was calculated:

$$\theta_{sink-amb} = \frac{T_{junc} - T_{amb}}{P} - \theta_{junc-case} - \theta_{case-sink} = 7.932\text{°C/W} \tag{4.9.7}$$

Fischer Elektronik manufactures a low cost T0 220 extruded heatsink that has a thermal resistance of 8°C/W, which was selected for use in the charging circuit.

4.9.2.4 Charge State Indicators

The UC3906 provides two outputs that can be used to determine which state the charger is in.

- **Float Charge Indicator**
  The internal transistor whose collector is common to pin 10 of the UC3906 is driven by the active low state level output, the state diagram of which is depicted in the appended datasheet extract. When in bulk or overcharge state, the state level output is on (active low), hence the transistor is off, and the pin 10 is pulled up to pin 13 which correlates to the internal reference voltage of 2.3V. When in float state, state level output is off (high), therefore the transistor is on, and pin 10 is pulled down to $V_{CE}(on)$ above ground, slightly less that 1V. A simple external comparator circuit is used to monitor and indicate the state of pin 10. A threshold of between $V_{CE}(on)$ and 2.3V is required to determine which state pin 10 is in; approximately one volt was chosen as a suitable value. It was decided to add a 5V regulator to the power supply circuitry as a constant reference voltage for state circuitry. In order to achieve the required 1V threshold, a voltage divider circuit was used with reference to the 5V rail. (Note that this 5V regulated reference is unaffected by varying power supply levels, and therefore all thresholds referenced to the 5V rail will remain constant.)

$$V_{thresh} = \frac{5 \times R_2}{R_1 + R_2} \tag{4.9.8}$$

  R_1 and R_2 were chosen as 10kΩ and 2.2kΩ respectively, to produce a threshold voltage of V_{thres}=0.9V. This threshold is made common to the non-inverting input of a LM311 comparator. Pin 10 is made common to the inverting input of the LM311. While pin 10 is above the threshold, the output of the LM311 will be high. If pin 10 falls below 0.9V, the LM311 output will be low. A LED in series with a current limiting resistor is connected to the output of the comparator, so that when the output is low, (charger is in float state), the LED will be on.

- **Over Charge Indicator** The internal transistor whose collector is common to pin 9 of the UC3906 is driven by the overcharge indicator output, the state diagram of which is also depicted in the appended datasheet extract. When the charger is in overcharge state, the overcharge indicate output is high, so the internal transistor is on, and pin 9 is low. The base of an external PNP 2N3906 transistor is connected to pin 9. A LED in series with a current limiting resistor is driven by 2N3906 transistor. When the internal transistor is on, it will sink the base current of the 2N3906, which will turn on the LED.

- **Bulk Charge Indicator** Bulk charge state is indicated by both float charge and over charge LEDs being off, while the charger is on.

4.9.3 Complete Power Supply

4.9.3.1 Integration of Mains and Battery Sources

It was decided to make the output of the power supply regulated, but adjustable for versatility. Both the mains supply, and battery output are fed into the same regulator, so that the output of the power supply is remains constant when switching between mains and battery. The mains supply and battery output cannot be connected together directly, but require a diode configuration to prevent undesirable feedback. To illustrate, consider the diagram shown in Figure 4.13. When there is a mains supply present at A, B will typically be in the region of 20VDC. The maximum possible voltage at C will be the overcharge voltage, 14.5VDC. Thus when mains is present, diode BD will always be forward biased, while diode CD will always be reverse biased. If the diode between C and D were not present, the voltage at B would be present at the battery terminals, nullifying the control of the charger. Similarly, if mains power at A was set to zero, and diode BD was not present, the battery would drain itself by supplying is own charger with power. The two diodes ensure a smooth transition between power sources. If mains power was to fail, the voltage at B would decrease as the filter capacitors discharged, and BD would gradually become reverse biased, while at the same time CD would become forward biased, allowing the battery to supply the regulator with power. As the output E is regulated, the end user would not detect any transitions between mains and battery supply until such time as the battery was drained to a level below the minimum threshold of the regulator.

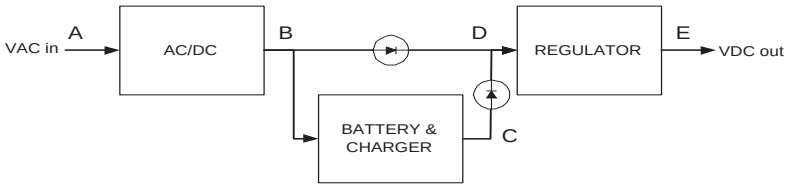


Figure 4.13: Power supply block diagram.

Both BD and CD are 3.0A rated diodes. The voltage drop over BD is both tolerable (because of the higher VA rating of the transformer) and desirable, in that it effectively reduces the differential voltage over the regulator when mains power is available, thereby reducing power dissipation in the regulator. The voltage drop over CD is not as desirable, as the battery voltage is somewhat lower than the mains level, however, it is unavoidable.

4.9.3.2 Output Regulation

A LM350 adjustable voltage regulator was selected for use, as it can handle the maximum current of 3A. The circuit configuration provides improved ripple rejection and is implemented as depicted in the datasheet. Resistor values were taken from the datasheet, R_1=5kΩ variable pot and R_2=120Ω. Adjusting the 5kΩ pot varies the output between 5 and 15VDC.

The LM350 has similar temperature characteristics to the TIP32C ($T_{junc} = 125°C$, $\theta_{junc-case} = 3°C/W$), and therefore the same type of heatsink was used for both pass devices, as both are subject to the similar current and voltage differentials.

4.9.3.3 Diagnostics

There are two diagnostic features that have been designed to allow the RTU to monitor the status of the power supply. Firstly, the 5V reference voltage is provided as a logic indicator signal for the microprocessor, as it is only on (high) when mains power is present. Thus, the PIC can monitor this voltage to determine when mains fails, or is restored, and alert the server accordingly. Secondly, the battery voltage level is also made available for monitoring by the PIC. This is done by means of a voltage divider, which translates the battery voltage of 0-15V to the 0-5V level which is the useable range of the PIC analogue/digital convertor. A variable resistor allows the voltage divider ratio to be set accurately. To calibrate the ratio, 15V was applied to the top of the voltage divider, and the varistor was adjusted until the output was 5V. Thus each volt that the PIC measures represents 3V at the battery. If the PIC uses 8 bit A/D to sample the translated battery level, a reading of 255 (5.00V) will represent a battery level of 15V, and a reading of 170 (3.33V) will represent a battery level of 10V. This is done to allow the PIC to monitor the battery level when mains power fails and so protect the battery. If the battery is drained to beyond a software set threshold, the PIC can prevent the battery from excessive draining by refusing to transmit until mains power is restored.

Both outputs are fused at 25mA, the maximum current that a pin on the PIC can sink, and fitted with a transient voltage suppressor which clamps at 6 volts to protect the PIC from any undesirable voltage spikes that may filter through the power supply.

4.9.4 Power Indicators

A "mains power on" indicator LED in series with a current limiting resistor was connected from the 5V regulated supply to ground. This LED also acts as a bleeder LED to discharge the filter capacitors when the power supply circuit is disconnected from mains supply and a load. A "supply power on" indicator LED was connected in the same manner, but to

the output of the LM350 regulator, thus indicating the presence of either mains or battery power.

4.10 Conclusion

This chapter described the selection of all hardware components and provided detailed design calculations of the entire prototype system. The resulting prototype consists of a 12VDC 3.0A power supply, complete with a lead acid backup battery and dual level charger, an RTU board controlled by the Microchip PIC 18F452, and an industry compatible I/O board to demonstrate the functionality of the system. The RTU board supports serial communication, and has additional SRAM used for the network protocol. The RTU board is interfaced to a Kenwood TK-3160 portable radio used for transmission of data. Modulation and demodulation is achieved with the FX469 MSK modem.

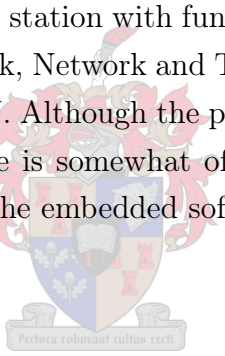
Complete circuit diagrams and PCB layouts are appended, together with relevant information using in the design process. Digital photographs are also appended.

Chapter 5

Embedded Software

5.1 Introduction

The program code loaded into the microprocessor is known as embedded software. Embedded software provides the RTU station with functionality. The objective of the PIC is to interface the Physical, Data-Link, Network and Transport layers, control all operations and monitor all inputs on the RTU. Although the program flow can be described in terms of the OSI model, the actual code is somewhat of an integration between layers. This chapter provides a description of the embedded software and its functionality.



5.2 General

The program code was written in C, using the Microchip MPLAB development environment. The CCS-C compiler was used to optimise and convert the C code into a .hex file. The .hex file is a sequence of any of 75 assembly instructions understood by the PIC. The .hex file was programmed into the memory of the PIC via the ICD2 in circuit debugger. CCS provides an extensive function library including basic commands like direct pin control, through to more advanced commands related to serial communication, ADC, timing and mathematical functions.

As stated in Chapter 4, the PIC18F452 has enough program memory space for memory 16K assembly instructions. The amount of available RAM (1.5KB) was extended by 64KB using external SRAM.

5.3 End to End Data Flow

The primary objective of the system is to provide a server with input information of remote stations, and provide the server with output control at the remote stations.

The most efficient way to achieve this is to map registers of the remote station to the registers in a data base at the server, and visa versa. The registers contain configuration settings for the station, as well as the statuses of all inputs. The program code at each remote station uses the values in the configuration registers to make operational decisions. The program code also ensures that the registers contain the real time statuses of all I/Os. Thus, if the server wishes to change a setting on the RTU, the relevant register at the RTU must be updated with the new value. Similarly, if the server wishes to learn the status of of an input, the value of the relevant register at the RTU is read.

The purpose of the various layers is to provide transparent data transfer between the server database registers, and the RTU registers, which can be considered as the end-points of the system. The server application is used to display selected values of the registers, and provide control over values of the configuration registers. The program code of the PIC is responsible for the link between onboard registers and I/O points.

The Transport layer is responsible for connecting the server database to the registers of any RTU. The server uses the RTU to which it is connected (via RS232) as an entry point into the network. It is of no concern to the server which RTU it is connected to. The server uses a simple protocol, similar to MODBUS (Crabtree, 2005), to carry out reading and writing of registers. The lower layers repack this protocol into the "air" protocol which contains a unique identifier, routing information, a checksum, and utilizes forward error correction.

To illustrate overall data flow, consider the following example. Assume the server issues a read command of a register at a certain RTU in the system. If the server is connected directly via RS232 to desired RTU, the register value will immediately be returned via RS232 to the server using the server protocol. If the server is not connected to desired RTU A, but instead to RTU B, the Transport layer of RTU B passes the message to its Network layer, which takes care of finding a route to RTU A. When a route has been found, the original server message is packed together with the route to RTU A, and transmitted to RTU A via the Data-Link and Physical layers. Upon reception at RTU A, the original server message is passed up to the Transport layer of RTU A by the Network layer. RTU A acknowledges the read command by loading the value of the desired register into a (server protocol) message that instructs the server to write the new value of the register into the database. The message is then passed to the Network layer for transmission back to the server via RTU B. When the Transport layer of RTU B receives the message from its lower layers, it simply transfers the message directly to the

server via the serial connection. Thus, the server has a virtual RS232 link to all stations in the network.

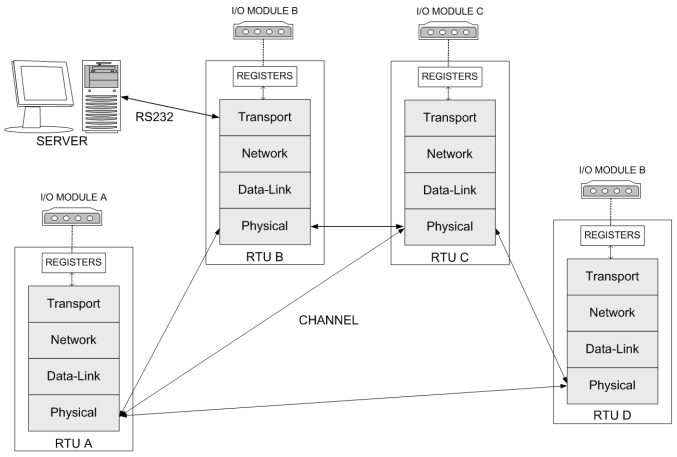


Figure 5.1: Overview of layers controlled by the embedded software.

5.4 Software Modules

The embedded software is responsible for the adminstration and integration of all layers. The code can be divided into various sections or modules. Each module has various sub functions, as shown in Figure 5.2. All modules are linked by the main software code. This section provides a more detailed description of the various modules of software, an describes the various sub functions of each section.

5.4.1 Transport Module

The Transport module, or Transport layer, is responsible for end to end connection between the server and the registers at any station in the network. The protocol which the server uses to communicate with the RTU is similar to MODBUS. It starts with the address of the destination, followed by a message type identifier. The type can either be read/write a selection of registers, or read/write a number of sequential registers. The data following the type byte contains the addresses of relevant registers, and values where applicable. The message type is sent as a sub type in the air protocol.

If the server wishes to send a message packet to a target, it initiates serial communication by issuing the RTU with a one byte service request. The USART activity triggers a hardware interrupt. The ISR for this port buffers the received byte, and sets a global flag indicating to the main code that a byte has been received on the serial port. When the main code detects the flag, it returns the service request byte to the server indicating it

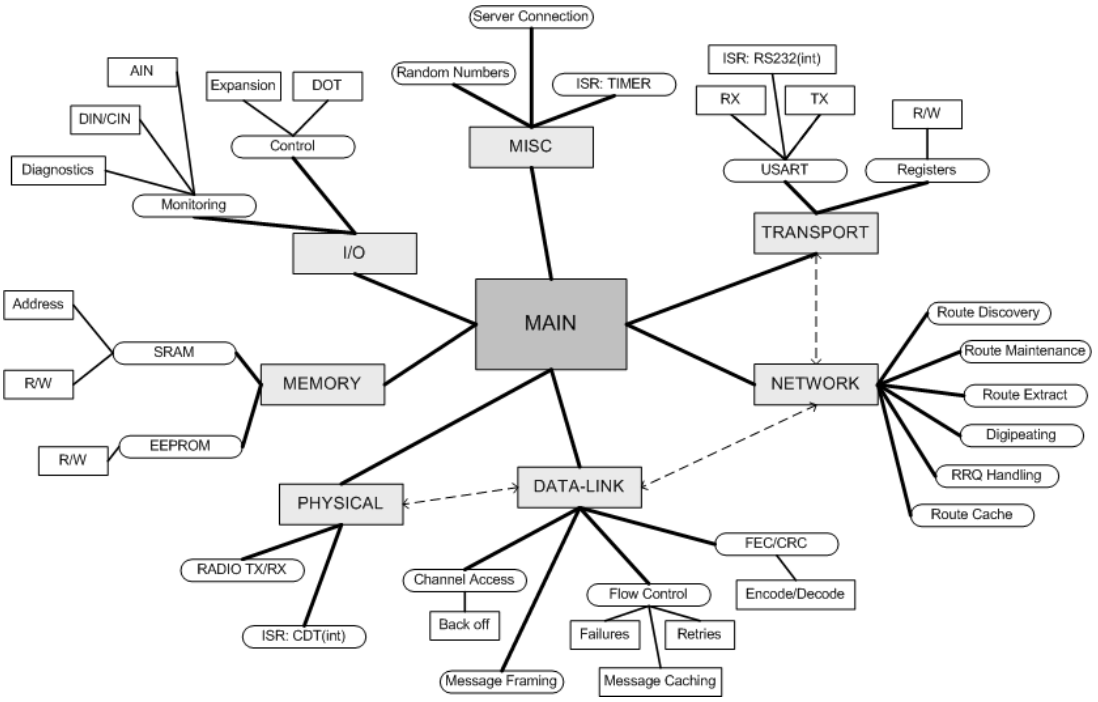


Figure 5.2: Software modules and sub-functions.

is ready to receive a message. The server immediately sends the message packet, which the RTU buffers until no further activity is detected on the serial port. The message is then passed to the Network layer for routing to the target.

When a target station receives such a message, the Transport module takes care of writing relevant registers, or generating a reply for read commands.

5.4.2 Network Module

The network section of the code is concerned with the routing protocol. The protocol is described exhaustively in Chapter 3, but a summary is provided here for completion.

When a message needs to be sent to a station, the code searches the route cache for a possible route. If a route is found, it is included in the message header. A UID is generated for the message, and it is passed to the Data-Link layer for transmission. If no route can be found in the route cache, the message is stored in the RRQBuffer, and tagged with its UID. A RRQ for the DST with the same UID is broadcast to all neighbouring stations. If the destination receives the message, it will respond immediately with a RRP. If the destination is not amongst the neighbours, the RRQ will be forwarded, or a neighbour with a route to the destination will respond with a RRP. The RRQ will not propagate further than the number of hops set by the initiator. Care is taken to prevent unnecessary transmissions and avoid loops in routes.

When a target station responds to a message it has received, it simply reverses the route in the received header to obtain a route back to the original source.

When a station is operating promiscuously, it investigates each received header packet, and attempts to learn any routes that it can.

If a station receives a message where its own address is in the header listed as a digipeater, the message will be updated and passed back to the Data-Link layer for forwarding.

If a digipeating station is informed by its Data-Link layer that a transmission failed to the next target, the station will create a LER message to the original SRC (new DST), listing the address of the faulty node, and inserting its own address as the new initiating SRC.

If a node receives a LER of which it is the target (the original SRC), a new route discovery for the original destination will be initiated.

Any station that learns of a faulty link, directly or indirectly, immediately remove such a link from their route cache.

5.4.3 Data-Link Module

The Data-Link section of the code is responsible for error detection and correction, message packaging, channel access control, flow control and transmission retry attempts. It also maintains a queue of messages waiting to be sent, and caches all sent messages that require confirmation of receipt. The system has been designed to handle asynchronous transmission cycles. In other words, multiple messages can be sent, the responses of which can come back in any order. The station does not have to have a response to one message before sending the next. This is critical for the success of the multi-hop network, so that even when a station is waiting for a response to a message it initiated, it can handle and forward messages from other stations without delay.

The operation of the Data-Link module is best described by explaining the transmit and receive sequences:

5.4.3.1 Message Transmission Handling

When a message is passed down from the upper layers for transmission, the Data-Link layer looks at what type of message it is, sets relevant flags and stores the message together with its flags in the FIFO queue of messages waiting to be sent (Send Queue). There are three flags relevant to each message, all pertaining to flow control. The first indicates whether or not the station should expect an end-to-end confirmation of receipt. The second indicates whether or not a passive acknowledgement is required. If there are digipeaters in the route, this flag will be set. If neither of the first two flags are set, then

the station will not expect a response for the message. This would be the case when a RRQ is forwarded. The third flag indicates whether the RTU should delay the message before sending, as would be the case when forwarding a RRQ, or replying with a route to a DST, giving the DST a chance to respond first. If its not set, the RTU will send the message immediately if the channel is clear.

The radio SQL is available to the code as an input on pin B1. This is used to determine whether the channel is busy or not. If the channel is clear upon sensing, the station will initiate transmission. In order to optimise processing, the code performs CRC, encoding, and packages the message waiting for the carrier to rise. Note that the RTU can be configured not to encode the data, but simply include a checksum. If the channel is found to be busy when a station wishes to transmit, it enters a random back off period before sensing the channel again. This is chosen as a Poisson random period, with a mean of an average transmission time. This process can be repeated to an amount of times definable in the RTU configuration.

After the PTT of the transmitting radio has been released, the code enters a delay period of 65ms. This procedure was followed because the squelch of a receiving radio takes approximately 65ms to close after the PTT of a transmitting radio is released. This delay prevents the transmitting radio from immediately seizing the channel again should it have a another message waiting to send. If there is no delay, it would sense the channel as clear, and initialize a transmission which would most likely result in a collision. The delay after transmission allows all radios to have an equal chance to access the channel.

Once a message has been sent, the original message (not the encoded one) is stored in the Sent Cache, along with its flags, and its retry timeout period (if applicable). As previously discussed, this period also follows a Poisson distribution, to minimize the possibility of synchronous collisions.

5.4.3.2 Message Reception Handling

When a message it is received, it is stored in the RXEncoded buffer. Upon completion of reception, the start and end bytes are located using autocorrelation (described in Section 5.4.3.4). Once the message length has been determined, the message is decoded. Note that each RTU is capable of receiving encoded and decoded messages. Before decoding starts, the software performs a CRC check on the received data. If it is encoded or contains an error, the check will fail and decoding will commence. However, if the check is valid, then the received data was obviously not encoded. A valid result is stored in the RXDecoded buffer for further processing.

If a response is received for a message in the Sent Cache, before the retry timeout is reached, the record will be erased from the Sent Cache. On the other hand, if no response

is received before the retry time expires, the PIC will increment the attempt counter, and add the message to the Send Queue for retransmission. Note that a response to a message whose reattempt is queued for transmission is considered valid, and the relevant retransmission will be removed from the Send Queue. If the maximum number of retry attempts is reached, the message will be deemed to have failed, and the Network layer will be alerted to take the appropriate action.

5.4.3.3 Message Framing: Zero Padding

Given a message of m bytes, plus two checksum bytes (CRC-16), the total message length is $m + 2$. In order to encode the message, encoder requires frames of 11 bits, thus the message bit length must be an integer multiple of 11. If it is not, zero bits must be added to the message. The number of zero bits required is calculated by

$$\text{zero bits} = 11 - \text{mod}(l, 11)$$

where l is the message length in bits, and mod is the remainder after division. Note that the CRC is encoded with the message to to indicate if the code failed to correct errors. The process is illustrated in Figure 5.3.

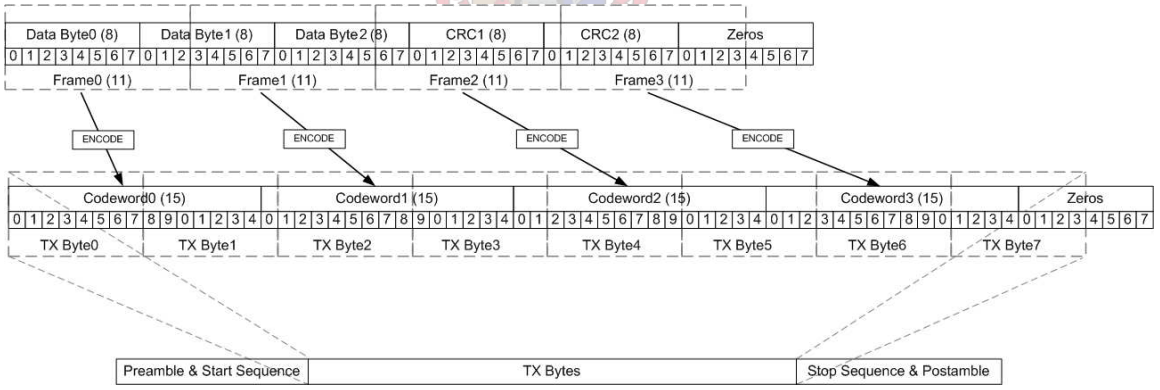
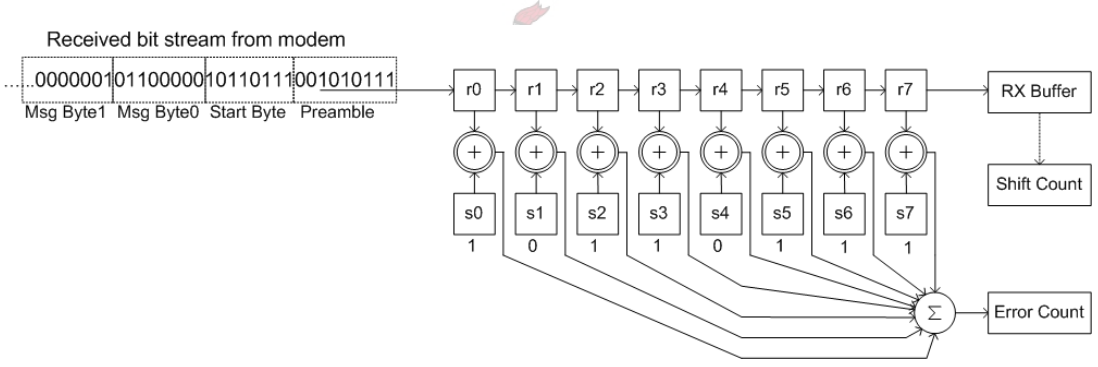


Figure 5.3: Example of how message framing is performed.

5.4.3.4 Message Framing: Start/Stop Bytes

The data sheet of the CML FX469 modem specifies that the modem requires a 16 bit preamble to acquire synchronization with the incoming sinusoidal signal. Once synchronization is obtained, the carrier detect (CDT) pin of the modem will go high to indicate

data is being received. The CDT pin is used as an interrupt signal to alert the processor of incoming data. However, the time that it takes for the CDT pin to go high varies slightly with the noise level. This presents a synchronization problem when receiving data, as the received preamble length may vary from time to time. This means that the data payload of each received message will not be received and buffered at exactly the same time relative to the CDT pin going high. Thus, a message start indicator is required. For robustness, it was decided to use autocorrelation of a 8 bit message start sequence. The 8 bit sequence was chosen as 10110111 (183), in keeping with autocorrelation methods. The start byte is inserted after the 16 bit preamble, and indicates the start of the data message. When the receiving station detects a CDT high from the modem, it immediately starts buffering the incoming bit stream. Once the entire bit stream has been buffered, the processor autocorrelates the start of the received bit stream with the known start byte (183), shifting the bit stream by one bit after each autocorrelation. The shift position at which the least number of errors occur indicates the position of most probable start byte. The process is similar for locating the stop byte of a message, also 183, except that the autocorrelation begins at the end of the bit stream and moves towards the front.



Start Byte = 183

	Register Contents								
Shift	r0	r1	r2	r3	r4	r5	r6	r7	Error Cnt
0	0	1	0	1	0	1	1	1	3
1	0	0	1	0	1	0	1	1	4
2	1	0	0	1	0	1	0	1	2
3	1	1	0	0	1	0	1	0	6
4	1	1	1	0	0	1	0	1	3
5	0	1	1	1	0	0	1	0	4
6	1	0	1	1	1	0	0	1	3
7	1	1	0	1	1	1	0	0	5
8	0	1	1	0	1	1	1	0	5
9	1	0	1	1	0	1	1	1	0
10	0	1	0	1	1	0	1	1	5
11	0	0	1	0	1	1	0	1	4
12	0	0	0	1	0	1	1	0	3
13	0	0	0	0	1	0	1	1	5
14	0	0	0	0	0	1	0	1	4
15	1	0	0	0	0	0	1	0	4
16	1	1	0	0	0	0	0	1	5
17	0	1	1	0	0	0	0	0	6
18	1	0	1	1	0	0	0	0	3
19	0	1	0	1	1	0	0	0	7
20	0	0	1	0	1	1	0	0	5

Start Byte = 182

	Register Contents								
Shift	r0	r1	r2	r3	r4	r5	r6	r7	Error Cnt
0	0	1	0	1	0	1	1	1	3
1	0	0	1	0	1	0	1	1	4
2	0	0	0	1	0	1	0	1	3
3	1	0	0	0	1	0	1	0	5
4	1	1	0	0	0	1	0	1	4
5	0	1	1	0	0	0	1	0	5
6	1	0	1	1	0	0	0	1	2
7	1	1	0	1	1	0	0	0	6
8	0	1	1	0	1	1	0	0	6
9	1	0	1	1	0	1	1	0	1
10	0	1	0	1	1	0	1	1	5
11	0	0	1	0	1	1	0	1	4
12	0	0	0	1	0	1	1	0	3
13	0	0	0	0	1	0	1	1	5
14	0	0	0	0	0	1	0	1	4
15	1	0	0	0	0	0	1	0	4
16	1	1	0	0	0	0	0	1	5
17	0	1	1	0	0	0	0	0	6
18	1	0	1	1	0	0	0	0	3
19	0	1	0	1	1	0	0	0	7
20	0	0	1	0	1	1	0	0	5

Figure 5.4: Example of the autocorrelation process followed to find the start/stop bytes.

5.4.3.5 Encoding/Decoding

As stated in Chapter 2, Hamming (15,11) is used when the RTU is configured to encode messages. The encoder uses 11 bit frame ($m_0 \rightarrow m_{10}$) to which it adds four parity check bits ($c_0 \rightarrow c_3$). The generator matrix was provided by the MATLAB Hamming function. The parity check bits are generated in software as follows:

$$c_0 = m_2 \oplus m_3 \oplus m_5 \oplus m_7 \oplus m_8 \oplus m_9 \oplus m_{10}$$

$$c_1 = m_1 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_7 \oplus m_8 \oplus m_9$$

$$c_2 = m_0 \oplus m_1 \oplus m_3 \oplus m_5 \oplus m_6 \oplus m_7 \oplus m_8$$

$$c_3 = m_0 \oplus m_3 \oplus m_4 \oplus m_6 \oplus m_8 \oplus m_9 \oplus m_{10}$$

The syndrome is generated in software as follows:

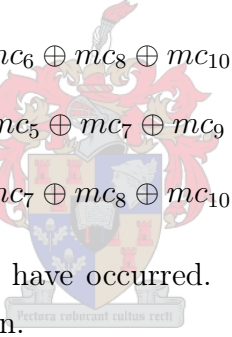
$$s_0 = mc_3 \oplus mc_6 \oplus mc_7 \oplus mc_9 \oplus mc_{11} \oplus mc_{12} \oplus mc_{13} \oplus mc_{14}$$

$$s_1 = mc_2 \oplus mc_5 \oplus mc_6 \oplus mc_8 \oplus mc_{10} \oplus mc_{11} \oplus mc_{12} \oplus mc_{13}$$

$$s_2 = mc_1 \oplus mc_4 \oplus mc_5 \oplus mc_7 \oplus mc_9 \oplus mc_{10} \oplus mc_{11} \oplus mc_{12}$$

$$s_3 = mc_0 \oplus mc_4 \oplus mc_7 \oplus mc_8 \oplus mc_{10} \oplus mc_{12} \oplus mc_{13} \oplus mc_{14}$$

If the syndrome is zero, no errors have occurred. The non zero value of the syndrome corresponds to an error bit position.



5.4.4 Physical Module

This section of the code is concerned with control of the physical communications medium, i.e. the radio and modem. Once again, the transmit and receive sequences are described.

5.4.4.1 Transmit Sequence

The PIC initiates the transmission sequence by setting pin B7 high, thereby keying the radio. The radio requires 150ms to reach its full power output, during which time the Data-Link layer encodes the message. After the run up delay, the PIC sets the modem into transmit mode by setting pin B3 low ($\overline{\text{TX Enable}}$). When in transmit mode, the modem generates a clock signal which is input to the PIC on pin B6. The modem samples the bit present on pin B4 of the PIC (TX Data) upon each rising edge of this clock. To ensure the data is valid for the modem, the PIC places bits (read sequentially from TXCurrent buffer) on this pin upon each falling clock edge. The length of the bitstream is passed to

the transmit function by the Data-Link layer. Once all bits have been set, the modem $\overline{\text{TX Enable}}$ is pulled high, and the PTT is released.

5.4.4.2 Receive Sequence

When the modem detects a MSK signal, it sets the CDT pin high. This output is common to pin B0 of the PIC, which is configured as an external interrupt input. Upon detection rising edge on this pin, the PIC calls an interrupt service routine, which checks the state of the radio squelch. If the radio is receiving, and the CDT pin is still high, a global flag is set which alerts the main code of an incoming MSK message, whereafter the Receive function is called. The modem provides a clock signal to the PIC on pin B5. It places received data on pin B2 upon each rising edge. Data is sampled by the PIC on each falling clock edge. The PIC packs the received bit stream into bytes, and stores the bytes directly into SRAM (RXEncoded Buffer) for processing by the Data-Link layer upon completion of the receive sequence. Note that the modem always remains in receive mode, because message arrival times are not fixed. This does not pose any problem, as the FX469 is capable of full duplex operation.

Variable length messages posed a problem in that a method was required by the PIC to indicate when to begin and when to end buffering a received bit stream, as the length was uncertain. It was not considered be viable to include a message length byte at the start of the message, as the message has to be processed in its entirety to check for errors before its contents can be deemed valid. Another option would be to use the squelch output of the radio. However, the squelch of the radio remains open for much longer than the actual transmission, due to the physical rise and fall times of the radio circuitry, so valuable processing time would be wasted. Thus, another method was required. The carrier detect output of the modem provided a good indication of when to start receiving a message, however it was found to be unreliable as an indication of when to stop, as its level changed several times during reception of a noisy message. Thus the PIC would incorrectly stop receiving the message at the first CDT dip. The next rise of the CDT pin would trigger another interrupt which would initiate a new reception sequence, and the message would be lost. Therefore it was decided to use an averaging technique on the carrier detect pin. For every bit that the PIC receives, a counter is incremented if the CDT was high during that bit period. On every eighth bit, a majority logic decision was taken. If the CDT is high for 50% or more of a byte, the counter is reset and the PIC continues to receive bits. On the other hand, if the counter is less than 4, the PIC decides the data sting has ended, and halts reception. Processing of the received message is carried out while the SQL is open, thereby preventing re-entry into the receive function.

5.4.5 Memory Module

As previously stated, external SRAM was employed to provide sufficient memory to keep track of all recent messages and routing information. The onboard EEPROM of 256 bytes was sufficient to store all necessary configuration information.

5.4.5.1 SRAM

The SRAM is divided into buffers of various sizes, each having an offset pointer defined in the header of the code. It is used to store all messages and routing information. As mentioned in the Chapter 4, the SRAM read/write cycles are much faster than the instruction cycles of the PIC, so no time is wasted in reading or writing to SRAM. 64KB of SRAM is available for use, correlating to 16 address lines, i.e. 16 bit addresses. The address lines are driven by an 8 bit data bus through two octal flip flops. For any read/write sequence, the correct address must be present of the address pins of the SRAM. Before the 16 bit address is set up the SRAM data bus is forced into high impedance (disabled) by setting pins C2 ($\overline{\text{OE}}$) and C3 ($\overline{\text{WE}}$) high. The first 8 of the 16 bits are placed on the data bus, and the flip flops are pulsed. The second 8 address bits are placed on the data bus, and the flip flops pulsed again. To write a byte to the address, the PIC places the byte on the data bus, and pulses $\overline{\text{WE}}$ low. To read a byte from the address, $\overline{\text{OE}}$ is set low, placing the byte on the SRAM data bus. The PIC reads the data bus, then sets $\overline{\text{OE}}$ high, forcing the SRAM back into high impedance. Either sequence takes less than 2μs.

Two functions were written to carry out the read an write procedures. The parameters passed to the write procedure are the 16 bit address and the data byte, while the read procedure parameter return the data byte when given the 16 bit address as a parameter.

5.4.5.2 EEPROM

The PIC onboard 256 byte EEPROM is used to store configuration and operational settings for the RTU. The EEPROM contains the registry that is mirrored in the server database. EEPROM is suitable for this as it holds its contents after power is removed from the PIC. All settings pertaining to timeouts, addresses, retry constraints, I/O alarm reporting and I/O status are stored in this memory. The program code uses the settings stored in EEPROM as instruction regarding what action should be taken at various points in the code. EEPROM read/write cycles are time consuming, thus at startup, the code assigns RAM variables to the relevant values stored in EEPROM, allowing for faster processing during program execution. If any changes occur in configuration settings the new values are written to EEPROM and the variables are initialized to the new values. The CCS C compiler provides functions with which one can read and write the EEPROM.

5.4.6 I/O Module

The I/O section of the code continuously monitors the status of configured inputs, and sets outputs as instructed. The DINs and DOTs are read and received via the logic expansion circuitry as explained in Chapter 4. AINs are connected directly to the PIC ADC. Values stored in EEPROM are used to decide which inputs are monitored, which outputs are set, and what action should be taken when the inputs change. Usually the action taken is in the form of event generation, i.e. the server is alerted of the change in input. For example, the eight DOTs are assigned to a specific byte in EEPROM. Each bit in the byte corresponds to the status of the output, i.e. if bit0 is set, DOT0 will be high. Should the server wish to change the status of the DOTs, it sends a message instructing the RTU to write a new value to the relevant EEPROM address. On the next cycle of program execution, the I/O software will read the new value from EERPOM and write this value to the DOTs.

Each digital input (DIN) requires several bits in EEPROM. The first bit, if set, tells the code whether the input should be monitored or not. The second bit, if set, tells the code that if the input changes from high to low, the change should be reported to the server. The third bit, if set, tells the code that if the input changes from low to high, the server should be informed. The fourth bit, if set, indicates that the input is configured as a counter input (CIN). If this is the case, the second and third bits do not apply. Each CIN has a corresponding 16 bit counter space stored in EEPROM. Counter inputs are checked every 10ms, and the relevant counter is incremented if a change is detected. Each counter has a configurable threshold, also stored in EEPROM. If the threshold is non-zero, and the counter exceeds this threshold, an event will be generated.

Like CINs, AINs also require several bytes in EEPROM. A specific byte indicates which AINs should be monitored. Each AIN has configurable low and high thresholds stored in memory. If the AIN is configured to be monitored, and the thresholds are non-zero, when the AIN value falls below its low threshold, or exceeds it high threshold, an event will be generated.

Once an event has been sent to the server, all events that occur before confirmation of receipt has been received are not transmitted, but rather logged until such receipt is received. Once this occurs, all logged events are transmitted in one message to the server.

5.4.7 Miscellaneous Functions

5.4.7.1 Onboard Time Counter

Every millisecond, a timer interrupt service routine is carried out. A 32 bit millisecond counter is incremented each time the routine is called. The program code makes use

of this counter for all timing functions, for example, retry and back off timeouts, CIN monitoring, radio run up times etc. When the millisecond counter reaches 86400000 (24hrs), it is reset. Thus the counter can be used as an indication of the time of day. It can be synchronized with a clock by the server application. Note that while code execution is inside an interrupt, the PIC disables all other interrupts. Thus, the interrupt service routines for the USART and CDT are kept as short as possible by simply setting flags, so that the counter does not lose significant time.

5.4.7.2 Random Number Generation

The CCS compiler provides a function that generates an 8 bit pseudo random number. Upon start up, the code uses the stations address (stored in EEPROM) as the seed for pseudo random generation. This ensures the sequence at each station will be different. A pseudo random number is used for the UID tag of each message sequence.

In order to generate a Poisson distributed random time, the following function is used.

$$t_{delay} = -\ln(r) \times t_{mean} \tag{5.4.1}$$

where r is a pseudo random number between 0 and 1. This is simply the inverse of the exponential function that describes an exponentially distributed (Poisson) arrival rate (Wolhuter, 2005). The function was tested on the PIC, and the results were validated as having a Poisson distribution by using Matlab's "poissfit" function. Poisson distributed numbers are used for back off and retry timeouts.

5.4.7.3 Server Connection

Although the server uses the "MODBUS type" protocol to communicate with the network, a private connection with the local RTU is also supported. This function was developed for diagnostic purposes. The server sends a periodic "ping" message to determine if an RTU is present on the serial port. An RTU that detects a "ping" responds accordingly. Thus the server and RTU are aware of each others presence. While the PIC is aware of a direct serial connection to the server, it sends status messages to the server, indicating the occurrence of events. For example, if a message was received, overhead or sent, the PIC informed the server of the message type, and route. Changes in the route cache were also sent to the server. This provided a means to monitor the channel activity at the local outstation.

5.4.8 Main

The main program runs in a continuous loop, and is responsible for integrating all sections of code together, and passing control to the relevant functions when required. After a function has performed its task, control is returned to the main program.

5.5 Conclusion

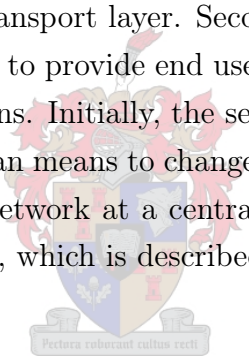
This chapter has described the function of the embedded software, in is role of integrating all layers together, and controlling all onboard operations. The code was divided into various sections and subsections, as depicted in Figure 5.2. Each section and subsection was explained. Appended flow diagrams describe the flow of data and operational procedures in more detail. Complete C code listing is also included in the appendix.

Chapter 6

Server Software

6.1 Introduction

The server software has two objectives. Firstly it provides users with an entry point to the network, interfacing with the Transport layer. Secondly, it provides a platform on which a custom GUI can be developed, to provide end users with a display of data, and present output control at remote locations. Initially, the server software was written to facilitate testing of the system, providing an means to change parameters on each station remotely, and record data regarding the network at a central point. Much of this functionality is carried over to the server proper, which is described in this chapter.



6.2 Server Description

The server code was written using Delphi, which is an IDE based on the object orientated version Pascal. Delphi uses components which are objects with a set of properties, functions and event handlers that are used to perform various tasks. Custom components can be created by inheriting from existing components to extend functionality.

The server communicates with an RTU via a serial port. There are various open source components available for Delphi to simplify use of the serial ports. The baud rate was selected as 9600, as this was found to be the highest that the link could reliably support. The server communicates with the network using a protocol similar to MODBUS, as mentioned in the previous chapter.

Basically, the purpose of the server software is to mirror the registers of all stations in the network. The registers of each RTU contains information pertaining to the local I/O points and configuration, therefore the mirrored registers at the server will contain the same information. Thus the server provides a means to store input data from remote stations at a central point. The data can be used to display system status information, or

for historical trend recording. It also provides a means to see the status of configuration settings and outputs, and provides control over these.

In order to keep a record of all this information, a database was created using Delphi's database components. Each RTU in the system has an entry in the database, indexed with is address. When the server receives information from an RTU, the relevant database entries are updated, and the database is saved. If a user changes a configuration setting at the server for a specific RTU, the server sends the information to the relevant RTU, and upon acknowledgement, updates the database accordingly. Each column of the database corresponds to an EEPROM address at each RTU.

In addition to the server protocol, the server also supports a private connection with the RTU to which it is connected. The protocol structure is identical, however the message types cannot propagate further than the local RTU. The purpose of this connection is to monitor the status of the local RTU. While the server is running, it sends a "ping" message via the serial port every few seconds, to query if there is a local RTU present. If an RTU is present, it will respond with its address and the ping message. While the PIC is aware of an active connection to a server, it will transmit status messages to the server for certain events. For example, it will inform the server each time it transmits or receives a message, along with the message type, and relevant routing information. This allows a user to connect to any RTU in the system, and monitor the network activity at that point. Note that the PIC will only transmit messages to the server if it is aware of an active connection.

The server has a GUI which was kept simple but informative. It has a panel in which all status messages from the local RTU are displayed, as and functional buttons to perform various tasks. The server allows a user to read or write a selection or a sequence of resisters. Configurable settings include:

- FEC on/off
- Mean Back Off Time
- Mean Retry Time
- Set Address
- Maximum Route Length
- Route Cache Size
- Maximum Retry Attempts
- Radio Run Up Time

- Address of server

I/O Settings include:

- Report selected DIN changes: H/L, L/H, ANY
- Monitor selected CINs, report on threshold
- Monitor selected AINs, report on threshold
- Set AIN high and low thresholds
- Set/Clear DOTs

Command Settings include:

- Reset Station
- Clear Route Cache
- Silence/Unsilence Station

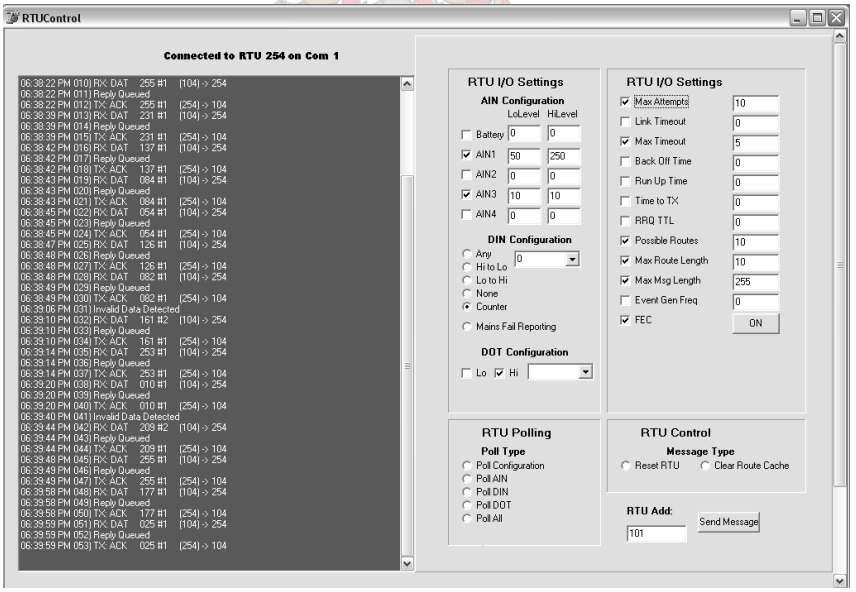


Figure 6.1: Screenshot of the original server application.

6.3 Data Display

The server software links the database to the RTU registers via the transport layer. Generally, the server would run in the background, and a further application would run using the database to display and control custom settings. This is the application that the client would see, and is the highest layer in the system. A simple example of such an application was written for purposes of completion.

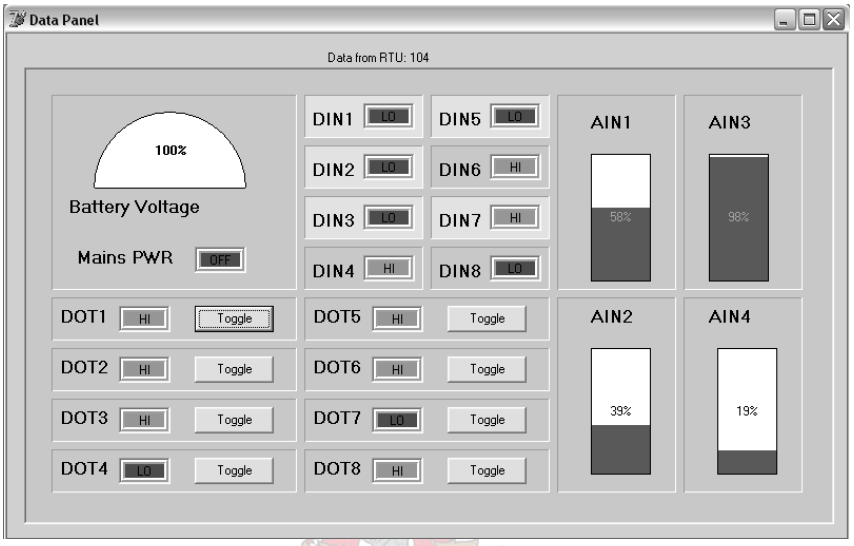


Figure 6.2: Screenshot of an example application displaying RTU I/O data.

6.4 Conclusion

This chapter has provided a description of the server software and described its function running above the Transport layer. The server application provides a means for a user to access the I/O points of any RTU in the system. The server maintains a database, the entries of which, corresponding to registers of remote RTUs. Also, the server allows a user to monitor all traffic at a local RTU by means of short status messages provided by the RTU.

A further demonstration application running above the server was described. This application uses data stored in the server data base and displays it graphically. Such an application could be customized to suit user requirements, as is common with most SCADA based systems. For example, changes certain inputs could be recorded and timestamped to facilitate investigation of trends in the system.

Chapter 7

System Performance Prediction

7.1 Introduction

In any communications system, utilization of the available bandwidth to achieve the optimal throughput is of primary concern. Bit error rate effects throughput when dealing with the physical layer, while queueing delay, or time spent in a buffer, is of concern when dealing with the Network layer. The time that data spends in the system should be kept to a minimum.

This chapter offers a prediction of the performance of the system for various scenarios. Initially, a prediction of routing setup times for a selection of network topologies is discussed. This is followed by an investigation of a round robin polling strategy strategy for various network topologies, data message lengths, and channel noise. The remainder, and bulk, of the chapter is dedicated to modeling the system performance when operating in the CSMA mode, using queueing theory. A selection of results is presented graphically at the end of the chapter.

7.2 General

When designing a new data network, it is critical for the designer to produce an optimal system. Prototypes could be built and measured, however adjusting parameters to find the best configuration can be extremely time consuming and exhausting. Simulation techniques could be used, but most available simulators are very general, and do not have enough input parameters to produce accurate predictions for specific networks. For example, many simulators do not provide for real world parameters like system overhead (e.g. rise times) or noise.

Apart from building prototypes or using simulators, queueing analysis is often used in system performance evaluation. Not limited to data networks, queueing theory is used in

a wide spectrum of areas, from transportation, to queues in a supermarket. When applied to data network, queueing theory can provide extremely accurate estimations of system performance. An excellent contribution using queueing analysis to theoretically model half-duplex contention type telemetry links, based on finite source systems can be found in Wolhuter (2002). This work provides for additional real world parameters, including back-off strategy, channel noise, channel sense time, and transmitter rise time. Although this work is for a finite source system, it provides an excellent starting point for creation of a model to predict the performance of the system under consideration. To the best of the authors knowledge, no model appears to have been developed for the said system.

As previously discussed, the system can be configured either in RRP or CSMA mode. Both strategies will be covered for separately for completion. In the text, the term "event generation" refers to an outstation sending a message to the server because of a specific change in one or more of its inputs. Note that in all cases, the master station refers to the RTU to which the server is connected via RS232. It is of no concern to the server software which RTU it is connected to, as all modules are identical, and all can function as the server entry point. Generally, however, the RTU which is connected to the server has the address 254, as all outstations that generate events target this address by default. However, this is configurable.

7.3 Routing Set Up Times

The time that the network takes to reach steady state, i.e. all required routes are known and active, depends on the network topology, traffic loading, and the knowledge that nodes have. For example if all stations were in range of each other, and one station sent a message, all stations would immediately learn a route to the initiating station. On the other hand, if the stations were in a pipeline configuration, the routing messages would be subject to the forwarding delay of being repeated at each station. In short, the exact set up time is difficult to quantify, as it depends on many variables. To illustrate, consider the network topology depicted in Figure 7.1

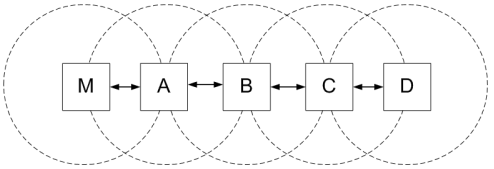


Figure 7.1: Pipelined network topology.

Assume the stations have no previous knowledge of the network, and the system is noiseless. If M wishes to find a route to A, it will broadcast a RRQ, to which A will respond

immediately with a RRP. B will over hear A's reply, and in so doing, learn a route to A.

$$T_{A-B} = t_{RRQ} + t_{RSP} \tag{7.3.1}$$

$$t_{RRQ} = t_{preamble} + t_{postamble} + t_{bits}[\text{RRQ bits}]$$

$$t_{RRP} = t_{preamble} + t_{postamble} + t_{bits}\times[\text{RRP bits}]$$

The initial RRQ length is 64 bits. It is simply the header length, plus the 16 bit CRC. The RRQ message length increases by one byte at each station that forwards the RRQ, as it appends its own address to the route. The RRP length is identical to the received RRQ length, as the same route is utilized.

If M wishes to find a route to D, it will broadcast a RRQ, which will be forwarded by A, B and C, after a small random delay which allows any stations with a route to the destination to respond first. When the RRQ reaches D, D responds immediately with a RRP. The RRP is sent back along the source route. Upon completion of the cycle, every station in the network has a route to every other station.

$$T_{M-D} = h \times (t_{RRQ} + t_{RSP} + 8t_{bits}) + \sum_{i=1}^{h}(i-1) \times (8t_{bits} + \sigma) \tag{7.3.2}$$

where h is the number of hops in the route, and σ is the small delay at the stations forwarding the RRQ leg. Note that the additional overhead due stations appending their addresses is minimal compared to the total transmission time, because of hardware rise times of the radio. The main delay is constituted of the digipeating delay.

7.4 Round Robin Polling Strategy

To simplify modeling, the system is considered stable. Routing messages are ignored, as they only contribute to overhead initially, when the network is setup up. Further routing messages will only be incurred upon a change in the network, if a station is moved or battery goes flat after a mains failure. As it is a nomadic (portable rather than mobile) system, it is unlikely that changes will occur very often.

Round Robin Polling would be the case when event generation at stations is disabled. A brief review of the operation of the strategy is provided. Channel activity can only be initiated by the master station. This means that an outstation will only transmit in response to a poll from the master station. The master station schedules all transmissions, so there will never be any collisions between stations. Thus, a retry will only occur as a

result of channel noise. The master station will poll each station in turn, and the relevant outstation will respond with the required data accordingly. It is not necessary for the master station to reply with an ACK or NAK to the outstation. Instead, the master station will retransmit persistently if no reply is received from the outstation after a fixed period. This period, t_{retry}, is chosen in the order of t_{rsp} for the RRP case, i.e. the time after which the master station should have received a response.

First consider a simple case, where master station A has a direct link to outstation B. Ideally, the minimum cycle time is given by:

$$t_{min} = t_{poll} + t_{rsp} \tag{7.4.1}$$

$$t_{poll} = t_{preamble} + t_{postamble} + t_{bits} \times (b_{header}) \tag{7.4.2}$$

$$t_{rsp} = t_{preamble} + t_{postamble} + t_{bits} \times (b_{header} + b_{data}) \tag{7.4.3}$$

If either poll or response message fails, a delay of t_{retry} and a new cycle will effectively ensue, in addition to the time taken to send the initial poll. Although the response message will always be longer than the poll request, and therefore more susceptible to noise disturbance, it must be emphasized that corruption of either message results in the same delay. It is thus valid to view noise disturbance as effecting a complete cycle, rather than the poll request or response message individually.

There are two approaches to include the noise factor. Either the BER can be calculated for different SNRs, or one can make an assumption that a certain percentage of cycles will be corrupted, resulting in delays. This is perhaps a more suitable approach, as it can provide for burst errors, for which no actual model exists, as well as channel noise (Wolhuter, 2002). If one assumes that a certain percentage, δ_n of the cycles will be corrupt by channel noise, then the maximum and mean times will be given by:

$$t_{max} = t_{poll} + t_{rsp} + \delta_n(t_{poll} + t_{retry}) \tag{7.4.4}$$

$$t_{mean} = \frac{(t_{max} + t_{min})}{2} = t_{poll} + t_{rsp} + \frac{\delta_n}{2}(t_{poll} + t_{retry}) \tag{7.4.5}$$

Next, consider the situation where intermediate nodes are required to digipeat a message. A digipeater receives a message entirely before retransmitting it, therefore each additional digipeater constitutes an extra cycle transmission. Ideally,

$$t_{min} = h \times [t_{poll} + t_{rsp}]$$

where h is the number of hops (number of digipeaters+1). As before, A will wait t_{retry} after transmitting a poll request for passive acknowledgment. Retries occur locally (at each node) instead of end to end, so the mean wait time is simply given by

$$t_{mean} = h \times [t_{poll} + t_{rsp} + \frac{\delta_n}{2}(t_{poll} + t_{retry})]$$

The mean wait time, per station, for N stations will be

$$t_{mean} = \frac{\sum_{i=1}^{N} h_i \times [t_{poll} + t_{rsp} + \frac{\delta_n}{2}(t_{poll} + t_{retry})]}{N} \tag{7.4.6}$$

where h_i is the number of hops in each route to each station. Thus, the mean wait of the system time of is clearly a related to the number of the digipeaters in the system. Referring again to Figure 7.1, the mean polling cycle time to each station is:

1. M⇔A : $1 \times t_{mean}$
2. M⇔B : $2 \times t_{mean}$
3. M⇔C : $3 \times t_{mean}$
4. M⇔D : $4 \times t_{mean}$

The collective mean time to poll A,B,C and D is thus $10t_{mean}$. The mean polling time per station in the system is the collective mean time for all outstations divided by the number of outstations, which is $2.5t_{mean}$. The RRP method is straightforward and deterministic as it is controlled by the master station. Thus its is simple to model and test. A selection of simulated results are depicted in Figures 7.6 through 7.8 for various length data payloads.

7.5 CSMA Strategy

Carrier Sense Multiple Access is a contention scheme whereby stations sense the channel when they wish to transmit. If the channel is found to be free, transmission is initiated. If the channel is found to be busy, a back off period is entered into, after which the channel is sensed again. This is used for cases where stations are configured to report events automatically. CSMA presents a more complicated operating mode, as collisions can occur especially due to the hardware rise times of analogue radios. The "hidden terminal" effect increases the collision rate even further. To illustrate this effect, consider the topology shown in Figure 7.2. M is in range of all four outstations A, B, C, and D. However, not one of the outstations are in range of each other, i.e. they are hidden form each other. Thus, even if A is transmitting, C will assume that the channel is clear, and could also transmit, causing a collision at M. The number of stations hidden from each other directly influences the possibility of collisions. Obviously, this becomes more pronounced as the event generation rate is increased.

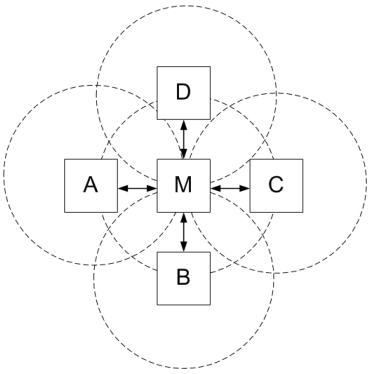


Figure 7.2: Hidden terminal network topology.

This effect, combined with the pipelining delay, poses a serious problem to a heavily loaded system, because of the increased latency due to collisions. For the purpose of the model development, it is assumed that all stations are in direct range of the master station, and in direct range of each other, i.e. no hidden terminals and no multi-hops. The result is then extended to incorporate multi-hops and hidden terminals.

Queuing theory is used to provide a theoretical prediction of performance, the basis being that, while a message is in a buffer, waiting for transmission, it is effectively in a queue waiting for channel access. Once a message has been transmitted, it is effectively placed in a second queue, waiting to be handled by the server. This can be considered as an overall queue, as messages from all stations are placed in this queue. Factors like collisions, back offs, and noise effectively increase the time that a message spends in the queue. It is intuitive that the busier the network, the longer a message will be in a queue. The time that it takes from an event occurring, until the time that the station receives confirmation of receipt of the event from the server is the latency, or the overall wait time of the message.

7.5.1 Review of Queueing Theory

A brief review of queueing theory is provided as a background before the model is presented. The simplest queuing system is described by an example as follows:

People (customers), from some population source, arrive at a bank ATM desiring to perform some transaction. If there is no one in the queue when a person arrives, they can use the ATM immediately, otherwise, the person must join the queue. Queueing theory can predict the average queue length and the average time a person will spend in the system, i.e. from when they join the queue, until they complete their transaction and leave the ATM. Queues are characterized by several parameters. The parameters are introduced in terms of the ATM example, and thereafter discussed in terms of the telemetry network under consideration.

- **Source Population**

  The source population is the source from where customers originate. In a large shopping mall, the source population can be considered infinite, as there are endless people who can join the queue. However, in a very small town, where only ten people lived, there are only ten people who could possibly queue for the ATM. This is known as a finite source population. For every person that joins the queue, there is once less person available in the town to join the queue. When a person completes a transaction, there is one more person available to join the queue.

- **Arrival Rate and Distribution**

  The rate at which people arrival at the ATM directly effects the queue length, and thus the average time spent in the system. Note that in a finite source system, the population available for arrival reduces by each additional person that joins the queue, thus the average arrival rate is proportionally reduced. In an infinite source system, the average arrival rate remains constant. The arrival rate can follow various probability distributions and depends on the type of system in question. Arrival patterns could be regular, or totally random.

- **Number of servers**

  The queue length is affected by the number of servers; if there were multiple ATMs available for use instead of just one, the time spent in the system would be shorter.

- **Queueing Discipline**:

  Although an ATM queue is first come first served, some queueing systems follow a first come last served structure, or some other prioritizing scheme whereby customers are served in order of importance.

- **Queue Capacity:**

  The stability of a queueing system depends on the maximum allowable queue length. If the queue length is less than the source population, it is possible that some customers may be lost when the queue reaches its maximum length. If the queue length is assumed to be infinite, the queue will grow boundlessly if items arrive faster than they are being served. Note that a finite source system will always be stable when the maximum queue length is equal to or greater than the source population. Generally the queue capacity is assumed to be infinite.

Modeling of queues is usually based on Markov chain theory. This means that the system is memoryless, so expected future occurrences are only based on the current system state, and not on history (Ng, 1996). Using this approach allows queues to be modeled mathematically using probabilities and statistics.

Figure 7.3 represents a basic queueing system with a single server. The rate at which customers arrive at the system is given by λ. The rate at which the server serves people is given by μ.

For a stable system, the utilization factor (ρ), which is the fraction of time that the server is busy, must always be less than one. The utilization factor is given by the arrival rate over the service rate.

$$\rho = \frac{\lambda}{\mu},\ \rho < 1 \text{ for stability} \tag{7.5.1}$$

The length of the queue for a single server memoryless queuing system is given by N, where

$$N = \frac{\rho}{1-\rho} \tag{7.5.2}$$

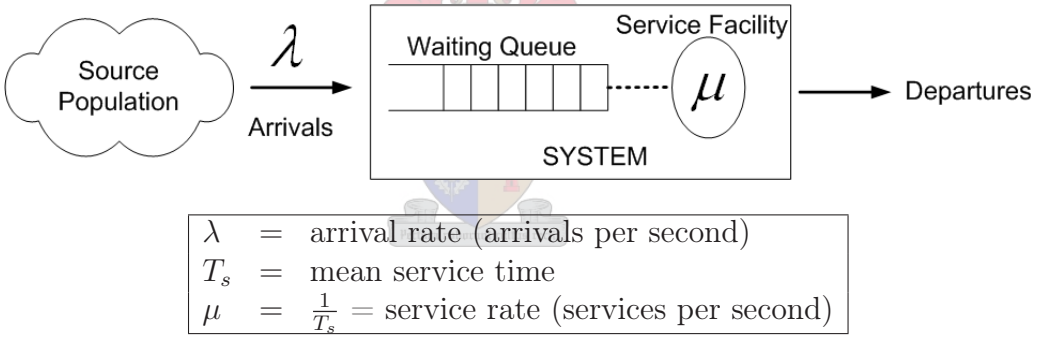


Figure 7.3: Simple single server queue.

Little's Theorem relates the average number of customers (N) in a stable queuing system to the average arrival rate (λ) multiplied by the average time a customer spends in the system (T).

$$N = \lambda \times T \tag{7.5.3}$$

From (7.5.2) and (7.5.3), the average time that a customer will spend in the queue, and being serviced is given by

$$T = \frac{1}{\mu - \lambda} \tag{7.5.4}$$

A discussion of queueing parameters relative to the telemetry system under consideration follows.

7.5.2 Number of Servers of a Telemetry System

The telemetry system has one server (master station) to which all data is directed. Thus, the overall telemetry system is considered as a single server queueing system.

7.5.3 Source Population of a Telemetry System

In a telemetry network customers in the system can be considered as events occurring at an outstation that require service by the master station. Assume an outstation is monitoring a critical input, such as a smoke detector in a building. The time of interest is how long will it take from when the outstation detects a fire in the building, until the outstation knows that the message has successfully reached the server at the fire station. Thus the customer requiring service is the actual event, and the time that this event remains in the system consists of the forward leg, and receipt confirmation leg, i.e. the time from when the event is generated, until the event is acknowledged by the server.

7.5.3.1 Finite Source Population

Remembering the simple ATM example, the basic property of a finite source system is that there are only a finite number of customers available to join a queue, i.e. there are only a finite number of events that can possibly join the queue to the server. Once an outstation places an event in the queue, it is effectively "out of play" until it this event has been dealt with. Thus, the maximum number of events a system can have is in fact equal to the number of event monitoring outstations in the network. For each additional outstation that places an event in a queue, there number of remaining outstations that can possibly generate events is reduced. The instantaneous arrival rate is reduced accordingly by the following equation, where N represents the total number of sources, and K represents the number of sources that already have an event in the queue.

$$\lambda_{Inst} = \lambda \left(\frac{N - K}{K} \right) \tag{7.5.5}$$

7.5.3.2 Infinite Source Population

In an infinite source system, the mean arrival rate is assumed to be fixed. There are assumed to be an infinite number of events that can occur, λ. This does not necessarily mean that there are in infinite number of outstations, but rather that each arrival at an outstation would constitute an immediate message to the server regardless of whether previous events have been acknowledged by the server or not. Returning to the ATM example, each outstation can be considered as one of a finite number of doors through

which customers (events) from an infinite source can enter into the room where the ATM is. People will arrive through each door at rate

$$\lambda_{Door} = \frac{\lambda}{N} \tag{7.5.6}$$

where N is the number of doors in the system.

7.5.3.3 Actual Telemetry System

In practice, a telemetry system is somewhat of a mix of both finite and infinite source cases, as it borrows characteristics from each. It would not be wise for an outstation to ignore any events that occur while it is waiting for a previous event to be acknowledged, but on the other hand, generating a message for every event would be impractical, especially at high arrival rates, due to the overhead of messages. Instead what happens in practice is that any subsequent events that arrive at an outstation after the first event has been placed in the queue are placed in a buffer at the outstation, and not transmitted immediately. As soon as the previous event (or message) has been handled, the entire buffer of subsequent events is added to the queue as one message. Thus messages may contain multiple events. All messages, regardless of the number of events they contain, are acknowledged by a single message from the server.

In effect, the stations follow the finite source property with regard to messages, by only placing one message in the queue at at time. However, as messages can contain multiple events, the system maintains a constant average event arrival rate at the server, which is a property of a infinite source system. Thus a typical telemetry system shares characteristics of both truly finite and truly infinite systems.

7.5.4 Arrival PDF and Service PDF of a Telemetry System

The arrival and service PDF's can be Markovian, General or Deterministic (M, G or D). The Markovian case is the most common choice, and has exponential inter-arrival times, or a Poisson PDF. Certain assumptions have to be made when using queueing theory to model a process mathematically. It should be kept in mind that an attempt is being made to model a telemetry system, and thus assumptions may not necessarily apply to the general case.

7.5.4.1 Arrival PDF

A Deterministic PDF implies that all arrival intervals are equal. This is not suitable to model a practical telemetry system, where the arrival pattern follows a random distribution. It is highly improbable that an absolutely constant arrival rate will occur. The

General distribution is difficult to handle analytically or model exactly (Wolhuter, 2005). As a result, modeling is commonly based on a memoryless, or Markovian process. In such a process, the future is totally independent of what has already occurred. Sufficient work exists concluding that the arrival pattern of a typical un-coordinated telemetry system can be described by the Poisson law, (7.5.7) (Wolhuter, 2002).

$$P_n(t) = \frac{(\lambda t)^n}{n} e^{-\lambda t} \tag{7.5.7}$$

Here, λ is the mean arrival rate and $P_n(t)$ is the probability that n (statistically independent) arrivals occur in an interval of length t. To conform to this law, the inter-arrival times must follow an exponential distribution:

$$A(t) = \lambda e^{-\lambda t} \tag{7.5.8}$$

7.5.4.2 Service PDF

It is most convenient to treat the service process as Markovian. It can be argued that the service time of a typical telemetry system is not fixed, but also follows a Poisson distribution, thereby validating this approach. The following points should be considered:

- Message lengths vary in accordance with Poisson distributed arrivals.
- Arrivals are random therefore channel usage will be random.
- Channel noise is random and it results in retries, reducing overall throughput. A reduction in overall throughput has the same effect as increasing the mean service time of the system (Wolhuter, 2002).

Kendall's notation, **A/B/m/k**, is commonly used to describe a queueing system, where:

- A=Arrival PDF
- B=service PDF
- m=No of servers
- k=Sub-class descriptor

Thus, the telemetry system can be defined as an M/M/1 queuing system.

7.6 Approach to Modeling an Infinite Event Source

The existing finite source state matrix model provides an indication of how long a *message* spends in a CSMA system, regardless of how many events the message contains. It gives no indication of what the average latency for an event is, i.e. it does not take into account the time that an event may spend waiting in a buffer at the outstation. However, it can be used as a starting point to determine this latency. The thought process is best illustrated by a hypothetical example.

Let events be represented by an infinite number of children wishing to travel by bus to school. Assume that the road to school is narrow, and only one bus can be on the road at a time, or else a collision will occur, which will increase the time spent in the system. Busses can either be on the road or stopped at a bus stop. A bus represents a data message, that contains children (events) and uses the road (channel) to get the children from the bus stop (outstation) to the school (master station). Children arrive at the bus stops following a Poisson distribution. As soon as the first child climbs aboard the bus, the bus tries to get onto the road. It must be kept in mind that the road may be busy because of busses from other bus stops. If this is the case, the bus must wait (back off) until the road is clear. If any more children arrive at the bus stop while the bus is waiting for the road to clear, they may also climb aboard the bus. Once the bus leaves the children continue to arrive at the same mean rate, so that when the bus gets back from the school, it is possible that there are several children waiting at the bus stop. Although children may arrive at school in batches, the average arrival rate at the school will be constant over a long period.

The purposes of the example is to illustrate that each child is subjected to two queuing systems, one at the bus stop, and one in the bus network. The output of the first queue is the input to the second, thus the queues are in tandem. The first queue has an infinite source population. The bus network is characterized by the finite source system. Jackson has shown that a complex network can be represented by decomposing it into smaller networks. Thus, a combination of a finite number of independent queues with a Poisson stream will produce another Poisson stream with a mean equal to the sum of all the individual streams. Little's result is applicable to the overall network, as well as each individual network. For tandem queues, the overall time in the system is equal to the sum of of the time in each sub system. Jackson assumes that there is at least one path out of the network, i.e. a customer will eventually leave the system with a probability of 1 (Ng, 1996).

The finite source model provides an indication of the mean time that an event will spend in the second network, i.e. once it has been transmitted. The mean time that an event will spend in the first queue, i.e. in the buffer waiting to be sent, is a function of the time that the message spends in the first queue, as the events in the buffer can only be

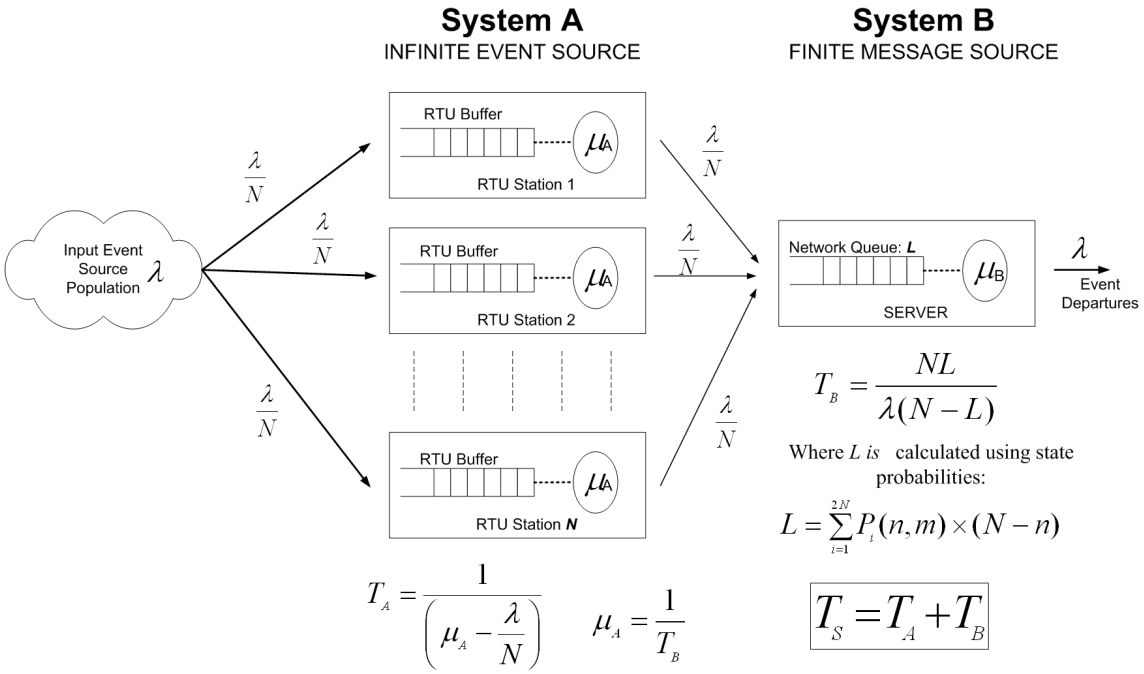


Figure 7.4: Overall event queueing system.

handled once the message in the second queue has been handled. It is proposed that the entire second system be represented as the servicing facility of the first queue. Stated differently, the first system has a mean service time equal to the mean time a message will spend in the second queue.

$$T_1 = \frac{1}{\mu_1 - \lambda}, \text{ where } \mu_1 = \frac{1}{T_2} \tag{7.6.1}$$

To motivate, the bus example is revisited. The time that a child has to wait at the bus stop is a function of the time that the bus is away from the bus stop. The mean time that the bus is away from the bus stop indicates the average rate at which the bus returns to the bus stop to fetch more children. This is simply the service rate of the first queue. The mean arrival rate at the bus stop is constant, as children keep arriving whether the bus is there or not. Note that because of the memoryless property of the Poisson process, no regard need be given to a child's arrival relative to when the last bus left. All arrivals are subjected to the same average time. Assume the average time from when a bus leaves a bus stop until when it returns is 10 minutes and that the bus follows a Poisson process. If child arrives at the bus stop 5 minutes after the bus left, the average amount of time until the bus returns will still be 10 minutes, not 5, due to the memoryless property of the process (Ng, 1996). The total time in the system will be the sum of the time in the first system and the time in the second system.

7.7 State Transition Matrix

A state transition matrix will be used to model the finite section of the network. The same procedure is followed as in Wolhuter (2002). It must be kept in mind that the finite source model is dealing with *messages*, and therefore only predicts the time that a *message* will spend in the system, and does not consider the time that an *event* may spend in a buffer waiting to be sent.

The message arrival rate, i.e. the rate at which messages are sent, at each station is constituted of new arrivals, and arrivals from backoff. As stated previously, retries due to collisions or channel noise have the same effect as increasing the average service time of the system. This has been validated by simulation (Wolhuter, 2002).

The finite source characteristic of a system allows for the definition of a finite set of system states. The system changes between these states with probabilities proportional to the service time and instantaneous arrival rates. The sum of all the state probabilities is one, and the system can only move to adjacent states. A state transition diagram is provided for the case of four outstations and one master station, as this configuration system was available for testing.

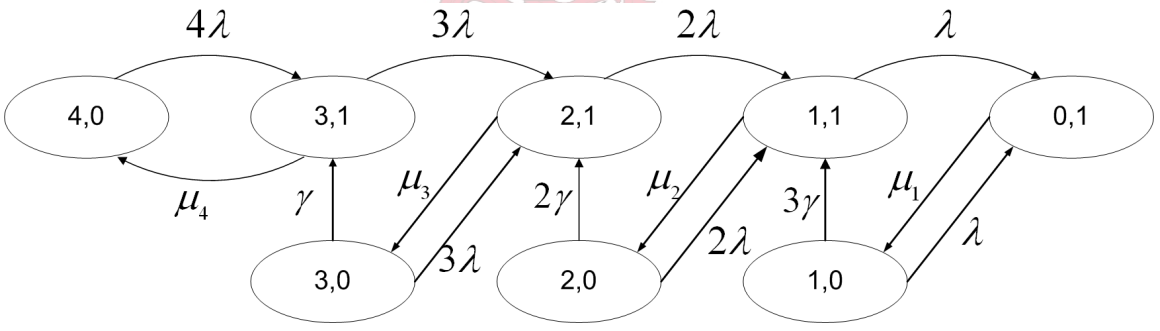


Figure 7.5: Finite source network state probability diagram, N=4.

The probability for each state is represented by $P(n, m)$ where: n represents the number of outstations that do not have a message to forward to the master station, and m represents the state of the server (channel) 1 for occupied, and 0 for idle. N represents the number of outstations in the system, λ is the message arrival rate, and μ is the service time. γ is the rate at which messages come out of back off and also follows a Poisson PDF. Note that in this case, λ represents the arrival rate *at each station*, not of the entire network. The arrival rate of the entire network is given by the sum of the arrival rate at each station. It is assumed that the mean arrival rate at each station is the same.

$$\lambda_{Network} = N\lambda_{Station} \tag{7.7.1}$$

If the system is in the first state $P(4,0)$, this means that none of the four outstations have a message send to the master station, and thus the channel is free (therefore the server is free). The system will move out of this state at a rate equal to the total arrival rate of the system (4λ) to the state $P(3,1)$. In this state one station has an event, and therefore the server is occupied. The system can either revert its original state $P(4,0)$ with rate equal to the effective service time, or another station can have an event occur, in which case the system will move to state $P(2,1)$. This will occur at rate 3λ, as only 3 of the 4 stations are available to generate a message. If the server cannot immediately handle a station's message because it is occupied, the outstation will enter backoff state, and upon completion of the service, the system will be in state $P(3,0)$, where the server is free, the first station's message has been dealt with, and the second station that still has a message for the server is in backoff. The system will move from this state, either when the backoff period expires at rate γ, or when one of the other stations generates a message at rate 3λ. It is possible that all stations can be in backoff, while the server is occupied, i.e. state $P(0,1)$. As soon as the server handles the message it is busy with, there will be one station available for events again, i.e. $P(1,0)$. Therefore it is impossible for the state $P(0,0)$ to occur.

The service time of the system is simply the average cycle time of a message. The actual software processing time of the message is negligible as the message transmission time is much longer than the processing time. The method for calculating the cycle time is exactly the same for as for the RRP case as detailed in Section 7.4, and includes hardware rise times. In this case, poll and response messages are replaced with DAT and ACK messages. The minimum cycle time, without regard to retries is given by

$$t_{cycle-min} = t_{preamble} + t_{postamble} + t_b \times (b_{DAT} + b_{ACK}) \tag{7.7.2}$$

As before, errors due to noise, given by a fraction δ_{noise}, will result in a delay waiting for the retry timeout to expire, plus another entire message cycle.

$$t_{cycle-noise} = t_{cycle-min} + \delta_{noise} \times t_{er} \tag{7.7.3}$$

$$t_{er} = t_{cycle-min} + t_{retry} \tag{7.7.4}$$

Note that in this case, t_{er} is chosen as the minimum cycle time, plus some mean delay t_{retry} to allow for the non deterministic nature of the contention network. One parameter that the RRP case does not include is overhead due to collisions. This is simply because collisions simply do not occur because of the deterministic strategy. With CSMA however, collisions most definitely occur, and the problem is aggravated because of the rise times of the radios.

Consider that is takes approximately 100ms from when one radio is keyed, until a nearby receiving radio detects the channel is busy, simply due to the hardware delays of the radio. If the radios are far apart, the propagation delay can increase this detection time. If station A begins to transmit, and any other station senses the channel up to 100ms after A began transmission, such a station will assume the channel is open, and could also commence transmission. Thus, there is a 100ms "collision possible" period on every transmission, which can be increased due to propagation delays (τ). Obviously the probability of a collision occurring is a function of how busy the channel is, i.e. the event arrival rate, and the rate at which events come out of backoff, and the hardware rise time. A collision will also constitute a retry cycle.

The fraction that the channel is free is given by $(1-\rho)$, where ρ is the utilization defined in (7.5.1). The propagation delay for the cycle is given by 2τ where $\tau = d/c$. γ represents the backoff return rate. The rise time is given by t_r, and there are two rise times in question in each cycle. Thus a fraction, σ_{coll}, can be produced using these parameters to increase the effective service time. Note that this factor is not constant, but changes in accordance with the instantaneous arrival rate, $n\lambda$, to produce a realistic result, Thus, it is a function of n, where n represents the number of stations that do not have a message for the server.

$$t_{cycle-coll} = t_{cycle-min} + \sigma_{coll} \times t_{er} \tag{7.7.5}$$

$$\sigma_{coll} = n\lambda 2t_r\gamma(1-\rho) + 2\tau \tag{7.7.6}$$

$$T_{s-eff} = \frac{1}{\mu_n} = t_{cycle-min} + (\sigma_{coll} + \delta_{noise}) \times t_{er} \tag{7.7.7}$$

Balance equations can be set up for the state transition diagram, and these can then be written in matrix form, and the probabilities of each state can be calculated using MATLAB.

$$\begin{aligned}
4\lambda P(4,0) - \mu_4 P(3,1) &= 0 \\
-4\lambda P(4,0) + [3\lambda + \mu_4]P(3,1) - \gamma P(3,0) &= 0 \\
-3\lambda P(3,1) + [2\lambda + \mu_3]P(2,1) - 3\lambda P(3,0) - 2\gamma P(2,0) &= 0 \\
-2\lambda P(2,1) + [\lambda + \mu_2]P(1,1) - 2\lambda P(2,0) - 3\gamma P(1,0) &= 0 \\
-\lambda P(1,1) + \mu_1 P(0,1) - \lambda P(1,0) &= 0 \\
-\mu_3 P(2,1) + [3\lambda + \gamma]P(3,0) &= 0 \\
-\mu_2 P(1,1) + [2\lambda + 2\gamma]P(2,0) &= 0 \\
-\mu_1 P(0,1) + [\lambda + 3\gamma]P(1,0) &= 0 \\
P(4,0) + P(3,1) + P(2,1) + P(1,1) + P(0,1) + P(3,0) + P(2,0) + P(1,0) &= 1
\end{aligned}$$

$$\begin{bmatrix} 4\lambda & -\mu_4 & 0 & 0 & 0 & 0 & 0 & 0 \\ -4\lambda & 3\lambda+\mu_4 & 0 & 0 & 0 & -\gamma & 0 & 0 \\ 0 & -3\lambda & 2\lambda+\mu_4 & 0 & 0 & -3\lambda & -2\gamma & 0 \\ 0 & 0 & -2\lambda & \lambda+\mu_2 & 0 & 0 & -2\lambda & -3\gamma \\ 0 & 0 & 0 & -\lambda & \mu_1 & 0 & 0 & -\lambda \\ 0 & 0 & -\mu_4 & 0 & 0 & 3\lambda+\gamma & 0 & 0 \\ 0 & 0 & 0 & -\mu_2 & 0 & 0 & 2\lambda+2\gamma & 0 \\ 0 & 0 & 0 & 0 & -\mu_1 & 0 & 0 & \lambda+3\gamma \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} P(4,0) \\ P(3,1) \\ P(2,1) \\ P(1,1) \\ P(0,1) \\ P(3,0) \\ P(2,0) \\ P(1,0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

The system queue length is calculated by the sum of each state probability multiplied by the number of stations that have events in that state.

$$L = \sum_{i=1}^{8} P_i(n,m) \times (4-n) \tag{7.7.8}$$

The average time spent in the system is given by Little's Therom. It must be remembered that instantaneous arrival rate is not constant, but rather a function of the queue length, as given by (7.5.5). Using (7.5.3),(7.5.5) and (7.7.1) the average time spent in the finite source system is:

$$T_B = \frac{L}{\lambda_{Station}(N-L)} \tag{7.7.9}$$

$$T = \frac{N}{\lambda_{Inst}}$$

$$\lambda_{Inst} = \lambda_{Network}\left(\frac{N-L}{L}\right)$$

$$\lambda_{Network} = N\lambda_{Station}$$

To extend this result to the infinite source system, this mean time spent in system B is considered as the service rate of the system A, as shown in Figure 7.4.

$$T_A = \frac{1}{\mu_A - \lambda_{Station}} \tag{7.7.10}$$

where

$$\mu_A = \frac{1}{T_B} \tag{7.7.11}$$

Note that $\lambda_{Station}$ is used, as the service delay μ_A is common to each station. The time that an event will spend in the entire system is simply a sum of both delays.

$$T_S = T_A + T_B \tag{7.7.12}$$

Thus the model can predict the mean time that it will take from when an input changes at an outstation, until the event has been acknowledged by the server.

The model was implemented in MATLAB to solve the system latency for various event arrival rates. It supports all system overhead parameters, including channel noise, retry time out, backoff times and radio rise times. The results of the model for various configurations are provided in Figures 7.9 through 7.13.

7.7.1 CSMA Extension to Multihop Systems

In the multihop case, each station that is required to act as a digipeater forwards messages immediately (when the channel is determined to be free), irrespective of whether the station has an event message of their own pending in the server queue. In other words, there is no significant queueing delay at each digipeater in a route. Modeling the a CSMA multihop network is complex because of collisions due to "hidden terminal" effects and pipelining delay. Referring to Figure 7.1, it is clear that the station farthest from the master station (D) in terms of hops will have the highest latency and the lowest traffic load, as channel activity depends mainly on its own events. The station nearest the master station (A) will have the lowest latency, but the highest traffic load, as traffic to and from all the outstations is routed through it. Thus, collisions will be more prominent nearer the master station, where the traffic load is higher. The mean system latency can be modeled on the approach used for the RRP multihop case, where the mean latency of the system is related to the mean route length of the system. The mean multihop delay can be incorporated into the state matrix model by increasing the effective service time by the mean route length of the system. Thus

$$T_{s-eff2} = \frac{1}{\mu_{eff2-n}} = h_{mean} \times [t_{cycle-min} + (\sigma_{coll} + \delta_{noise}) \times t_{er}] \tag{7.7.13}$$

where h_{mean} is the mean number of hops per route in the system.

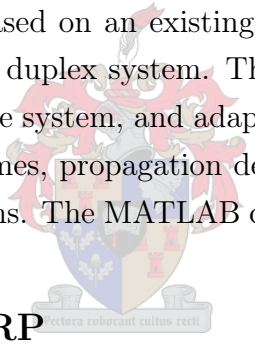
The hidden terminal effect presents a far more severe case, as the period in which a collision can occur is extended to the entire transmission sequence, not just the rise time. It is proposed that the mean service time be further increased to incorporate the mean hidden terminal effect by replacing the rise time (t_r) with a factor related to the transmission time in the calculation of σ_{coll}. Note that this is simply a proposal, and provides an estimation

the network performance for high event rates. The actual performance would depend on the exact network topology, and number of hidden terminals in the system. The model was found to agree with initial testing, although only four prototype outstations were available.

Results of the extended model are shown for a selection of network configurations, as depicted in Figure 7.15. The arrows in the figure indicate the only links in the system, i.e. terminals without arrows linking them are hidden from each other.

7.8 Conclusion

This chapter has provided a theoretical performance prediction of the system for RRP and CSMA modes of operation. Routing messages were ignored in the modeling process, as stated at the outset, due to the low mobility of the network. Results have been presented graphically for various scenarios and network topologies. The RRP modeling process was straightforward as the nature of the strategy prevents collisions from occurring. The CSMA modeling process was based on an existing state transition matrix approach for a finite source narrow band half duplex system. The model was extended to predict the latency of an infinite event source system, and adapted to cater for multi-hop paths. The model includes hardware rise times, propagation delays, and noise, and makes provision for hidden terminals and collisions. The MATLAB code of the CSMA model is appended.



7.8.1 Comments on RRP

The results obtained from the RRP results were as expected, because of the deterministic nature of the strategy. It follows that the system performance responds linearly to a variation in parameters like noise, data length and the mean route length. Note that the mean cycle time per station does not depend on number of stations in the system. The total mean time for the system is the simply the cycle time given in Figures 7.6 through 7.8 multiplied by the number of stations in the system.

7.8.2 Comments on CSMA

In an environment where there is a shared (communications) medium, the system performance typically responds exponentially in accordance with the event rate (Ng, 1996). The model results demonstrate this exponential response. The model offers a reasonable prediction of system performance as it responds to variations in parameters that would affect actual performance. Other than noise, message length and rise times, another such

parameter is the backoff time. It should be pointed out that varying the backoff time affects the wait time, as shown in Figure 7.12. The variation is more pronounced at higher event rates, as too long a backoff period would increase the wait time unnecessarily, while too short a backoff period would increase competition for the channel, therefore increasing collisions, and the wait time. Thus the model offers a solution whereby an optimal backoff period can be selected. For the system under consideration, it was found to be around the minimum cycle time.

Figure7.14 indicates instability for a high event rate, where the mean route length (MRL) of the system is 2.5. This is valid, because at this point, the mean service time has exceeded the mean inter-arrival time, and the queue grows without bound.

Note that the model includes incorporates the retry time as part of the mean service time. In effect this means that the mean latency is proportional to the retry time, as shown in Figure 7.13. While valid for longer retry times, in reality, if the retry time is too short, competition for the channel would be increased, resulting in an increased latency. This is perhaps an area were the model could be further refined.

7.8.3 Comparison of Strategies

For systems with a low overall event rate, regardless of the number of stations in the system, CSMA is the obvious choice, as the channel is only utilized when required. On the other hand, the deterministic nature of RRP offers a positive advantage of stability. It is interesting to note the substantial delay that multi-hopping introduces. This is more pronounced in the CSMA case, because of collisions caused by hidden terminals. In networks with a high event rate, extensive multi-hopping could present an unacceptable latency. In such networks, it is recommended that the mean route length of the system be limited. The choice of strategy will depend on the type of application for which the system is used, and the system has been designed to cater for both modes of operation simultaneously.

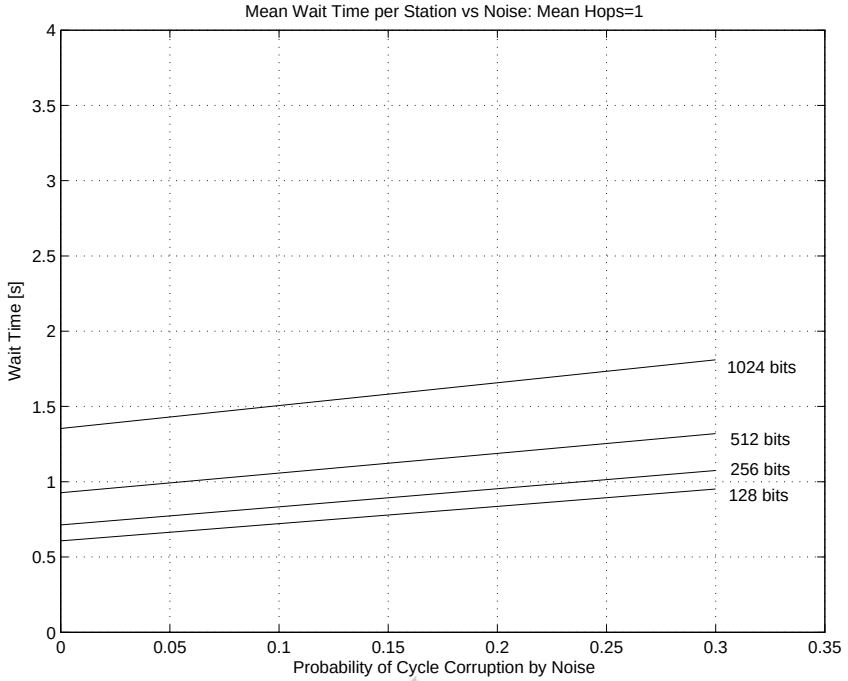


Figure 7.6: RRP mean cycle time for various noise levels, mean hops=1.

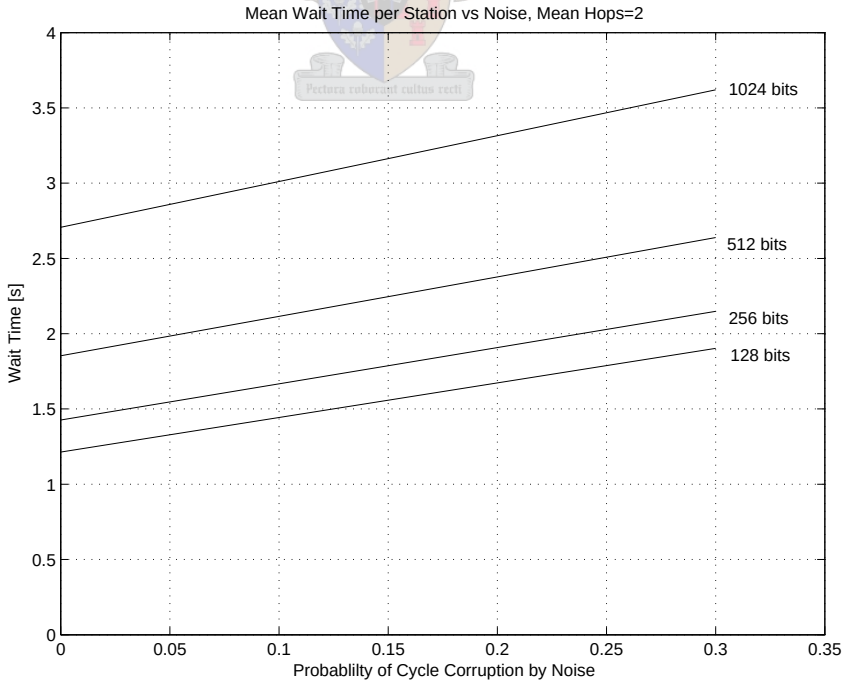


Figure 7.7: RRP mean cycle time for various noise levels, mean hops=2.

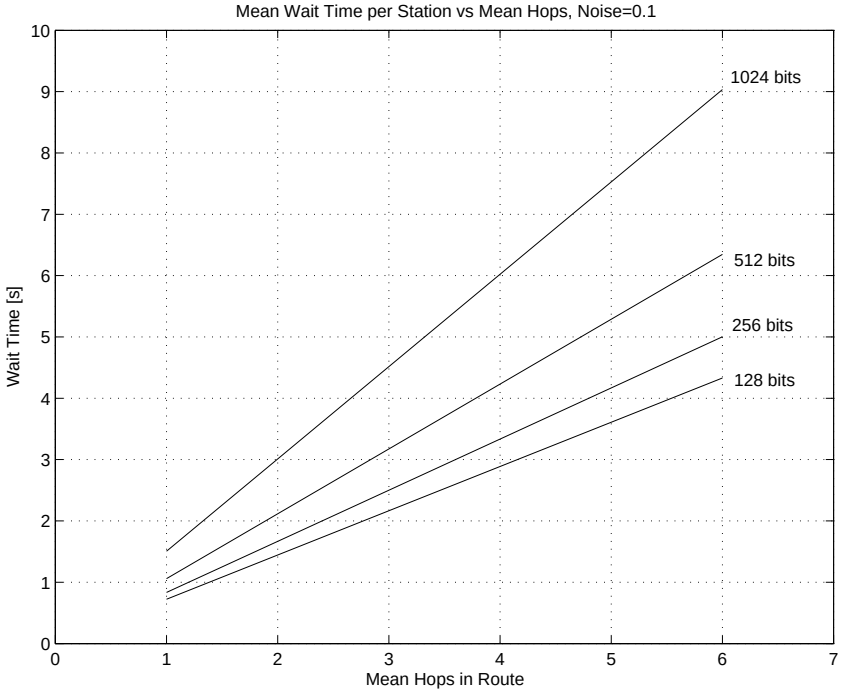


Figure 7.8: RRP mean cycle time for various mean route lengths.

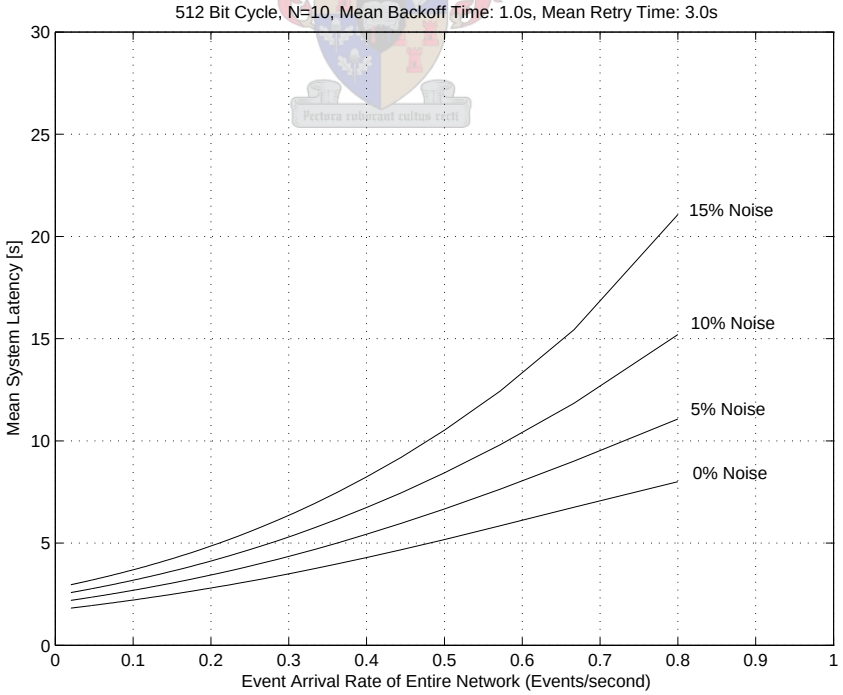


Figure 7.9: CSMA Theoretical Mean Latency for various noise levels (mean hops=1, no hidden terminals).

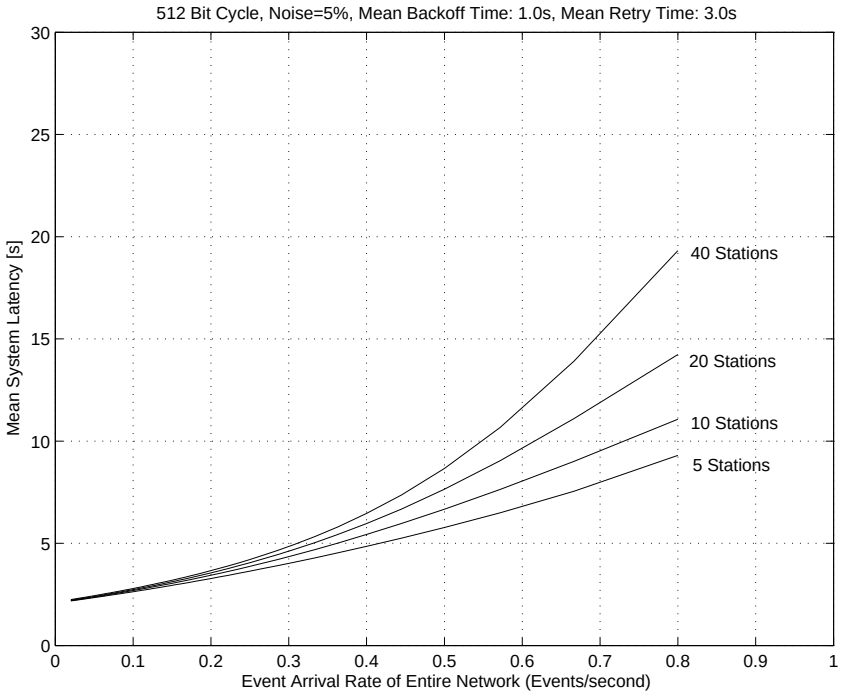


Figure 7.10: CSMA Theoretical Mean Latency for various noise levels (mean hops=1, no hidden terminals).

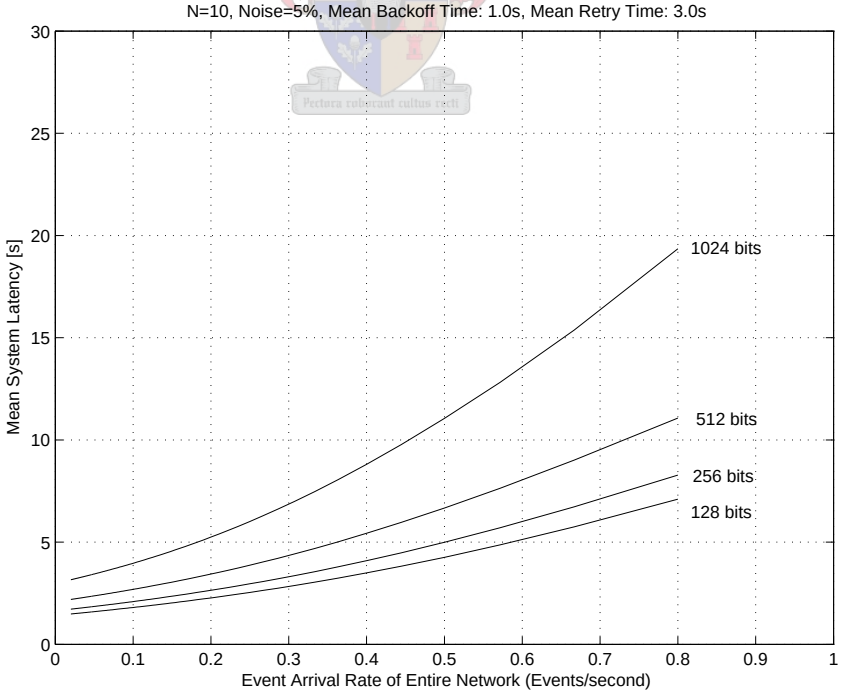


Figure 7.11: CSMA Theoretical Mean Latency for various bit cycle lengths (mean hops=1, no hidden terminals).

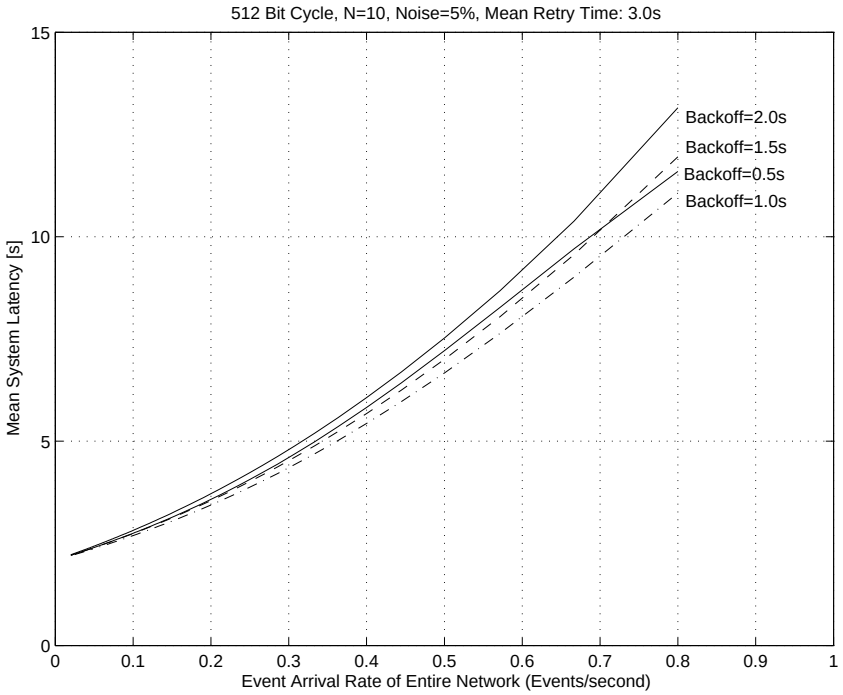


Figure 7.12: CSMA Theoretical Mean Latency for various mean backoff periods (mean hops=1, no hidden terminals).

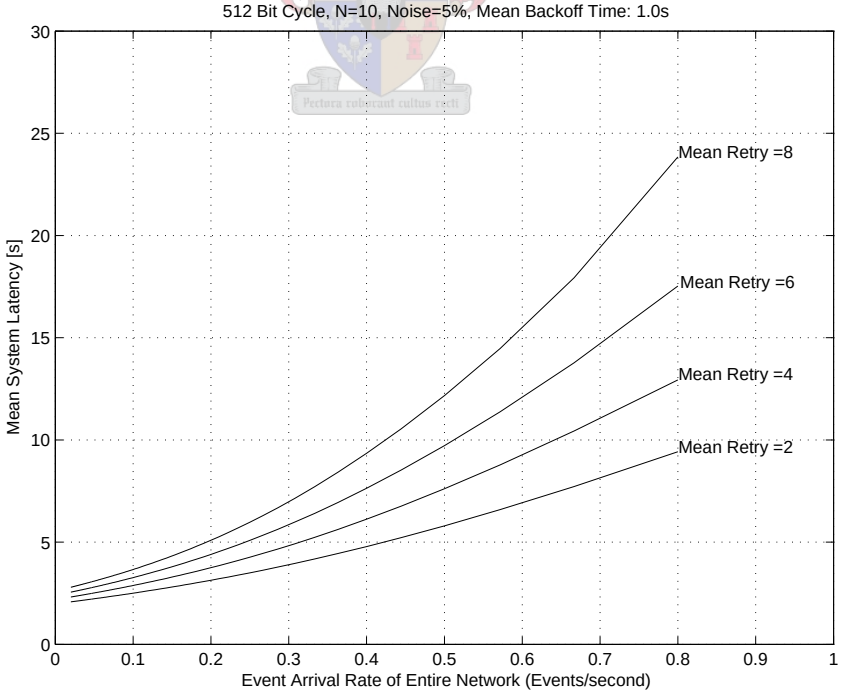


Figure 7.13: CSMA Theoretical Mean Latency for various mean retry periods (mean hops=1, no hidden terminals).

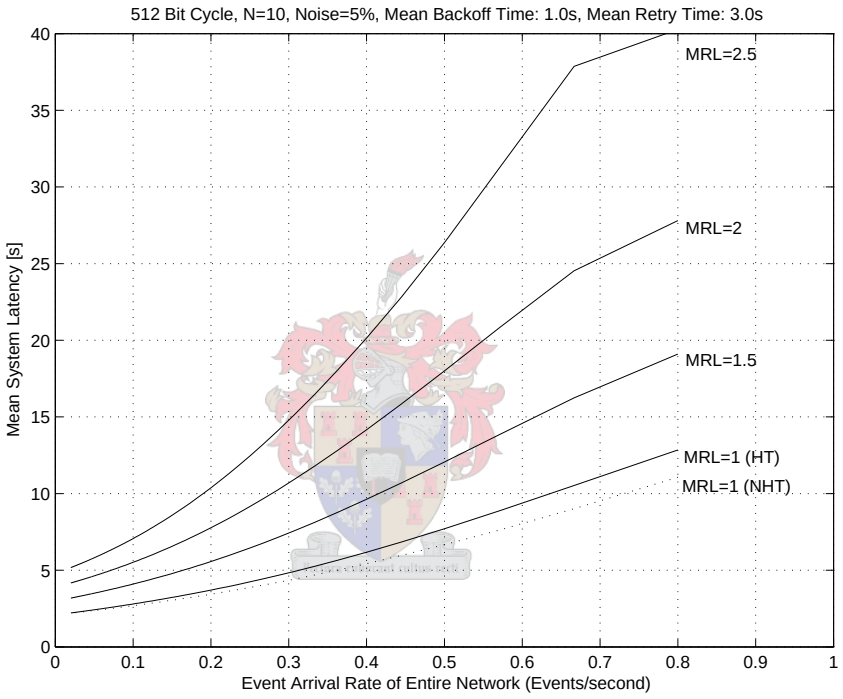


Figure 7.14: CSMA Theoretical Mean Latency for various mean route lengths (mean hops), N=10.

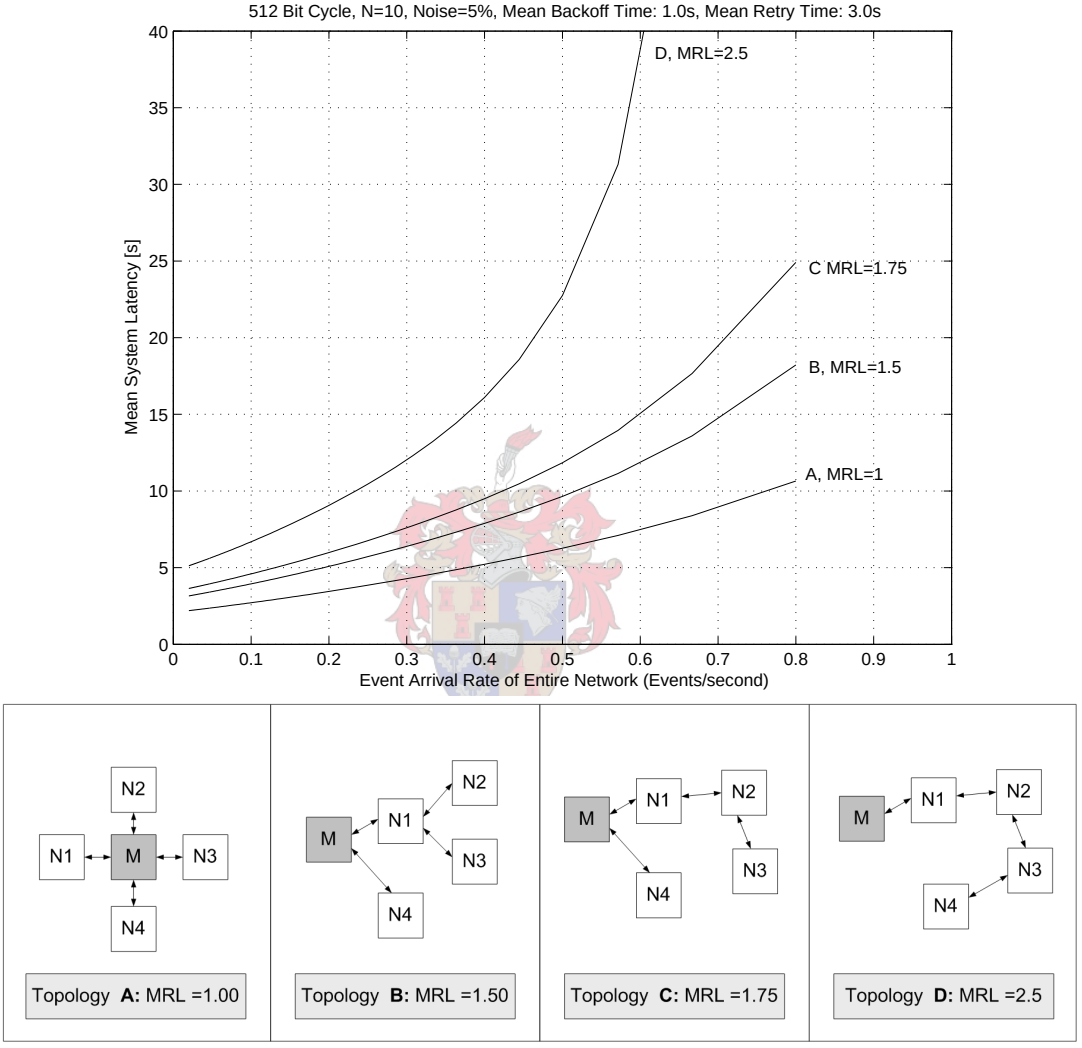


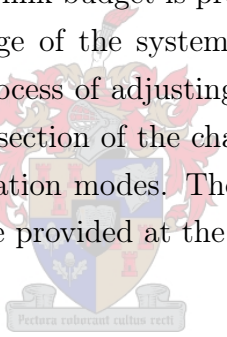
Figure 7.15: CSMA Theoretical Mean Latency for various mean route lengths (mean hops), N=4.

Chapter 8

Measurements and Results

8.1 Introduction

The first section of this chapter deals with the with the results and measurements of the communications infrastructure. A link budget is provided, which is a detailed calculation that predicts the operational range of the system. The measured performance of the radio and modem follows. The process of adjusting the audio input level for the correct deviation is detailed. The second section of the chapter deals with the measured system latency for RRP and CSMA operation modes. The chapter concludes with a discussion of results. All graphical results are provided at the end of the chapter.



8.2 Link Budget

After an appropriate communications infrastructure was selected, a theoretical link budget was compiled to predict the operational range of the system. An estimation of the SNR at the input to the receiving modem was required. All stations use the same hardware, i.e radios, and antennas at each station are identical. This simplifies the process somewhat.

The theoretical prediction applies to each link in a multi-hop route. This is because each routing node will receive the message, process it and then retransmit it, so the SNR at the output of the digipeating node will be identical to the SNR at the output of the source. This is not the case with conventional repeaters, where the noise figure of the repeater is cascaded through to the destination node.

When referring to digital signals, the E_b/n_0 parameter is commonly used. However, when referring to analogue signals, such as ones that voice grade radios are designed to carry, E_b is replaced with the carrier power, S. E_b is used for baseband signals, or the modulating signal before carrier modulation. These can best be described in the current project as the audio tones that modem produces and passes to the radio. S is used to describe the

pass band signals, i.e. the power of the carrier modulated with the audio tones. As shown in Chapter 2, E_b and S are related by the bit rate,

$$S = E_b \times R_b \tag{8.2.1}$$

The carrier power per hertz (unit bandwidth) is given by

$$\frac{S}{n_o} = \frac{E_b}{n_o} R_b \tag{8.2.2}$$

It follows that carrier power in the entire bandwidth is given by

$$\frac{S}{N} = \frac{E_b}{n_o} \frac{R_b}{B} \tag{8.2.3}$$

The radio can be characterized by its noise figure F.

$$F = \left(\frac{S_i/N_i}{S_o/N_o} \right) \geq 1 \tag{8.2.4}$$

Here, S_i and N_i are the input power levels, i.e. at the antenna, and S_o and N_o are the output power levels, i.e the audio output of the radio, or input to the modem. By definition, the input noise power is assumed to be the noise power from a matched load (antenna), at T_0=290K, that is

$$N_i = kT_0B_f \tag{8.2.5}$$

B_f is the IF filter bandwidth (7-8kHz in a 12.5kHz channel (Wood, 2004*a*)) in which the noise is available, and k is Boltzmann's constant. Once F_{radio}, S_i and N_i have been calculated, the SNR at the input to the modem can be determined by rewriting (8.2.4) as

$$\text{SNR}_{modem} = 10 \log_{10} \left(\frac{1}{F_{radio}} \frac{S_i}{N_i} \right) \tag{8.2.6}$$

$$\text{SNR}_{modem} = -F_{radio}(dB) + S_i(dB) - N_i(dB) \tag{8.2.7}$$

The E_b/n_0 ratio of the modem itself can be calculated by rewriting (8.2.3) as

$$E_b/n_0(dB) = \text{SNR}_{modem} + 10 \log_{10} \left(\frac{B}{R_b} \right) \tag{8.2.8}$$

where B is the input filter bandwidth of the modem, 4.8kHz. This bandwidth is justified in Section 8.4.

8.2.1 Determining Input Noise

N_i is simply calculated from (8.2.5), where $k = 1.38e^{-23}$, T_0=290K B_f=8kHz,

$$N_i(dB) = -164.95\ dB$$

8.2.2 Determining the Radio Noise Figure

In order to calculate the noise figure, the equivalent noise temperature of the radio required calculation first. The equivalent noise temperature can be calculated using the receiver sensitivity, which is specified for a certain SINAD in the radio datasheet. SINAD is a measure of the quality of an audio signal that comes out of the radio receiver. It is defined at the total audio power (signal+noise+distortion) over the undesired audio power (noise+distortion), and is expressed in decibels.

$$\text{SINAD} = 10\log_{10}\left(\frac{S_o + N_o + D_o}{N_o + D_o}\right) \tag{8.2.9}$$

Generally, SINAD is measured in the following manner:

1. A carrier signal modulated with a 1kHz sinusoidal audio tone is applied directly to the antenna input of the radio under test.
2. The total power in the audio output from the radio is measured $(S_o + N_o + D_o)$
3. The signal in passed through a notch filter which removes the 1kHz tone, leaving only noise and distortion, which is then measured $(N_o + D_o)$
4. Using (8.2.9) the SINAD is calculated from the two power measurements
5. The power of the input signal is varied until the desired SINAD is reached. The input power at this level defines the sensitivity of the receiver, and is usually expressed in μV_{rms}.

This process can be automated using a Communications Test set, as described in Section 8.3.1. A SINAD of either 12dB or 20dB is generally used, as this range is accepted to be the threshold for reasonable intelligibility of voice. The lower the input power to the receiver required to achieve this SINAD, the better the receiver. The radio datasheet states that the radio receiver has a sensitivity $0.32\mu V$ at 12dB SINAD for 12.5kHz channel operation. The minimum detectable signal power (MDS) is related to to the sensitivity by the following equation:

$$V_{i_{min}} = \sqrt{2Z_0 S_{i_{min}}} \tag{8.2.10}$$

where $V_{i_{min}}$ is the sensitivity in V_{rms}, and Z_0 is the antenna impedance (Pozar, 2001), in this case 50Ω. If follows that for the given sensitivity, the minimum signal power is

$$S_{i_{min}}(dB) = 10\log_{10}\left(\frac{V_{i_{min}}^2}{2Z_0}\right) = 10\log_{10}\left(\frac{(0.32e^{-6})^2}{100}\right) = -149.90 \text{ dB}$$

or

$$S_{i_{min}}(dB) = 119.90 \text{ dBm}$$

The SNR available to the discriminator of the receiver is known as pre-detection SNR, or SNR_{in}. It is a ratio of S_i to noise that the radio adds itself, N_{radio}. Depending on the quality of narrow band receiver this value typically ranges from 8dB to 12dB to produce a SINAD of 12dB (Miller, 1988). Although SNR_{in} and SINAD are related, their relationship is not linear (Bensky, 2000).

The noise power of the radio is given by:

$$N_{radio} = kT_{radio}B_f \tag{8.2.11}$$

By manipulating (8.2.11), the equivalent noise temperature of the radio can be calculated as

$$T_{radio} = \frac{N_{radio}}{kB_f} \times \frac{S_i}{S_i} = \frac{S_i}{kB_f} \div \frac{S_i}{N_{radio}} \tag{8.2.12}$$

Rewriting (8.2.12) in dB:

$$T_{radio}(dBK) = S_{i_{min}}(dB) - 10\log_{10}(k)10\log_{10}(B) - \text{SNR}_{in} \tag{8.2.13}$$

$$= -149.90 - (-189.57) - 8 = 31.67 \text{ dBK}$$

The equivalent noise temperature of the radio is thus

$$T_{radio} = 1469 \text{ K}$$

It follows that the noise figure of the radio is given by:

$$F_{radio}(dB) = 10\log_{10}\left(\frac{T_{radio}}{T_{ambeient}} + 1\right) = 10\log_{10}\left(\frac{1469}{290} + 1\right) = 7.83 \text{ dB} \tag{8.2.14}$$

Using this result, and the result for N_i in Section 8.2.1, the SNR_{modem} can be calculated from (8.2.7) for $S_{i_{min}}$ as

$$\text{SNR}_{modem}\ (for\ S_{i_{min}}) = -7.83 + (-149.90) - (-164.95) = 7.22\ dB$$

This illustrates the non-linear relationship between SNR and SINAD.

8.2.3 Determining Received Signal Power

At the transmitter, the output SNR is assumed to be excellent.

$$\mathrm{SNR}_t = 10\log_{10}\left(\frac{S_{TX}}{N_{TX}}\right) \geq 40dB \tag{8.2.15}$$

Although present, N_{TX} is very small compared to the signal power. Both signal and noise are subjected to the same losses in the transmission channel and thus N_{TX} is negligible.

The received signal power S_i is related to the transmitted signal S_{TX} by two equations. A comparison of both is provided.

1. **Friis Equation:** The Friis equation accounts for free space loss, antenna gains, and loss in equipment. Equipment loss includes cable loss, connector loss and antenna impedance mismatch and misalignment. However it does not account for loss from diffraction or multipath effects, and therefore is only suitable for line of sight links.

$$S_i = S_{TX}G_{TX}G_{RX}\left(\frac{\lambda}{4\pi d}\right)^2 LF_{equip}\ \mathrm{W} \tag{8.2.16}$$

$$\lambda = \frac{0.95c}{f}\ \mathrm{m} \tag{8.2.17}$$

2. **Path Loss Equation:** A more realistic approach is to use the Path loss equation (Pozar, 2001). The Path loss equation is an estimation accounting for multipath effects, mainly from ground reflections. It generally applies to frequencies above VHF. h_1 and h_2 are the respective heights above ground of the transmit and receive antennas. d is the distance between the antennas. Typical values of n for various environments are given in Figure 8.1.

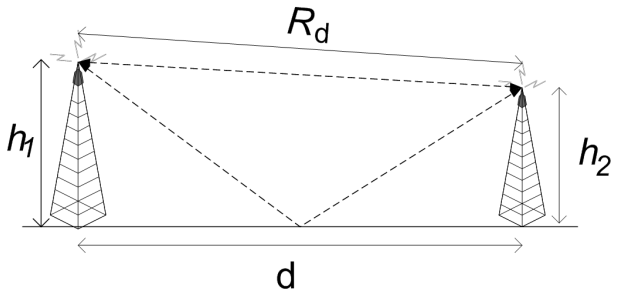
$$S_r = S_{TX}G_{TX}G_{RX}\left(\frac{h_1h_2}{R_d^n}\right)^2 LF_{equip}\ \mathrm{W},\ \text{where } R_d = \frac{(h_1-h_2)^2}{2d} + d \tag{8.2.18}$$

Values of variables in the above equations are provided for the current system:

S_{TX}=1W=30dBm

$G_{TX} = G_{RX}$=2.2dBi

$c = 2.998 \times 10^8\ m/s$



Environment	n
Free space	2
Urban	2.7-3.5
Shadowed Urban	3-5
Building LOS	1.6-1.8
Building Shadowed	4-6

Figure 8.1: Diagram relevant to the Path Loss Equation.

f=433MHz

d=variable distance in meters

$h1 = h2$=2m

LF_{equip}=0.535

The loss factor for the external equipment LF_{equip} is calculated as follows:

1. The antenna cable used is RG58 coaxial. The RG58 datasheet specifies the cable loss as approximately 22.9dB/100m. Assuming a total cable length (TX and RX) of no more than 4m, the maximum cable loss is 0.92dB.

2. An impedance mismatch between the antenna and radio will reduce the delivered power by a factor $(1 - |\Gamma|^2)$, where Γ is the reflection coefficient between the radio and antenna. The theoretical impedance of the type of antenna used, Predator 400, a half wave end fed antenna is approximately $Z_{ant} = 72\Omega$ (Pozar, 2001). The radio impedance is standard at $Z_{radio} = 50\Omega$.

$$\Gamma = \left| \frac{Z_{radio} - Z_{ant}}{Z_{radio} + Z_{antenna}} \right| = 0.180 \tag{8.2.19}$$

The loss, applies to both transmitter and receiver, so the total loss factor is calculated as

$$LF = (1 - |\Gamma_{TX}|^2)(1 - |\Gamma_{RX}|^2) = 0.936 \tag{8.2.20}$$

The loss is approximately 0.3dB.

The cable has the same impedance as the radio, so there is no mismatch between the two. If antenna feed cable has a length equal to an exact multiple of $\lambda/2$, it

will have no effect on the impedance. Thus, it would effectively be equivalent to connecting the antenna directly to the radio, with no feed cable.

3. Connections (connectors and adaptors) between the radio, cable and antenna also cause a significant loss due to signal reflections from mismatches (insertion loss). A loss of 0.25dB per connector can be assumed. There are three connections of each station, so a nominal total system loss of 1.5dB is assumed.

This results in an overall external equipment loss of 2.72dB.

$$LF_{equip}(dB) = (-0.92 - 1.5 - 0.30) = -2.72\ dB$$

This correlates to a loss factor LF_{equip} of 0.535, a reduction of almost half the linear signal power.

Figure 8.4 depicts the received signal power level in dBm versus the distance between the stations in kilometers. Figures 8.5 and 8.6 depict the SNR at the output of the receiving radio, versus the distance between the stations in kilometers. Figure 8.7 depicts the E_b/n_0 of the modem, versus the distance between the stations in kilometers.

8.3 Measured Performance of Kenwood TK-3160

8.3.1 Sensitivity and Noise Figure

In order to measure the actual performance of the radios, a HP8920 Communications Test Set was used. The test set provides an RF output that is connected to the antenna of the radio under test. The audio output of the radio is connected to the audio input on the test set. The test set calculates the SNR of the audio signal for various RF outputs. Three radios were measured, and the average performance was recorded.

Table 8.1: Measured SNR and SINAD performance of the Kenwood TK-3160

RF input [dBm]	SNR 1.2kHz [dB]	SNR 1.8kHz [dB]	SINAD 1kHz [dB]	Distortion %
−115	11.04	6.80	10.87	27.67
−110	15.53	14.93	15.03	17.33
−105	20.43	19.20	20.23	10.60
−100	25.00	23.20	23.27	7.03
−95	29.01	27.73	25.56	5.00

Table 8.2: Measured rise and fall times of the Kenwood TK-3160.

Parameter	Mean Time [ms]
TX PTT ON → RX SQL ON	100
TX PTT ON → RX DATA OK	150
TX PTT OFF → to RX SQL OFF	65

The average minimum receiving threshold of the radios was found to be -117dBm. From the measured results, it was clear that the radio requires an input signal of approximately -114dBm to produce a SINAD of 12dB at the audio output, somewhat greater than the -119.9dBm specified in the datasheet. From (8.2.4), this correlates to a radio noise figure of 13.16dB. This noise figure detracts marginally from the predicted operational range, however the range depends on many variables, and Figures 8.4 through 8.7 provide a good indication nonetheless.

8.3.2 TX/RX Rise and Fall Times

Radio rise and fall times were measured using an external PIC configured as a timer, with start and stop inputs. The start and stop inputs were connected to outputs from transmitting and receiving units. When the first station pulled the PTT of its radio low, a signal instructed the external PIC to "begin timing". Meanwhile the receiving unit waited for a SQL open signal from the radio, upon which, it issued a "stop timing" signal to the external PIC.

It was also found that data could only be reliably sent (received) once the PTT of the transmitting radio had been low for more than 150ms. These results provided input parameters for the theoretical modeling process of the previous chapter, namely, pre-amble (150ms), post-amble (65ms), and rise time (100ms).

8.3.3 Radio Deviation Adjustment

The level of the audio tones inserted into the radio has a direct influence on the carrier deviation. If the audio level is too high, the carrier will be over modulated, and will deviate more than the desired amount. Stated differently, the level of audio input to the radio, effects the modulation index, and therefore system bandwidth.

The radio interface circuitry as described in Chapter 4 provides a means of adjusting the level of the modem output by using a multi-turn resistor. The desired input level was found by using a RS Spectrum Analyzer to measure the deviation of a received signal. An antenna was connected to the RF input of the spectrum analyzer. An RTU board was programmed to transmit a bit stream, via its modem and radio. The Spectrum

Analyzer center frequency was selected as the carrier frequency of the radio. The received modulated carrier was then viewed on the Spectrum Analyzer.

As detailed in Chapter 2, The deviation is related to the modulation index by the following equation:

$$m = \frac{\triangle f}{f_m} \tag{8.3.1}$$

MSK has a modulation index of 0.5, and $f_m = R_b$ which is 1200. Thus the required deviation is $\triangle f$=600Hz. By selected a scan width and band width wide enough so that all significant sidebands are covered, the "max hold" and "min hold" functions of the Spectrum Analyzer are used to store the highest and lowest deviation levels to the screen (Agilent Technologies). The spacing between these levels is equal two twice the peak deviation, i.e. $2 \times \triangle f_{peak}$, or 1200Hz. This is shown in Figure 8.2. The measurement was performed for a range of audio input levels until the desired deviation was achieved. Screen shots from the Spectrum Analyzer are appended. An input level of around 20mV_{p-p} was found to produce the required deviation. Note that in this case, the sidebands fall away faster than in the case where the audio level is higher, indicating the bandwidth efficiency of MSK with a modulation index of 0.5. This level is consistent with the data sheet of the radio, which specifies an a level of 16mV_{p-p} on the microphone input.

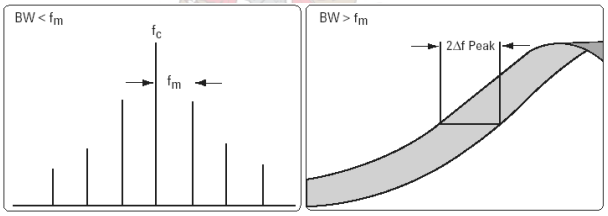


Figure 8.2: Extract from Agilent Application Note Agilent Technologies (2000).

8.4 FX469 Modem Testing

The datasheet of the FX469 specifies the modem to have a BER= 2.5×10^{-4} at a SNR of 12dB in the bit rate bandwidth, i.e E_b/n_0, where R_b=1200.

In order to test the performance of the modem, a test circuit was set up in the following manner:

1. Two modems were connected directly to each other (hard wired), and a noise signal added to the transmission channel. A predefined random bit stream was sent repeatedly from a computer to the first modem, across the channel to the second

modem, and back to the computer. The modulated signal had noise added to it in the transmission channel before demodulation at the receiving modem. The computer compared the received bit stream to the transmitted bit stream to calculate the errors that occurred due to noise in the transmission channel.

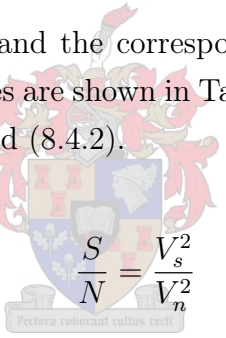
2. A rms noise generator was used to generate a Gaussian white noise signal.

3. Both transmitting modem signal and the noise signal were buffered using a op-amps in a voltage follower configuration to provide isolation between the sources.

4. The outputs from the buffering op-amps were then combined using a third op-amp in a mixer configuration.

5. The output signal level of the FX469 modem was measured to be 850mV$_{rms}$. The specified optimal input signal is 230mV$_{rms}$. With the noise source turned off, the mixer op-amp was adjusted to attenuate the signal to the desired level of 230mV$_{rms}$. The ratio of the signal to noise voltages remained the same before and after the mixer, as they were subjected to the same attenuation.

6. The noise level was varied, and the corresponding bit error ratios calculated and recorded. The recorded values are shown in Table 8.3, together with the SNR, which is calculated using (8.4.1) and (8.4.2).



$$\frac{S}{N} = \frac{V_s^2}{V_n^2} \tag{8.4.1}$$

$$\text{SNR}_{measured} = 20 \log_{10} \left(\frac{V_s}{V_n} \right) \tag{8.4.2}$$

The signal to noise ratio SNR$_{measured}$ has been calculated before input to the modem and therefore contains noise over a wide bandwidth. This is not the same signal to noise ratio that the modem detector will see. The modem input filter reduces the available noise power bandwidth to its own bandwidth, and this must be taken into account when calculating the modem performance. To do so, the SNR was converted to the E_b/n_0 ratio. Results were then comparable to the values as given in the FX469 datasheet.

$$\frac{E_b}{n_0} = \frac{S}{N} \frac{B}{R_b} \tag{8.4.3}$$

$$\frac{E_b}{n_0}(dB) = \text{SNR}_{measured} + 10 \log_{10} \left(\frac{B}{R_b} \right) \tag{8.4.4}$$

It was found that for a filter bandwidth of B=4800Hz, and R_b, the measured E_b/n_0 corresponded exactly with the performance specified in the modem datasheet. This is shown

Table 8.3: Measured performance of the FX469 modem.

V_n	SNR (dB)	E_b/n_o(dB)	BER
0.30	9.05	15.07	1.0×10^{-8}
0.35	7.71	13.73	2.9×10^{-6}
0.40	6.55	12.57	4.0×10^{-5}
0.45	5.52	11.54	1.4×10^{-4}
0.50	4.61	10.63	6.0×10^{-4}
0.60	2.67	8.70	6.2×10^{-3}
0.75	1.09	7.11	1.7×10^{-2}
1.00	-1.41	4.61	9.2×10^{-2}
1.50	-4.93	1.09	3.7×10^{-1}

in Figure 8.3, along with the theoretical BER plots for coherent and non coherent MSK. It it therefore safe to assume the modem has an input filter bandwidth of approximately 4.8kHz when the baud rate is 1200.

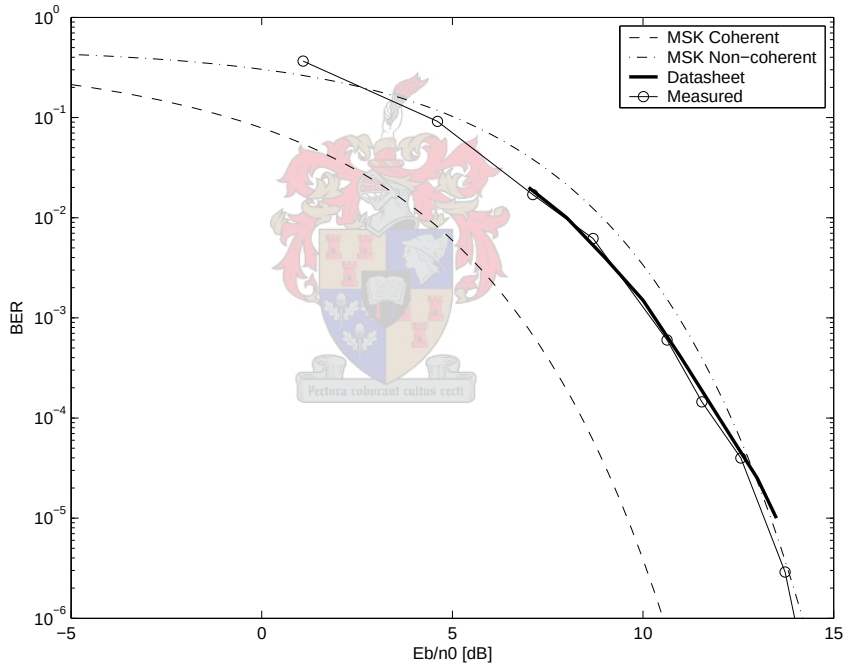


Figure 8.3: Measured FX469 modem BER performance at 1200 baud in a white noise channel.

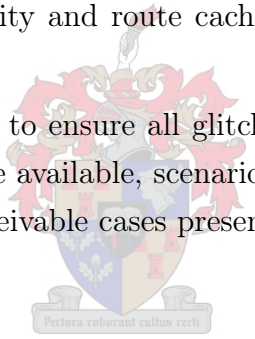
The same test was performed with the modems operating at 2400 baud. Measurements corresponded with the datasheet specification for 2400 baud operation, however the 2400 baud performance is far worse than 1200 baud, and was therefore not pursued.

8.5 Testing of the Routing Protocol

8.5.1 Description of the Testing Process

In order to test the auto-routing capabilities of the system, two types of antennas were used. The master station and three of the four outstations were fitted with dummy antennas, thereby limiting their range. The remaining outstation was fitted with a standard antenna. Initially, all stations were placed in the same proximity, i.e. all in range of each other, and the outstations were set to report events to the master station. One of the outstations with a dummy antenna was then removed so that it was out of range of the master station, but still in range of the outstation with the standard antenna. When this outstation detected that it no longer had a route to the source, it initiated a RRQ to the server, to which the station with the standard antenna replied. The station with the standard antenna thus acted as a digipeater between the outstation and the master station. When the outstation was moved back into range of the master station, it immediately detected a direct route to the station by overhearing the master station's transmissions. Monitoring of the channel activity and route caches of each RTU was provided by the server GUI.

Exhaustive testing was required to ensure all glitches were removed from the code. Although only five prototypes were available, scenario specific code was written to test the protocol thoroughly, for all conceivable cases presented by the system.



8.5.2 Results

Using the prototypes, the tests were repeated for various network configurations and scenarios, and all aspects of the protocol proved to operate as desired.

8.6 Measurement of System Latency

8.6.1 RRP Strategy

Due to the deterministic nature of Round Robin Polling, the strategy was very simple to test.

8.6.1.1 Description of Testing Process 1

One of the five stations was connected to the server, acting as the master station, while the other four stations were configured as slave stations. The master station polled each

slave station in turn. Upon initiation of a poll, the server station set a timer running. Upon reception of a valid response to the poll, the time was logged. To calculate the mean latency for all polls, the poll times were added together and divided by the number of poll cycles. The test was repeated for various message lengths, with FEC on and off, as well as for various network configurations where the system had to use multihop routes.

8.6.1.2 Results

After the initial routing overhead, where the master station found routes to all the stations, the system converged to the predicted theoretical latency. A selection of results along with the various mean route lengths due to network configurations is provided in Figure 8.8. It was found that the coding gain of the FEC could not be gauged using the complete prototypes, as there was no means to establish the SNR for each case. Thus, a similar test set up used to test the modems was used to measure the performance of the FEC.

8.6.1.3 Description of Testing Process 2

Two prototype modems, one master station and one outstation, were connected to directly to each other exactly as explained in Section 8.4, so that a known noise power could be added to the signal. For various noise levels, the master station attempted to poll the outstation a predefined number of times, and counted the number of valid responses. The probability of a cycle failing for each noise level could thus be measured. The master station kept track of the mean time it took to complete a successful cycle. This time included all system overhead (carrier rise/fall time, SQL open close/time retry timeout, start and stop bytes), as the prototypes were configured to operate exactly as if the radios were present.

8.6.1.4 Results

The test was run numerous times for various data payloads with FEC on and off. The efficiency, or effective data throughput, of the system was calculated by the data payload time over the mean cycle time, i.e. the ratio of useful data over the total time that it took to get the useful data. A selection of results is shown in Figures 8.9 through 8.12. The noise levels used in each case are those shown in Table 8.3. Included in each figure are results of a MATLAB simulation which was mentioned in an earlier chapter. The simulation included all system overhead parameters, and used the theoretical performance of both coherent and non coherent MSK to determine the error probability for various message lengths. In each case, the measured result falls in between the two theoretical results, which is consistent with the modem performance.

It is clear from the results that the FEC provides coding gain in areas of low SNR, but adds (marginally) to system overhead in areas of high SNR. The efficiency of the system is low, especially for the shorter data payload, because of the system overhead; particulary contributed to by the rise times of the hardware. The figures indicate an interesting trade off between efficiency and performance with regard to data payload length. Although the system is more efficient with a longer data payload, the longer messages are more susceptible to errors, which leads to cycle corruption, and retries. Assuming the minimum acceptable efficiency is two thirds of the maximum, in Figure 8.9 this efficiency occurs at E_b/n_0 ≈10dB. When FEC is used, the same efficiency occurs at Eb/n_0 ≈8dB, as depicted in Figure 8.10. Thus, the coding gain can be gauged as approximately $2dB$. The same is true for Figures 8.11 and 8.12.

8.6.2 CSMA Strategy

The CSMA strategy proved more difficult to test. Note that a system with a very low overall event rate was of no interest, as the latency is easily predicted since collisions are highly unlikely. Rather a heavily loaded system was considered. A range of mean overall (system) inter-arrival times from one to four seconds ($\lambda = 1 \rightarrow 0.25$) was chosen for testing. With Little's Result in mind, a system event inter-arrival time of one second was considered to be about the maximum that the system could handle before demonstrating instability, as this is close to the minimum cycle time (service time) of a point to point link when all overhead is included. For system stability, the time mean inter-arrival time must be greater than the mean service time, as explained in Chapter 7. The implemented testing process is described in the following text.

8.6.2.1 Description of the Testing Process

Code was written to allow the PIC to generate exponentially distributed interrupts at a specified rate. The rate at which the interrupts occur is the mean event frequency. The mean time between the events is simply the inverse of the frequency. The mean time between events is stored in the EEPROM of each PIC, and is adjustable remotely from the server. The code thus simulates Poisson arrivals with a mean inter-arrival time that is stored in memory.

Upon generation of an event, the station initiates a message to the server. The time at which an event occurs is logged on the RTU and tagged with the UID of the message. The RTU compares the time at which an ACK is received to the time at which the event occurred to calculate the latency. If further events occur while the RTU is waiting for an ACK for the previous event, the times at which these events occur are logged together. At the next opportunity, the RTU sends all the buffered events to the server as one message.

Thus, each additional event increases the message length accordingly. A single ACK from the server services all the events in the message, and the latency of each individual message can be calculated, as the time of each was logged. The mean latency is calculated by the sum of all the individual event latencies divided by the number of events that occurred. Each RTU keeps track of its mean latency, and includes it in each message to the server.

The server maintains a database containing the latencies at each RTU, which is updated with each message it receives. The mean system latency can be calculated from this database, by the sum of the latencies at each station divided by the number of stations.

As well as keeping track of its mean event latency, each RTU also maintains a count of the number transmissions, retries, cycle failures and invalid messages it has received. All this data is included in each message to the server, and is stored in the server database.

8.6.2.2 Results

Numerous tests were run for various event rates, retry timeouts, backoff periods and network topologies using the five prototype stations. Each test was run for a time sufficiently long enough to ensure event rates converged to the mean values. Collective information stored in the database was used to produce the results shown in the figures at the end of the chapter. The figures are plotted using the mean event inter-arrival times, as this was the parameter stored in the data base.

Figures 8.13 through 8.18 depict results for the case where all stations are in range of the master station, and of each other, for various retry times, and mean backoff periods. These results correspond with the model predictions. However, at high mean event rates (short mean inter-arrival times), the results do show some deviation from the model, as this is where events began to fail. Some messages reached the maximum retry count without receiving a valid response, thus the system reached instability. Only the latency of successful cycles were stored in the database, thus the recorded latency deviates from the model. Nevertheless, this result is valid as the model indicates a wait time striving to infinity. Included with each latency result are sub-plots indicating the measured probability that a message would have to be resent, the measured probability that a station would enter backoff, and the measured mean number of events included in each message. All of these decrease, as expected, in accordance with a decrease in the event rate, as the traffic density becomes less. It is interesting to note the high probability of retries at high event rates, due to congestion. Further, the probability of backoff is reduced noticeably in case were hidden terminals exist, as collisions become more likely.

Figures 8.19 through 8.20 indicate results for the hidden terminal and multi hop cases. The model provides a fair estimation of the expected latency for each case. Only four outstations were available for testing which hampered the ability to gather more conclusive

results for cases when more stations are involved in the network. This could provide an opportunity for future work. Such testing would also allow the model to be further refined.

8.7 Operation as a Telemetry System

After all testing was complete, the stations were configured for the intended use; a dynamic telemetry network, capable of multi-hopping and automatically discovering and learning routes to other stations. The system provides the full functionality of similar equipment in the form of CINs, DINs, AINs and DOTs. The server was found to operate as intended, mapping its database entries to the registers of outstations in the network. A simple application was written to demonstrate the system functionality by displaying the status of an outstation's I/O points, and allowing a user to change outputs and configuration settings remotely.

8.8 Conclusion

This chapter has provided a detailed link budget to predict the operational range of the system. The radio sensitivity was measured, from which a noise figure was calculated for comparison with the theoretical value. The measured radio performance was found to be marginally worse that the theoretical value, however not so much as to nullify the predictions of the link budget.

The BER performance of the FX469 modem was tested and found to correspond closely with the values specified in the data sheet. The results fall in between the theoretical performances of coherent and non-coherent MSK demodulation. Using the same test set up, and the RRP strategy, the FEC coding gain of the system was measured to be approximately $2dB$, and provided and acceptable efficiency at E_b/n_0 ≈9dB. This value can be used to determine the range, from the plots produced from the link budget, for various loss factors.

Ultimately, the operational range of the system will depend on the geographic location of the stations, but it is clear that the desired range of several kilometers is achievable in good conditions. This was validated up with a physical test in the Stellenbosch area, where a range of several kilometers was achieved.

The routing protocol protocol was tested and found to work successfully using the five prototype stations. The system adjusted accordingly to changes in the network topology.

The CSMA strategy was tested thoroughly and showed a good correspondence with the results of the model. As expected, the system became unstable (cycles began to fail) at high event rates especially when multi-hop configurations were tested, because of the

pipelining delay, and resulting collisions due to hidden terminals. Nevertheless, operation is found to be satisfactory for intended system applications, where a high event rate is not expected.

The testing of the protocol and CSMA strategies demonstrated the systems ability to send messages asynchronously, and operate promiscuously, learning routes from all received packages. Finally, the system was configured as an actual telemetry system, complete with a server and a user interface. All aspects of the system were found to function together as desired.

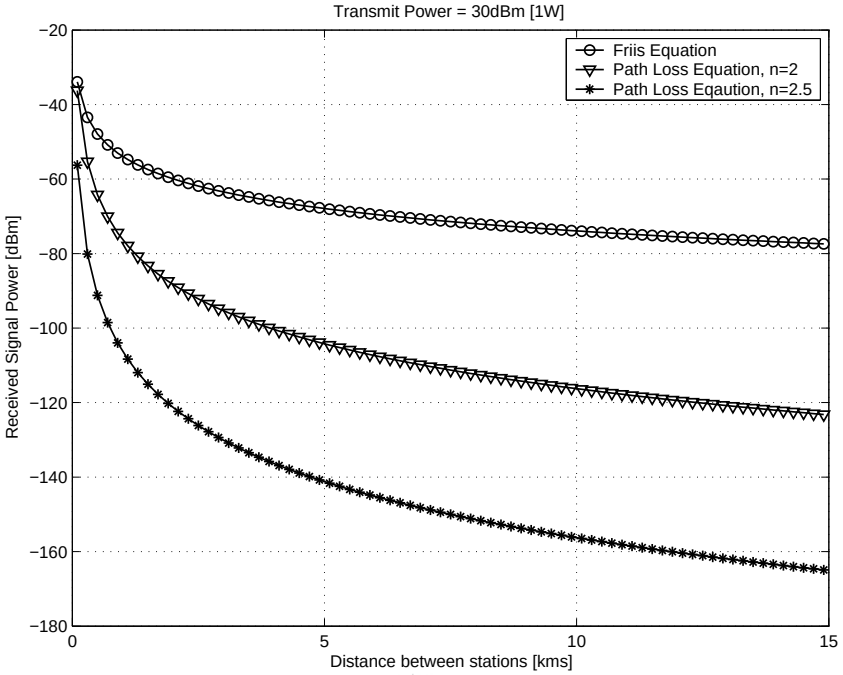


Figure 8.4: Received signal power input vs distance.

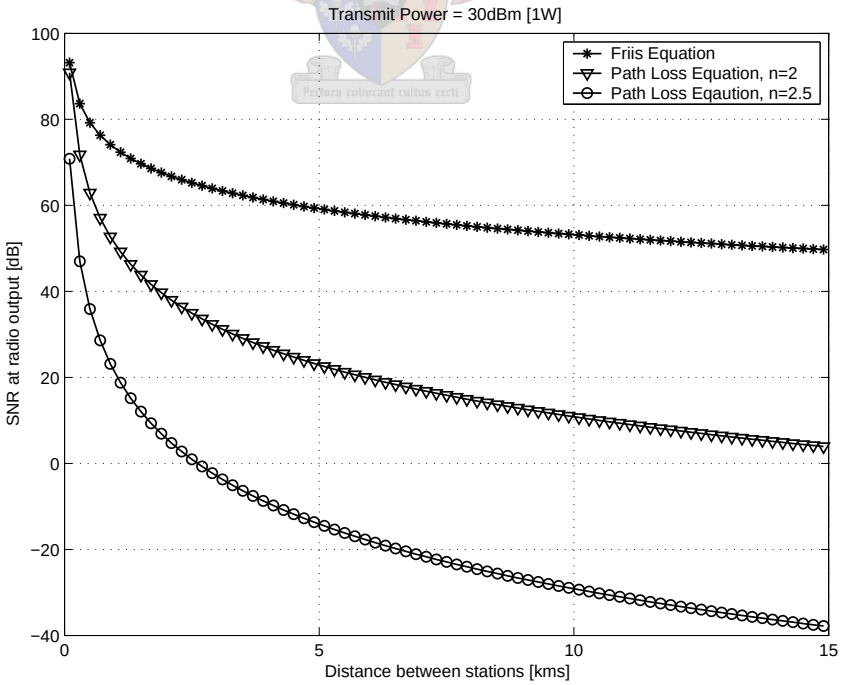


Figure 8.5: SNR at receiver modem input vs distance.

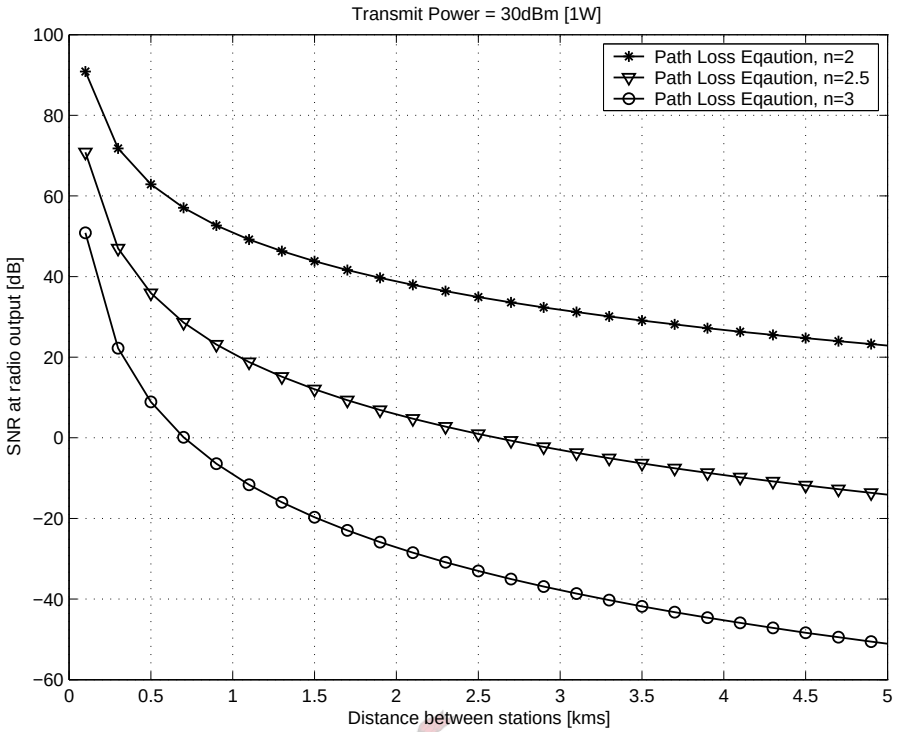


Figure 8.6: SNR at receiver modem input vs distance for various n.

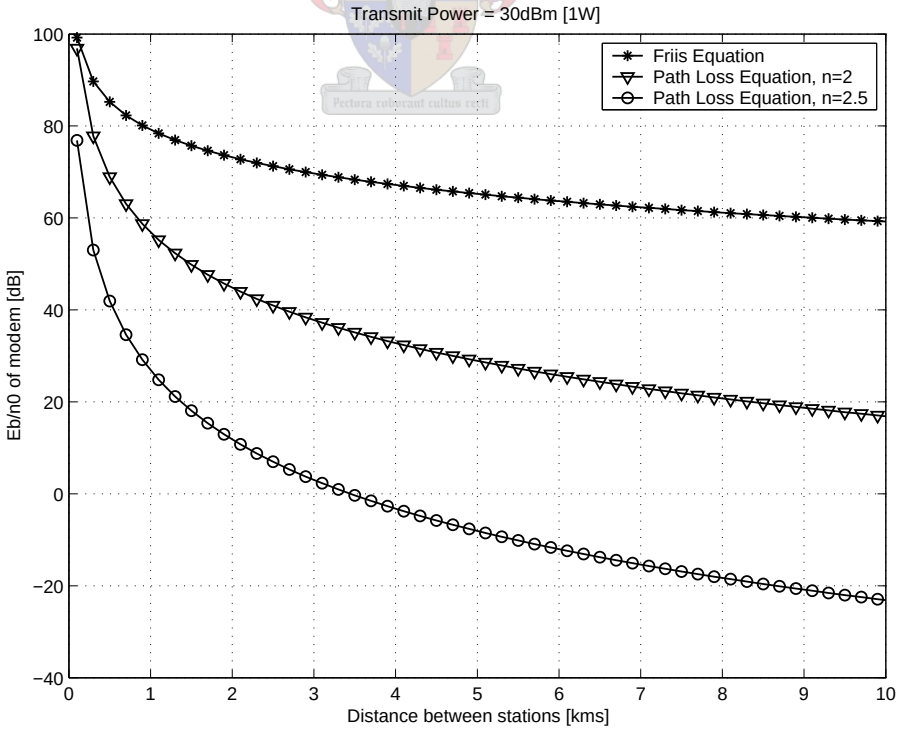


Figure 8.7: E_b/n_0 of modem vs distance between stations for various n.

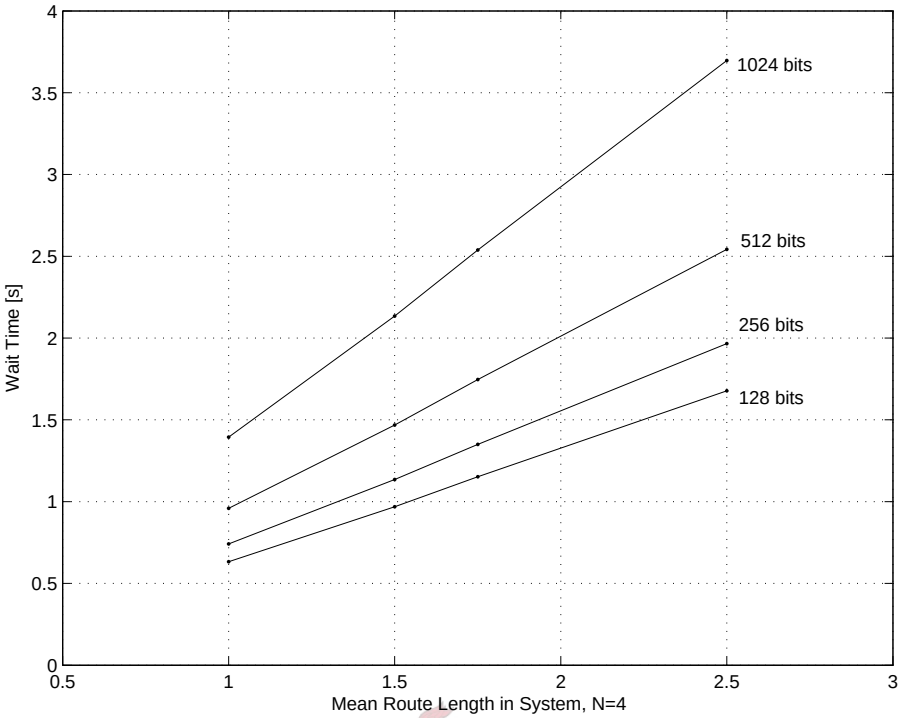


Figure 8.8: Measured Results: RRP for various network topologies and MSG lengths.

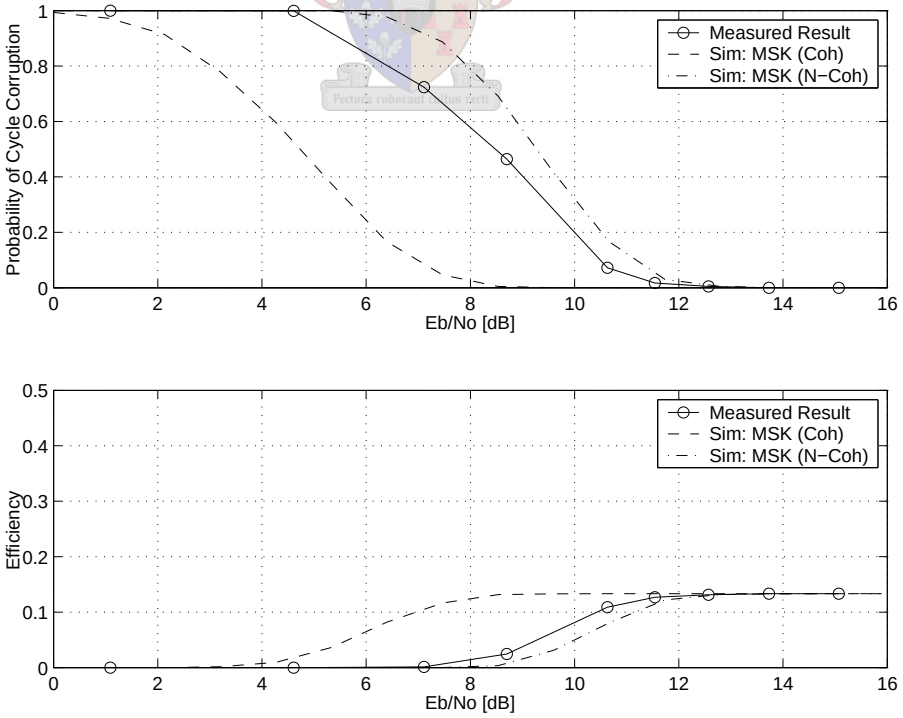


Figure 8.9: Measured Results: 16 data bytes without FEC.

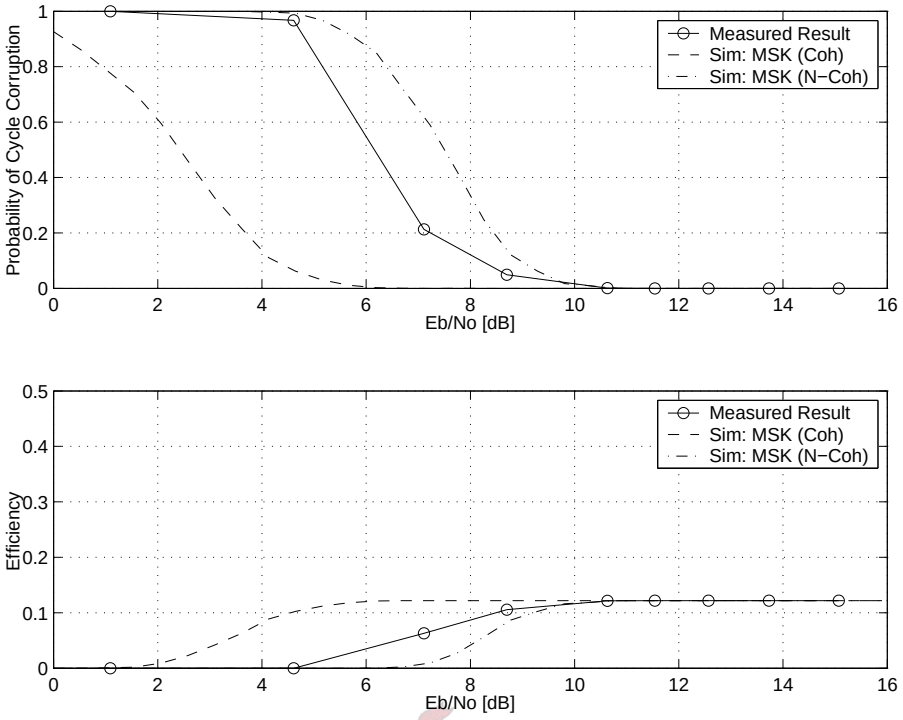


Figure 8.10: Measured Results: 16 data bytes with FEC.

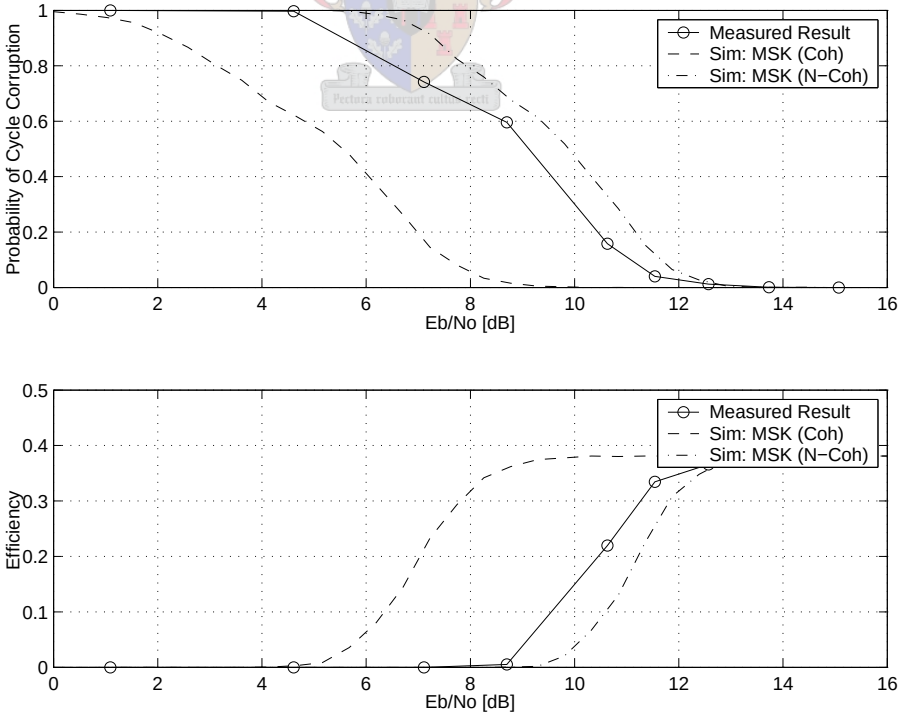


Figure 8.11: Measured Results: 64 data bytes without FEC.

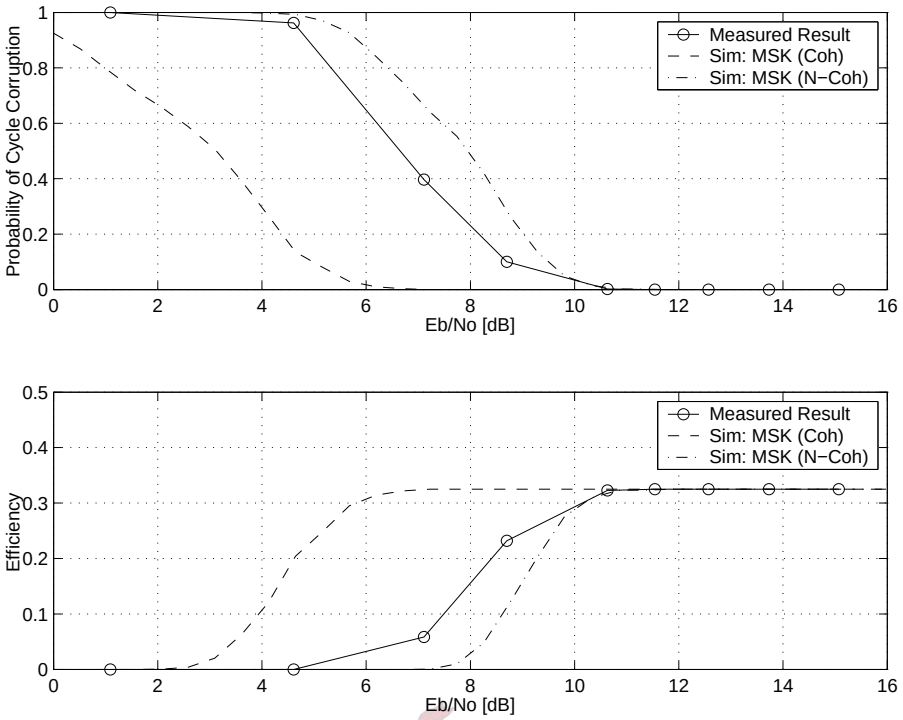


Figure 8.12: Measured Results: 64 data bytes with FEC.

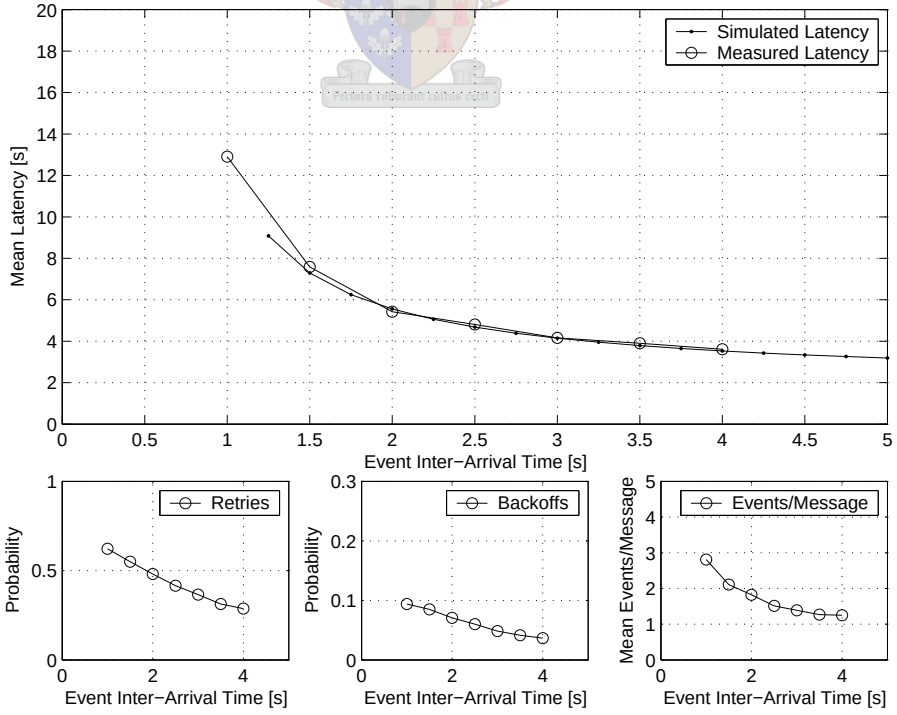


Figure 8.13: Measured Results: N=4, Retry Timeout=3.5s, Mean Backoff =0.8s.

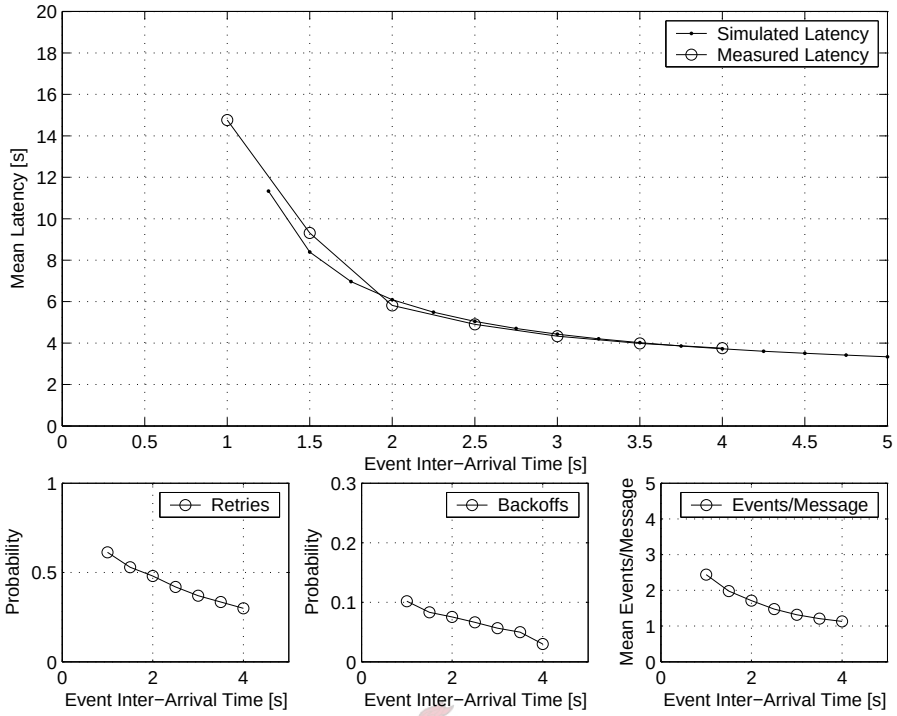


Figure 8.14: Measured Results: N=4, Retry Timeout=3.5s, Mean Backoff =2.0s.

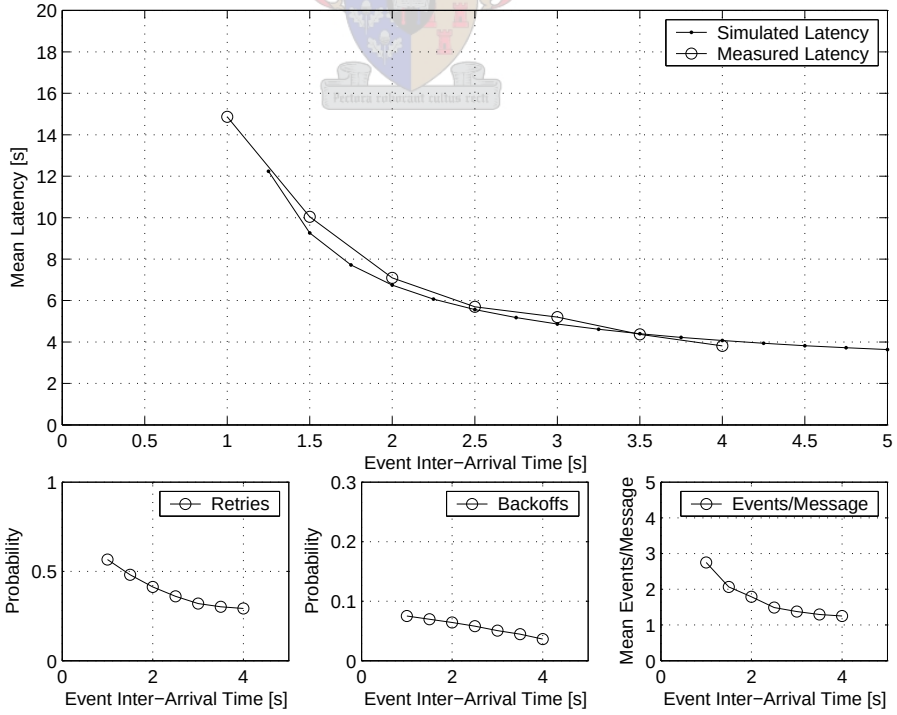


Figure 8.15: Measured Results: N=4, Retry Timeout=5.0s, Mean Backoff =0.8s.

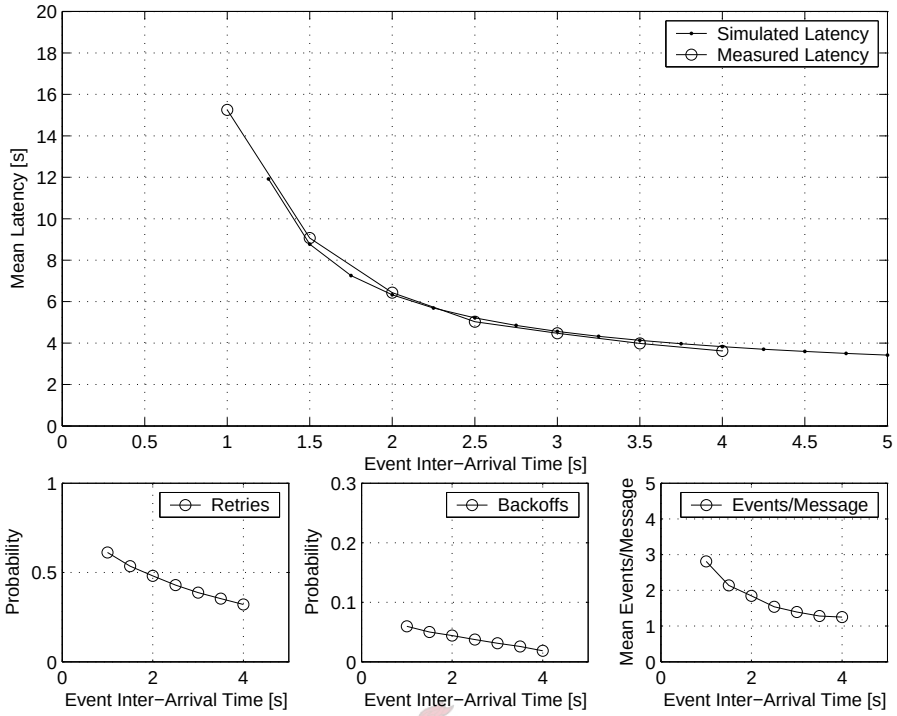


Figure 8.16: Measured Results: N=4, Retry Timeout=5.0s, Mean Backoff =1.5s.

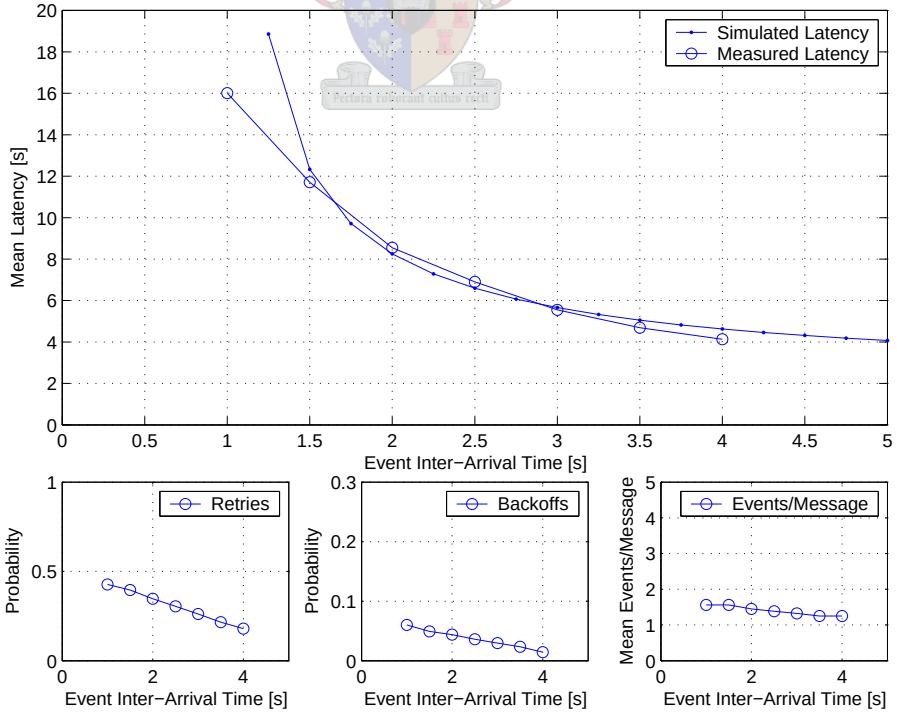


Figure 8.17: Measured Results: N=4, Retry Timeout=7.0s, Mean Backoff =0.8s.

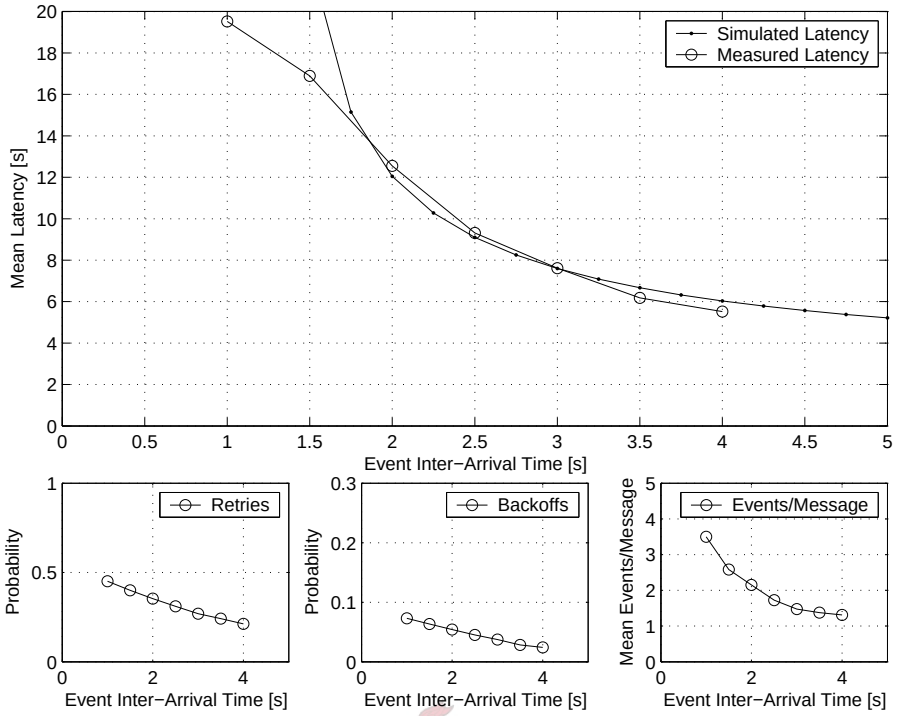


Figure 8.18: Measured Results: N=4, Retry Timeout=10.0s, Mean Backoff =0.8s.

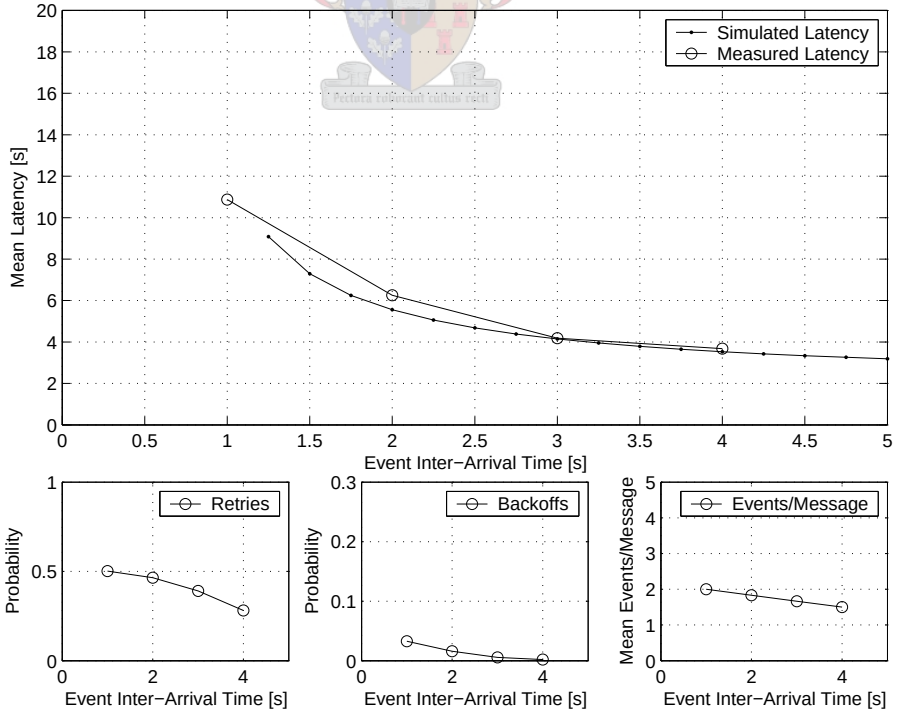


Figure 8.19: Measured Results: N=4, RT=3.5s, BO=0.8s MRL=1 with hidden terminals.

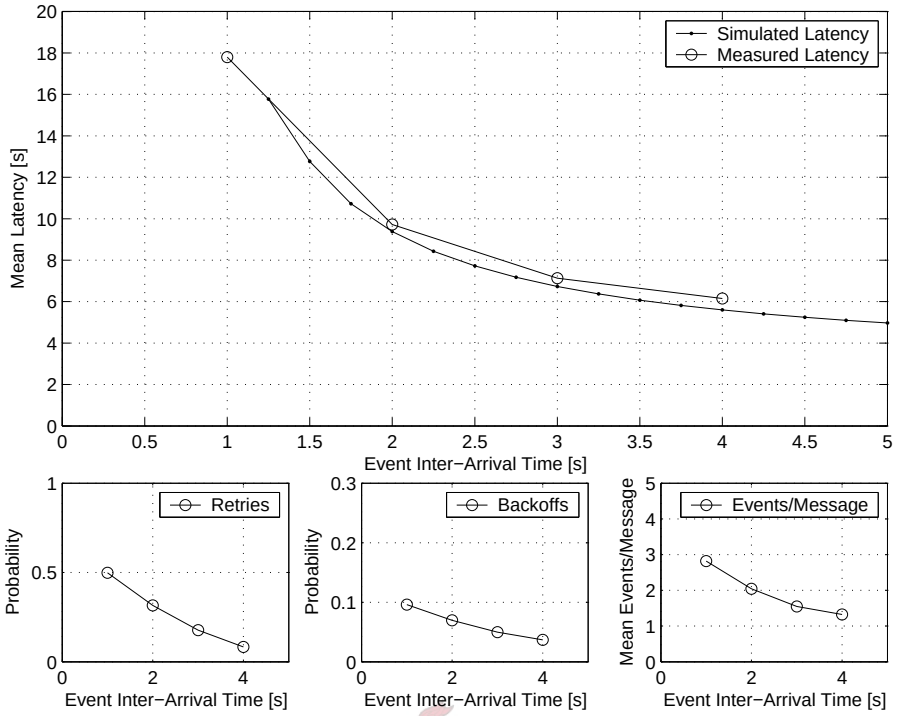


Figure 8.20: Measured Results: N=4,RT=3.5s,BO=0.8s MRL=1.5.

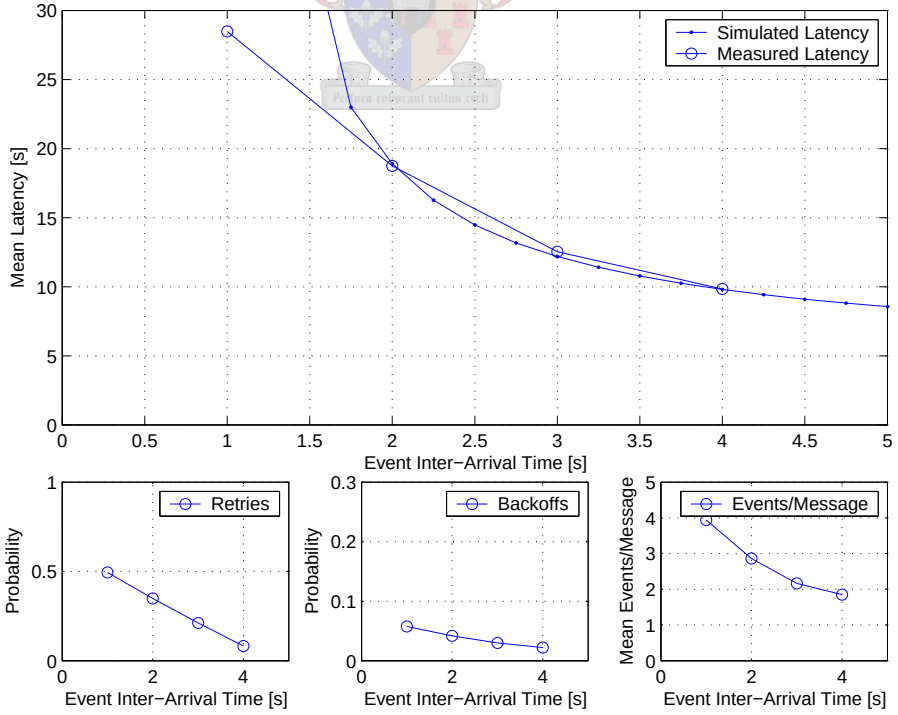


Figure 8.21: Measured Results: N=4, RT=3.5s, BO=0.8s MRL=2.5.

Chapter 9

Summary and Conclusions

9.1 Motivation

Although much of the focus of wireless networks lies on high speed networks, a definite need nevertheless exists for low-end telemetry type networks. Typically, these networks do not demand an extremely high data throughput, and often the superior propagation characteristics offered by lower frequency (narrow band) transceivers provide a far more viable option than their high speed counterparts. The primary objective of this project was to prototype a low cost telemetry system with automatic multi-hop and routing capabilities. The motivation is summarized in the following paragraph:

A unique opportunity was felt to exist in the design of dynamic telemetry system. The design would utilize multi-hopping and auto-routing techniques to extend the operational range of the telemetry network and make it suitable for rapid deployment and nomadic monitoring, thereby providing a reliable and cost effective replacement for stand alone data loggers. In addition to monitoring inputs, it would provide output control at remote locations.

9.2 Summary of Objectives

Subsequent to the motivation for the project, the following objectives were formalized:

1. At the outset, it was essential that typical telemetry systems and related hardware be investigated in order to determine the amount of data the system would be required to carry. Factors included the typical I/O count and circuitry, user interface characteristics, as well as the data collection strategies of such systems.

2. It was necessary to compare existing communications hardware and modulation techniques on the basis of cost, operational range, efficiency, data throughput and bandwidth, in order to select such hardware optimally.

3. Investigation of system optimizing forward error correction (FEC) strategies was considered important, as such a strategy would improve performance in areas of low SNR.

4. As the primary motivation for the project was a routing protocol implementation, an investigation of existing wireless routing protocols was considered vital.

5. A critical requirement was the selection of various hardware components that would enable support of the multi-hop auto routing protocol, while operation as a telemetry system was maintained. It was decided that a selection of industry standard I/O points be included in the prototype design, in order to demonstrate functionality.

6. Although not essential, it was considered that the design and development of a suitable battery backed up power supply and charger would contribute to the completeness of the prototype.

7. It was necessary to provide a user interface to the telemetry system, capable of maintaining a database of the I/O points of all monitoring stations, and allowing the user to control outputs and configuration settings.

8. It was felt that the performance of the telemetry network should be compared to theoretical results, in order to provide a platform on which to base predictions for larger networks and various scenarios.

9.3 Summary of Thesis

9.3.1 Initially, a background to telemetry networks in general was provided, followed by the formalization of project motivation and objectives. Some typical examples of where the system could be used were listed. The approach of following the layered OSI model was outlined.

9.3.2 The two lowest layers of the OSI model were handled first, namely the Physical and Data-Link layers. Development of the Physical layer entailed a comparison of available communication hardware, motivation for the selection of narrow band analogue radio, and a review of relevant modulation techniques and corresponding bit error probabilities. These techniques were compared on the basis of bandwidth efficiency and BER performance. Motivation for the selection of 1200 baud MSK was provided. Thereafter, the

Data-Link layer was considered, with the focus on error detection and correction strategies. A review of relevant schemes was provided, followed by a comparison on the basis of efficiency and error correcting capabilities. Motivation for implementation of CRC-16 and Hamming(15,11) strategies was provided.

9.3.3 Next, the Network Layer was presented, which focussed on the routing protocol. Initially, a review of some common protocols used for mobile ad hoc networking was provided. Dynamic Source Routing was found to be the most suitable protocol for the system under consideration, and the rest of the chapter was dedicated to describing the development and detailing the operation of the implemented protocol. The structure of the message header, which is common to all messages, was also explained in this chapter.

9.3.4 After the three lower layers were dealt with, the hardware design of the prototype units was presented. Design choices were made to enable the prototype to function as a fully operational telemetry system while supporting the routing protocol. All design calculations and motivations for component selection are included.

9.3.5 A complete description of the embedded software and its functionality was provided. The the flow of data, I/O control and the integration of all layers was explained. The Transport layer was introduced.

9.3.6 Subsequently, the Transport layer was expanded upon with a description of the server and its functionality. The concept of register mapping was explained. A basic user interface providing full remote control of all stations in the network was described. The server and user interface were considered the upper layers of the system.

9.3.7 At this point, all components and operation of the system in its entirety had been described, and a prediction of the system performance was provided. Initially, an indication of the routing setup times was provided for various cases. Thereafter two operation strategies were considered individually, namely RRP and CSMA. The RRP case was simple to predict because of its deterministic property, and theoretical latency values were provided for various network topologies and noise levels. CSMA, however, proved far more demanding. A theoretical model based on the state matrix approach of queuing theory was defined, following a documented approach for a finite source half duplex narrow band CSMA network. The finite source model was extended to provide for an infinite event source multi-hop system. The model allowed for incorporation of system overhead parameters, including hardware rise times and backoff periods, propagation delay, noise and retry times, in order to provide a realistic result that included the possibility of

collisions. The model produced an estimate of mean system latency vs mean system event rate, for an infinite event source. Variation in parameters that are known to effect actual systems (e.g. mean back off time, retry time) produced a corresponding effect on predicted performance.

9.3.8 All measurements and results were documented in Chapter 8. A link budget was provided, to predict the operational range of the system, using measured radio parameters. The BER performance of the FX469 was tested and found to correspond with data sheet specifications. The same test set up was used to gauge the performance of the implemented FEC, and found to be comparable with theoretical results. The process used to test the auto routing and multihop capabilities of the system was explained. The procedure for testing both RRP and CSMA strategies was described, and results were presented for comparison with the theoretical predictions. As expected, the RRP results showed an exact correspondence with the theoretical predictions. The CSMA results showed a satisfactory correspondence with the model results, especially for the case where all stations had a direct link to the master station. For the multihop case, the results still showed an acceptable correspondence. If was felt that the ability to refine the model to provide an accurate extrapolated prediction of larger multi-hop networks was somewhat limited because only five prototypes were available to produce results to compare to the model. Finally, an overview of the system as whole was provided.

9.4 Future Work and Recommendation

One of the interesting aspects of the work described in this thesis is the extension of an existing state transition matrix queuing model to encompass the type of network designed. Although acceptable tracking between model and implementation was obtained, it would be of interest to establish the limiting parameters, to which the model would still be accurate. Fine tuning the model would allow for additional hardware and network parameters, such as a wider range of protocol settings, which would also present an interesting research opportunity.

9.5 Conclusion

In closing, it can be said that the benefit of this project is two fold. Firstly, a fully functional prototype telemetry system, suitable for low-end use was developed. The uniqueness of the system, and the area on which the main emphasis lies, is that it is capable of multi-hopping, and auto-routing, thereby extending the potential range of the network, and making the system ideal for rapid deployment and nomadic monitoring. Each station

in the network acts both as a host, and a router, and is capable of sending multiple messages asynchronously. Each station operates promiscuously, learning information about the network topology from all received messages. The system operates using half duplex narrow band links at 1200 baud, and utilizes standard voice grade radios for data transmission. This keeps the cost of the system relatively low, and allows users to make use of both licensed of unlicensed frequency bands, as it depends solely on the radios used. The system employs FEC to optimize the system performance in areas of low SNR. Prototype testing indicated a point to point range of several kilometers was achievable using 1W UHF radios. The system is complete with an industry standard I/O interface, and each station is fully configurable remotely, with a GUI via a server. A power supply with back up battery and dual level battery charger is provided, complete with diagnostic outputs which allow the system to prevent battery damage by excessive draining. Each station can be configured to report to a master station at predefined times, report the occurrence of events, or simply wait to be polled by the master station.

Secondly, an existing model for finite source, half duplex narrow band CSMA type networks, which included system overhead parameters, was extended to provided a prediction of the mean latency for an multihop infinite events source. The model corresponded to measured results using five prototype stations, and is felt to provide a platform on which further theoretical analysis of low speed multi-hop networks could be based.

It should be kept in mind that, although hardware design played a big part and was necessary to support testing, the main emphasis of this thesis was the development of a multi-hop protocol. Further hardware modifications could be implemented without difficulty to provide failsafe data handling. These modifications could include an on-board clock for time-stamping purposes, and external memory to cater for data logging, should communication failures occur. Nevertheless, the prototype successfully demonstrates the routing protocol and meets the initial objectives. Such a protocol, implemented either in this, or similar hardware, is felt to have great potential in the telemetry field. Therefore, it is considered a success in every aspect.

Appendices

Appendix A

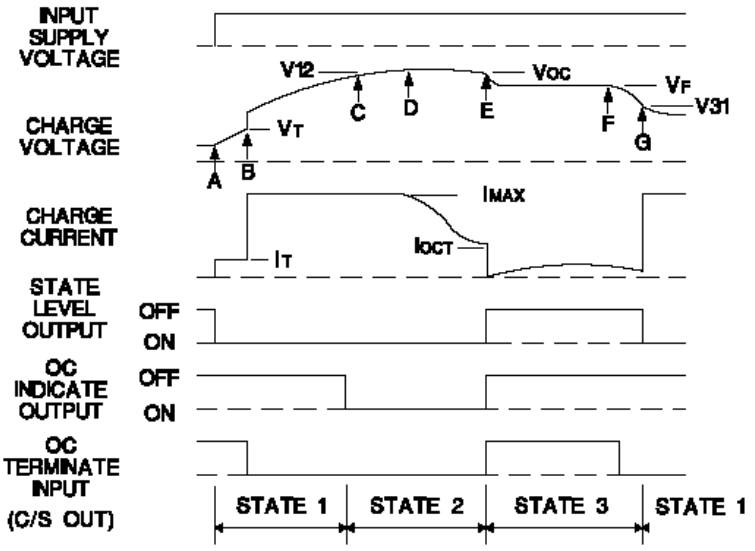
Additional Design Information

Table A.1: Voltage levels relevant to PIC18F452 I/O pins.

Voltage level	**Low**	**High**	**Unit**
TTL (input)	$0 \rightarrow 0.15V_{DD}$	$2 \rightarrow V_{DD}$	VDC
Schmitt Trigger (input)	$0 \rightarrow 0.2V_{DD}$	$0.8V_{DD} \rightarrow V_{DD}$	VDC
CMOS (output)	$0 \rightarrow 0.6$	$(V_{DD} - 0.7) \rightarrow V_{DD}$	VDC

Table A.2: SRAM Truth Table.

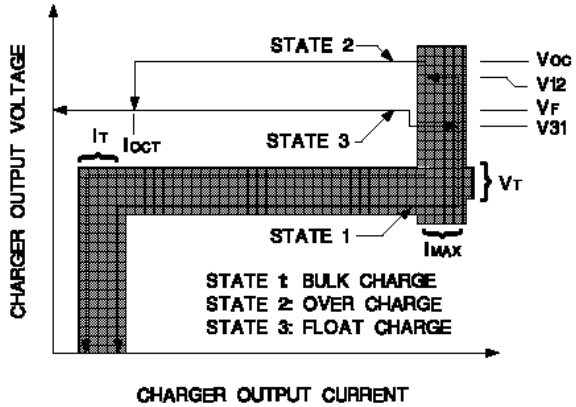
$\overline{\textbf{OE}}$	$\overline{\textbf{WE}}$	$\mathbf{D_0 - D_7}$	**Mode**
L	H	Data Out	Read
X	L	Data in	Write
H	H	High Z	Outputs disabled



Explanation: Dual Level Float Charger

A. Input power turns on, battery charges at trickle current rate.
B. Battery voltage reaches V_T enabling the driver and turning off the trickle bias output, battery charges at I_{MAX} rate.
C. Transition voltage V_{12} is reached and the charger indicates that it is now in the over-charge state, state 2.
D. Battery voltage approaches the over-charge level V_{OC} and the charge current begins to taper.
E. Charge current tapers to I_{OCT}. The current sense amplifier output, in this case tied to the OC TERMINATE input, goes high. The charger changes to the float state and holds the battery voltage at V_F.
F. Here a load (>I_{MAX}) begins to discharge the battery.
G. The load discharges the battery such that the battery voltage falls below V_{31}. The charger is now in state 1, again.

Figure A.1: Extract from UC3906 datasheet explaining the dual level charging cycle.



Design procedure

1.) Pick divider current, I_D. Recommended value is 50µA to 100µA.
2.) $R_C = 2.3V / I_D$
3.) $R_A + R_B = R_{SUM} = (V_F - 2.3V) / I_D$
4.) $R_D = 2.3VR_{SUM} / (V_{OC} - V_F)$
5.) $R_A = (R_{SUM} + R_X)(1 - 2.3V / V_T)$
where: $R_X = R_C R_D / (R_C + R_D)$
6.) $R_B = R_{SUM} - R_A$
7.) $R_S = 0.25V / I_{MAX}$
8.) $R_T = (V_{IN} - V_T - 2.5V) / I_T$

Note: $V_{12} = 0.95V_{OC}$
$V_{31} = 0.90V_F$
$I_{OCT} = \frac{I_{MAX}}{10}$
For further design and application information see UICC Application Note U-104

Figure A.2: Extract from UC3906 datasheet explaining the design process for the dual level charger.

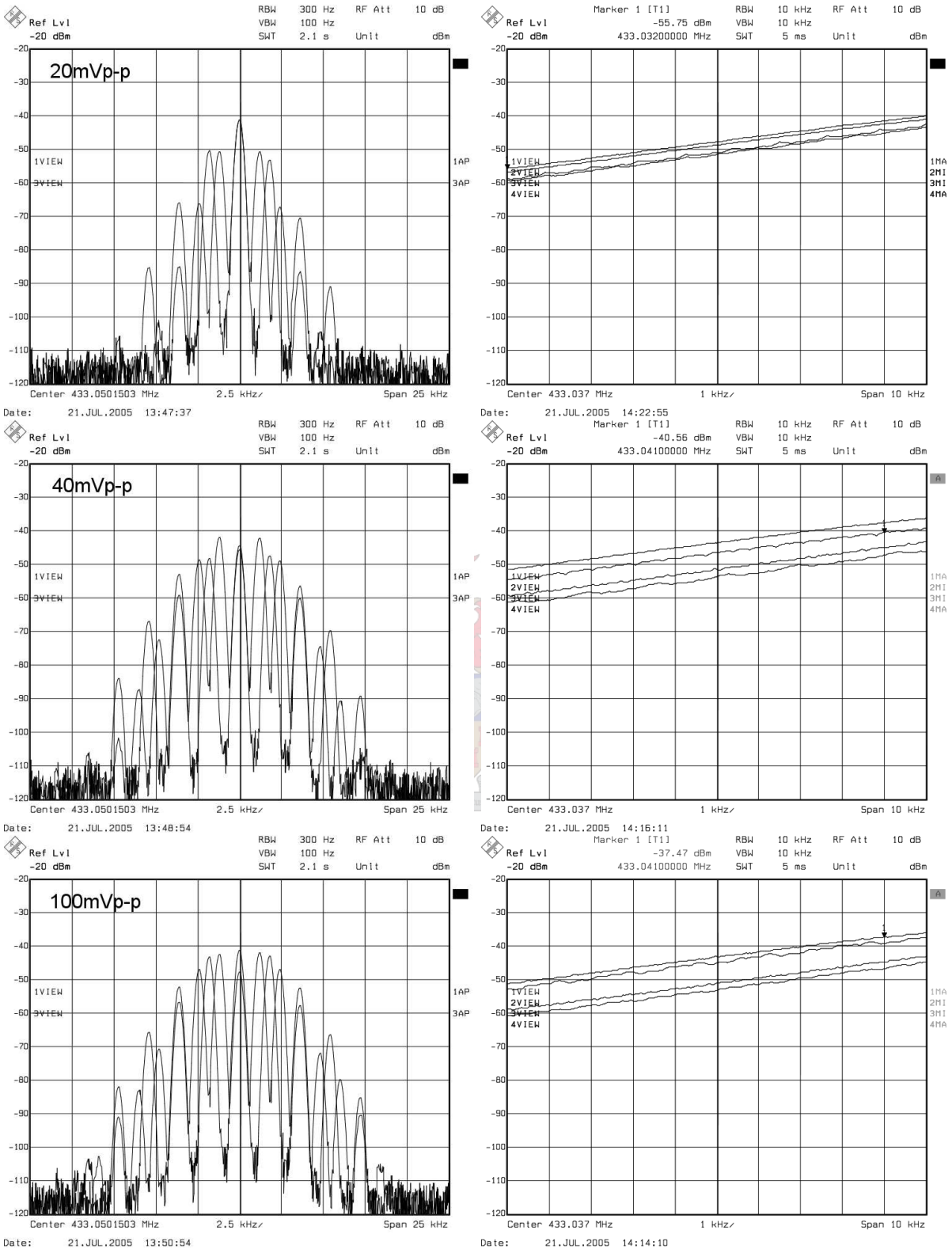


Figure A.3: SA: Modulated carrier for 20mV$_{p-p}$, 40mV$_{p-p}$, 100mV$_{p-p}$ (1.2kHz & 1.8kHz superimposed).

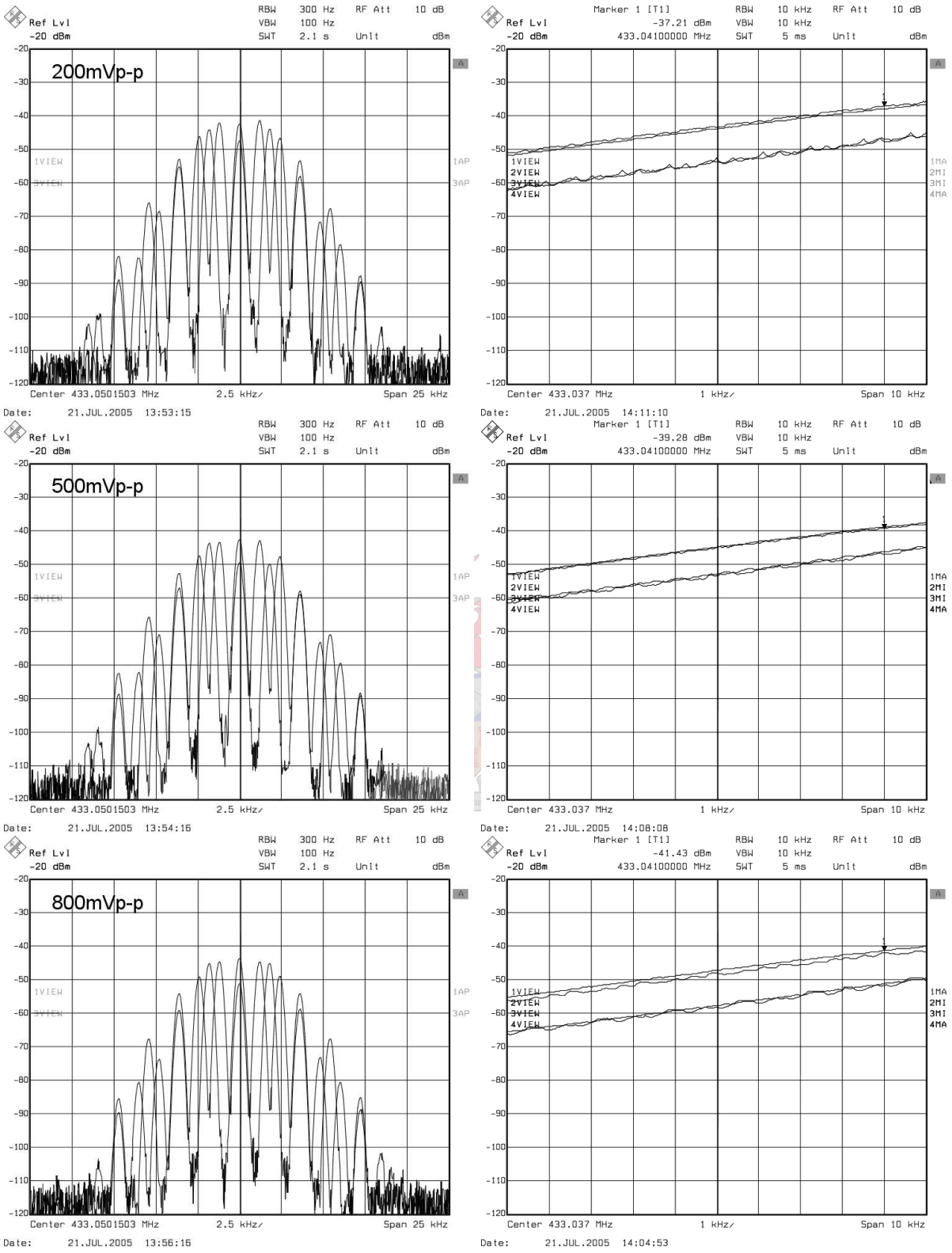


Figure A.4: SA: Modulated carrier for 200mV$_{p-p}$, 500mV$_{p-p}$, 800mV$_{p-p}$ (1.2kHz & 1.8kHz superimposed).

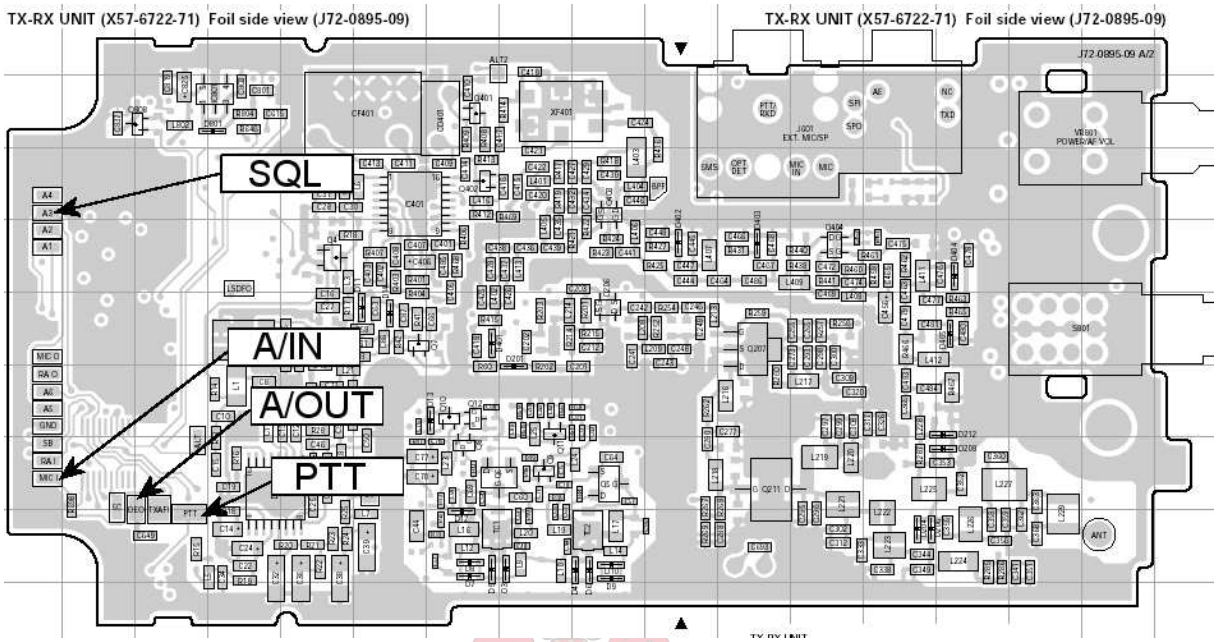


Figure A.5: Connection points to Kenwood TK-3160 PCB.

Table A.3: Radio DB9 Connections

PIN	Connection
1	V_{IN} 7 to 13.8V
2	V_{IN}7 to 13.8V
3	Squelch
4	PTT
5	TX Audio
6	GND
7	GND
8	N/C
9	RX Audio

Appendix B

System Flow Diagrams

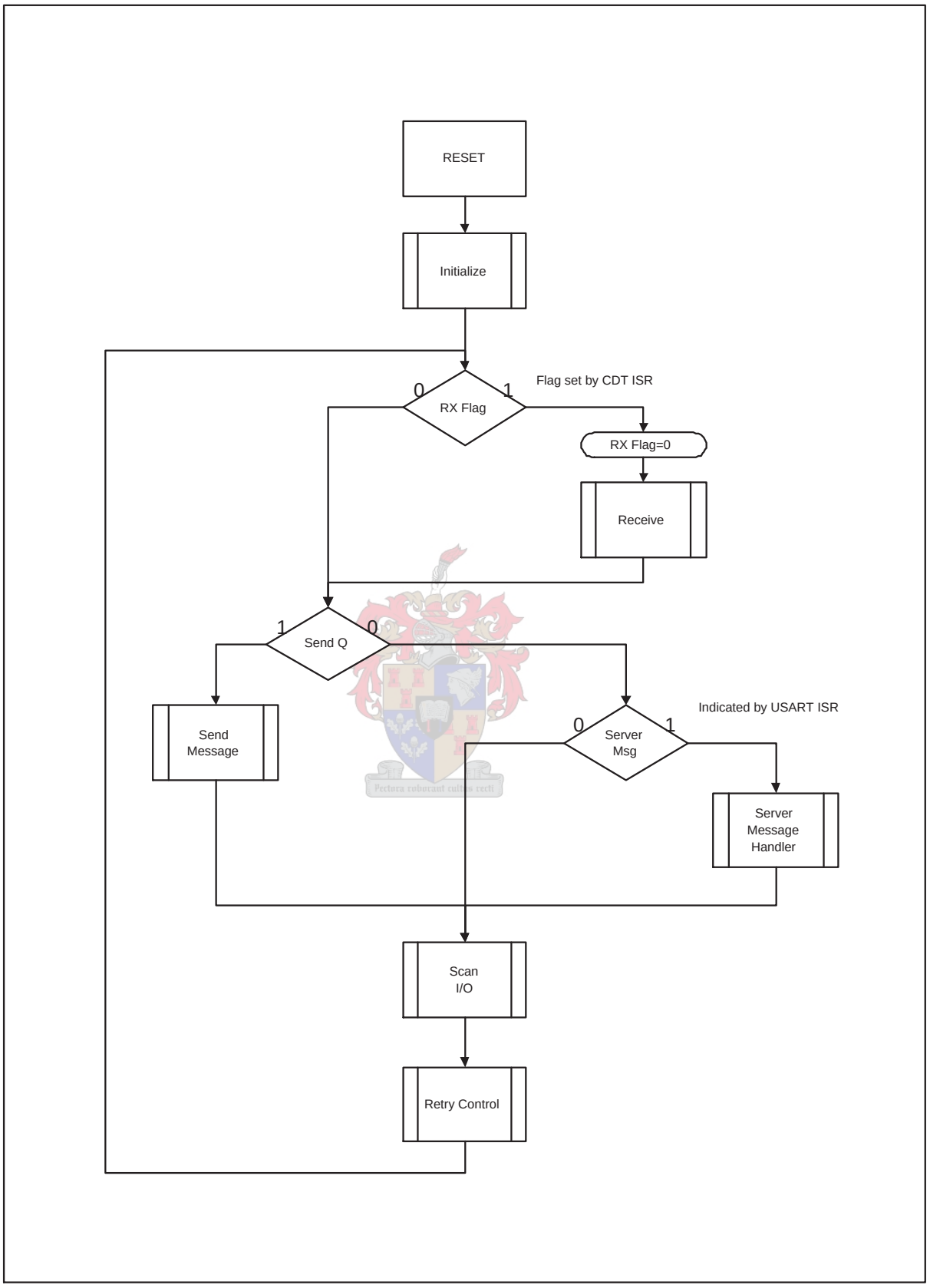


Figure B.1: Main program loop.

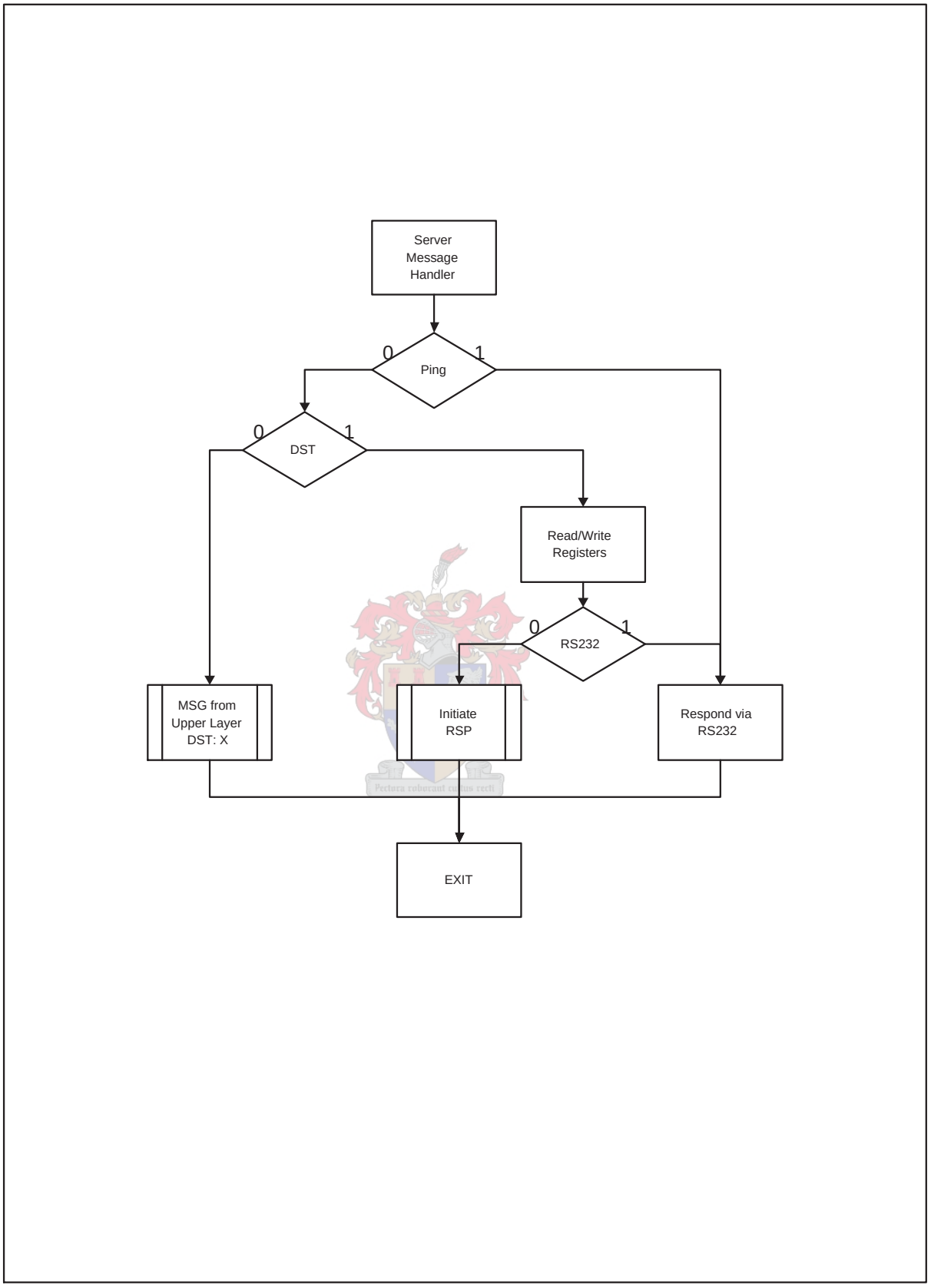


Figure B.2: Server Message Handler.

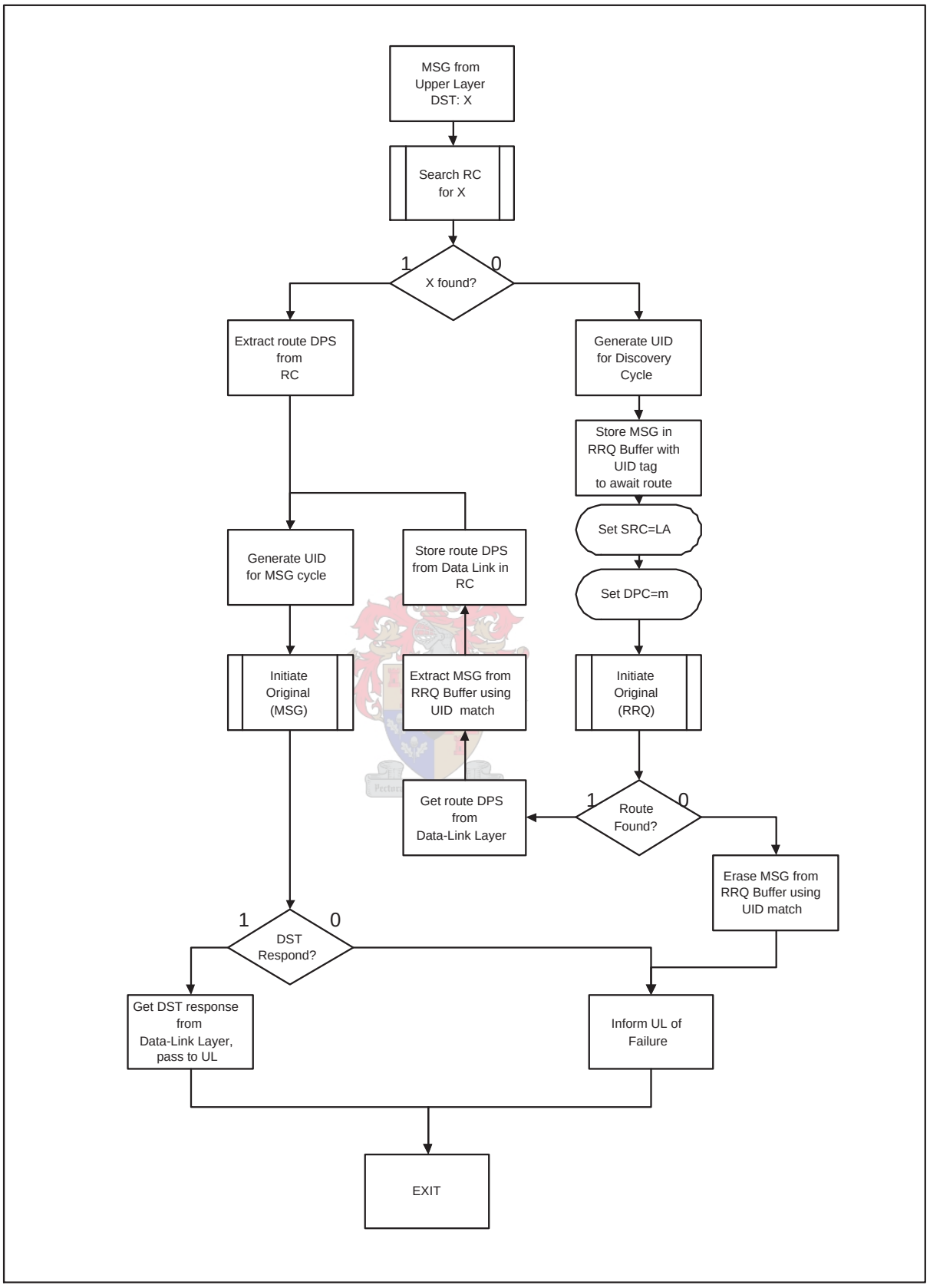


Figure B.3: RX message from server to forward to another station.

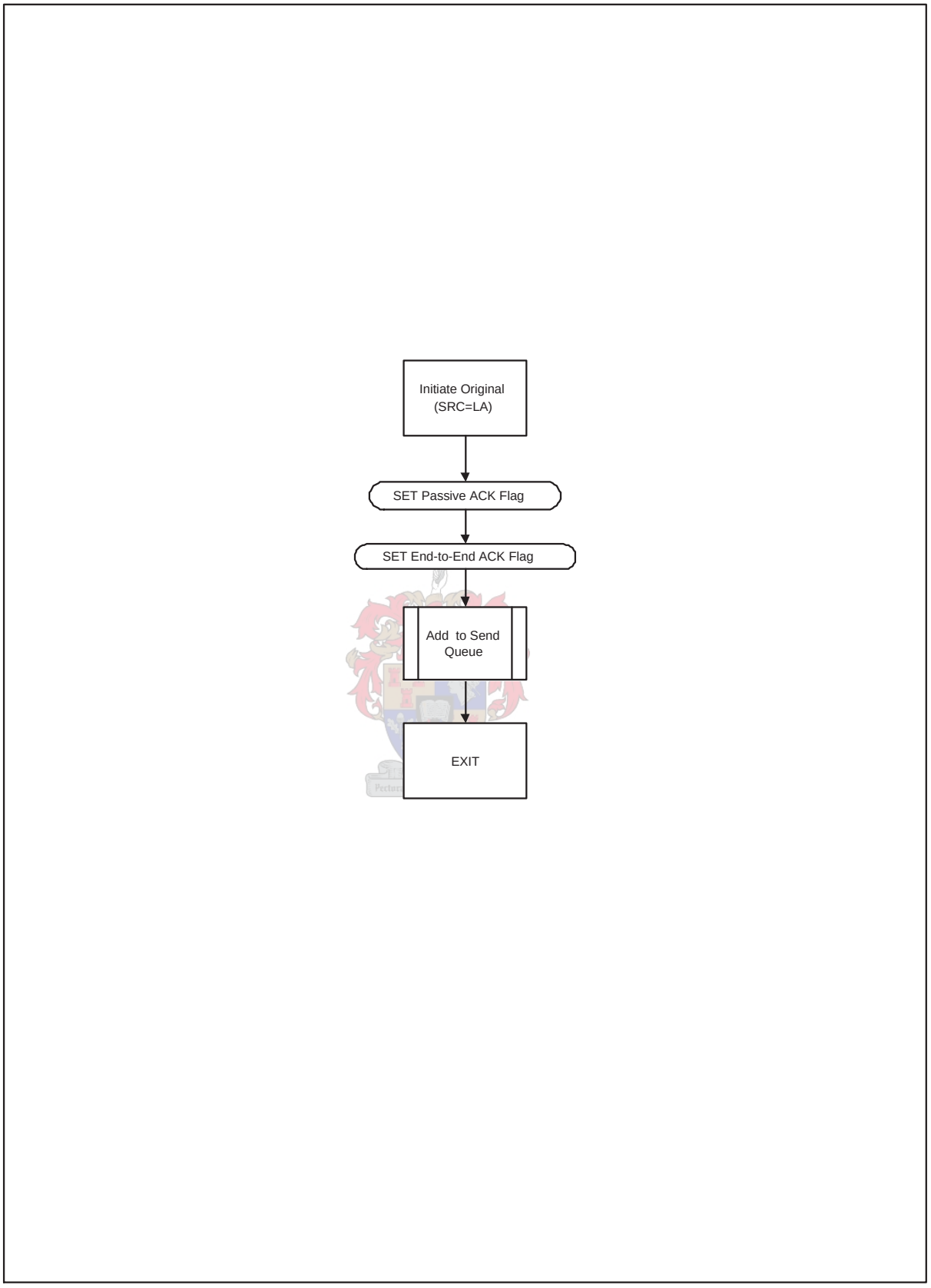


Figure B.4: Initiate Message.

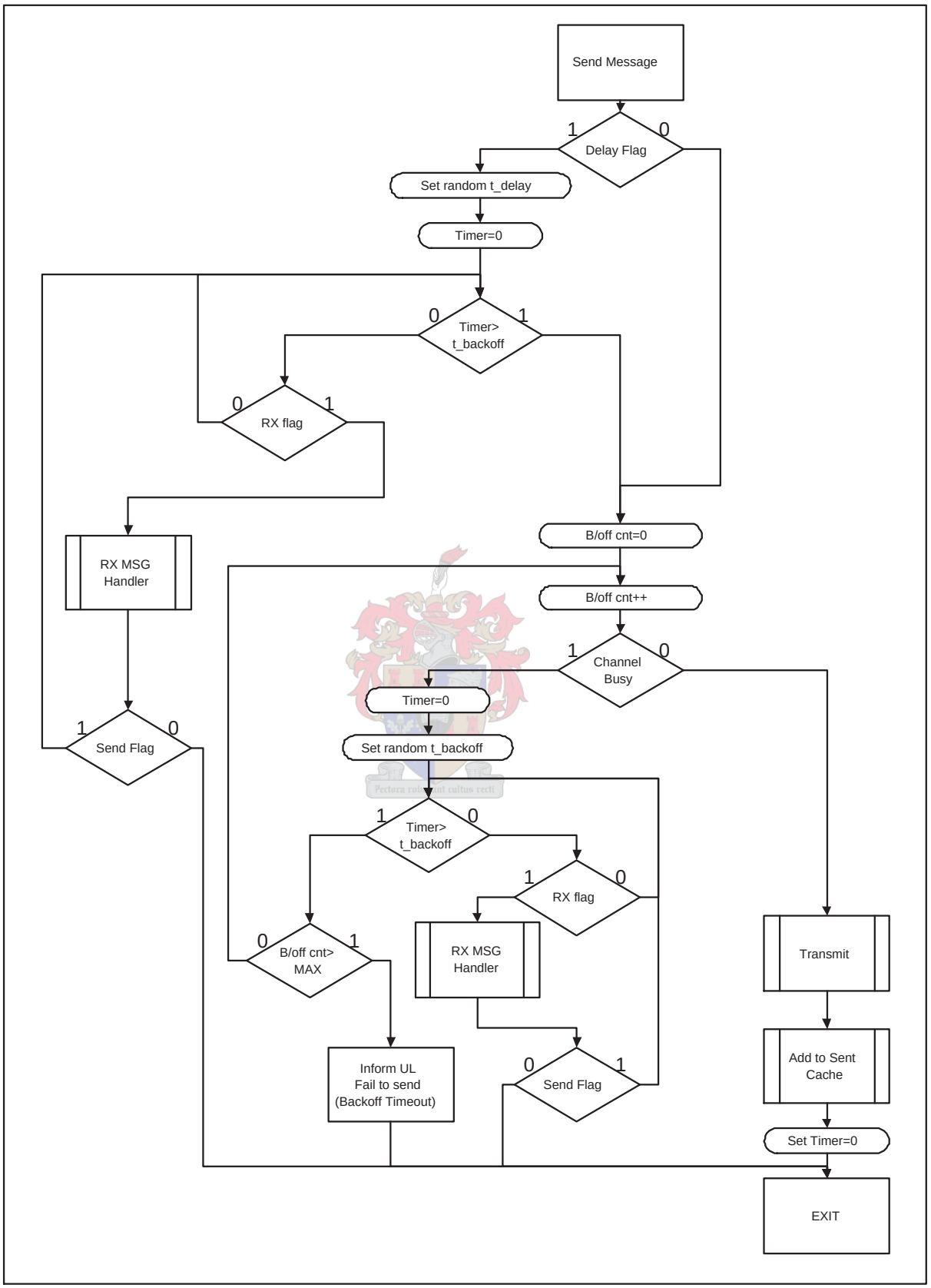


Figure B.5: Send Message Handler.

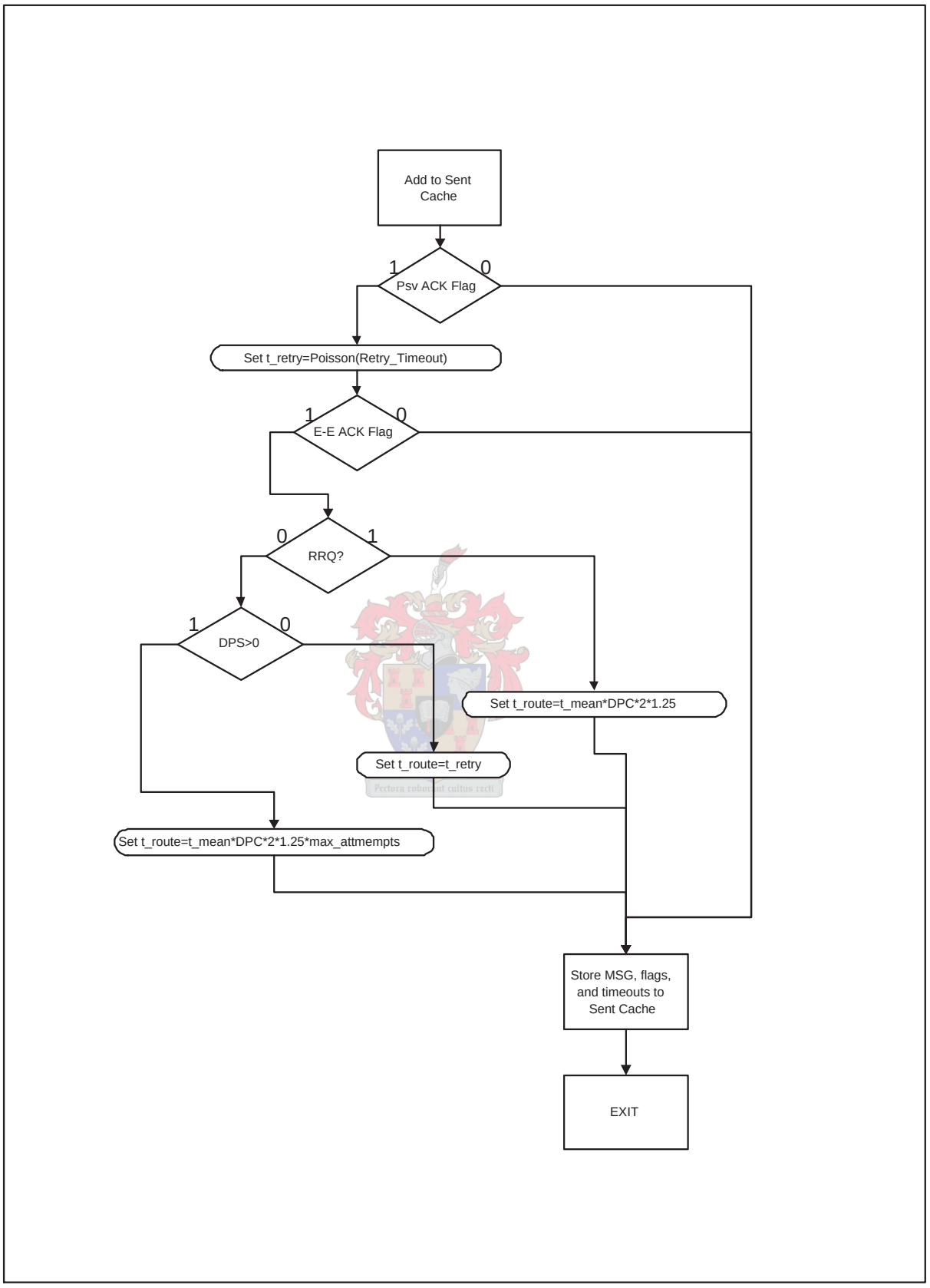


Figure B.6: Store message to Sent Cache.

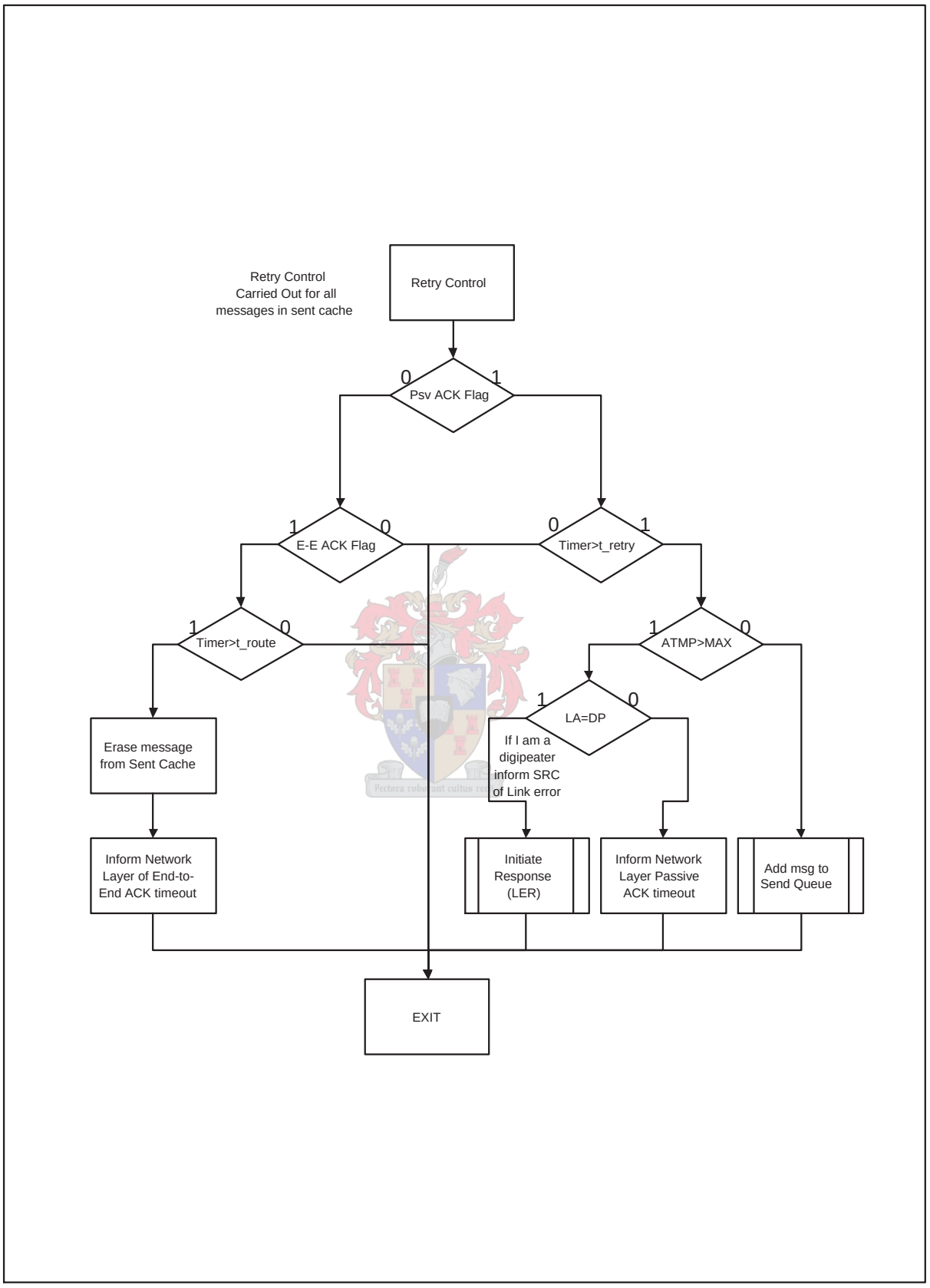


Figure B.7: Retry Control

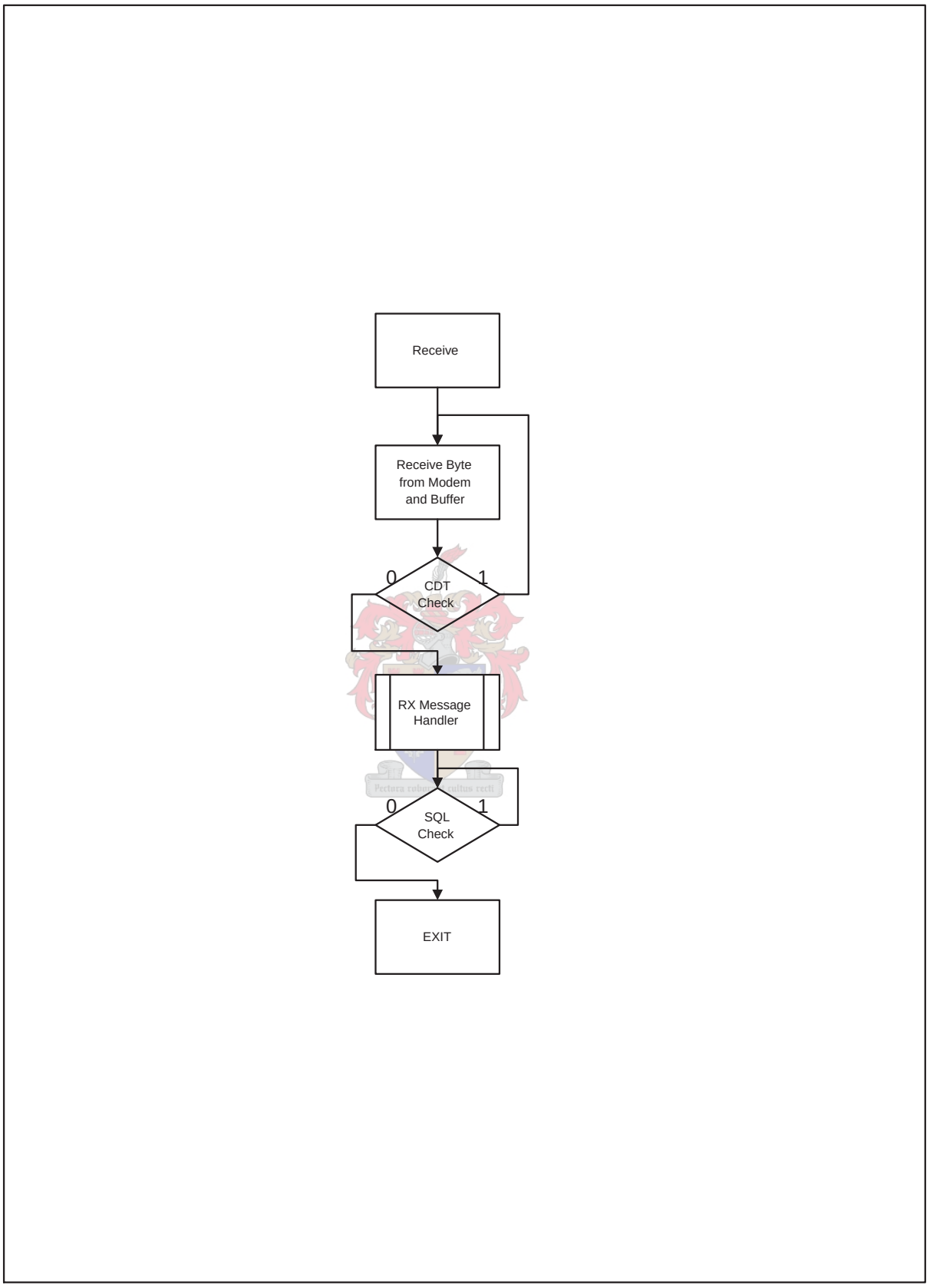


Figure B.8: Receive Message.

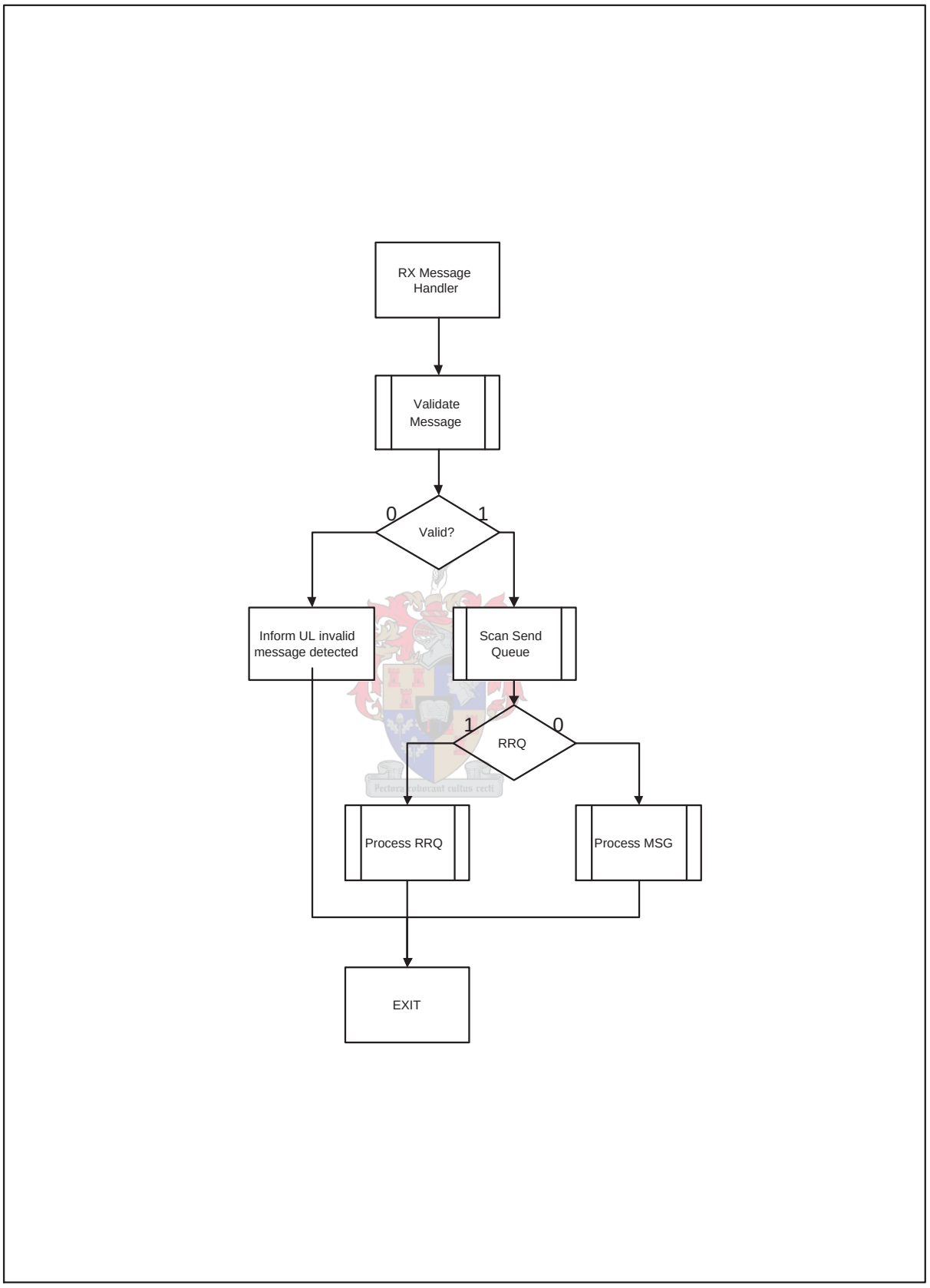


Figure B.9: RX Message Handler.

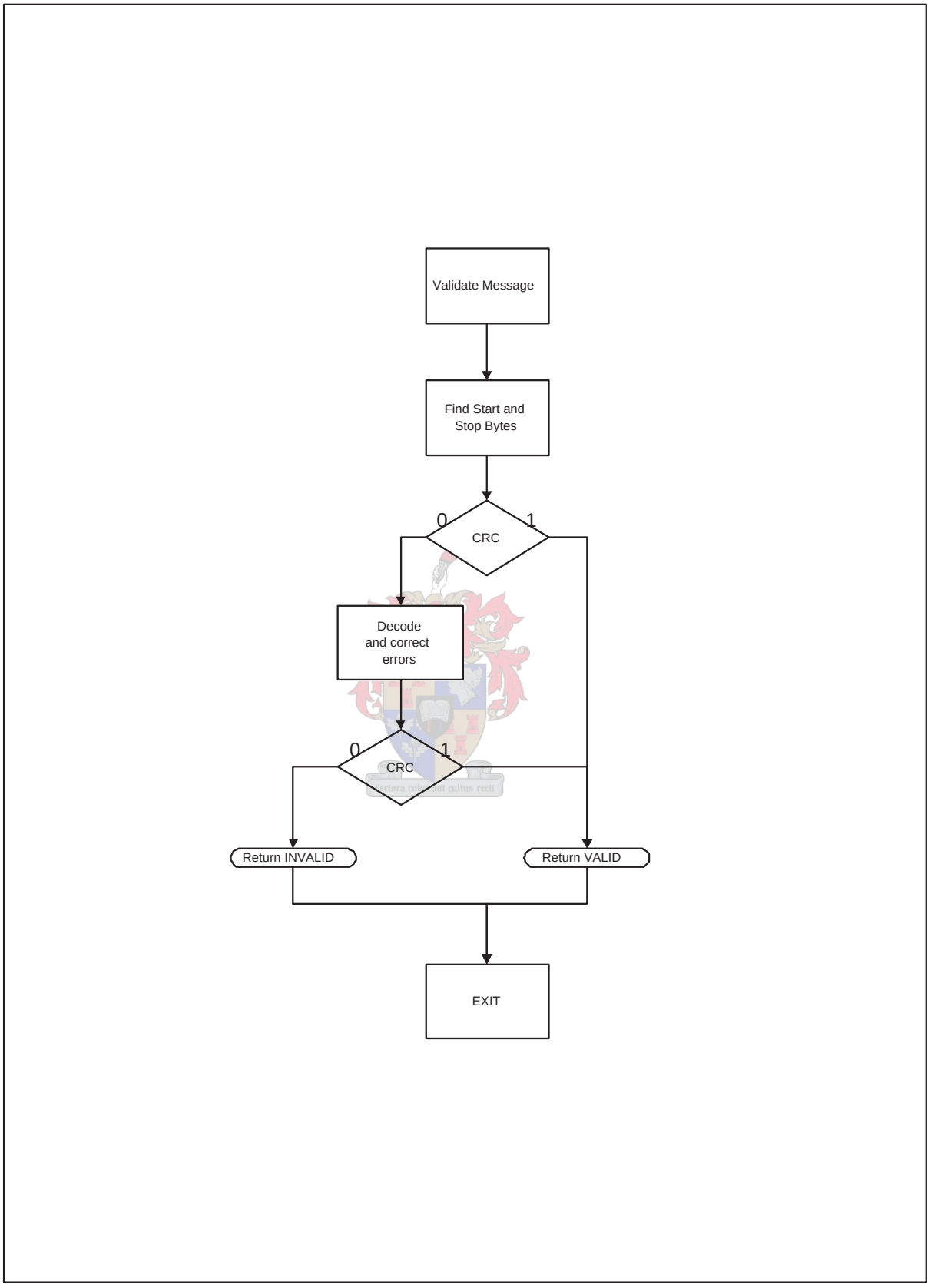


Figure B.10: Validate Message

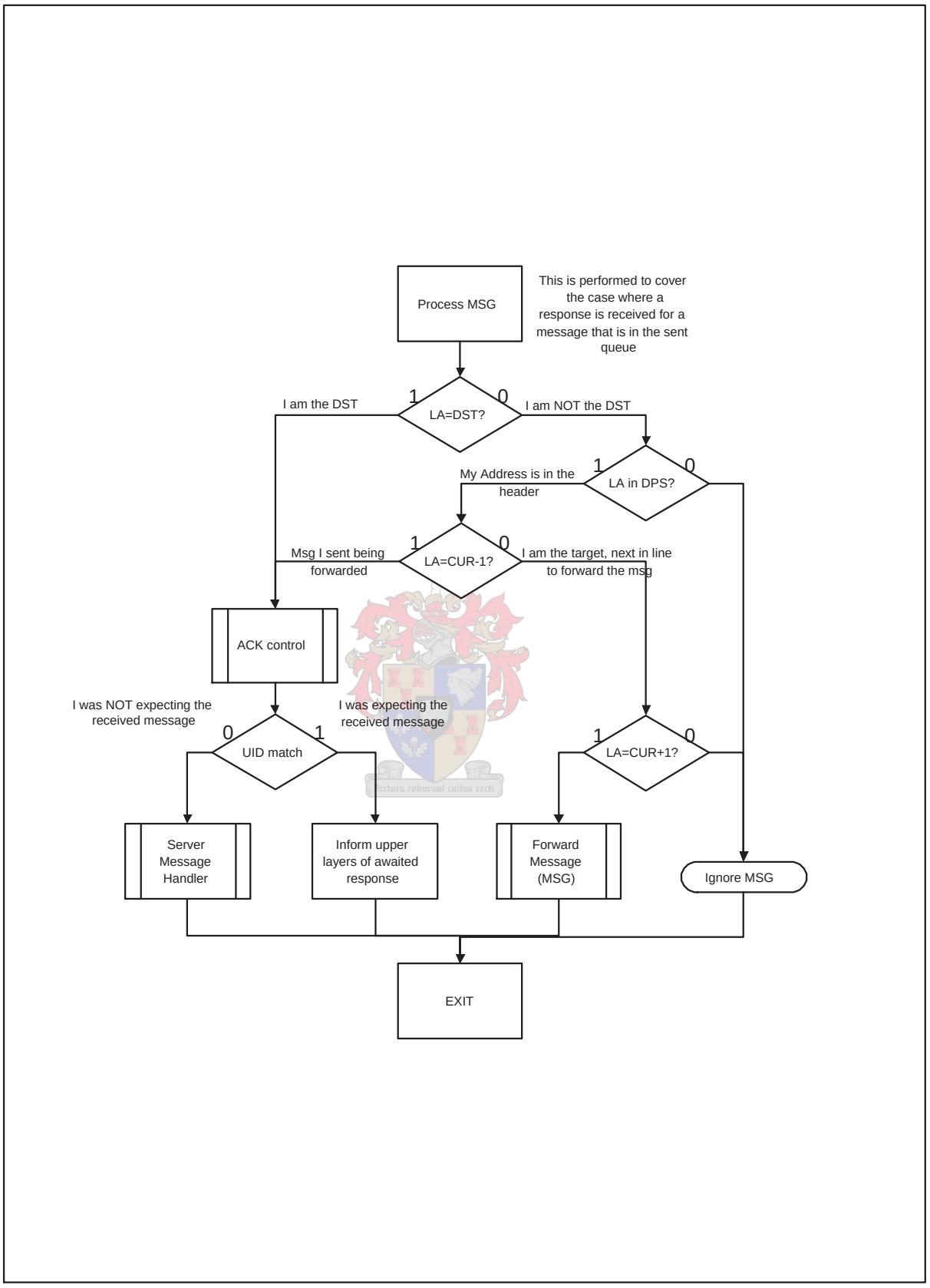


Figure B.11: Process Received Message.

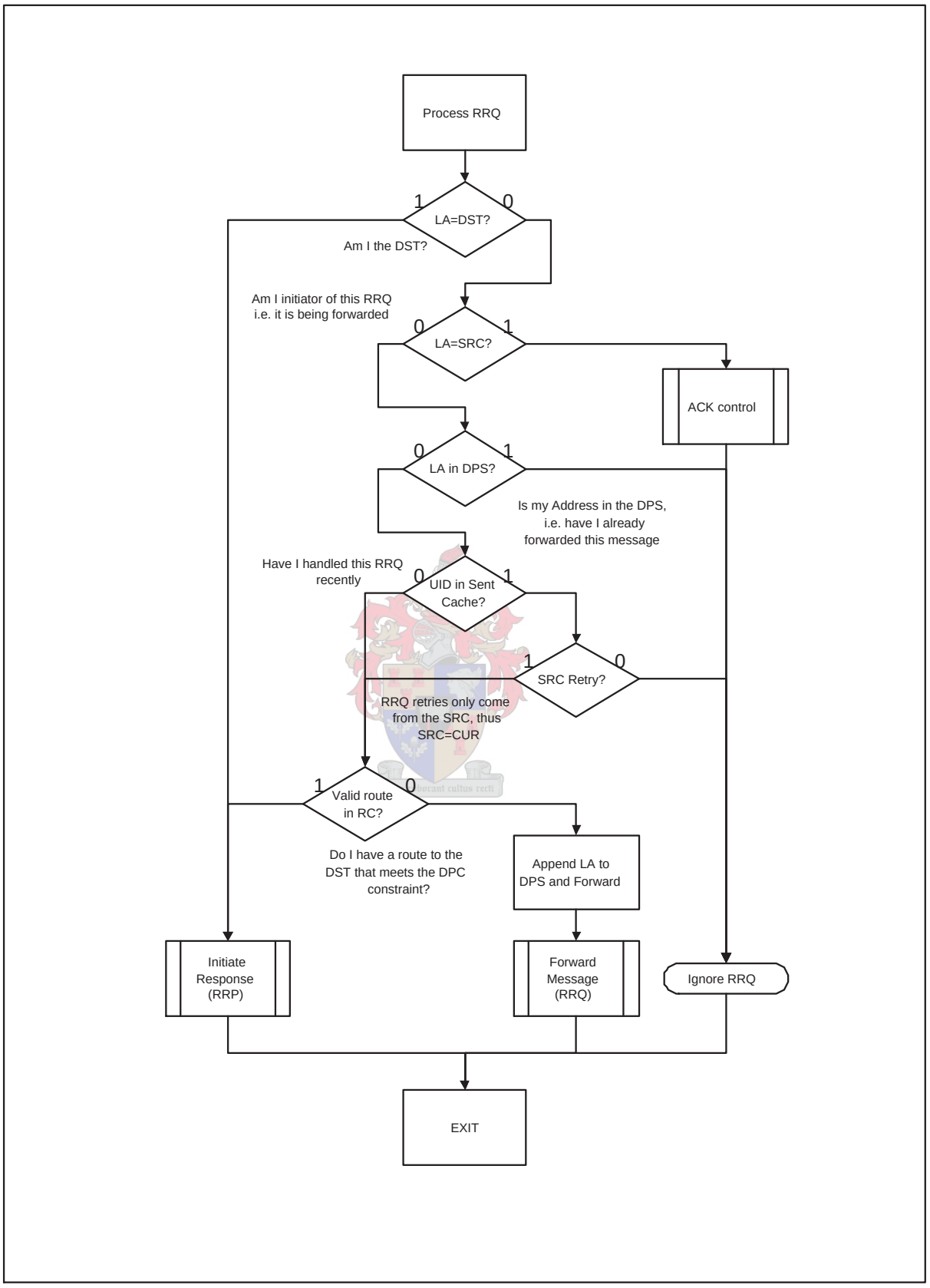


Figure B.12: Process Received RRQ.

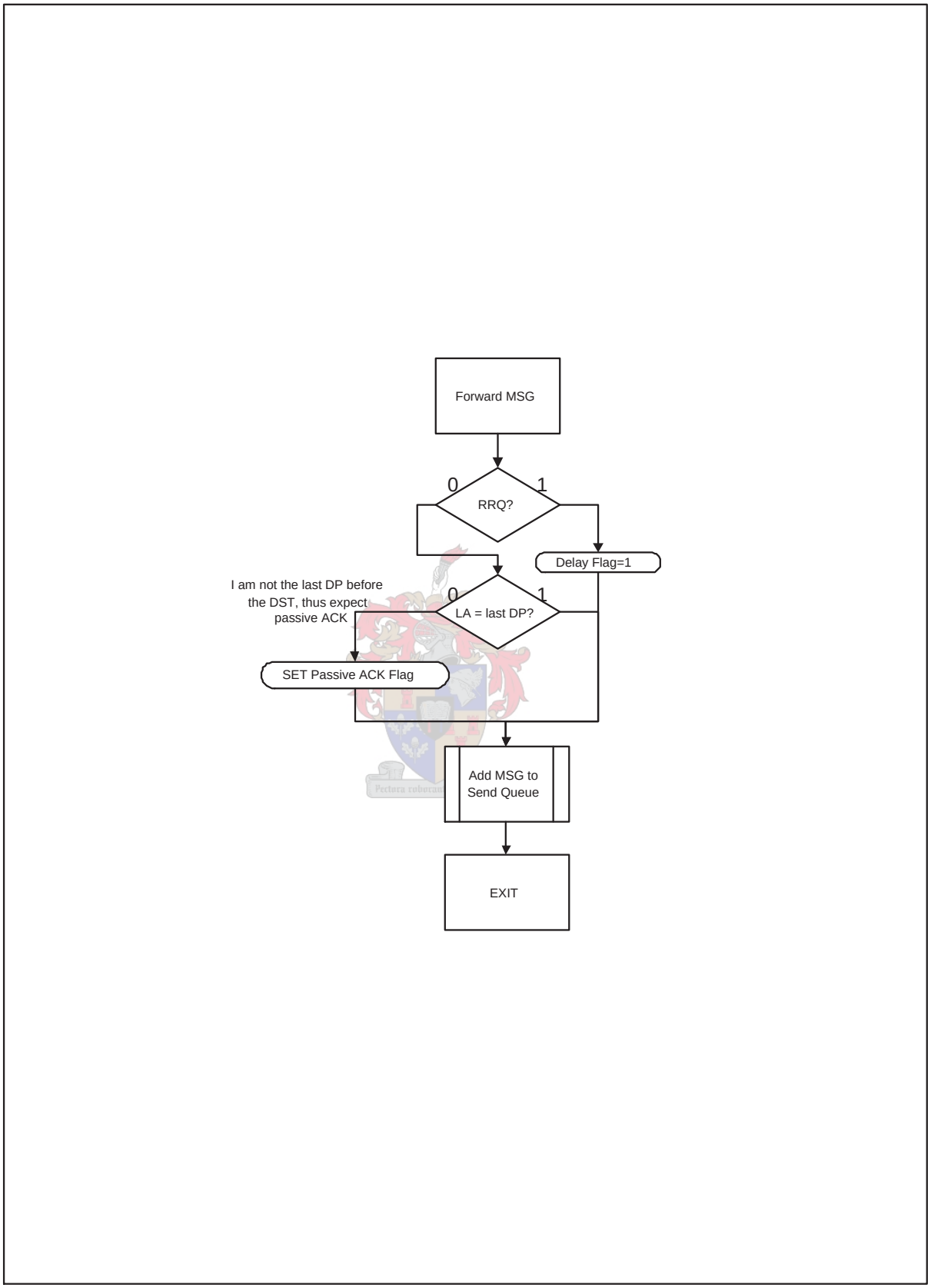


Figure B.13: Forward Message.

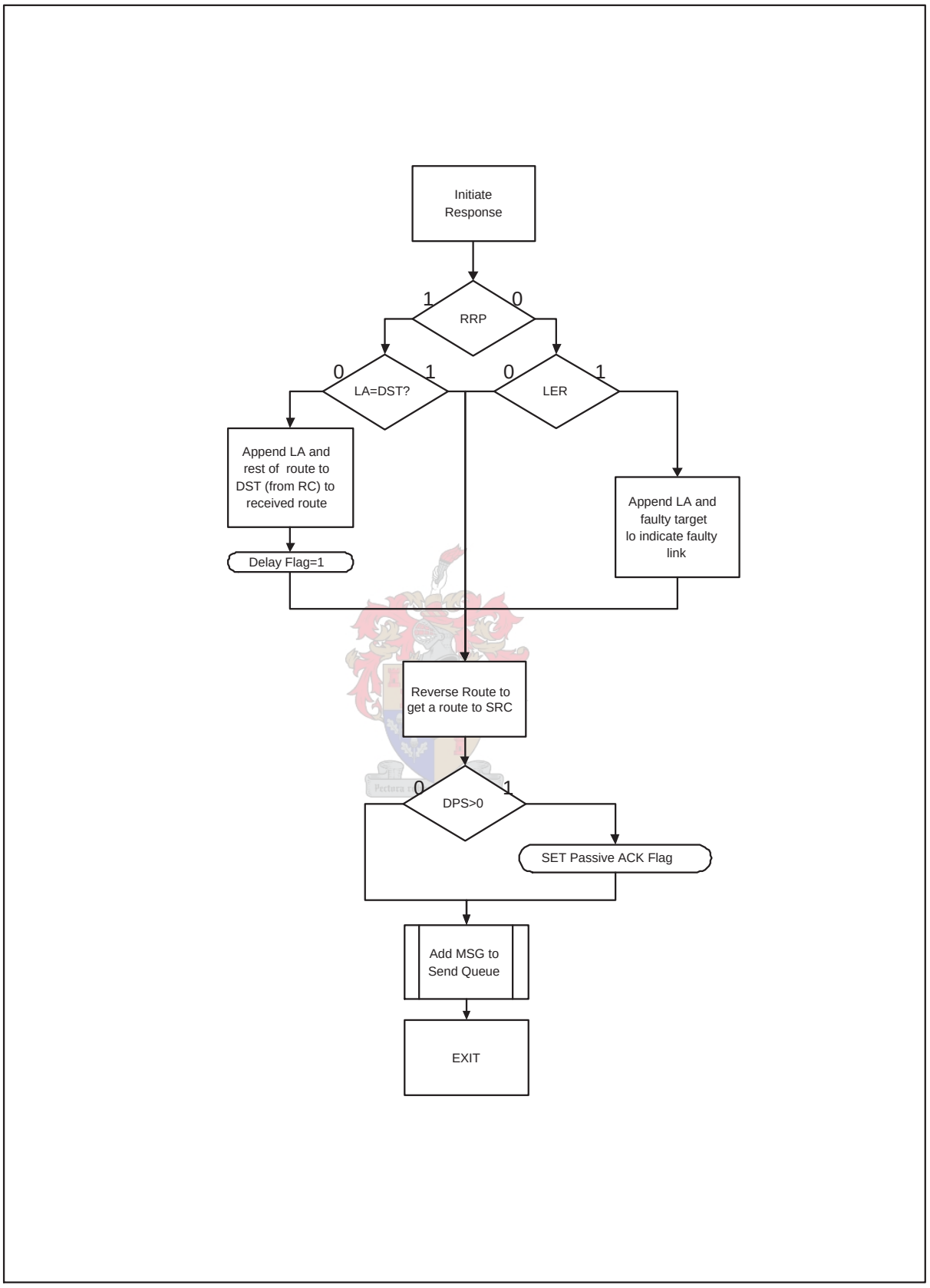


Figure B.14: Initiate Response.

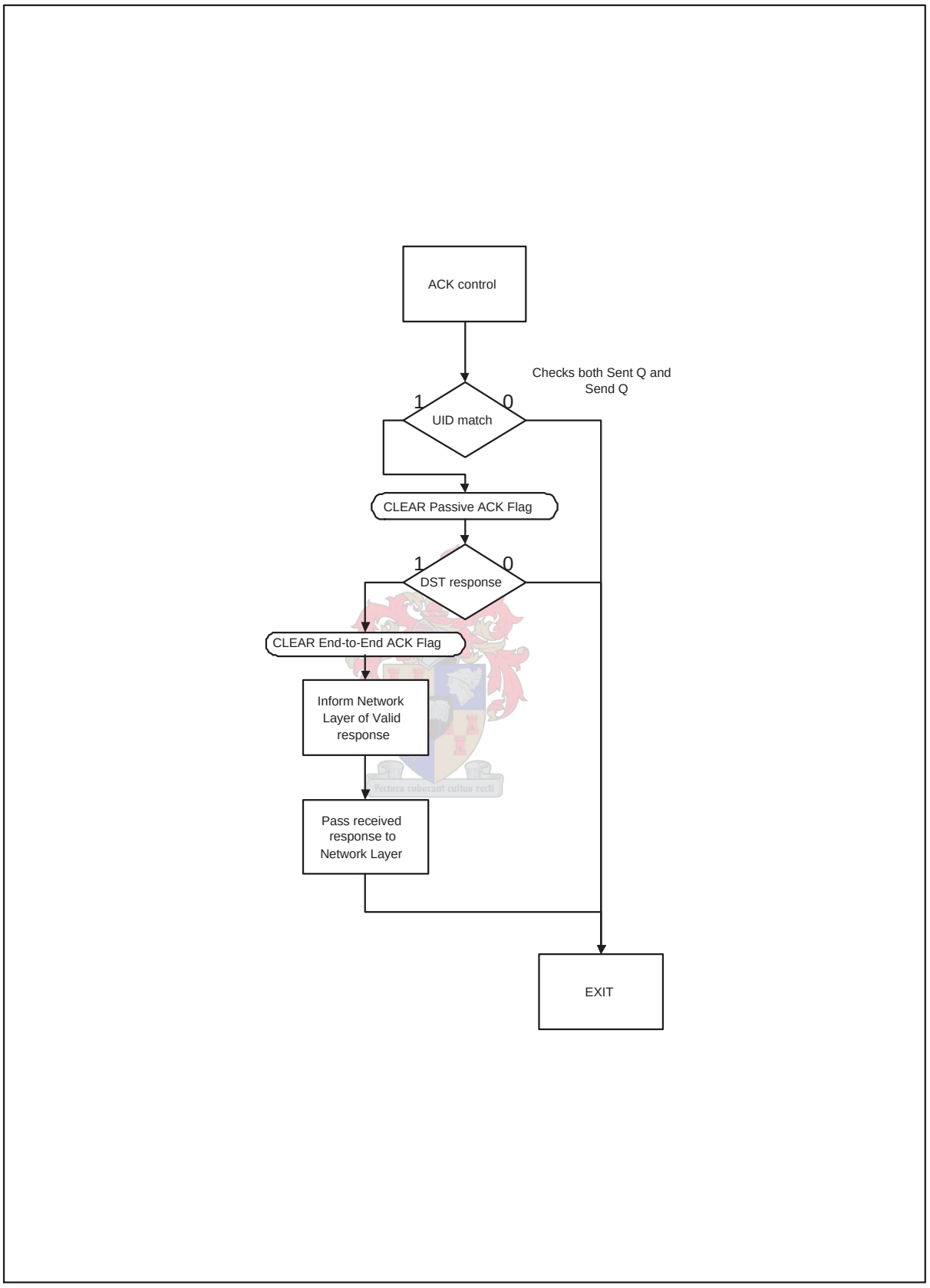


Figure B.15: Acknowledgement Control.

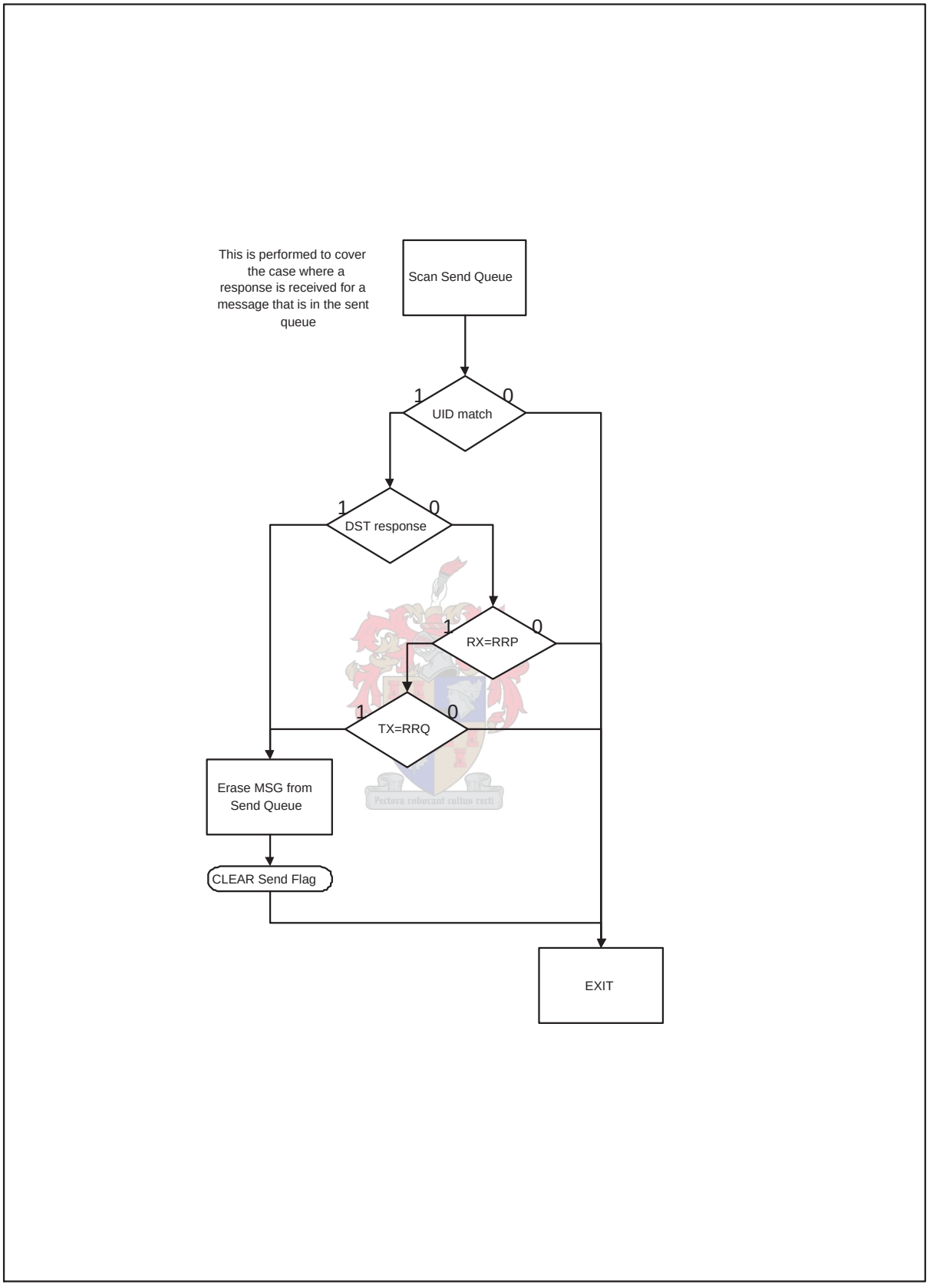


Figure B.16: Scan Send Queue.

Appendix C

Circuit Diagrams and PCB Layouts

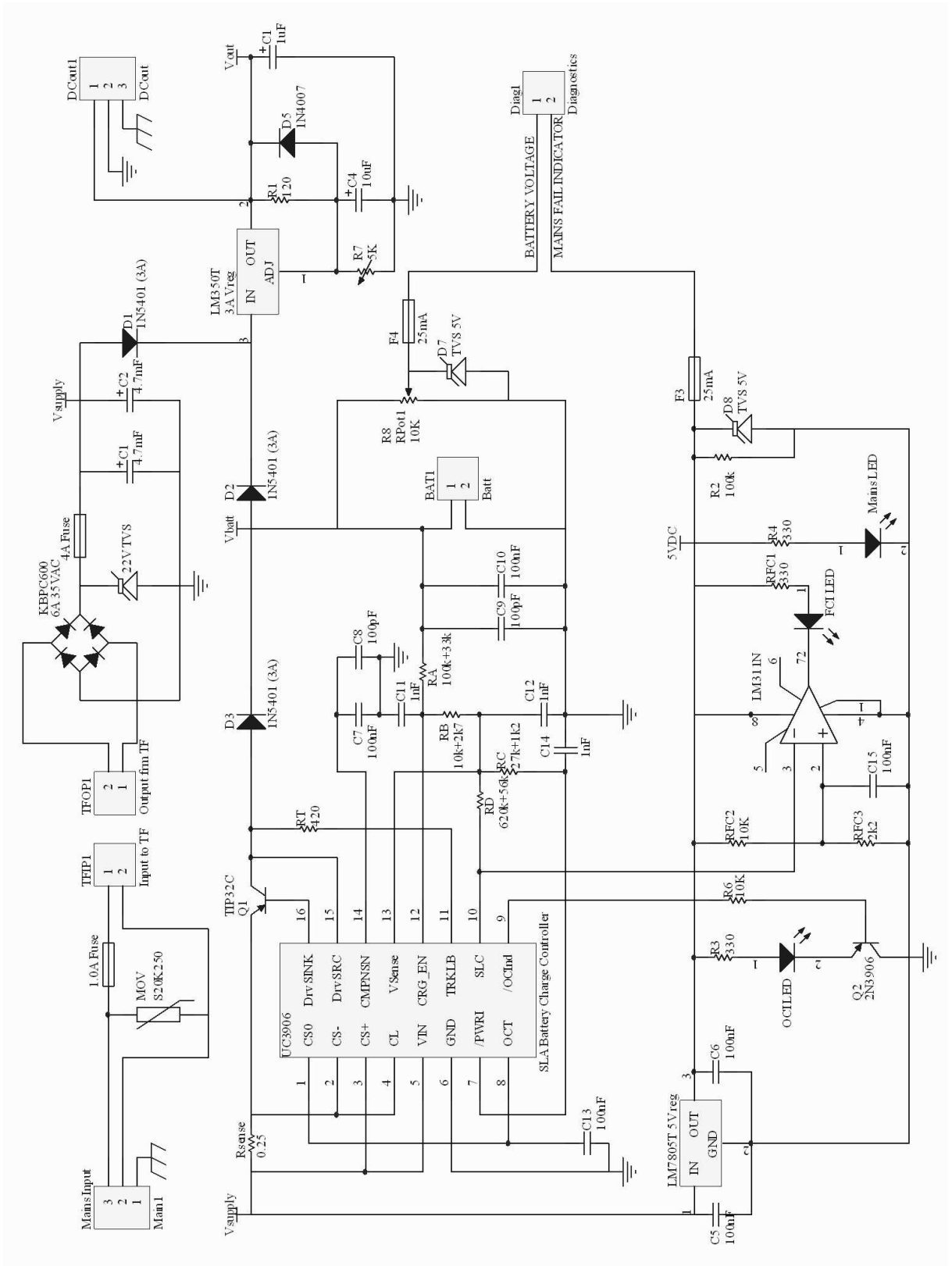


Figure C.1: Complete schematic diagram of the Power Supply circuitry.

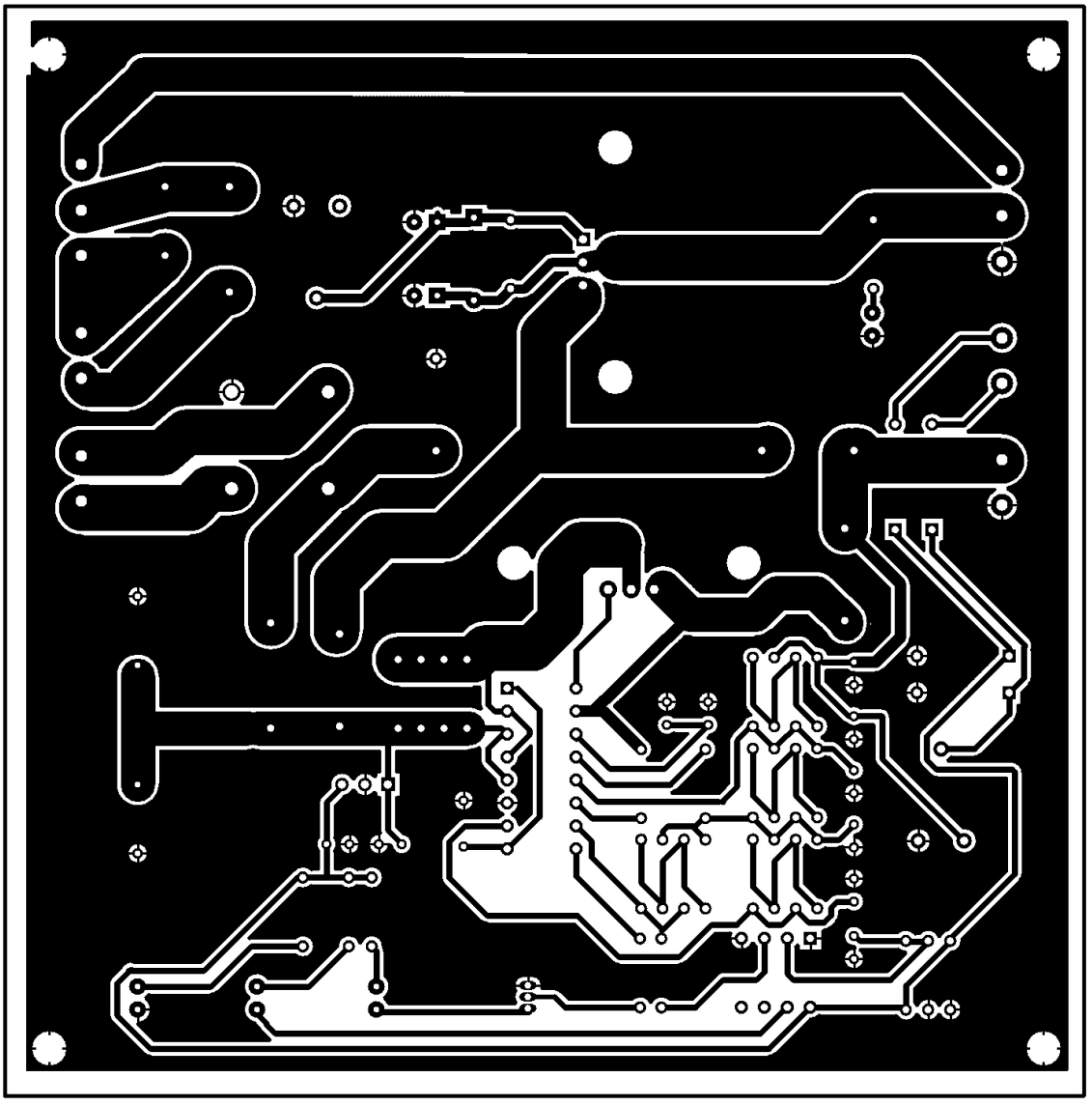


Figure C.2: Power Supply PCB Top View, Scale 1:1.

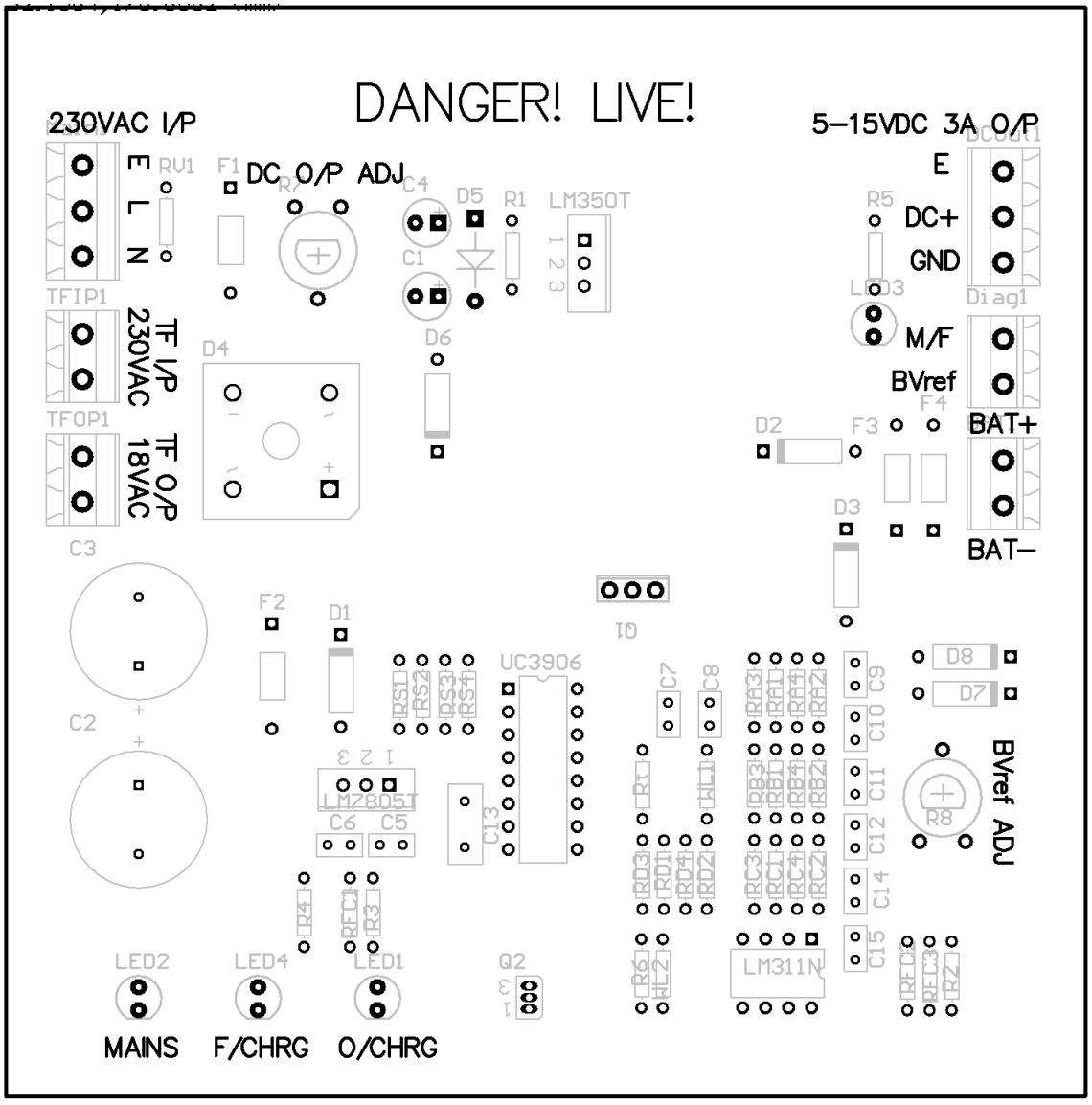


Figure C.3: Power Supply PCB Bottom View, Scale 1:1.

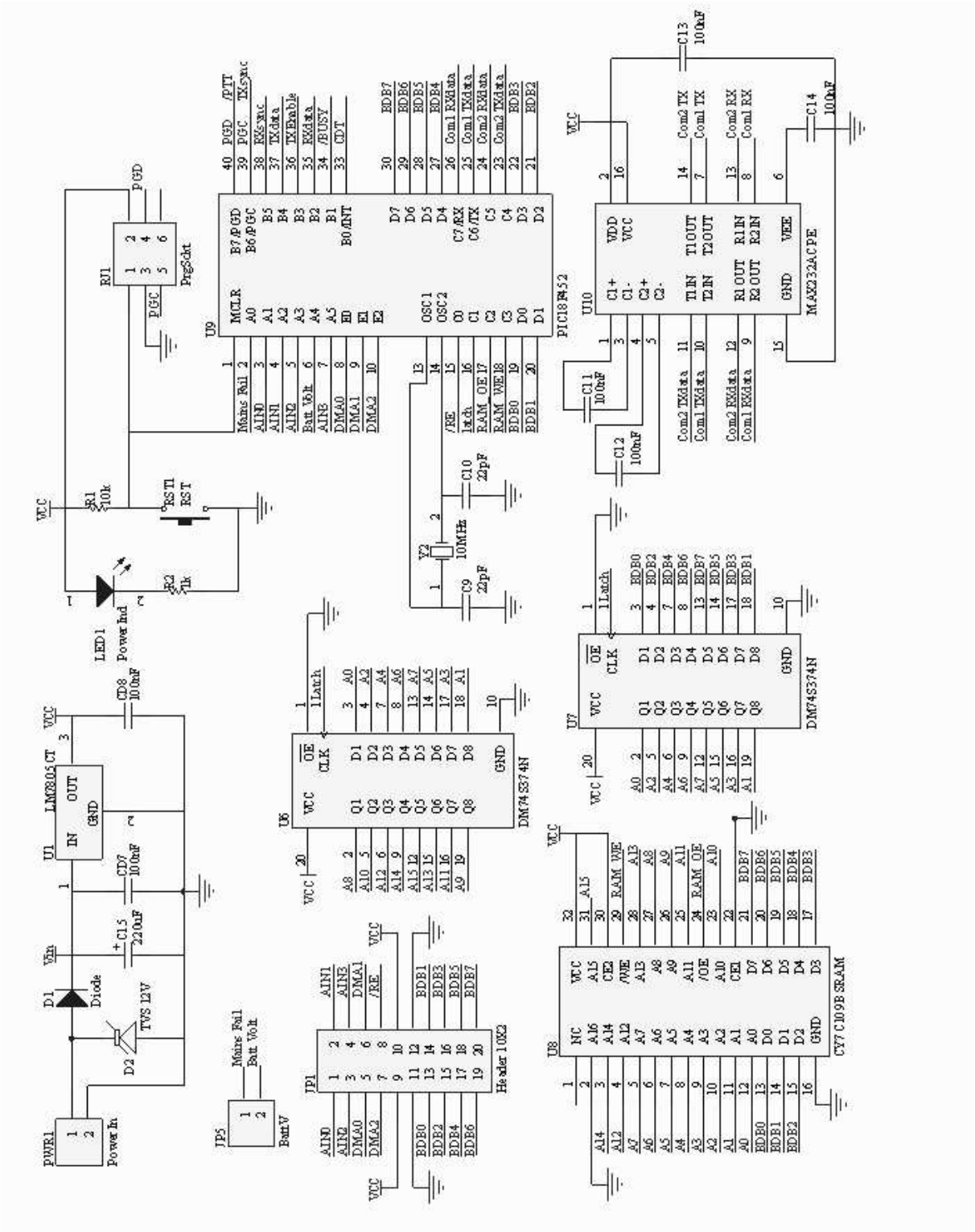


Figure C.4: Part A of complete schematic Diagram of the RTU circuitry.

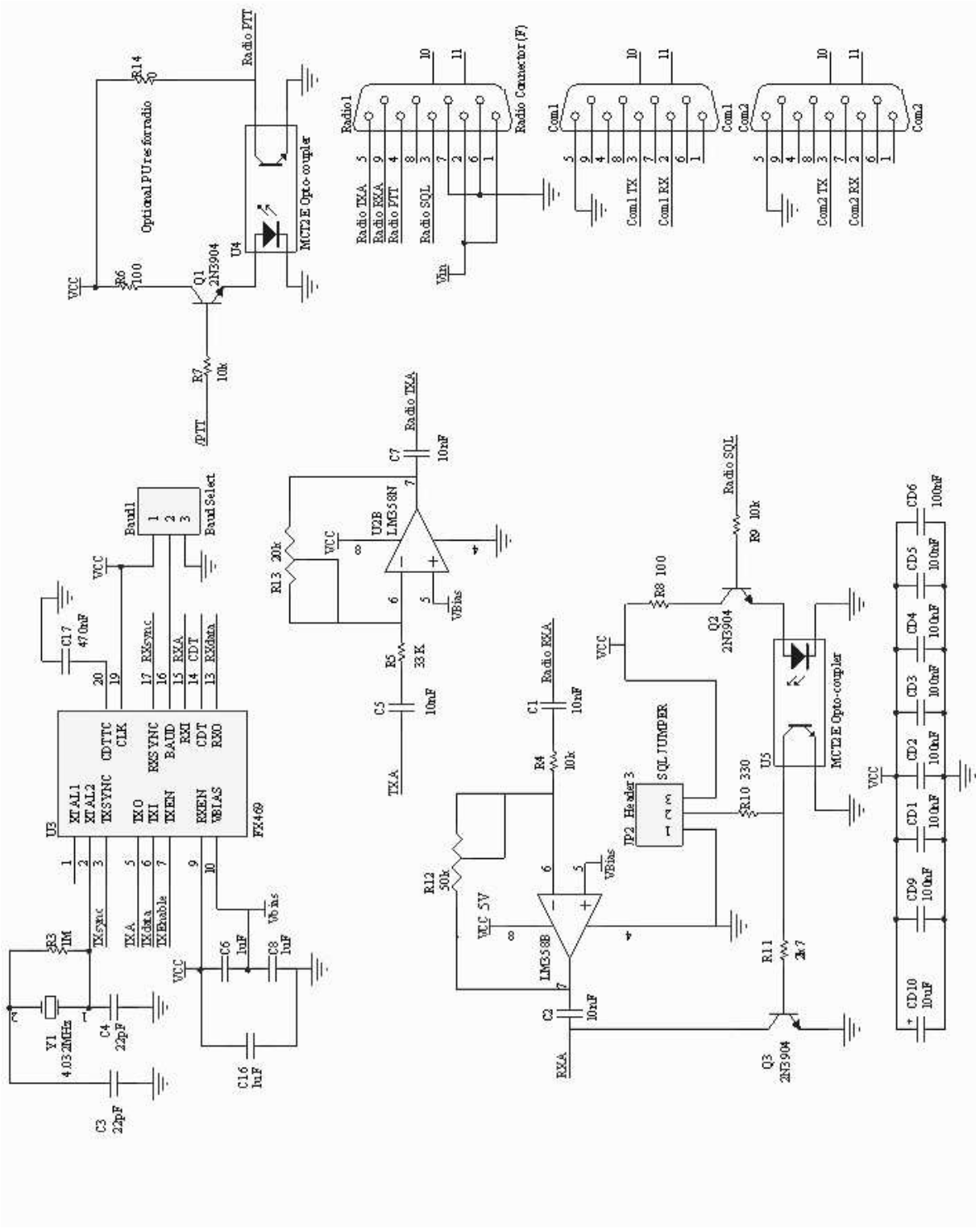


Figure C.5: Part B of complete schematic diagram of the RTU circuitry.

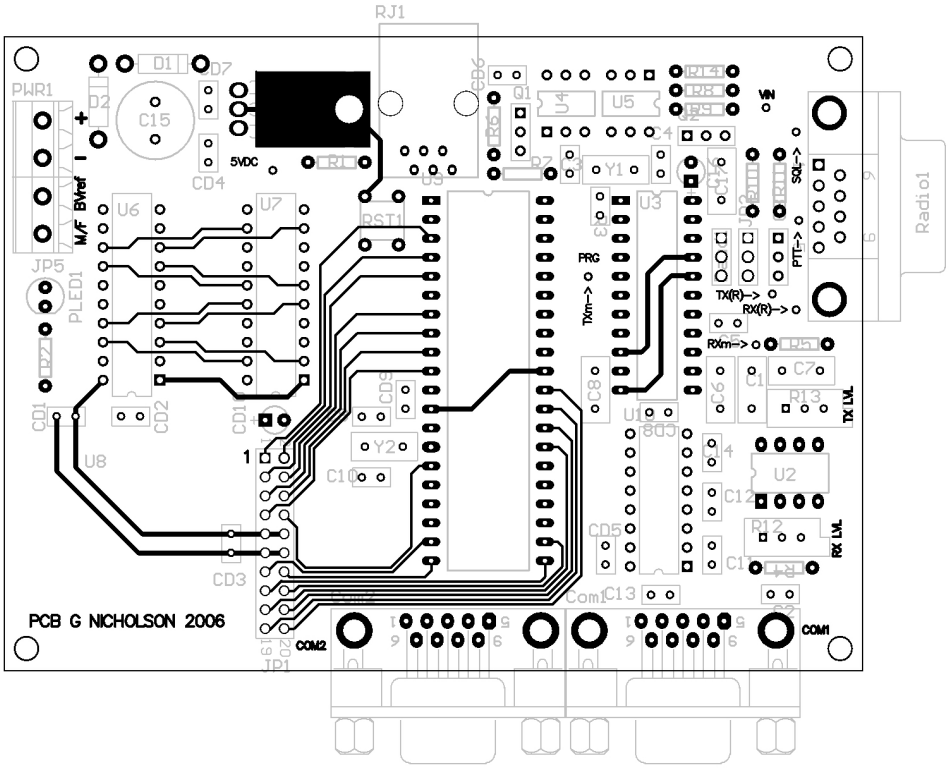


Figure C.6: RTU PCB Top View, Scale 1:1.

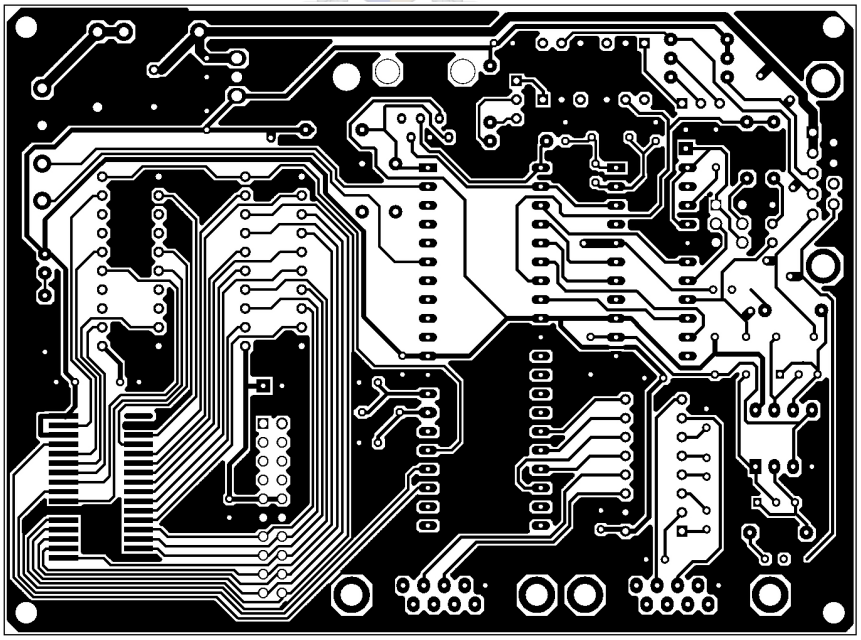


Figure C.7: RTU PCB Bottom View, Scale 1:1.

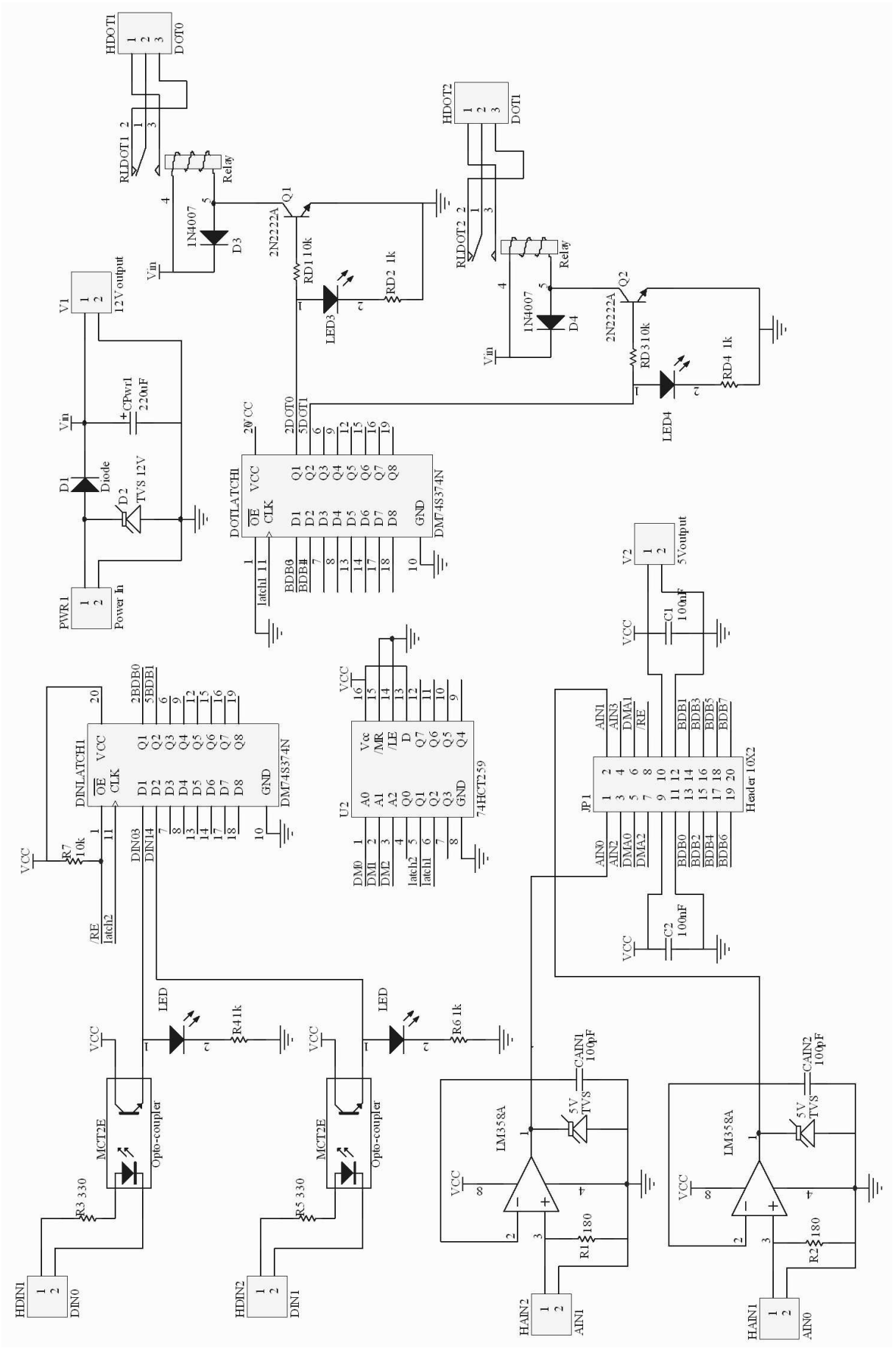


Figure C.8: Complete schematic diagram of the I/O Module circuitry.

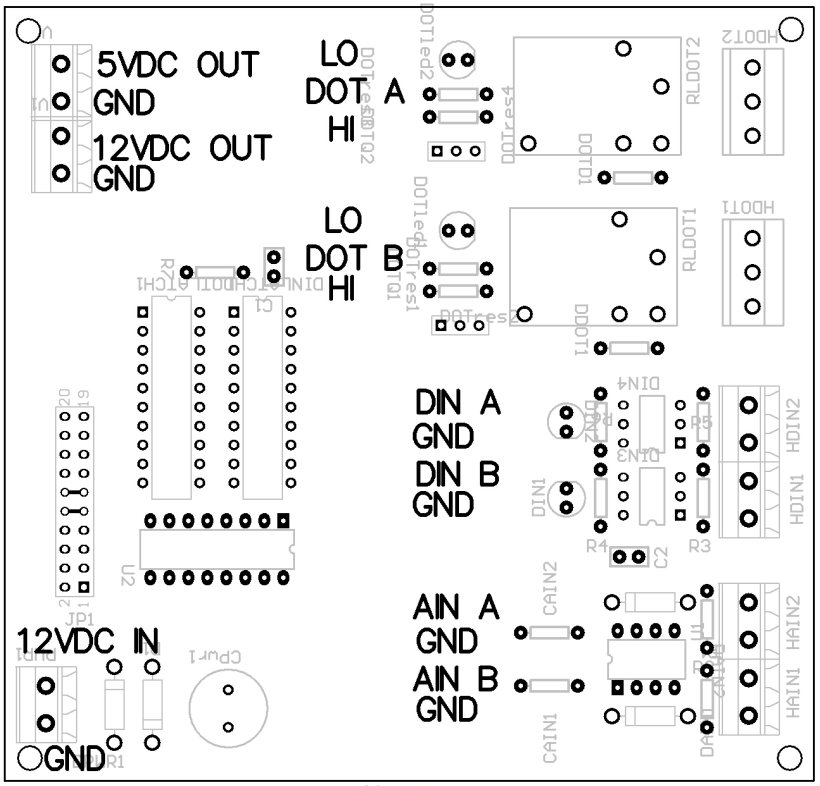


Figure C.9: I/O PCB Top View, Scale 1:1.

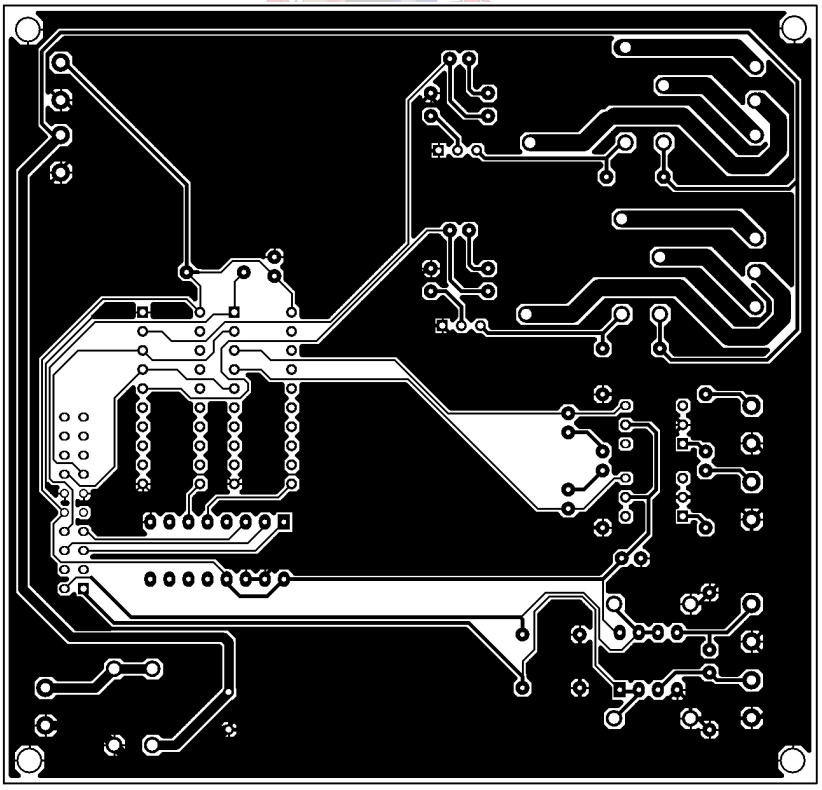


Figure C.10: I/O PCB Bottom View, Scale 1:1.

Appendix D

Prototype Pictures

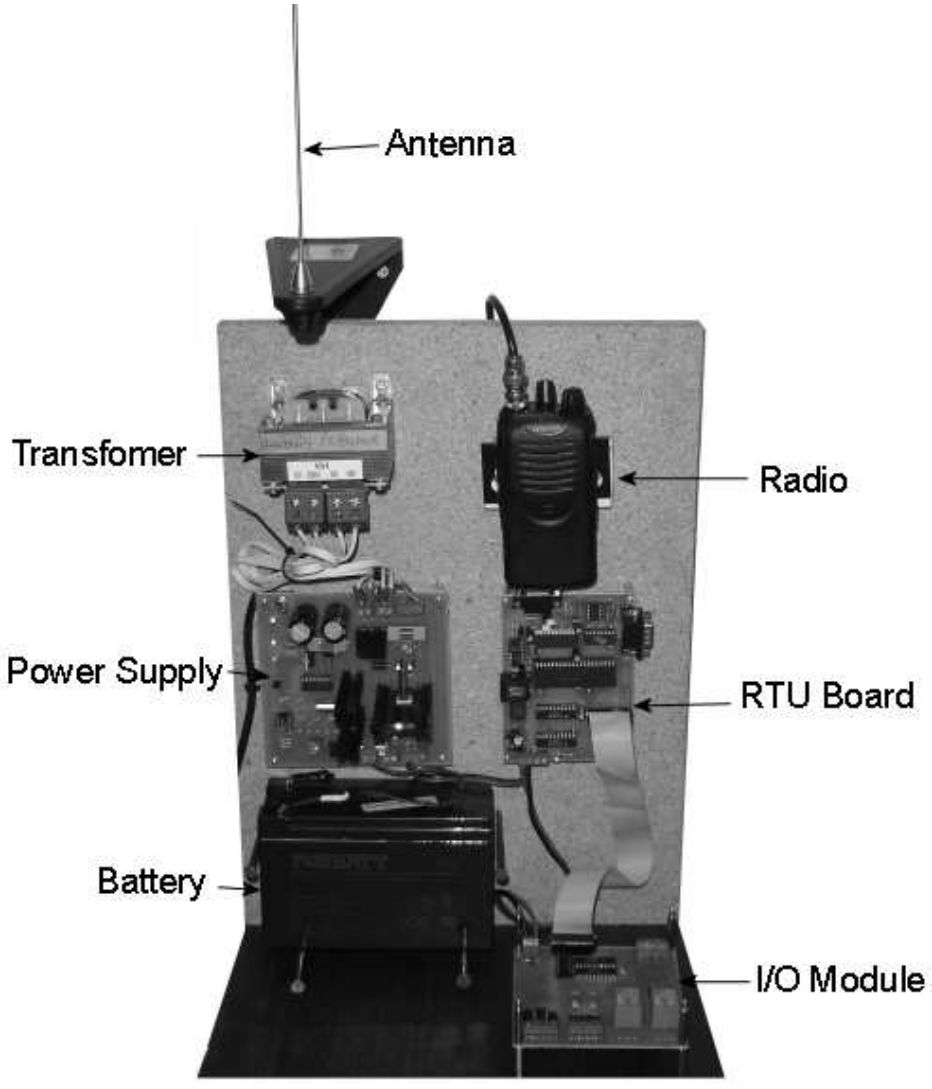


Figure D.1: Complete prototype.

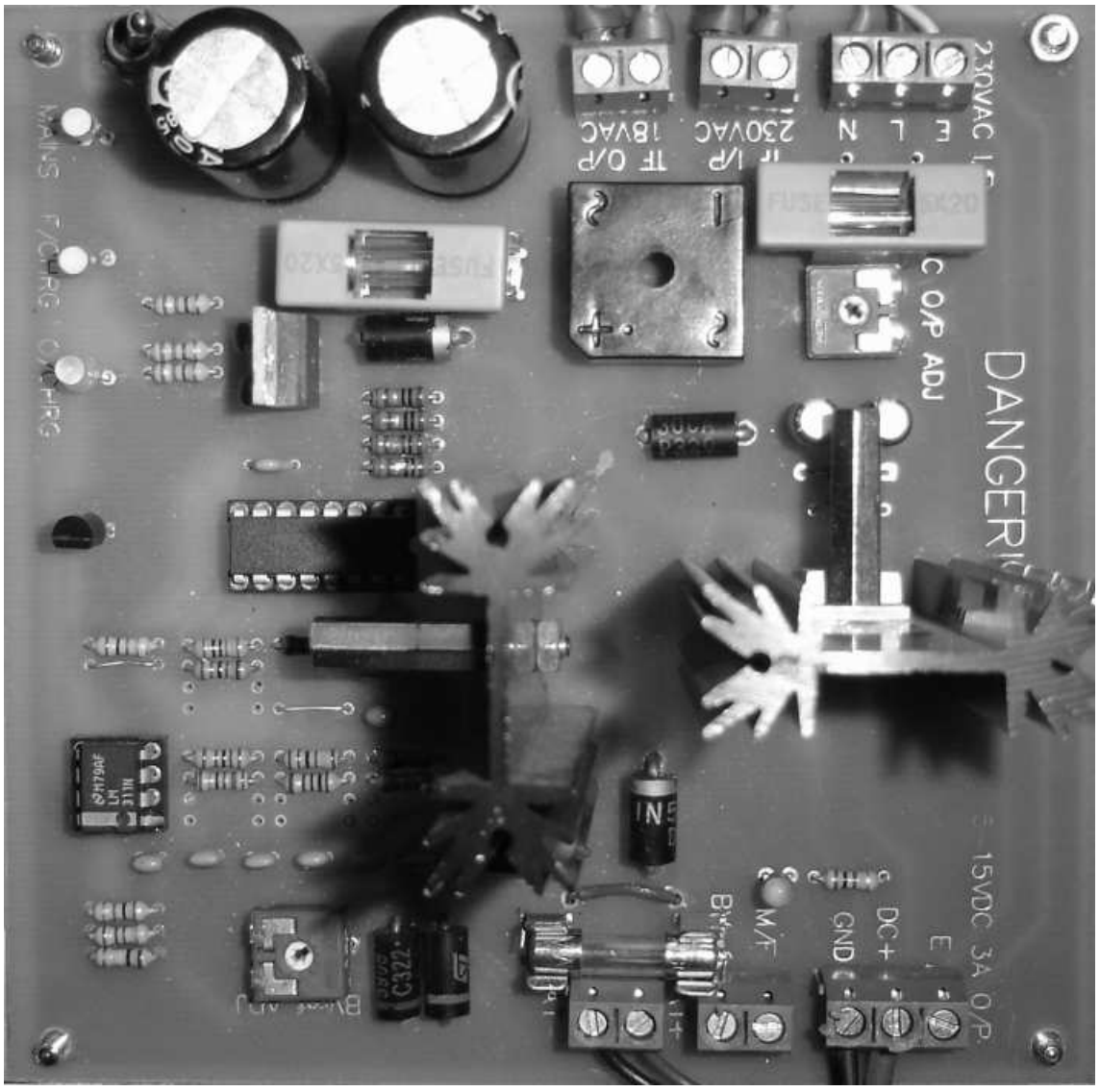


Figure D.2: Power Supply board.

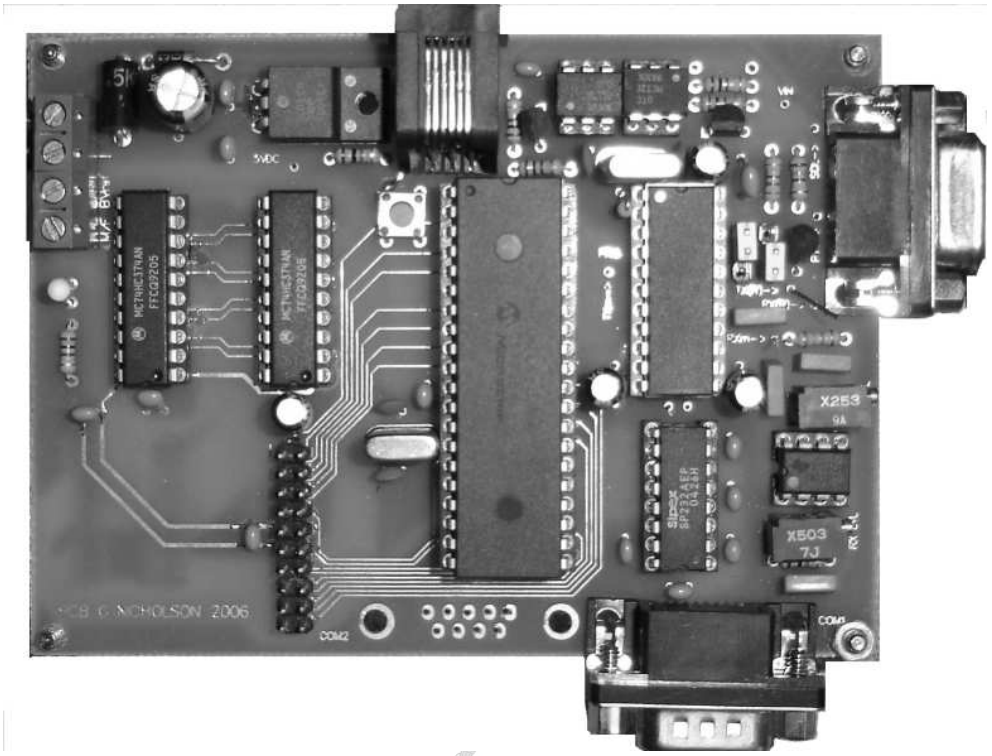


Figure D.3: RTU board.

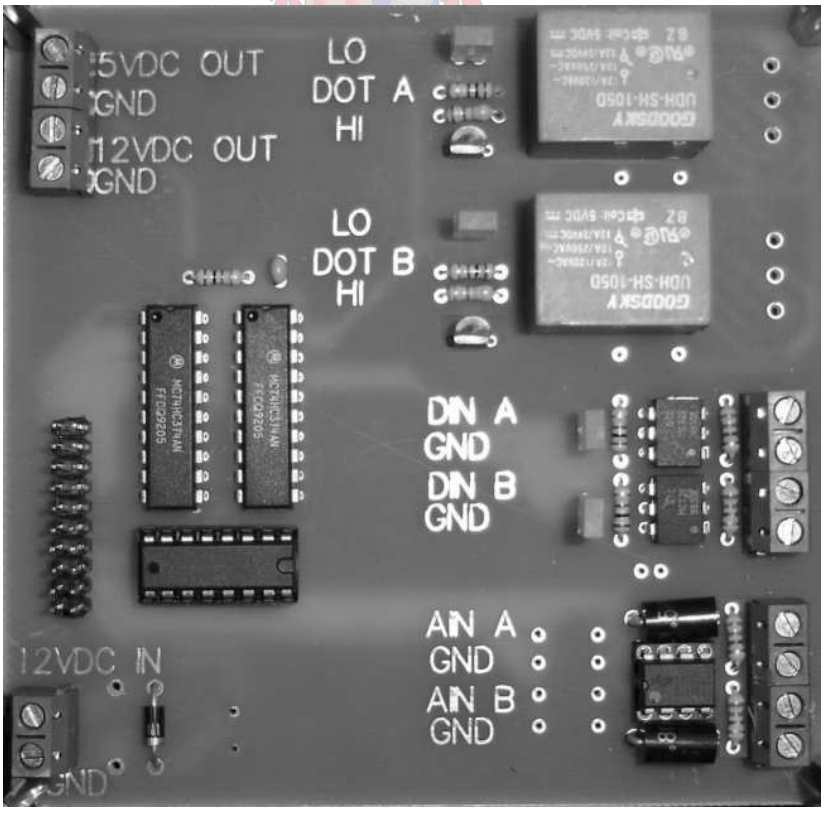


Figure D.4: I/O Module board.

Appendix E

CD-ROM

E.1 Embedded Program Code

A version of the embedded program code as written in the MPLAB IDE for the PIC18F452 is listed.

E.2 Server Application Software

A version of the server application used to interface to the RTU is provided.

E.3 MATLAB Script Files

The MATLAB script files included here present examples of the CSMA state matrix model, and RRP simulation, used to generate the theoretical performance plots contained in this thesis.

E.4 Relevant Datasheets

Datasheets pertaining to the hardware components used in the design have been included.

List of References

Agilent Technologies (2000). Digital modulation in communication systems- an introduction. Available at: `http://www.agilent.com`. Application Note 1298.

Bensky, A. (2000). *Short Range Wirless Communication.* LLH Technology Publishing, Virginia,USA.

Blahut, R. (2003). *Algebraic Codes for Data Transmission.* Cambridge Universtiy Press, Cambridge,UK.

Broch, J., Maltz, D., Johnson, D., Hu, Y. and Jetcheva, J. (1998 Oct). A performance comparison of multi-hop wireless ad hoc network routing protocols. In: *In Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 85–97. Available at `http://www.monarch.cs.cmu.edu`.

CML Microcircuits (2003 Jan). Application note: Wireless modem trouble shooting guide. Available at: `http://www.cmlmicro.com`.

Crabtree, M. (2005). *Mick Crabtree's Industrial Communications Handbook.* 2nd edn. Crown Publishers, Bedfordview,RSA.

Hac, A. (2003). *Mobile Telecommunications Protocols For Data Networks.* John Wiley & Sons, INC., New York,USA.

Heap, N. (1993). *An Intoduction to OSI.* Blackwell Scientific Publications, Oxford, London.

Horowitz, P. and Hill, W. (2002). *The Art of Electronics.* Cambridge Universtiy Press, Cambridge,UK.

Hunter, R. and Kostedt, F. (2000). Using two-point modulation to reduce synthesizer porblems when designing dc-coupled gmsk modulators. Available at: `http://www.cmlmicro.com`.

IEEE Computer Society LAN MAN Standards Committee (1997). *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std 802.11.* The Institute of Electrical and Electronics Engineers, New York, USA.

Johnson, D., Broch, J. and Maltz, D. (2000 Nov). The dynamic source rotuing protocol for mobile ad hoc networks. Internet Draft, draft-ieft-manet-dsr-00.txt. Work in progress.

Johnson, D. and Maltz, D. (1996). Dynamic source routing in ad hoc wireless networks. In: Imielinski, T. and Korth, H. (eds.), *Mobile Computing*. Kluwer Academic Publishers, Boston MA,USA.

Kostedt, F. and Kemerling, J. (1998). Application note: Practical GMSK data transmission. Available at: `http://www.cmlmicro.com`.

Langton, C. (2002). An intuitive guide to principles of communication. Available at: `http://www.complextoreal.com`.

Lee, E. and Messerschmitt, G. (1988). *Digital Communications*. Kluwer Academic Publishers, Boston MA,USA.

Ling, S. and Chaoping, X. (2004). *Coding Theory, A First Course*. Cambridge Universtiy Press, Cambridge,UK.

Miller, G. (1988). *Modern Electronic Communication*. 3rd edn. Prentice Hall, New Jersey,USA.

Mohan, N. (2003). *Power Electronics: Convertors, Applications and Design*. John Wiley & Sons, INC., New York,USA.

Neamen, D. (2001). *Electronic Circuit Analysis and Design*. 2nd edn. McGraw-Hill, Boston,USA.

Ng, C. (1996). *Queueing Modelling Fundamentals*. John Wiley & Sons, INC., New York,USA.

Pozar, D.M. (2001). *Microwave and RF Design of Wireless Systems*. John Wiley & Sons, INC., New York,USA.

Raaphorst, S. (2003 May). Reed-muller codes. Available at: `http://www.sebandthecity.com/academic/mat5127paper.pdf`. Carelton University.

Wolhuter, R. (2002). *The Determining of Optimum Protocol Strategies for Half-Duplex Telemetry Communication Links*. Ph.D. thesis, Electrical and Electronic Engineering, Stellenbosch University, Stellenbosch, South Africa.

Wolhuter, R. (2005). Advanced telecommuncations. Course Notes (Advanced Telecommuncations 823), Stellenbosch University.

Wood, A. (2004 Feb*a*). Channel spacing and occupied bandwidth. Application Note:7000 0019.

Wood, A. (2004 Feb*b*). A summary of modem techniuqes. Application Note:7000 0018.

Ziemer, R. and Tranter, W. (2002). *Principles of Communications, Systems Modulation and Noise*. 5th edn. John Wiley & Sons, INC., New York,USA.