Electricity Theft Detection on a Low Voltage Reticulation Environment

Thesis presented in partial fulfillment of the requirements for the degree of
Master of Science in Electrical and Electronic Engineering
at the
University of Stellenbosch

RIAAN (W.A.) DOORDUIN

Supervisor: Prof. H. dT. Mouton

December, 2004

Declaration

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

.....
Riaan (W.A.) Doorduin
November 23, 2004

Abstract

Electricity theft in South Africa has become a major problem. This led to several developments from both industries and research institutes to counter these actions. Since equipment is already installed and major capital has been invested to provide electricity for a broad spectrum of consumers, the challenge is to find a low cost solution harnessing current investments and technology to detect electricity theft more accurately.

This thesis investigates into the electricity theft topic. Two different methods, Time Domain Pulse Reflectometry and a data driven platform based on the Theory of Constraints philosophy, were investigated to provide means to detect and determine the impact of illegal electricity usage. Both methods required detailed designs to conduct preliminary proof of concept tests in a laboratory environment.

These methods are evaluated against their economical viability, possible practical implications and applications. This thesis presents a practical approach to electricity theft detection and provides the basic tools for management of this ever-increasing problem.

Opsomming

Suid Afrika se elektrisiteit diefstal statistiek het die afgelope jare skrikwekkend gegroei. Dit het die industrie genoop om baie meer navorsing in die area te doen. Met reeds gevestigde toerusting en tegnologie om dié energie medium so effektief moontlik te versprei, is die uitdaging juis om ’n ekonomiese oplossing te vind om reeds beskikbare tegnologieë meer doeltreffend aan te wend.

Die doel van die tesis is om die gebied van elektrisiteit diefstal na te vors. Twee verskillende metodes is ondersoek, naamlik Tydgebied-pulse-reflektometrie en ’n informasie gebaseerde stelsel wat op die Randvoorwaarde Teorie gebaseer is, om effektief die omvang van elektrisiteit diefstal in ’n mikro, asook makro omgewing te bepaal. Die twee metodes is in ’n beheerde omgewing getoets sodat die konsepte wat ontwikkel is bewys kon word.

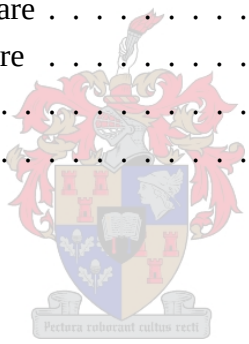
Die metodes is ge-evalueer in terme van die ekonomiese lewensvatbaarheid daarvan met inagneming van die praktiese implikasies. Die tesis bied bestuur die nodige kennis om elektrisiteit diefstal in die praktyk doeltreffend die hok mee te slaan.

Acknowledgements

Thank you to all who helped and encouraged me through my studies:

- My heavenly Father for His providence.
- My family for their encouragement.
- Prof. H. dT. Mouton for his guidance.
- Dr. H.J. Beukes for his insights and contributions.
- Dr. and Miss. D.E. Steinmann for the splendid time I had during my two year stay in Stellenbosch.
- ESKOM, the NRF and Thrip for providing funds for the project.
- Stellenbosch Municipality for providing test facilities and allowing the use of their data.
- Actaris for providing the data concerning the Mooiwater test site linked to the Stellenbosch Municipality.
- Prof. R. Herman and Mr. L. Swart for assistance with the site analysis.

Contents



List of Figures

List of Abreviations

ABC	Aerial Bundled Conductor
API	Application Program Interface
AC	Alternating Current
CT	Current Transformer
DC	Direct Current
DM	Data Mining
DSP	Digital Signal Processor
ET	Electricity Theft
GUI	Graphical User Interface
GSM	Global System for Mobile Communications
HHT	Hand Held Terminal
HV	High Voltage, e.g. $\pm 100\ kV$ to $400\ kV$
IDE	Integrated Desktop Environment
ISM	Industrial, Scientific and Measurement
LSB	Least Significant Byte
LV	Low Voltage, e.g. $\pm 230\ V_{LN}$
MCU	Micro Controller Unit
MIDP	Mobile Independent Device Profile
MOSFET	Metal-Oxide Semiconductor Field-Effect Transistor
MOV	Metal-Oxide Varistor
MV	Medium Voltage, e.g. $\pm 11\ kV$
PCB	Printed Circuit Board
PDA	Personal Digital Assistant
PDF	Probability Density Function
RMS	Root Mean Square
SCADA	Supervisory Control And Data Acquisition
SMS	Short Message Server
SPI	Serial Peripheral Interface
SQL	Structured Query Language

TDR	Time Domain Pulse Reflectometry
TOC	Theory of Constraints
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver-Transmitter
XTAL	Crystal Oscillator

List of Symbols

C_{DS}	Drain Source Capacitanse
f_{sample}	Sample Frequency
$I_{D_{max}}$	Continuous drain current
$i_{network}$	Network current
I_{ϕ}	Phase current
i_s	Source current
i_{Switch}	Switch current
$\Im(Z)$	Imaginary part of complex impedance
$L_{network}$	Network inductance
L_s	Source inductance
P_{tot}	Power dissipation
$R_{DS(on)}$	Drain source on resistance
$t_{d(off)}$	Turn-off delay time
$t_{d(on)}$	Turn-on delay time
t_f	Fall time
t_r	Rise time
$V_{DSS_{max}}$	Drain Source breakdown voltage
V_{GS}	Gate-Source voltage
V_{LL}	Line-to-Line voltage
V_{LN}	Line-to-Neutral voltage
v_s	Source voltage
Z_{Load}	Load modeled as an impedance

Chapter 1

Introduction

1.1 Background

Although various figures are available, the non-technical losses for Eskom are estimated at almost R250 million per annum [60] (2003). Compared to Eskom's annual profit of R2561 million in 2001 this can be estimated at roughly 10% of their annual profit. Electricity theft does not only concern losses in monetary terms, but poses an extremely dangerous health and safety risk [75].

Although electricity theft is mostly concentrated in urban areas and on bulk users (see Appendix J.3), Eskom was introduced to a new market with the introduction of prepaid systems and the electrification initiative in the early 1990s (see Appendix G.6). It can be deduced that more consumers had unmonitored access to electricity and hence the increase in non-technical losses.

1.2 Electricity Theft Defined

During the course of the study it was found that, in general, electricity theft is confused with cable theft (see Appendix F.3.1). This prompted the definition of electricity theft to be clearly defined before addressing it in the thesis.

The energy usage in an electricity reticulation system can consist of the energy used by consumers, technical losses and non-technical losses.

The energy consumed by the users is measured by the electricity distributor at a consumer measuring point, such as a conventional or a prepaid meter. The utility is responsible for delivering quality power up to that measuring point within strict specifications [67, 68].

The technical losses consist of transmission and transformer inefficiencies in the transmission system to the consumers in the transmission system [33].

Non-technical losses are best defined as [11]:

> ...non-technical is a euphemism encompassing poor management or human fail-

ing related reasons such as supplies that haven't been metered, meters that measure inaccurately or don't work properly, meters that are tampered with, bad or fraudulent accounting by vendors, illegal connections to the distribution system and other similar circumstances.

Electricity theft is then defined as the non-technical losses, encompassing both criminal activity and non-criminal actions and circumstances, in a reticulation system.

1.3 Scope of the Thesis

The goal of this thesis is to investigate methods of detecting electricity theft in Low Voltage (LV) networks. Only the case from the LV transformer up to and including the meter measurements will be included. The processing and back office will not be considered or discussed, for example revenue losses due to inadequate administration. The focus will be on the physical tapping of electricity and tampering with metering equipment.

From [15] only the following non-technical losses will be considered:

- Unauthorized line tapping and diversion
- Losses due to faulty meters and equipment
- Inadequate or faulty metering
- Inadequacies of meter reading
- Inaccurate estimation of non-metered supplies, for example public lighting, agricultural consumption and rail traction.

Two methods were suggested for study, namely Time Domain Pulse Reflectometry (TDR) and a Data Driven Solution based on the Theory of Constraints [36].

1.4 Methodology

To gain an understanding of the LV networks, Time Domain Pulse Reflectometry (TDR) was studied and evaluated. This approach looks into the unauthorized line tapping mentioned in the section above.

Goldratt's Theory of Constraints [36] is used as a tool to address the Check Meter and Data Mining study of the thesis, which encompasses the rest of the points mentioned in the previous section. Although designed for production processes it can be applied to electricity theft in the sense that it optimizes the detection process. Using all the available resources the problem area is identified and addressed in a cost-effective manner.

Each of the studied methods is evaluated according to the following criteria:

Capital Layout Cost The total cost of the system is evaluated.

Running Cost How expensive it will be to run the system, especially in terms of man-power.

Accuracy To what extent can the solution determine electricity theft and under what conditions are these results expected.

1.5 Outline of Thesis

The thesis consists of four different parts.

The first part, of which this chapter forms part, consists of some background and an introduction to the topic of electricity theft. Chapter 2 discusses some additional issues regarding electricity theft and provides the required statistics.

The second part, Chapter 3, discusses Time Domain Pulse Reflectometry as an electricity theft detection method.

The third part is built on the Theory of Constraints, discussed in Chapter 4, and is held together using the Data Mining package of Chapter 5. Two data capturing methods are investigated. Chapter 6 looks at the design and implementation of a Mobile Remote Check Meter and Chapter 7 developed software for deployment in a hand held device to capture meter readings. The simulations and results of the integrated data driven solution are documented in Chapter 8.

The thesis is finally summarized and concluded with Chapter 9, providing an overview of the work done and discusses possible future research opportunities.

Chapter 2

Background and Insights into Electricity Theft

> The reasons for fraud are complex and revolve undeniably around the prevalent social patterns. The question facing utilities is to determine cost justifiable strategies to overcome the losses. [70]

2.1 Introduction

Electricity theft directly affects the bottom-line of a utility and is directly associated with revenue protection (see Appendix G). The outstanding debt due to non-payment runs into billions per area, together with the millions of rands lost per year to electricity theft. This chapter is an overview of the current electricity theft situation in South Africa in terms of available statistics, the law, forms of electricity theft and provides a look at current solutions for addressing electricity theft.

2.2 Background and Statistics

Electricity theft has grown rapidly in the last ten to fifteen years since the introduction of prepaid meters (see Appendix G) and the rural electrification drive initiated in the early 1990s [25], whereby from 1994 to 1999 roughly 300 000 new connections were added to the national grid by Eskom.

Although electricity theft occurs in such small amounts relative to the total energy consumption of the utilities, the problem must be addressed firstly, to promote safety and secondly, to avoid any unreasonable prices. Combined with the non-payment levels, for example in Soweto which in 1995 had a $51\ \%$ payment level, the cost for faithful paying consumers increased, which certainly is unfair [19]. Consumers in Philippi, Cape Town, are reported to be removing these illegal connections in order to prevent this effect (see Fig. 2.1).

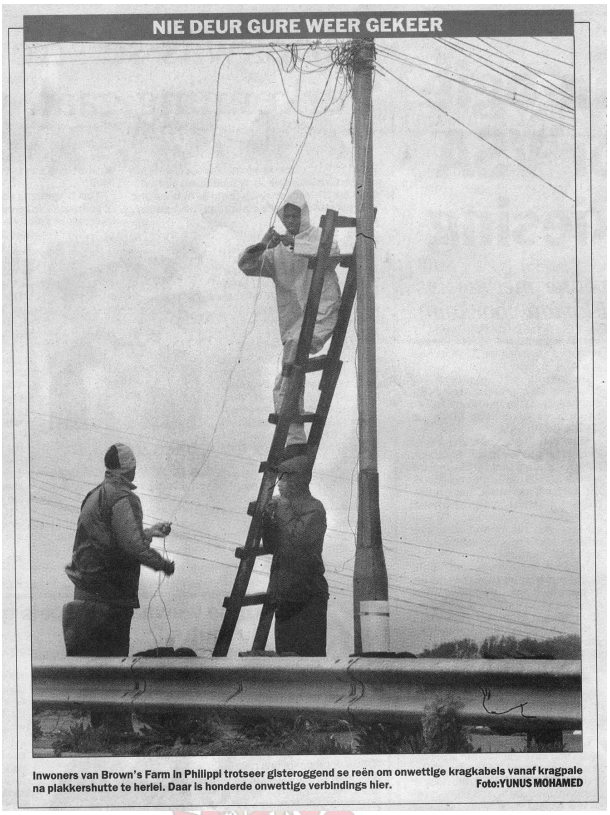


Figure 2.1: *Removal of illegal connections [55]*

As reported by Shingai [70] the problem in South Africa is a social one. As electricity is supplied to people from previously disadvantaged groups the assumption is that electricity should, like roads and health services, be provided free of charge. The local government is in the process of phasing in a $50\ kWh$ basic electricity grant [53, 17] to provide relief for the poor.

Roughly $38\ \%$ of electricity in South Africa is supplied to households by municipalities [24].

Openshaw reported that electricity theft cost Eskom more than R250 million per year in South Africa [60].

Shingai reports the following statistics [70]:

- The United States of America reports a $4\ \%$ distribution loss on total revenue.
- The United Kingdom reports revenue losses of £250 million per year.
- Pakistan's 30 transmission feeder's losses exceed $30\ \%$.
- In certain areas in South Africa up to $38\ \%$ non-technical losses are reported.
- Non-technical losses in India amount to $10\ \%$ per year.

Chapman [13] reports $36\ \%$ non-technical losses in Molofo, Soweto.

Region	R'mil	%
Northern	0.45	3
North-Eastern	0.5	4
Central	7.6	58
Eastern	1.7	13
North-Western	1.5	11
Southern	1.1	8
Western	0.3	2

Table 2.1: *Breakdown of revenue loss in 1998 [11]*

Municipality	2001	2002	2003
Stellenbosch	2.60 %	2.05 %	0.98 %
Matzikama	7.95 %	16.55 %	

Table 2.2: *Breakdown of electricity losses in distribution network*

In 1998 the non-technical loss statistics broken down into the various provinces are shown in Table 2.1 [11], and do not nearly add up to the R250 million mentioned earlier [60].

Information from the annual report of local municipalities was compared in Table 2.2. Stellenbosch and Khayelitsha (see Section 2.5.2) show that losses can be reduced to at least $2\ \%$.

2.3 The South African Law

2.3.1 Theft

According to the South African law, Law 41 of 1987 article 27 (2-3), states that if a person is found guilty of tapping electricity they will be liable to experience the penalties indicated in the following extract:

> (2) Iemand wat sonder 'n wettige reg (waarvan die bewyslas op hom rus) elektriese stroom uitneem, aftak of uitkeer of laat uitneem, aftak of uitkeer of sodanige stroom wat wederregtelik uitgeneem, afgetak of uitgekeer is, verbruik of gebruik, wetende dat dit wederregtelik uitgeneem, afgetak of uitgekeer is, is aan 'n misdryf skuldig en by skuldigbevinding strafbaar met die strawwe wat vir diefstal opgele kan word.
>
> (3) Iemand wat sonder 'n wettige reg (waarvan die bewyslas op hom rus) 'n apparaat vir die ontwikkeling, oorstuur of voorsiening van die elektrisiteit afsny of beskadig of hom daarmee bemoei, is aan 'n misdryf skuldig en by skuldigbevinding strafbaar met 'n boete van hoogstens die bedrag wat van tyd tot tyd by regulasie bepaal of met gevangenisstraf vir 'n tydperk van hoogstens 12 maande of met

daardie boete sowel as daardie gevangenisstraf. [73]

Eskom uses a form as seen in Appendix A.1 to give a written notice to users with unlawful connections as currently used by Andre Truter (see Appendix F.4). Without this completed document no legal action can be taken against the perpetrator.

Although this seems fairly straightforward, a judgment in the Cape High Court stands [50]:

Theft of electricity is not possible as it is not tangible.

2.3.2 Safety

Another important law and issue regarding electricity theft concerns the safety risk introduced due to illegal connections. The utilities and distributors are therefore in terms of the Occupational Health and Safety Act forced to remove any dangerous and illegal connections [75]. In eThekwini Electicity, Durban, 14 deaths have been reported, of which more than half are under the age of 10 [12], due to illegal connections.

When technicians try to remove illegal connections the, communities threaten the lives of the personnel removing illegal connections [11]. This clearly poses a problem.

2.4 Forms of non-technical losses

Although many methods exist to obtain illegal electricity, this section mentions a few obvious methods detected during this study. As an indication of the vast range of possibilities available and the ingenuity of the perpetrators a list from [45] with more methods, has been included in Appendix A.2.

1. Obvious tapping of electricity from a nearby electricity point, Fig. 2.1, Phillipi, Caledon.
2. Posing an extreme safety hazard is the distribution of electricity to individuals from another premises, Fig. 2.2, Grabouw.
3. Tampering with conventional magnetic disc meters, Fig. 2.3, Caledon, for example forcing the disc to rotate slower with a pin.

2.5 Available solutions and implementations

2.5.1 Sweeps

The simplest way to combat electricity theft is by using sweep teams to audit meters and check for illegal connections. Depending on the community this is sometimes extremely dangerous for the personnel [11].

Figure 2.2: *Secondary Distribution in Grabouw*

Figure 2.3: *Tampering with Disc at Caledon*

Depending on the resources available within the local distribution authority the auditing function can be outsourced to private companies.

2.5.2 Case Studies

Ekurhuleni

The subsection is based on a report from Jamieson [44, 43] as well as documentation received during a visit to Ekurhulni in August 2003 (See Appendix I).

History In 1995 the Tembisa reticulation network became the responsibility of Kempton Electricity. With the responsibility Kempton inherited 31 500 consumers and a bad debt exceeding $R\,250$ million and with an additional $R\,3$ million per month unpaid usage due to a non-payment campaign within the community. This, with a neglected network due to the ”self-help” culture required intense maintenance to comply with safety regulations. A plan was approved to upgrade the network and to add an automatic meter reading SCADA system which exceeded $R\,100$ million in cost.

Community buy in Due to the political instability at the time, the time being just after the first election in 1994, the municipality’s new government expected to get paid for services delivered. It was also a high priority to ensure the community’s involvement in the project. A survey was done to develop the market strategy and a marketing campaign was launched to help initiate the new metering system.

Metering implementation The first meters were installed in November 1996. Due to further unrest in the area the need to protect the meters with $10mm$ thick steel protective structures was required, costing $R\,24$ million.

Results With all structures and meters in place credit control was implemented and payment levels rose from $R\,0.9$ million to $R\,8.9$ million per month.

Khayelitsha

The progress seen in Khayelitsha is phenomenal. Within six years the 70% losses are down to 1.5% and this area boasts the lowest non-technical losses in the country [11]. This progress can be attributed to excellent management of the network.

Cape Town

In Keith Strober’s paper, Cape Town [80] sets out his practical approach to revenue protection. The following are a few main points from his report:

- He assigned a team of whom three members were master electricians. This enabled the team to enter into all possible installations.
- His approach is not to disconnect, but to encourage users to remain connected and pay the outstanding debt at an additional rate of $14\ \%$ added to their electricity purchases.
- In order to find perpetrators a printout of all the consumers who have not purchased electricity is used.
- $76\ \%$ of the tampering is found in low-cost areas.
- From the $10\ 000$ annual inspections done only 15% are apprehended.
- By approaching electricity theft in an understanding manner his efforts recovers nearly $R\ 3$ million per year.

2.5.3 DC Reticulation

Since electricity for domestic purposes is provided at $230\ V$ AC it is easy to tap from the distribution company's lines.

Hence, Molepo [57] proposed and developed a solution whereby the electricity is distributed using $750\ V$ to $1\ 000\ V$ DC. At the consumer's connection, a high voltage DC to $230\ V$ AC inverter is installed to allow the power to be used by the consumer. The proposed system can be implemented on the existing aerial bundled conductors used for rural electrification.

The solution's cost is estimated to be under $R\ 1\ 000$ per unit or consumer. The solution is almost fail-safe and would definitely reduce tampering due to the extremely high risk of electrocution when attempting to tamper [34]. This also indicates the one negative aspect of this solution: In the event of equipment failure innocent people may also be electrocuted.

2.6 Conclusion

From the contradictory statistics, figures and background it can be seen that no conclusion about the extent of electricity theft in South Africa can be made. However, the fact that publications were presented and figures calculated by authors indicates that electricity theft in South Africa is of a growing concern. Due to these efforts successes such as Ekurhuleni and Khayelitsha are recorded.

Chapter 3

Time Domain Pulse Reflectometry

3.1 Introduction

One problem with electricity theft is the problem of identifying an individual perpetrator. This chapter investigates the possibility of extracting information about the distribution network using Transmission Line Theory and Time Domain Pulse-Reflectometry (TDR) to identify perpetrators.

Reflections caused by loads and mismatched cables[1] are exploited and studied. By determining the time it takes for a signal to travel to and from a mismatched load, the distance to the load can be determined. The magnitude of the reflections can give an idea of the load situation at each reflection in order to detect possible faults and tampering. The problem with using TDR on a LV reticulation network is that uninterrupted power, meaning $230\ V/50\ Hz$ AC signal, must be delivered to the consumer at all times, while a sharp pulse is sent on the network and images are generated.

This chapter investigates the practical aspects of using TDR as a way of detecting electricity theft.

3.2 TDR background

TDR is a technique used to exploit reflections caused in transmission lines by mismatched loads or imperfections. A pulse propagating in a transmission line at a definite speed [39] is reflected back and forth by discontinuities. Pulses reflected back to the measuring point is used to build an image of the network and transmission line. The discontinuities are caused by mismatched cables, connections and loads due to their characteristic impedances. The propagation equations (3.1,3.2), reflection- (3.3) and transmission (3.4) coefficients shown,

$$V_0^- = V_0^+\Gamma \tag{3.1}$$

[1]It is assumed that cables and loads in TDR have roughly the same characteristics

$$V_r^+ = V_0^+ T \tag{3.2}$$
$$\Gamma = \frac{Z_r - Z_0}{Z_r + Z_0} \tag{3.3}$$
$$T = 1 + \Gamma \tag{3.4}$$

with,

$V^{+,-}$ the amplitude of the pulse propagating in the positive or negative direction,

x the distance from the connection of the two transmission lines,

Z the respective surge impedance of the transmission lines,

Γ the reflection coefficient and

T the transmission coefficient.

are illustrated in Fig. 3.1, where the two transmission lines are denoted by using subscripts $0, r$. These equations are derived from the telegraphist equation [46, 65, 39, 35]. Further explanations and derivations are given in Appendix B.

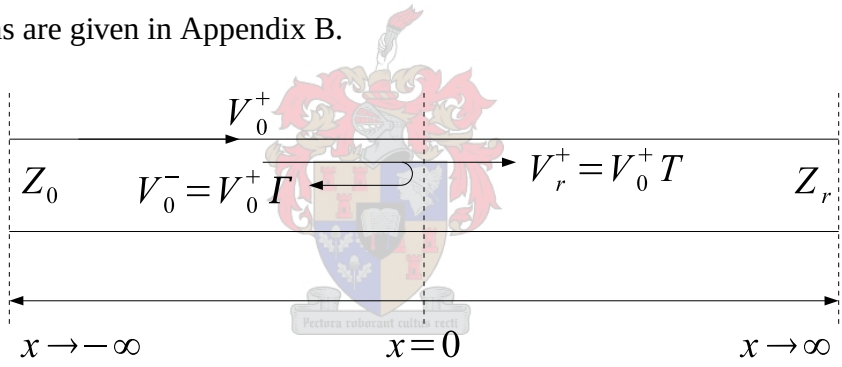


Figure 3.1: *Reflection and Transmission Coefficients of a Transmission Line Configuration*

3.3 Pulse Generator Topology

As mentioned earlier in this chapter the power supply may not be interrupted, and the power provided to the consumers should be within regulations [67, 68]. Various pulse generators were evaluated, see Appendix D, and the parallel-switch topology shown in Fig. 3.2 was found to be a practical solution.

The idea with this topology is to produce an inverted pulse on the $50Hz$ signal of the power signal as shown in Fig. 3.3 by dissipating energy, instead of injecting energy into the system. The advantage of this topology is that the device can be clipped onto the grid at existing split pole's terminals, without interrupting the supply.

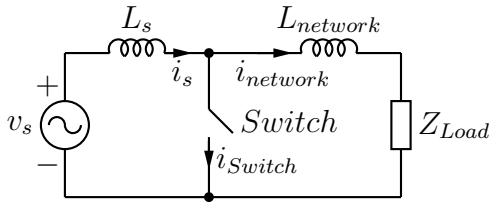


Figure 3.2: *Pulse Generator Topology*

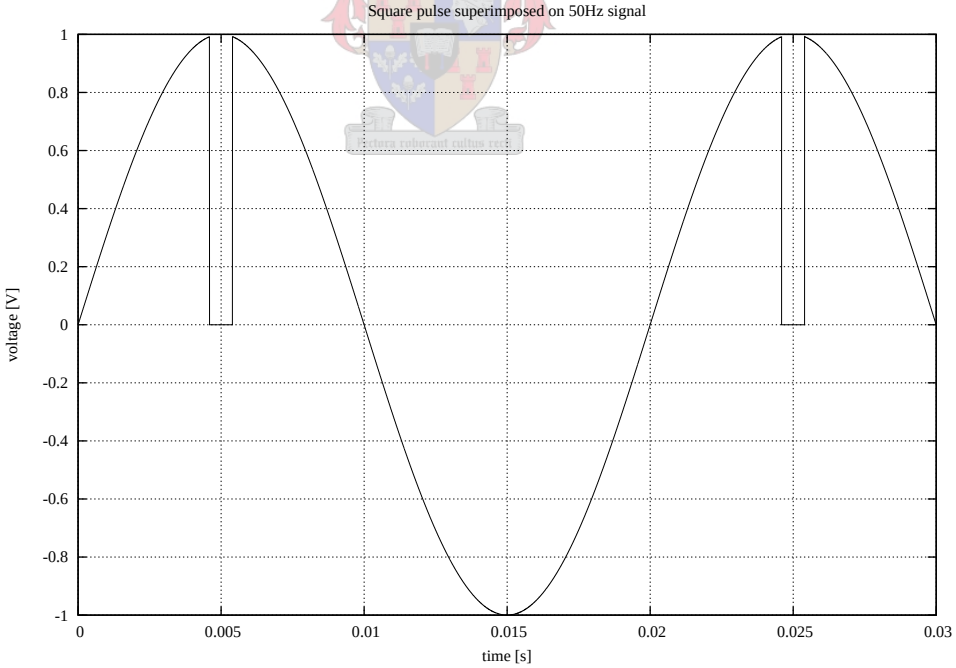


Figure 3.3: *Pulse Generation*

3.4 Design of Prototype Parallel-Switch

The pulse must be characterized by high dV/dt's and a small pulse width to increase resolution of the measurement, while simultaneously reducing harmonic content. A pulse width of between $40\ ns$ to $100\ ns$ was found to be practical. The $100\ ns$ pulse width is small enough compared to the $20\ ms$ period of the power signal and therefore the $50\ Hz$ component can be ignored.

The impact on power quality should be considered and preferably not be affected in any way. This implies that the pulse repetition rate should be kept as low as possible. The device implemented should not disturb the power flow of the current reticulation installation drastically during operation and installation.

3.4.1 System Specification

An Infineon CoolMOS 20N60S5 [41] was selected to perform the function of the main switch. The pulse should be initiated when a signal is received via an optic fiber interface in order to isolate the user and signalling equipment.

The CoolMOS MOSFET from Infineon [41] is considered with the following highlighted specifications:

$$V_{GS} = \pm 20\ V \tag{3.5}$$
$$V_{DSSmax} = 600\ V \tag{3.6}$$
$$I_{D_{max}} = 20A@25\ ^oC \tag{3.7}$$
$$= 13A@100\ ^oC \tag{3.8}$$
$$P_{tot} = 208\ W \tag{3.9}$$
$$R_{DS(on)} = 0.19\ \Omega \tag{3.10}$$
$$t_{d(on)} = 120\ ns \tag{3.11}$$
$$t_r = 25\ ns \tag{3.12}$$
$$t_{d(off)} = 195\ ns \tag{3.13}$$
$$t_f = 45\ ns \tag{3.14}$$

With the time limits on the MOSFETs the minimum pulse width can roughly be calculated as $100\ ns$. A maximum possible pulse width of $t_{pw} = 200\ ns$ is used in our calculations. Relative to the $200\ ns$ pulse the $50\ Hz$ can be considered as DC.

Fig. 3.2 shows the loads and currents flowing in the system. Using Kirchoff's current law $I_s = I_{network} + I_{switch}$. Due to the inductive nature of the transmission lines high $\frac{dI(t)}{dt}$ is not possible. This implies that when the switch is closed the additional current of I_{switch} will start flowing in L_s. When the switch is opened this additional current must be dissipated by the

snubber circuit. It is also clear that the main load current does not pass through the switch. Therefore, for the design of the switch, the load conditions in the circuit is not considered.

The following assumptions are summarized:

- $20\ ms >> 200\ ns$, therefore power source is considered to be DC.
- I_{switch} is absorbed by the snubber circuit.
- L_s is considered to include the transformer inductance and line inductance as shown in Fig. 3.2, where L_s is larger than the minimum inductance specified in (3.16).

3.4.2 Circuit Layout and Design

The total circuit layout is shown in Fig. 3.4. Components were selected from available stock in the laboratory and available samples from Texas Instruments. Due to the small amount of components only a few design considerations were taken. What is important, however, is to determine the specifications of the external circuit and how external components will influence the circuit behavior. The design is split into the following sections:

- The optical interface receives a signal from a computer or signal generator.
- The glitch generator is responsible for the pulse width.
- The driver circuit ensures that the CoolMOS has sufficient gate current to switch on and off.
- The switch circuit is used to generate the $0\ V$ pulse on the line.

Switch Circuit and Snubber Design

The switch and snubber circuit are considered as the main components of the system and will operate within the LV reticulation network as shown in Fig. 3.2.

Consider the network in operation as shown in Fig. 3.2. The current in L_s and in $L_{network}$ will be equal. Inductors cannot have discontinuous current flowing through them. When the switch is on for $40\ ns$ to $200\ ns$ the current in $L_{network}$ can be assumed to be continuous and will draw the required current from L_s for that instance in time. Therefore in the snubber design we need to compensate for the additional energy stored in L_s during the on state when the network is in full operation and when no load or $L_{network}$ is present.

Since L_s can change for each different setup, the minimum L_s should be specified. The device will probably be operated outside and can easily be exposed to maximum sunlight. Therefore, the maximum current at $100\ ^oC$ that may flow through the switch is $I_{max} = 13\ A$ from

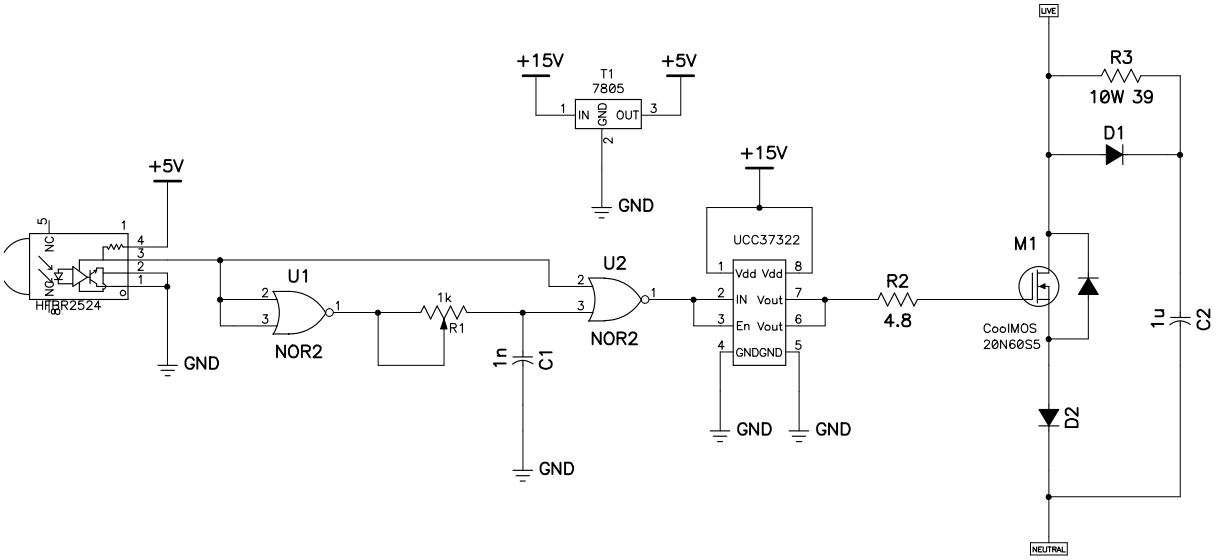


Figure 3.4: *Circuit Diagram of Complete Parallel Switch*

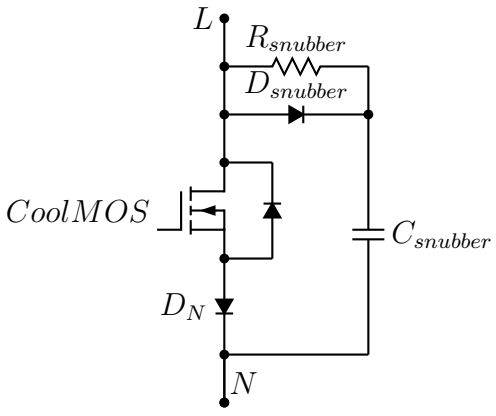


Figure 3.5: *Switch and Snubber Circuit*

(3.8), with a peak voltage of $V_{peak} = \frac{\sqrt{2}}{\sqrt{3}} 400\ V = 327\ V$, $t_{pw} = 200\ ns$ the maximum pulse width, is used.

$$L_s \approx \frac{V_{peak} t_{pw}}{I_{max}} \tag{3.15}$$
$$\approx 5.025\ \mu H \tag{3.16}$$

The device must be able to absorb the total energy stored in L_s to ensure that the proper functioning will occur under no load conditions. The energy stored in the inductor is

$$E_{s(min)} = \frac{1}{2} L_s I_{max}^2 = 424.578\ \mu J \tag{3.17}$$

when the MOSFET is conducting. The over voltage snubber capacitor sinks this energy when the MOSFET is turned off. The voltage over the capacitor is the same as the bus voltage, which can be taken as V_{peak}, just before the MOSFET is turned on again.

$$E_{snubber} = \frac{1}{2} C V^2 \tag{3.18}$$
$$C_{snubber(min)} = \frac{2E_{s(min)}}{V^2} = 7.941\ nF \tag{3.19}$$

A high frequency snubber capacitor with a value of $1\ \mu F$ was available in the laboratory. The energy storage available in this capacitor is

$$E_{snubber} = \frac{1}{2} C V^2 \tag{3.20}$$
$$= 53.465\ mJ \tag{3.21}$$

The problem however with the current configuration is that there is no real way to dissipate the energy stored in the capacitor. From the capacitor a resistor is connected to the live bus to drain the capacitor. The resistor should be chosen so that the capacitor is discharged when the next switching cycle starts. The following approximate identity holds [56] and $R_{snubber} = 39\ \Omega$ is chosen due to availability, therefore

$$t_{empty} > 2.3 C_{snubber} R_{snubber} \tag{3.22}$$
$$> 100\ \mu s \tag{3.23}$$

Considering that the stored energy in the inductor decreases as the inductor size increases, for the configuration, as indicated in (3.26),

$$E_L = \frac{1}{2} V I_{max} t_{pw} \tag{3.24}$$
$$= \frac{V^2 t_{pw}^2}{2L_s} \tag{3.25}$$
$$\Rightarrow \lim_{L \to \infty} \frac{V^2 t_{pw}^2}{2L_s} = 0 \tag{3.26}$$

Cable Inductance The minimum inductor size must not be less than $5.025\ \mu H$. To translate this into a physical cable distance from a source, neglecting source inductance, a $150\ mm^2$ ABC [2] cable is used[2] and

$$X_{L(min)} = 0.082\ \Omega/km\ @\ 50\ Hz \tag{3.27}$$
$$L_{min} = \frac{X_L}{2\pi f} \tag{3.28}$$
$$= 261.014\ nH/m \tag{3.29}$$
$$\ell = 19.252\ m \tag{3.30}$$

Transformer Inductance To calculate the $11\ kV$, $450\ V$ distribution transformer inductance the fault current is estimated at 10 times the transformer rating. With the line to phase voltage the impedance is calculated. Considering that the inductance is approximately ten times higher that the resistance, the inductance is calculated. For example:

$$S_{rated} = 800\ kVA \tag{3.31}$$
$$I_{\phi fault} = 10(I_{rated}) \tag{3.32}$$
$$= 11.594\ kA \tag{3.33}$$
$$Z = \frac{V_{rated_{LN}}}{I_{\phi fault}} \tag{3.34}$$
$$= 19.739i + 1.974\ m\Omega \tag{3.35}$$
$$L_{800\ kVA} = \frac{\Im(Z)}{2\pi f} \tag{3.36}$$
$$= 62.831\ \mu H \tag{3.37}$$
$$L_{100\ kVA} = 252.6\ \mu H \tag{3.38}$$
$$\tag{3.39}$$

The values are calculated to give a general idea of the impedances using the rule of thumb, and should be verified with the data of the actual transformer, prior to testing. The graph in Fig. 3.6 gives an idea how the rating of the transformer influences the inductance.

Driver Circuit

In order to switch with sharp rising and falling edges, which enhances the resolution, it is required to switch the MOSFET as hard as possible. The driver should be capable of providing all the required current for the MOSFET. A driver circuit, the UCC37322 (non-inverting) MOSFET with enable [90] from Texas Instruments was selected. The operating voltage range of

[2]The inductance according to the datasheets decrease as cable size increase. Since the minimum distance is calculated the largest available conductor is used.

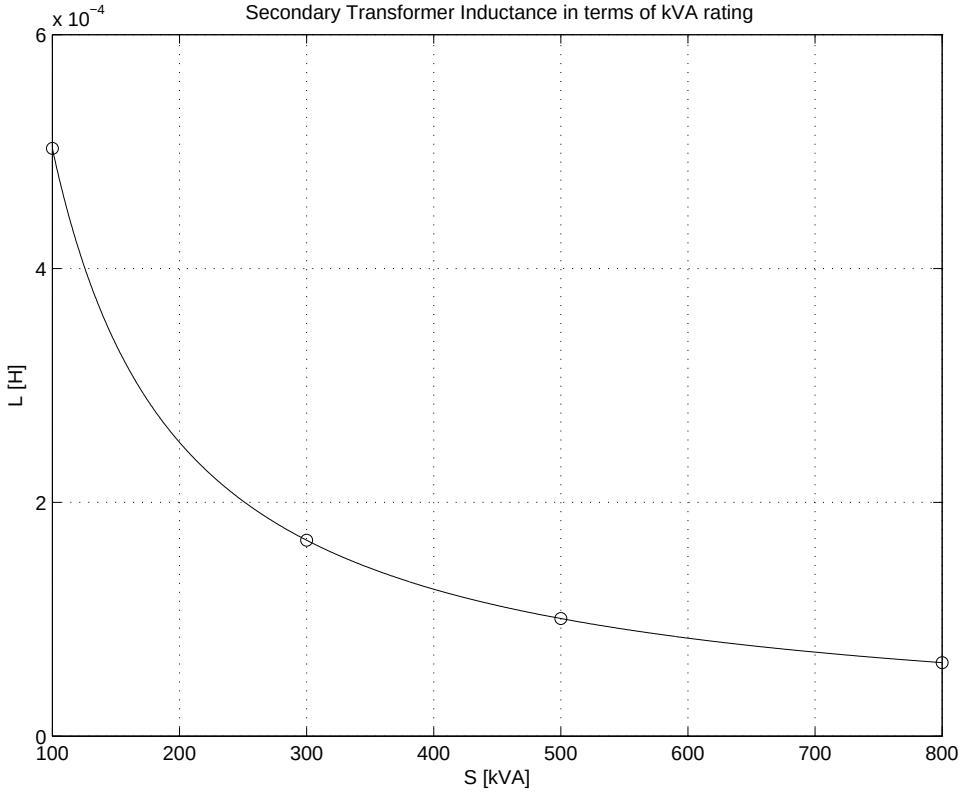


Figure 3.6: *Transformer inductance as a function of kVA rating*

about $15\ V$ can be applied, which is in the operating range for V_{GS}. The chip can sink $9\ A$ and have typical rise times of about $20\ ns$.

The gate resistor must be chosen such that the maximum current requirement for the driver is not exceeded and the time constant of the gate capacitor and gate resistor combination does not exceed the switching times of the MOSFET. The minimum resistance of R_G is calculated:

$$R_G > \frac{V_{on}}{I_{on}} \tag{3.40}$$

$$> \frac{15}{9} \tag{3.41}$$

$$> 1.6\ \Omega \tag{3.42}$$

The shortest switching time of the MOSFET is $t_s = t_{d(on)} + t_r = 145\ ns$ and the input capacitance $C_{iss} = 3\ nF$, according to [41]. The maximum resistance is then calculated to be:

$$R_G < \frac{t_s}{C_{iss}} \tag{3.43}$$

$$< 48\ \Omega \tag{3.44}$$

A value of $R_G = 4.8\ \Omega$ was chosen to drive the MOSFET.

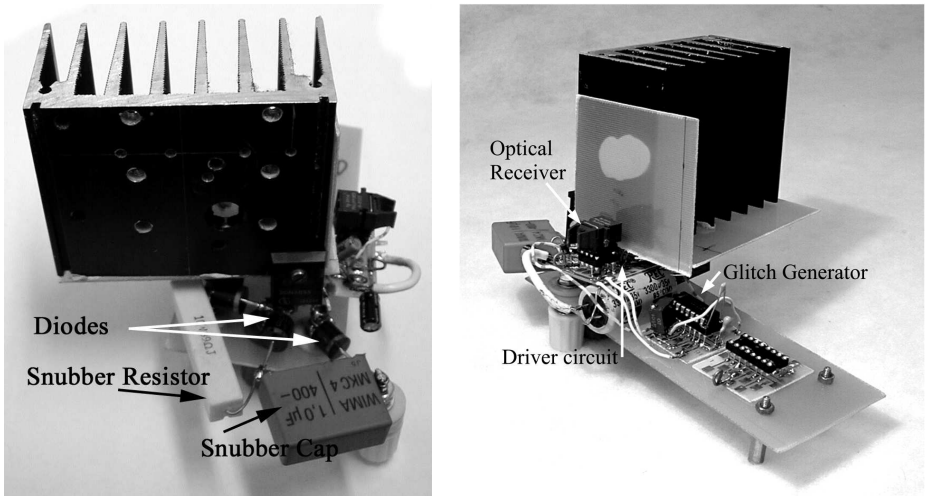


Figure 3.7: *Parallel Switch*

Glitch generator and circuit timing

To generate a pulse with small a enough pulse width, a Schmidt inverter logic circuit was used to construct the glitch generator. The idea is that the NAND logic is driven with two different signal paths, which can be achieved by running the one signal through another NAND gate. The circuit's time constant is increased using a RC circuit with a variable potentiometer, with a maximum of $R = 1\ k\Omega$ and a capacitor with $C = 1\ nF$. This allows the pulse width to be tuned between roughly $8\ ns \leq t_{pw} \leq 1\ \mu s$.

Optical Fibre input

The optical fiber input, HP R-2522, is used to start the pulse generator. An optical signal is received from either a signal generator or computer. This ensures that the computer and signal generator are isolated from the mains power. The optical fiber input feeds the glitch circuit.

3.4.3 Results

The final prototype is shown in Fig. 3.7. The device was tested by connecting it directly to the utility power. A signal was generated using a computer and the optical interface. The pulse width was set to around $80\ ns$. The result is shown in Fig. 3.8.

3.4.4 Practical Implementation Considerations

High Frequency Model

Due to the signals propagating in the transmission lines at high frequencies it is important to know how the pulse generator reacts under these conditions.

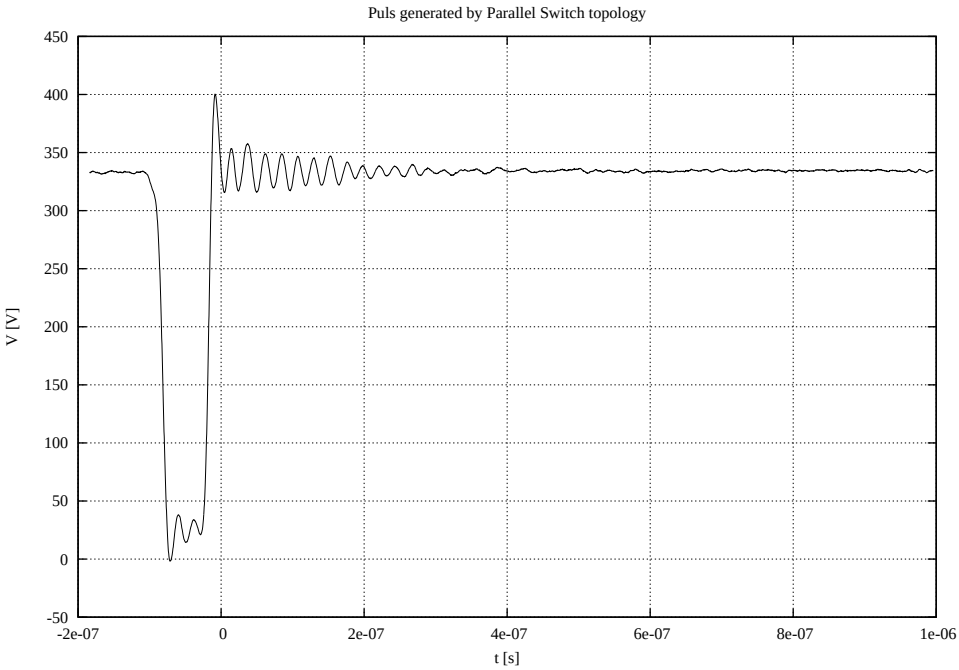


Figure 3.8: *Pulse Generated by Parallel-Switch*

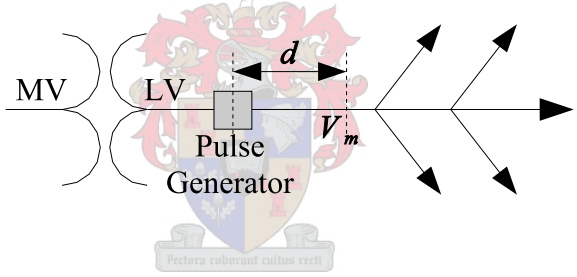


Figure 3.9: *Proposed Pulse Generator Position*

The model used for the parallel-switch is obtained by considering the snubber circuit and switching components. As shown in [58], for a diode it can be seen that the capacitance present acts as a short under HF conditions, which poses a problem. The MOSFET with C_{DS} present can also be modeled roughly as a short as indicated in [58], under high frequency conditions.

System Position within Reticulation network

Fig. 3.9 shows a generalised overview of a low voltage reticulation network. From the high voltage or medium voltage distribution network the transformer brings the voltage to an acceptable voltage, $V_{LL} \approx 400\ V$. From the transformer the power is distributed in a typical tree structure scheme[3]. The placement of the pulse generator and the measurement device must be considered. Due to the fact that the high frequency characteristics of the pulse generator react

[3]See Appendix E

as a short circuit and, therefore, the measurement device must be put some distance from the pulse generator as indicated by V_m and distance d. Placing the pulse generator at the end of the network won't have the desired effect, since the transmission line will have a hyperbolic effect on the pulse as it moves nearer to the transformer [46]. The distance from the transformer must also be considered. Using an inductor L_{min} it is calculated in (3.30) using (3.28) that the minimum length to the transformer should be $\ell = 16.5\ m$.

3.5 Experiments

Experiments were conducted to test the Parallel Switch Pulse Generator and verify the theory in previous sections. The tests are conducted at a low voltage, $V_{peak} = 327\ V$, at the end of the supply line. Therefore, a large inductance L_s, Fig. 3.2, is expected.

Two experiments are presented to illustrate the parallel switch functionality. The experiments are sketched using signal flow graphs [65]. Please refer to Section B.4 in the Appendices for a summary of the Signal Flow Graphs Theory.

3.5.1 Measurement Position

This experiment investigates the positioning of the measuring device relative to the pulse generator.

The pulse generator is connected to the power outlet (V_{50Hz}) at measuring point V_{switch}. The $15\ m$ flex cable[4] ($|A|\, e^{-\gamma\ell}|_{15m}$) extends the transmission line to the second measuring point V_{15m}. A $50\ m$ parallel wire transmission line ($|A|\, e^{-\gamma\ell}|_{50m}$) is added in the second experiment set.

The signal flow graph in Fig. 3.10 shows the expected propagation paths of the pulse through the setup. The open circuit at the end of the $50m$ parallel wire is indicated with the reflection coefficient, $\Gamma \approx 1$. The pulse generator's reflection coefficient is shown by $\Gamma \approx -1$. Although some small reflections occur at V_{15m} these are small enough to be discarded for this experiment's purpose.

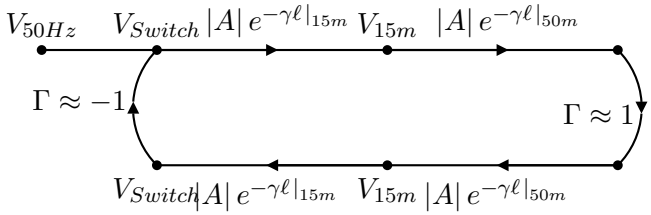


Figure 3.10: *Signal Flow graph for measurements at switch*

[4] 3-core $1.5\ mm^2$

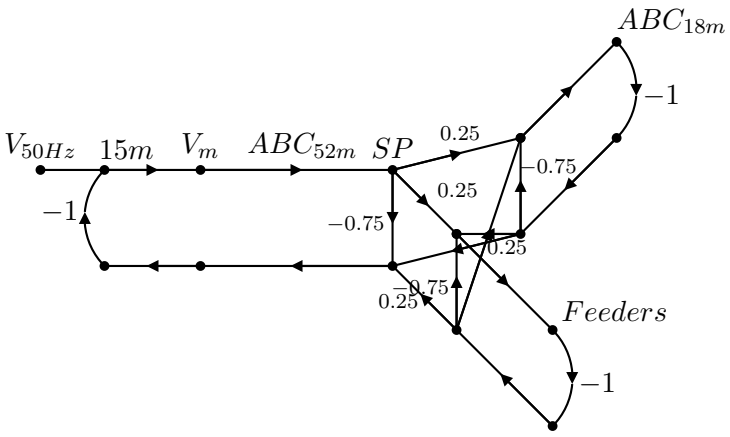


Figure 3.11: *Signal Flow graph for ABC measurements*

3.5.2 ABC and Split poles

For theft detection it is important to measure changes when new connections are made. The setup of this experiment is based on a simple LV network and therefore a slightly different configuration is used than in the previous experiment (Section 3.5.1). This experiment investigates the possibility of TDR to picking up changes in an LV network configuration.

The measurements are all taken $15\ m$ from the pulse generator. This is done to ensure that pulse widths do not overlap when the signal is propagating towards the PG and reflected back. The measurement distance from the pulse generator is calculated using (3.45)

$$\ell_{back\&forth} = \frac{v_c t_{pw}}{2} \approx 15\ m \tag{3.45}$$

with,

ℓ the distance from the pulse generator to the measuring point,

v_c the propagation speed of the pulse in the transmission line and

t_{pw} the pulse width of the initiated pulse.

The $50\ m$ parallel wire is replaced by a $70\ m$ Aerial Bundled Conductor (ABC) which is used for low-cost LV reticulation networks. At $52\ m$ a split pole configuration was added, with 4 feeders with open circuits at approximately the same length. The feeders can be disconnected from the ABC by a circuit breaker. This experiment is shown by the signal flow diagram in Fig. 3.11, which includes the added feeders at $52\ m$ at the split pole.

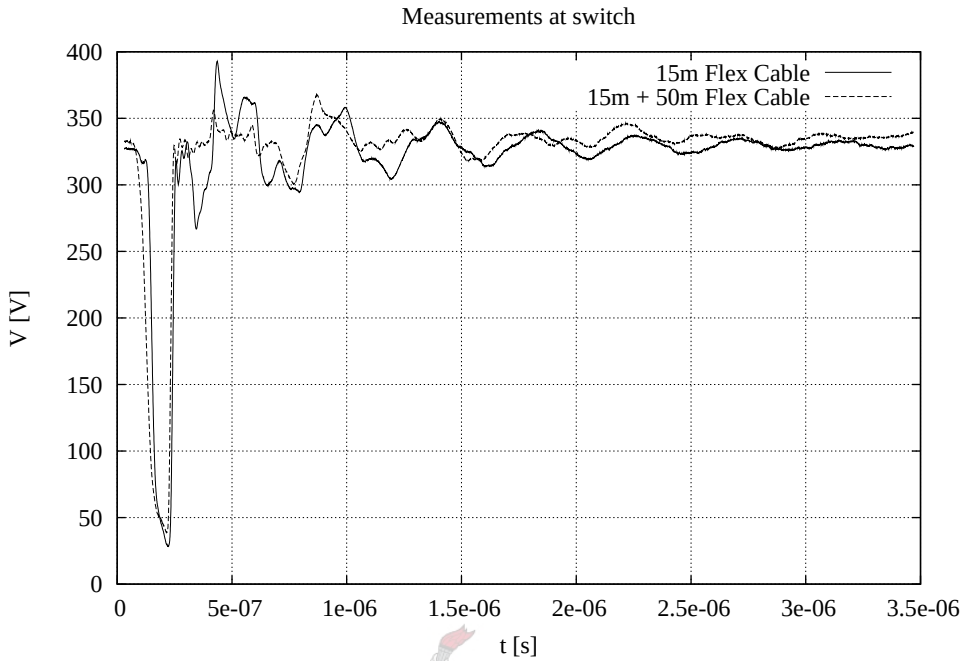


Figure 3.12: *Results from measurements at switch*

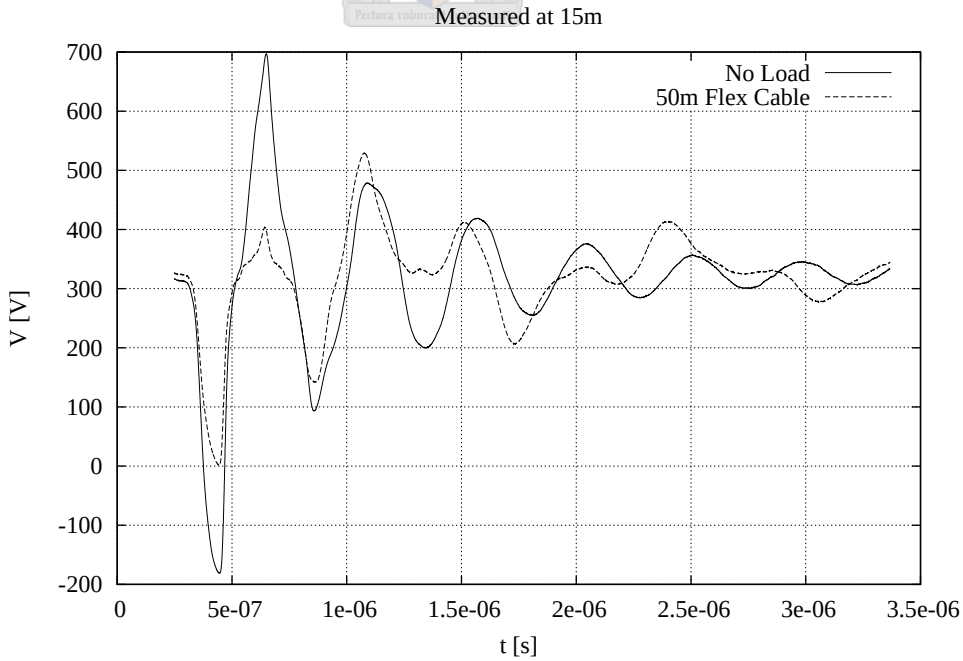


Figure 3.13: *Results from measurements* $15\ m$ *from switch*

3.6 Results

3.6.1 Measurement Position

It can be seen from the results in Fig. 3.12 that no significant signal can be detected when measuring at V_{switch} due to the capacitance of the pulse generator. By moving the switch and comparing the measurements at V_{15m} substantial differences in results can be seen in Fig. 3.13. The cable lengths and setup, in terms of the signal flow diagram, are indicated in the legend of Fig. 3.10.

3.6.2 Aerial Bundled Conductor and Split pole

The results of the aerial bundled conductor (ABC) and split pole experiment are shown in Fig. 3.14. For the measurements with only the ABC connected to the pulse generator the small reflected pulses can be attributed to bends and non-uniform characteristics of the ABC. The second measurement shows the image with the four feeders connected to the ABC configuration. A substantial difference can be seen when the two images are subtracted from each other in the third data set shown.

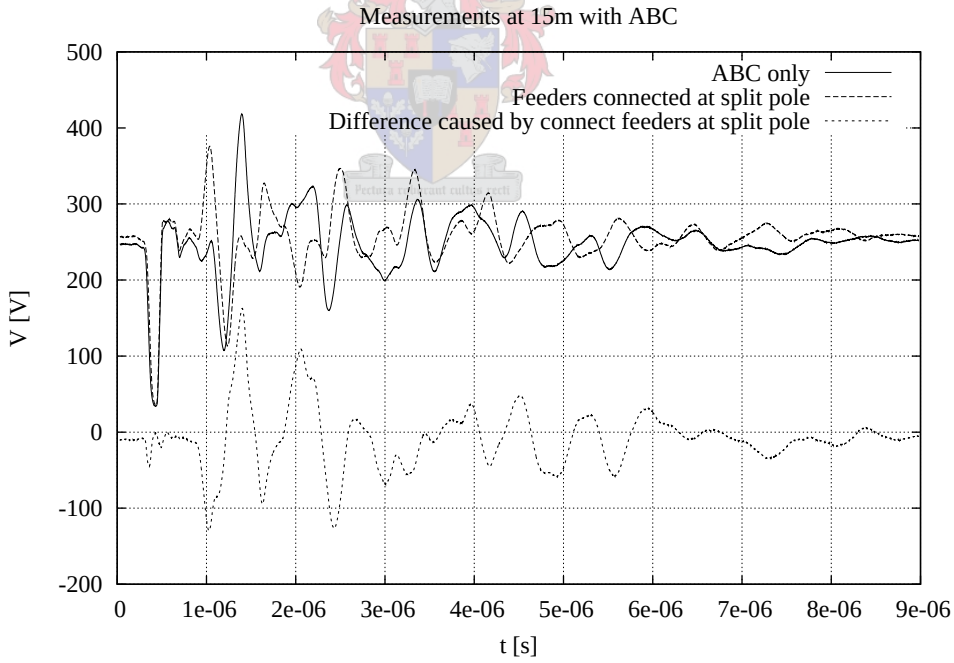


Figure 3.14: *Results from measurements with aerial bundled conductor*

3.7 Simulation

By confirming the models with a SPICE[5] package, larger and more complex networks can be simulated. The parameters used were calculated using the equations provided in Appendix B.5 and the results from an experiment with coaxial cable in Appendix D.2.3.

3.7.1 Cable Parameter Calculation

The data from the experiment performed in Section 3.6.1 was used to determine the parameters. In order to simplify the problem the assumption is made that both cables express similar characteristics and, therefore, they have the same per unit length parameters.

The surge impedance is taken as $Z_0 = 105.545\ \Omega$ from (D.13). From the results of the experiment, the propagation time for an additional $50\ m$ section is measured at $15\ m$ from the switch at V_{15m}, $\tau = \frac{420}{2}\ ns$. The attenuation A as calculated using $V_{t=0ns} = 323\ V$, $V_{t=420ns} = 187\ V$, where $t = 0\ ns$ corresponds with first peak on the results shown in Fig. 3.13, and where the relation $V_{t=420ns} = V_{t=0ns}\ A^2\Gamma$ is used, the attenuation is calculated as

$$A = \sqrt{\frac{V_{t=420ns}}{V_{t=0ns}\Gamma}} \tag{3.46}$$

$$= 0.7609 \tag{3.47}$$

Using the available physical values the following parameters are calculated using the equations in Appendix B.5:

$$R = 571\ m\Omega \tag{3.48}$$

$$L = 439\ nH \tag{3.49}$$

$$C = 40\ pF \tag{3.50}$$

$$G = 52\ \mu S \tag{3.51}$$

$$\tag{3.52}$$

3.7.2 Single Line Simulation

The transmission lines are setup in a simulation which is similar to the experiment in Section 3.5.1. The model used is shown in Fig. 3.15.

When the results in Fig. 3.16 are compared with the results of the actual $50\ m$ Flex Cable measurements in Fig. 3.13, it matches fairly well and confirms the model and measurements. Although the timing and the magnitude of each pulse are roughly in the same area the model does not introduce much distortion and, therefore, gives results that are much better than expected.

[5]Microsims' PSPICE Evaluation Version 9.1

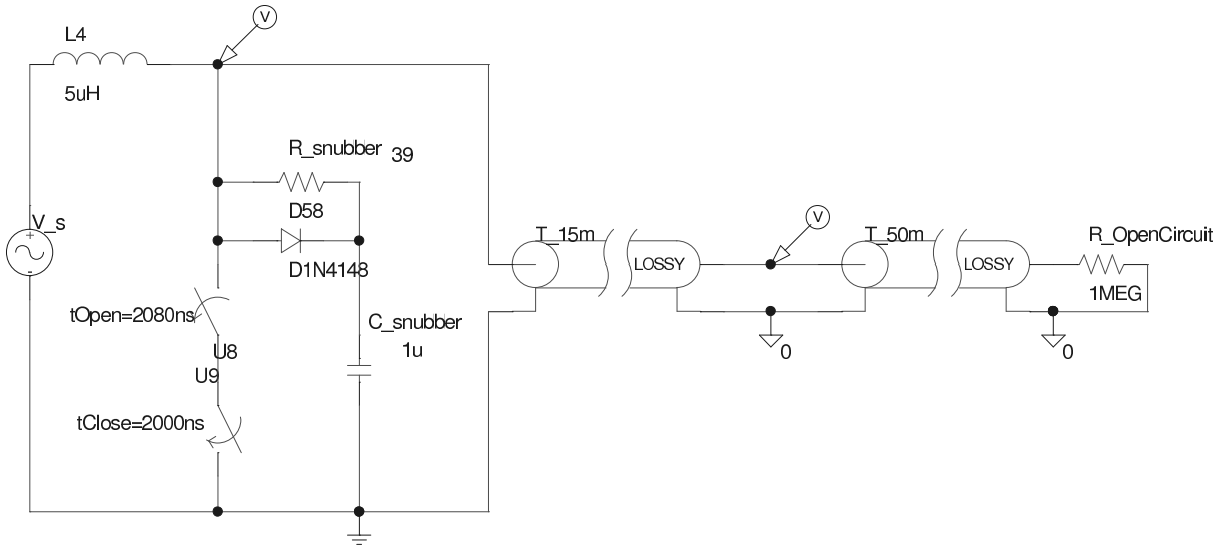


Figure 3.15: *Circuit used for simulation of parallel-switch connected to transmissionlines*

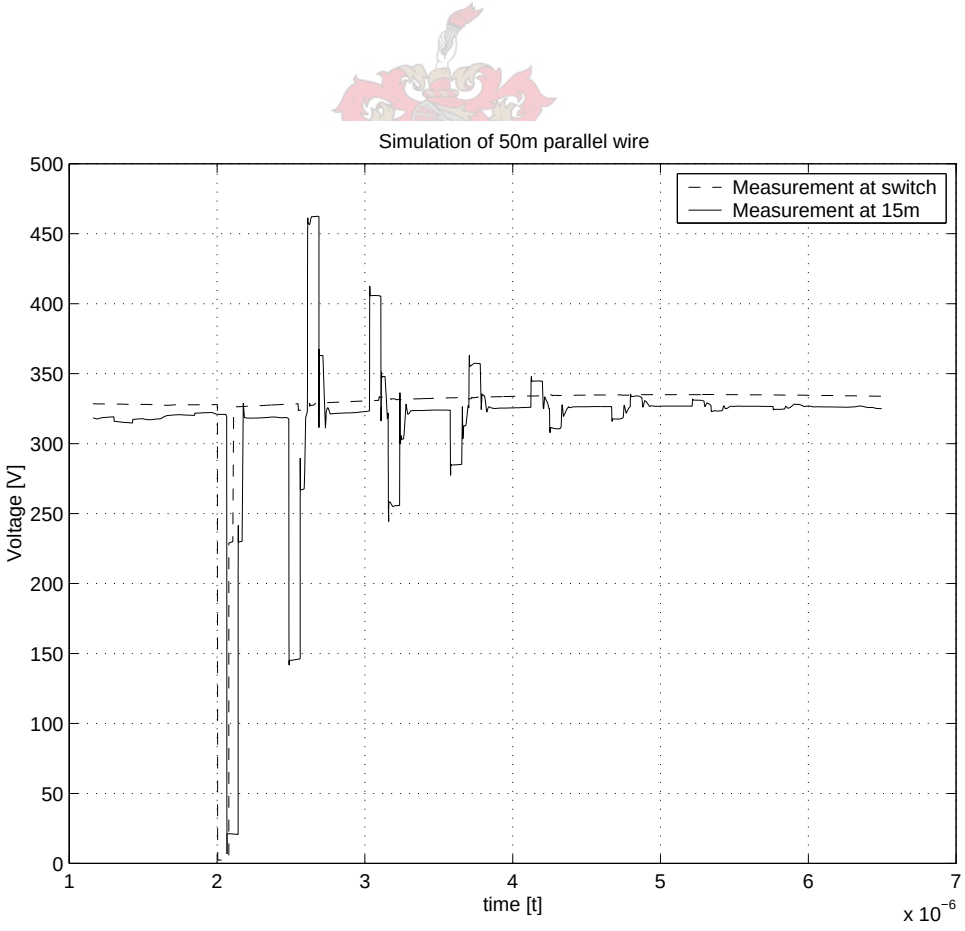


Figure 3.16: *Results from Parallel Switch Simulation*

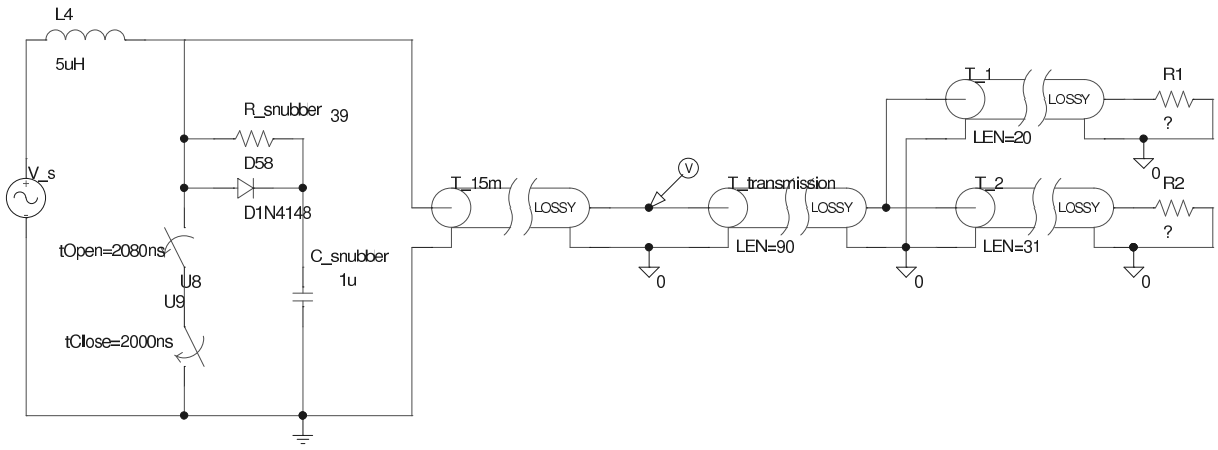


Figure 3.17: *Extended Small Network Circuit*

3.7.3 Split pole Simulation

A second simulation was done to see what effect more connections would have. The simulation performed is a split pole scenario with two connections. The two connections of different lengths, $T_1 = 45\ m$ and $T_2 = 31\ m$, were added to a $90\ m$ cable, as shown in Fig. 3.17. The lengths were chosen to be close multiples of the total length of the first cable. Resistors R_1 and R_2 were used to terminate the two transmission lines, T_1 and T_2 respectively. The measurements were taken after the $15\ m$ transmission line.

The resistor values for the four experiments are shown in Table 3.1.

Experiment	R_1	R_2
1	$1\ M\Omega$	$1\ M\Omega$
2	$1\ M\Omega$	$0\ \Omega$
3	$0\ \Omega$	$1\ M\Omega$
4	$0\ \Omega$	$0\ \Omega$

Table 3.1: *Resistor values for split pole simulation experiment*

The results of the simulation are shown in Fig. 3.18, with the plot only showing data from the first reflection from the split pole connection. The first reflection is found at $t = 2.823\ \mu s$ and travels at a speed of $v = 241\ Mms^{-1}$. The influence of the change in the impedances at R_1 and R_2 can now be seen when the reflection from the switch, $t = 2.948\ \mu s$, meets the reflection from R_1 at $t = 2.992\ \mu s$. The reflection of R_2 is seen at $t = 3.067\ \mu s$. The trailing end of the first reflection from the switch can be seen at $t = 3.0255\ \mu s$.

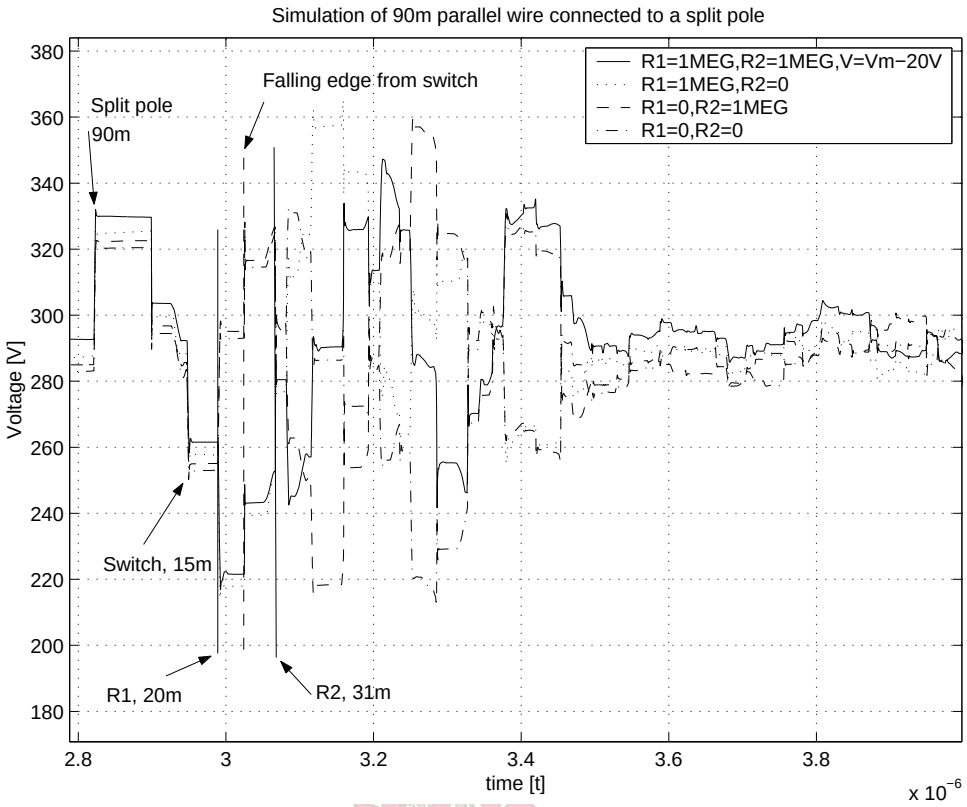


Figure 3.18: *Results of Simulation on small network*

3.8 Evaluation

From the measurements and simulations it can be seen that only after a few hundred meters the accuracy of TDR declines rapidly due to the superposition of multiple reflections. It is also extremely difficult to interpret these signals and map them to a two-dimensional network. Although, as Nielson (see Appendix H.3) proposed, the use of directional couplers may reduce the effect of the reflections against the switch , the problem of echoes however will remain at other split connection further down the line. In the simulation the effect of distortion is not even present. It can, therefore, be expected that field measurements will be even less interpretable as the results presented here. Due to these anticipated inaccuracies it may be required that many units be placed over the whole network section to increase accuracy. This has a direct impact on the total cost of this solution.

An expensive part of implementing this system will be the high voltage, high sample rate, high accuracy data acquisition component. Combined with the running costs of a data link, this could become an expensive solution, even per unit. Since DSP techniques are not yet developed to decode the signals, human interpretation is required. This adds to the running cost of the system. At this stage any automated process would require an accurate model of the implemented network.

With a densely populated network it can be seen from the measurements and simulation that the signal complexity will grow and will become distorted beyond recognition.

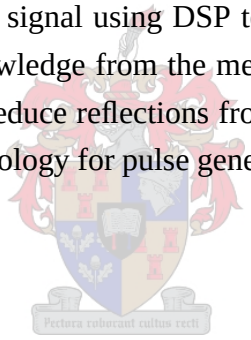
Although not suitable for electricity theft detection, this concept can be used for fault detection [21] on a live network, and possible implementation for cable theft detection on MV lines can be considered.

3.9 Conclusion

This chapter shows that TDR can be applied to an LV reticulation system without interrupting the supply. This enables real time on-line TDR imaging of the network. The placement of different system components is crucial. The capacitance of the PG and pulse width must be taken into account when assembling the system.

Although differences can be detected, the extraction of data from the one-dimensional signal for a two-dimensional problem is too complex to implement in a working system for accurate theft detection.

Modeling and interpreting the signal using DSP techniques can be considered for investigation purposes to gain extra knowledge from the measurements. The use of directive bridge couplings can be investigated to reduce reflections from the pulse generator. Possible applications, using the parallel switch topology for pulse generation, are on-line medium voltage cable theft and fault detection.



Chapter 4

Theory of Constraints applied to Electricity Theft

> If the 1980's were about quality and the 1990's were about re-engineering, then the 2000's will be about velocity. About how quickly the nature of business will change. . . The successful companies of the next decade will be the ones that use digital tools to re-invent the way they work. - Bill Gates, 1999

4.1 Introduction

To address the problem of electricity theft Goldratt's Theory of Constraints (TOC) [36] is used. The initial focus of this publication was at optimization of processes in production plants. However, since its publication in 1992 it has found many other applications and is still regarded as highly relevant in change management today [9].

4.2 The Goal

The main idea behind the TOC philosophy is to establish the goal of an operation, and to optimize the operation towards achieving its goal. Consider a general electricity distribution company. Such a company should be managed in a sustainable manner. This points to one main goal. The company should generate money and profits for the shareholders. In order to achieve this goal the company needs to supply customers with a product or service. In this case, cost effective high quality power. In order to deliver the product as defined, the company should apply a Continuous Improvement Strategy to improve on their past performances and provide affordable electricity to both industry and consumer.

For the purpose of the thesis the goal is extrapolated further and narrowed down to our constraint, namely the "Reduction of Non-Technical Losses". This can be seen as the goal for this thesis as shown in Fig. 4.1.

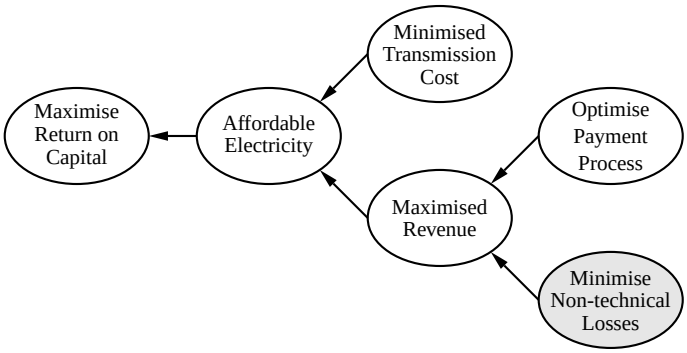


Figure 4.1: *Identifying constraints acting on the goal*

4.3 The TOC Process

According to [31] there are five steps to ongoing improvement. These steps are reduced to four steps for our purposes[1] of handling electricity theft. The process included in these steps is shown in Fig. 4.2.

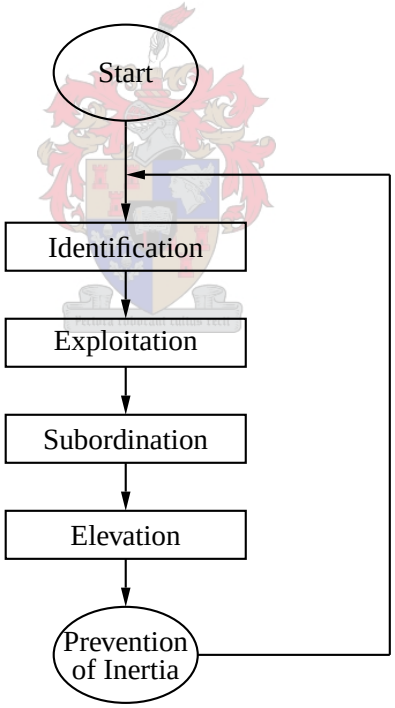


Figure 4.2: *TOC: a Process of Ongoing Improvement*

[1] Elevation is not relevant

4.3.1 Identification

The identification process must be seen in the context of the goal. Anything that brings the whole process closer to the goal is good, and anything that pushes it further away from the goal can be seen as a constraint. For electricity theft, the problem can be subdivided into locations based on the area each different transformer provides with power. Since electricity flows in one direction this makes the search for the weakest link a fairly straight-forward process. The weakest link can be seen as the area where the most electricity theft occurs. This area is where the focus for optimisation should be.

The decision of area size and how many transformers are included in this area will depend on the various situations.

4.3.2 Exploitation

With the facts known with regards to the specific area, for example the losses is known, various methods can be implemented. Below is a list of some known methods:

- If an individual is identified, prosecution is an option (see Appendix F.4).
- Implementing a full-scale remote measurement system has proved to be effective for Ekurhuleni Municipality (see Appendix I).
- Meter sweeps and meter audits can also be done.
- The use of another energy form, for example a $700\ V$ DC, has already been developed[57] for the purpose of eradicating electricity theft.

For each method a business plan can be drawn up in order to show the sustainability prior to initiating such a project. The evaporative cloud can be used here to determine the specific conflict to be addressed for solving this area's problem. Such a cloud is presented in Fig. 4.3, where the prerequisites are in conflict with each other.

An example of such a conflict could be the following:

In order for the electricity utility to provide cost effective electricity requires payment of electricity bills. The perpetrators do not have money and therefore cannot pay their accounts. This clearly results in a direct conflict.

4.3.3 Subordination

Depending on the nature of electricity theft a certain priority and urgency must be given to the relevant project [74]. Depending on the priorities of the organization, that is in terms of distributing the electricity, everything must subordinate in order to reduce electricity theft in the area.

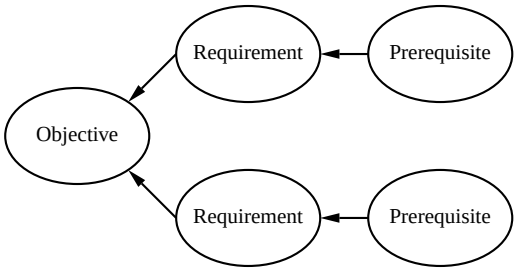


Figure 4.3: *Evaporative cloud for the general case*

4.3.4 Prevention of Inertia

When the situation in the specified area is under control, a new area can be identified as the constraint. The process will now repeat itself when the new constraint is identified.

4.4 Gap analysis and proposed process

As can be seen in the above section there are solutions to address electricity theft and these are available for implementation when an electricity theft hot spot is identified. Management should take care of *subordination* and *prevention of the inertia*. The critical factors for selecting a method is the return on investment made or effort spent and the local context of the electricity theft problem. What is lacking is an effective way or method of monitoring and detecting electricity theft.

With the help of TOC a process has been defined. This process is illustrated in Fig. 4.4.

The process proposed in the thesis consists of the following steps:

1. With all the available data on the various levels one can investigate the current losses within the network and determine if it is feasible to initiate a further investigation concerning the losses of the system in monetary terms. This would typically involve the electricity bill from the incoming feeders into the network and the total consumption of all the customers on the network. The statistical meters on the medium voltage network can also be used. If it makes sense, continue with the process.

2. Identify the measurement points and how energy flows relative to these points.

3. Determine possible “hot spots” within these measurement points and try to reduce or eliminate known technical losses as much as possible by choosing the measuring points carefully, for example:

   - Measure on the LV side of the transformer, to reduce the transformer losses.
   - Reduce the human factor in the information-gathering chain.

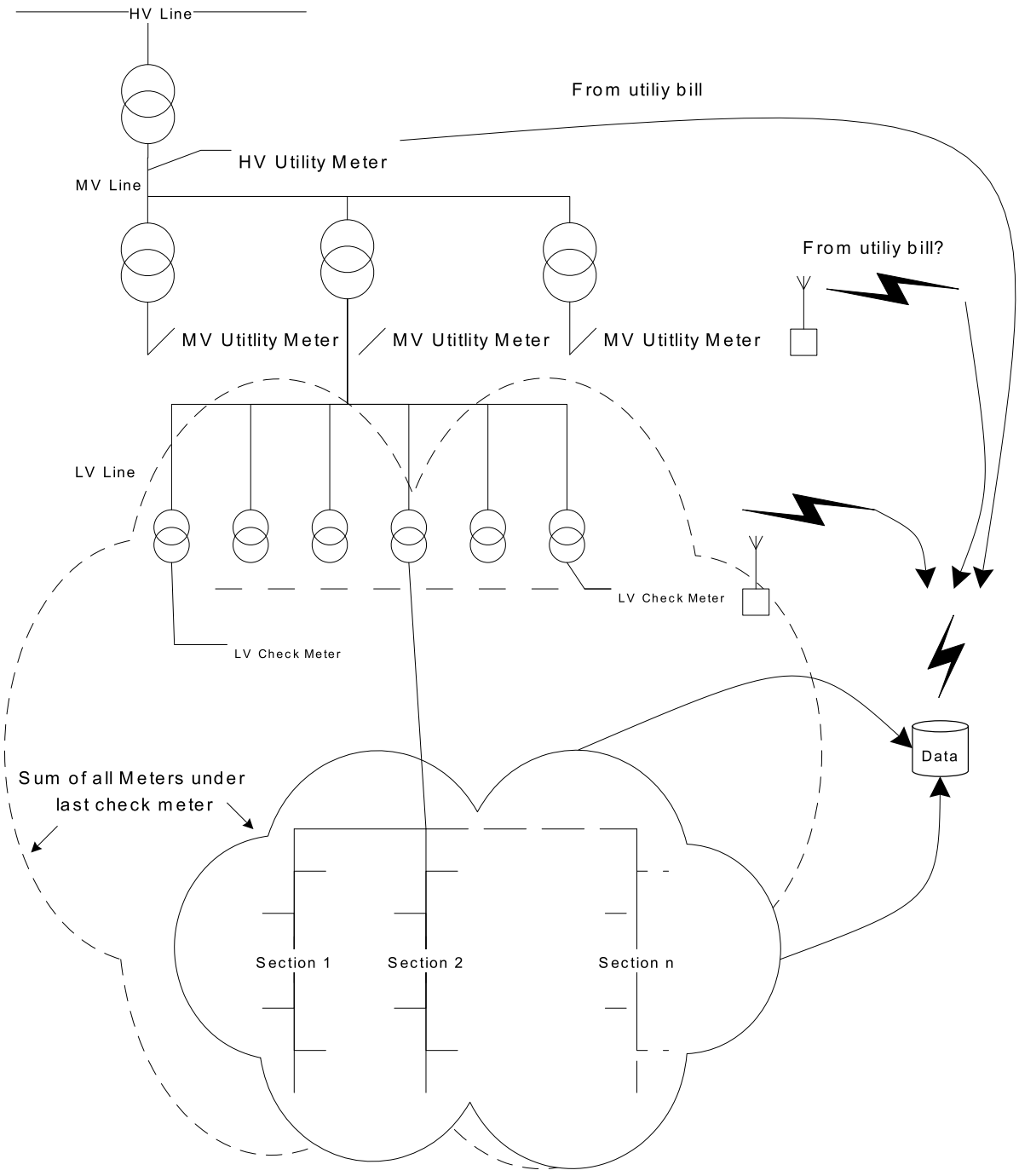


Figure 4.4: *Finding the constraint in the network*

4. With all the decisions that need to be made, keep the costs and benefits in mind through the whole process, and focus on the goal in terms of TOC.

5. If the area is successfully determined and electricity theft has dropped significantly, using *exploitation*, *subordination* and *prevention of inertia*, repeat the identification process again.

To aid this process the following needs to be investigated:

1. The available resources at hand, within the current network.

2. A platform for storing the appropriate data.

3. Investigation and possible development into an efficient system in the information chain. If these systems are required the specifications should be determined.

4.4.1 Resources

Available

In the general distribution network, power flows into the network and all power is consumed in some form. The distributor pays for incoming power and the consumers pays for the used energy. For the utility to charge the distributor, the energy flowing into the distribution network is measured and charged. In some networks, statistical meters may be installed on the MV lines using VT-CT converters or units, as seen in Grabouw in Fig. E.3, which can be used to identify the energy flow into one specific area.

The available resources can be summarized in the following points:

- Incoming meter readings on the HV lines.
- Consumed meter readings (e.g. all the paying consumers)
- Possible statistical measurements on the MV lines.

Required

With all the data available, from the consumers, statistical meters and the utility accounts, the net losses can be calculated. Since all the data is stored in various formats and in different databases a software package is required to represent this data in a useful format, taking into account the possible delays in measurements and stochastic sample times.

Lacking in this setup is a meter on the LV side of the distribution network. Since electricity theft is a dynamic occurrence, the meter should be extremely mobile and weatherproof.

With a platform in place where losses for an area can be calculated one last area is not addressed. The meter readings for the conventional meters are done using a hand system for

data collection. This means that the meter readers write the readings into a notebook and the clerks type the data into the municipality's debtor system. This leaves a lot of room for human error.

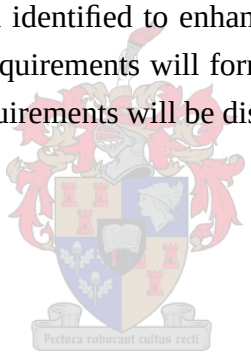
The following are identified to be lacking in the system:

- A data representation package or light-weight data mining application
- An LV mobile remote check meter
- A mobile data collection or meter reading solution

4.5 Conclusion

In this section a process for detecting and identifying and addressing electricity theft has been discussed. Implementing the TOC will elevate revenue protection to a new level. This process is in line with the South African Government Municipal Systems Act [74].

Three requirements have been identified to enhance the TOC thinking process, when applied to electricity theft. These requirements will form a platform for further research. In the following chapters these three requirements will be discussed and expanded.



Chapter 5

Data Mining Software Development

5.1 Introduction

With available technology vast amounts of data is captured daily. Data appears in a variety of formats and locations in distributed systems. These data sets are stored with inconsistent time stampperiods. This requires special methods to extract and represent the information from these data sets.

> Data Mining is the process of extracting previously unknown, valid and actionable information to make crucial business decisions. [10]

The art is then to effectively use the available data and turn the data into information. This is normally done by implementing simple short rules [51], instead of complicated models. This in short is the essence of this chapter: to extract and model the required data to determine the losses in the network to as close a perimeter as possible by accurately presenting the information retrieved.

5.2 Methodology

A data mining solution can be broken up into three different components[10]:

- Data warehousing and collection.
- Data mining or modeling
- Presentation

5.2.1 Measurement and Data Collection

As explained in Section 4.4 in the reticulation environment, incoming and paid consumption is known. The following data points are available for evaluation:

- Incoming feeders from the utility
- Usage of bulk and conventional meters
- Consumption of prepaid meter users

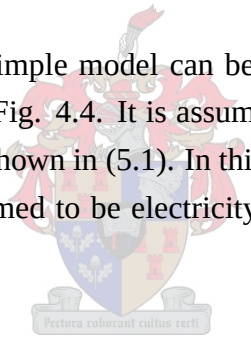
The data coming into the network from the utility is contained in the bills from the utility and is fed into the administration or financial system.

The consumed values for conventional metering and bulk metering is assumed to be stored in a debtor or administration system. These systems normally have some flat file export function or a direct database link via SQL (Structured Query Language). This means that these records can be accessed by external software.

The prepaid customers records are kept in a data base. From a site visit to Actaris in Cape Town, these systems are built on SQL servers and the data can be extracted using an SQL connection.

5.2.2 Modeling

With the data point identified a simple model can be built up using the power flow within a reticulation network as shown in Fig. 4.4. It is assumed that power is only consumed and not generated within this network as shown in (5.1). In this context E_{in} is the total power delivered to the network, $E_{consumed}$ is assumed to be electricity paid for, and E_{losses} includes technical and non-technical losses.



$$E_{in} = \sum E_{losses} + \sum E_{consumed} \tag{5.1}$$

In general, most distribution companies have statistical meters at substations or redistribution points on the medium voltage networks, covering some transformer area. Equation (5.1) can now be expanded.

$$E_{in} = \sum_{area=1}^{areas} E_{losses_{area}} + \sum_{area=1}^{areas} E_{consumed_{area}} \tag{5.2}$$

$$= \sum_{area=1}^{areas} (E_{losses_{area}} + E_{consumed_{area}}) \tag{5.3}$$

$$= \sum_{area=1}^{areas} E_{in_{area}} \tag{5.4}$$

where

$$E_{in_{area}} = E_{losses_{area}} + E_{consumed_{area}} \tag{5.5}$$

In the same manner $E_{in_{area}}$ can be broken down into smaller areas covered by a transformer or an area within a transformer area. This breaking up of the areas reveals the same pattern

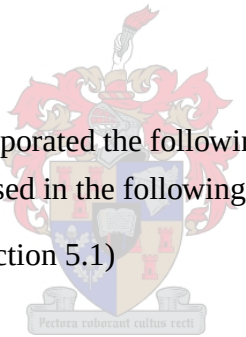
repeating itself recursively, for example, by comparing (5.1) with (5.5). This can easily be implemented into a tree structure as shown in Fig. 5.1 where the area is represented by a node with a relation or connection to a parent and child.

Each measurement point or area can be seen as a node within the network. The node is responsible for extracting information from the data warehouse. Each node is linked to a parent node and can have children nodes. The node is responsible for importing , pre-processing and formating the data for use by the rest of the software elements and nodes.

5.2.3 Presentation

To present the data, a graphical user interface is used. The nodes should be accessible and easy to use. From the modeling of the network a tree structure fits this representation criteria perfectly. Each node should represent the losses in the structure below it, in terms of its children or connected database. To determine trends the software should be capable of plotting the data contained in each node.

5.3 Design



The design of the application incorporated the following phases [10] in the data mining process into a layered design and is discussed in the following subsections:

1. Objective determination (Section 5.1)
2. Data preparation
3. Data mining
4. Analysis and presentation

The software is developed in the Java environment using the Java SDK [83] and Netbeans.org IDE [59] for the following reasons:

- Easy integration and re-use with already developed software classes
- Support and documentation on the internet
- Available classes on the internet

Objective determination

The objective of this application is to determine the losses in an area as effectively as possible, in order to detect abnormalities. These abnormalities will indicate either a highly inefficient system on the technical side or, as expected, an indication of electricity theft in the specific transformer area.

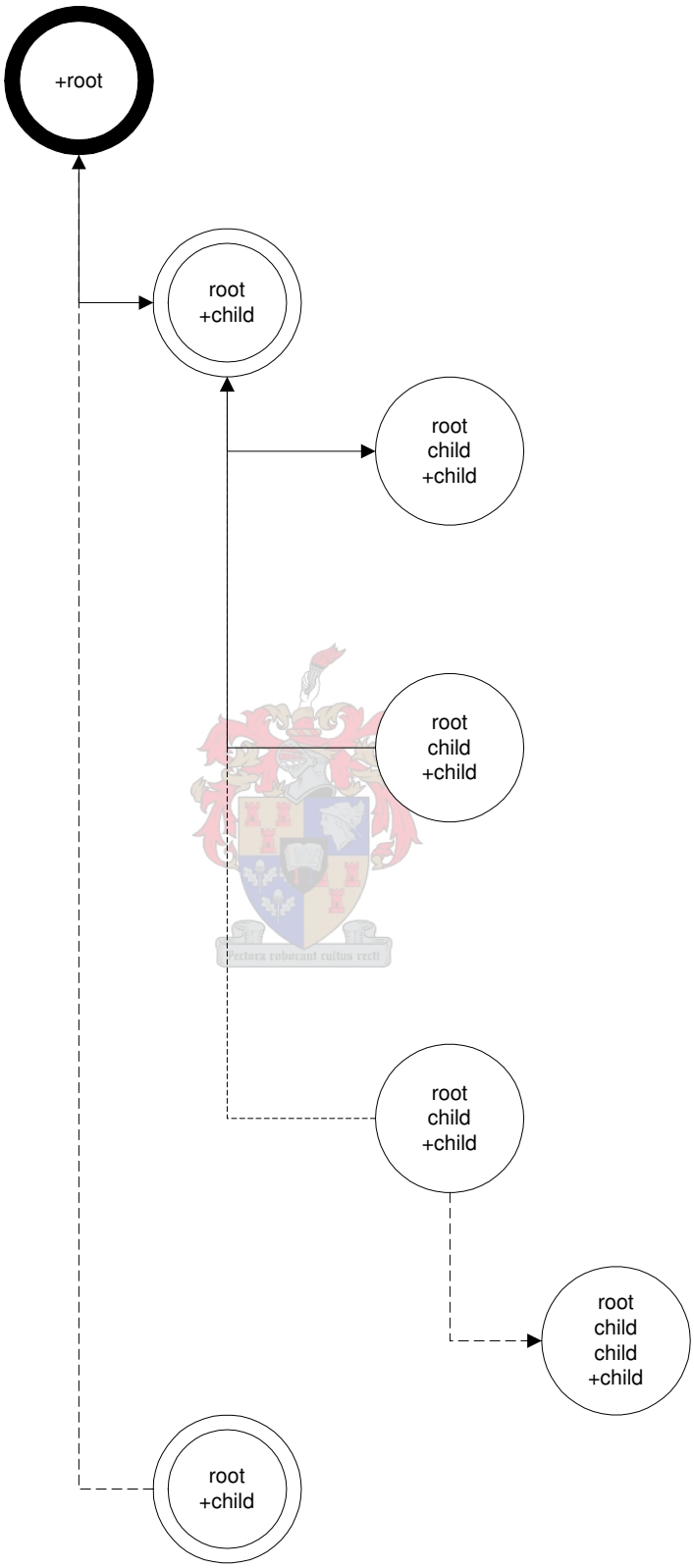


Figure 5.1: *Tree structure diagram where + indicates the depth in the structure*

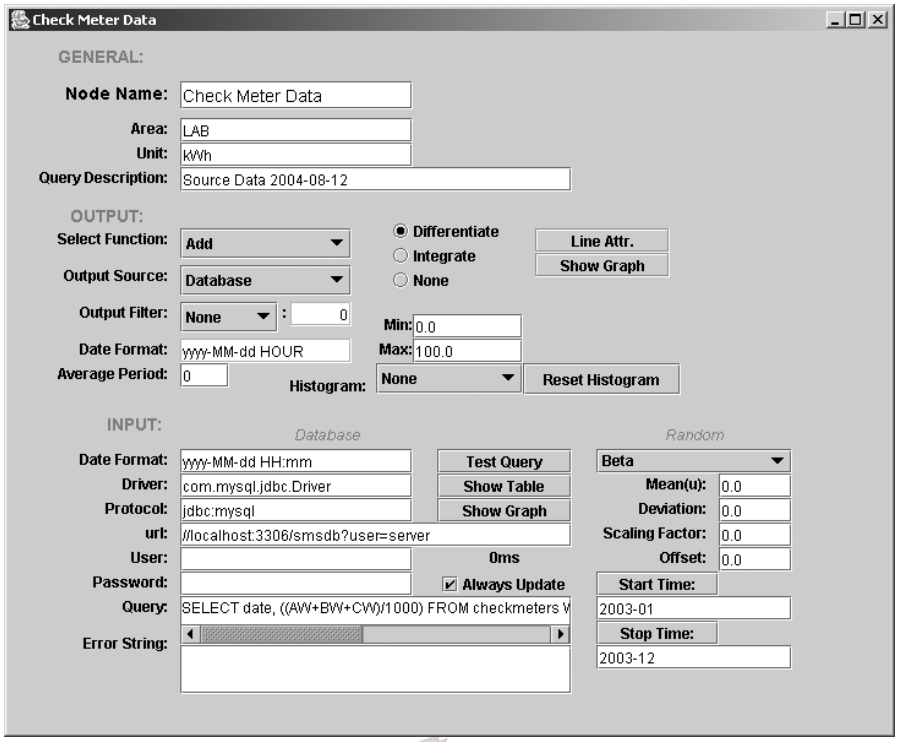


Figure 5.2: *Node Setup Screen Shot*

Data preparation

The data is available in various different forms. The data could be in a flat-file or preferably in a database with SQL capabilities. If the data is available in a database, the correct setup and drivers to access the database are required. Since there are so many available forms, it was decided to implement the software using a middleware client such as JDBC (Java Data Base Connectivity) [85]. To further broaden the scope of the possible data connections a JDBC-ODBC bridge is available to interconnect with ODBC (Open Data Base Connectivity) [94] driven databases.

The data retrieved from measurements in the field is sampled at extremely variable intervals. A grouping function should be implemented to scale the time axes to the relevant interval period, typically a monthly interval. Filtering and other mathematical functions such as integration and differentiation are also required, since the energy readers tend to accumulate their values from sample to sample.

The data preparation is configured in each node. A screen shot of the node setup is shown in Fig. 5.2.

For scenario planning, using the current data to sketch a situation under various circumstances can be extremely helpful, therefore two simulation functions are added to the data preparation part. A histogram function is added, to gather the data of several iterations and a random function generator is included, as shown in Fig. 5.2

An overview of how the data preparation is handled is shown in Fig. 5.3 and is implemented in `public dataCollection toDataCollection()` in Appendix Q.7.2.

Data mining

The data mining part is based on the area subdivision model developed in Section 5.2.2. Although very simple, the various nodes should have the capability to add and subtract the specific samples. As mentioned all the nodes are placed in a tree structure. This allows a routine to search through the tree structure recursively and calculate the value of each sample as specified by the node.

Presentation

Each node is identified in the tree with text. The text gives the name and the context of the node. The average of the total of the node is represented in the text as well as an indication of how efficient the node is with regards to its children. The data of each node can be plotted, therefore allowing a user to identify trends and to react to those trends. The plots give an idea of the current nodes data as well as the children. This screen shot can be seen in Fig. 5.4. For the user interface generation two mechanisms are used namely:

1. The tree structure
2. Data representation using graphs

The tree structure is represented by `javax.swing.JTree` [82] class provided in the standard Java classes.

The graph representation is created using the Scientific Graphical Toolkit provided by EPIC [23].

5.4 Implementation

5.4.1 Overview

The code for the application is held together and coordinated by the `DataMinor.myTree` class. An overview of all the interactions between the classes is given in Fig. 5.5.

5.4.2 Data Connectivity

To connect to the databases the `peg.sql.DBConnect` class is developed. This class uses the JDBC API. JDBC uses drivers provided by the database vendors. These drivers are available either from the vendor's website or from [85].

To setup the database the following parameters are required:

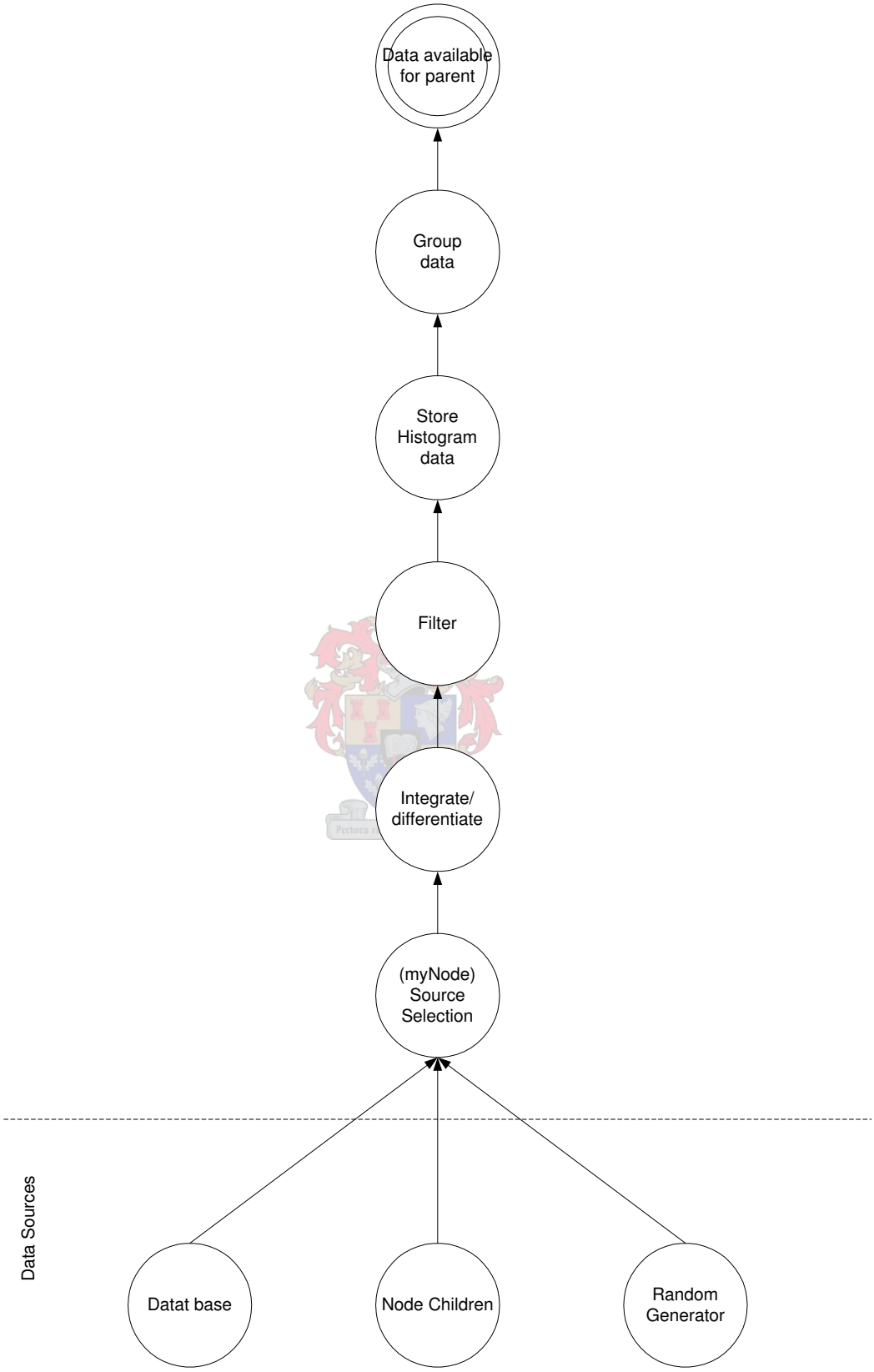


Figure 5.3: *Data Preparation Flow*

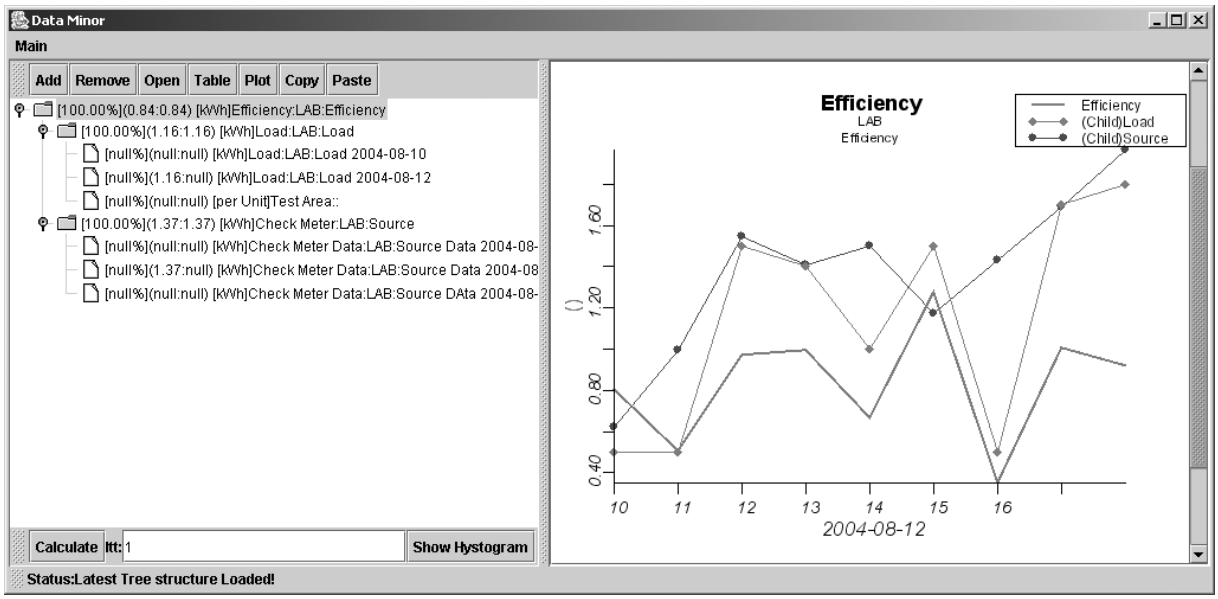


Figure 5.4: *Screen Shot of Application*

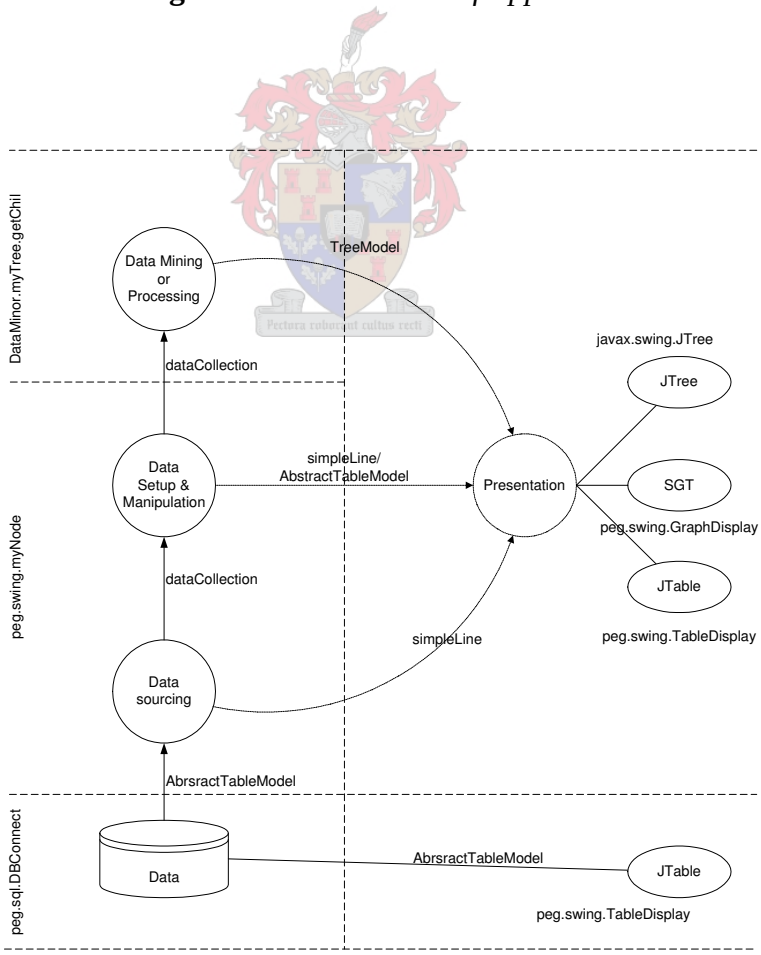


Figure 5.5: *Software Overview*

- the driver class, for example `com.mysql.jdbc.Driver`
- a protocol, for example `jdbc:mysql`
- a connection url to locate the database on the network or PC, such as `//localhost:3306/smsdb?user=server`
- a username, for example server[1]
- a password, such as somepassword[2]
- the query to be sent to the database in order to extract the required information.

This class interfaces to other classes using the `javax.swing.table.AbstractTableModel`. This model can be used to present the data in a table format and can easily be accessed by other classes.

5.4.3 Nodes

The node is represented by `peg.swing.myNode` class. The class is responsible for retrieving the data from the `peg.sql.DBConnect` class and preparing the data for processing in the rest of the software.

The `myNode` class relies on the `peg.utils.dataCollection` class to store the data and handle all the conversions and manipulations of the data. This class keeps the data in a `Vector()` of data-points. The `dataPoints` consist of the time and value of each point. The `dataPoints` can be manipulated in this environment as indicated by the functions in Fig. 5.6.

The data is retrieved from the node using the `toDataCollection()` method in the `myNode` class. The logic of this method is shown in the flowchart in Fig. 5.7.

The node is also responsible for the random processes. These random numbers are generated using classes provided by [71]. The node also keeps track of all the passed values in a `peg.utils.hystogram`(see Appendix Q.8.4) class. This can provide simulation functionality as seen in Section 8.2.

5.4.4 Data Calculation

Each node has the capability of storing the result of all the children's values as a `dataCollection` object. This is used by the recursive procedure to calculate the various values of each of the nodes, which are shown in Fig. 5.8.

---

[1] mySQL requires the user name to be in the connection url.

[2] mySQL requires the password to be in the connection url.

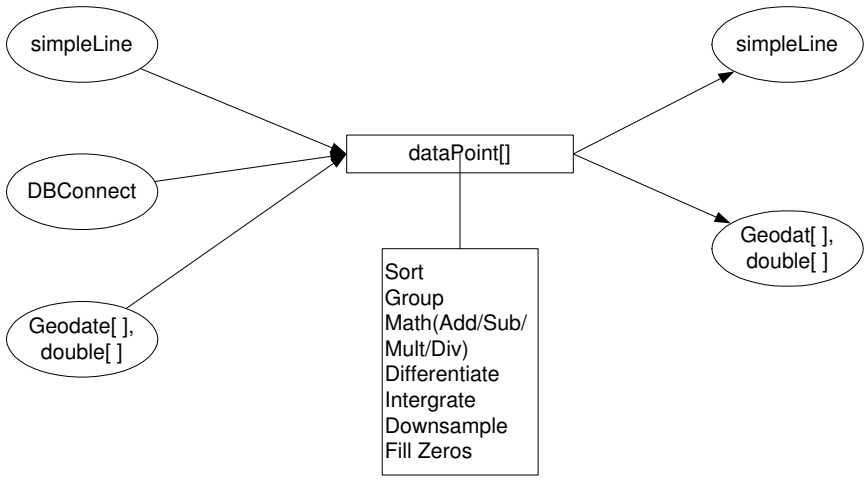


Figure 5.6: *dataCollection Class Functional Overview*

5.4.5 Presentation

The presentation part is implemented using the `javax.swing` classes available in the J2SE [81].

The `JTree` class keeps all the data together. From the `JTree` all the nodes are accessed and controlled, for example getting data and plotting the data and tables.

Two extra classes are developed to extend the `JPanel` class and can be imported into the forms designed in the Netbeans environment.

The `GraphDisplay` class relies completely on the Scientific Graphic toolkit, and provides a simple interface for the `myTree` class. The `TableDisplay` class is written with the same idea, but implements the JTable to display all data extracted from the database associated with a node.

Plotting the data for the user is initiated by pressing the PLOT button on the user interface. Depending on the type of node and how the node acquires its data the plot is performed according to Fig. 5.9.

5.5 Summary

This chapter develops software to assist managers and distribution personnel by providing a tool to determine the losses in transformer areas. By intelligently placing check meters and adapting the consumed data within the node, it is possible to zoom in and identify the cause of the high losses in the area.

By adding random processes in the nodes simulation capability is provided and combined with available data. From this models can be generated to aid in scenario planning. This can be used prior to developing an area to simulate possible network layouts. With an already existing network the expansion possibilities can be investigated.

The node analysis currently being done is based on comparing the averages of the various

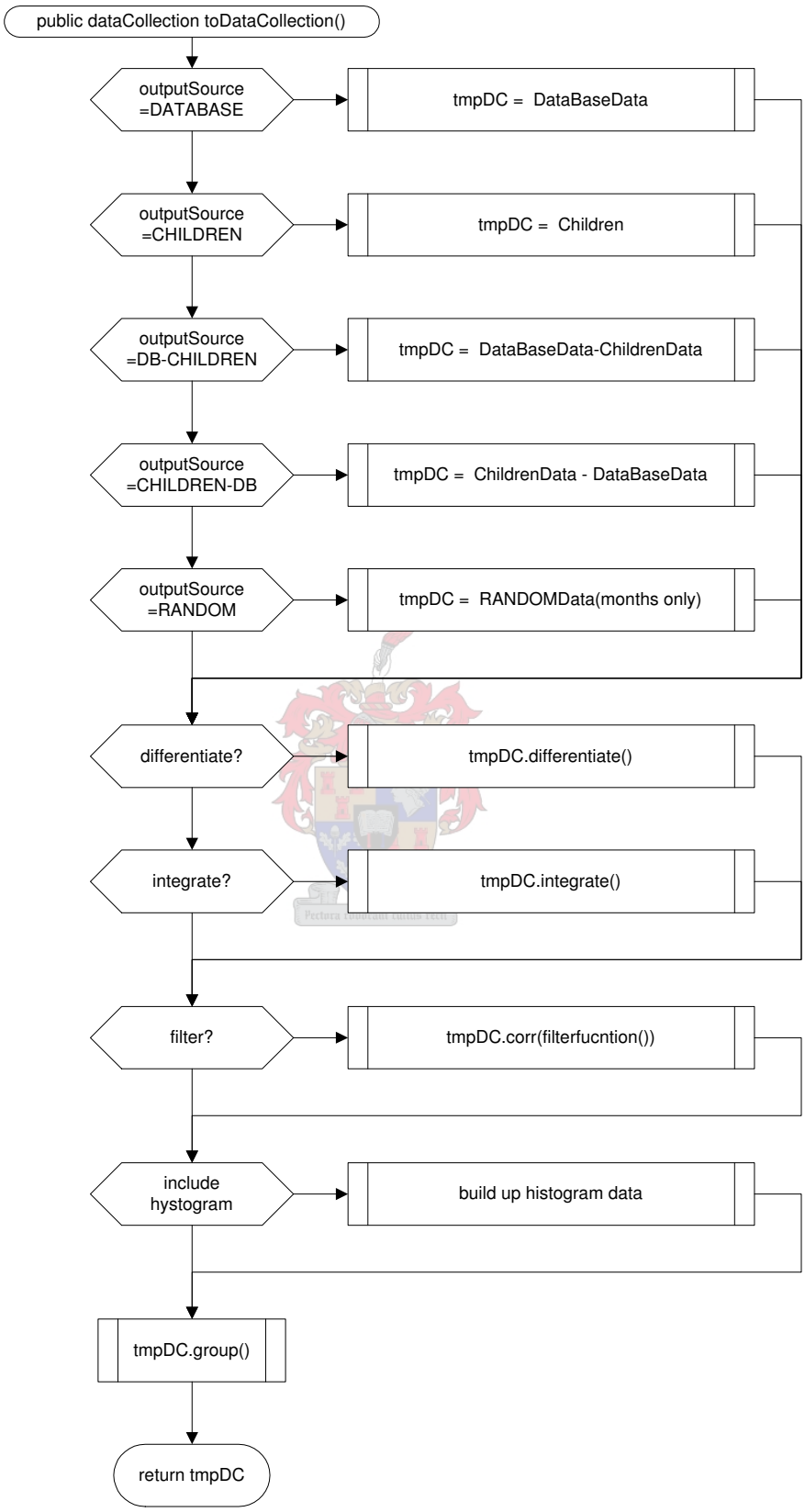


Figure 5.7: *Node Data Output Program Flow*

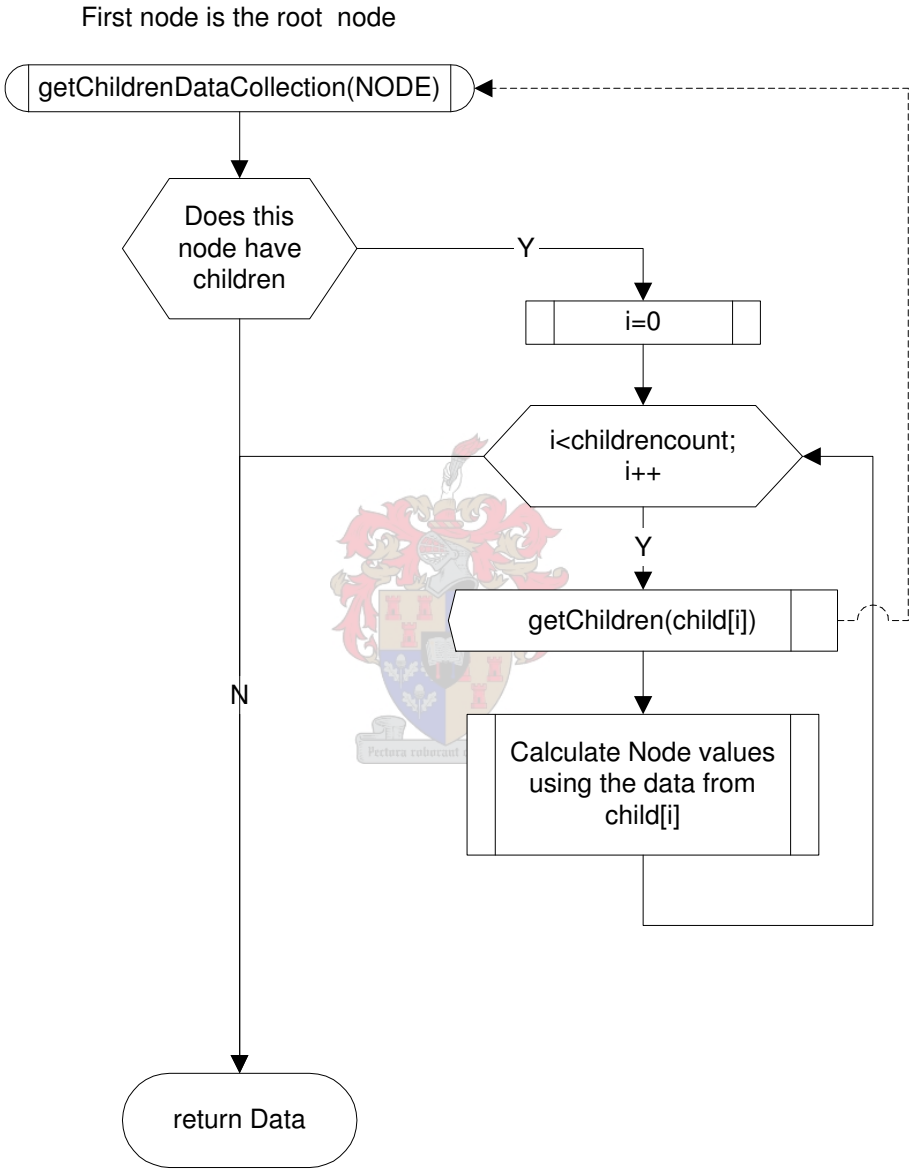


Figure 5.8: *Recursive Node Calculation Procedure*

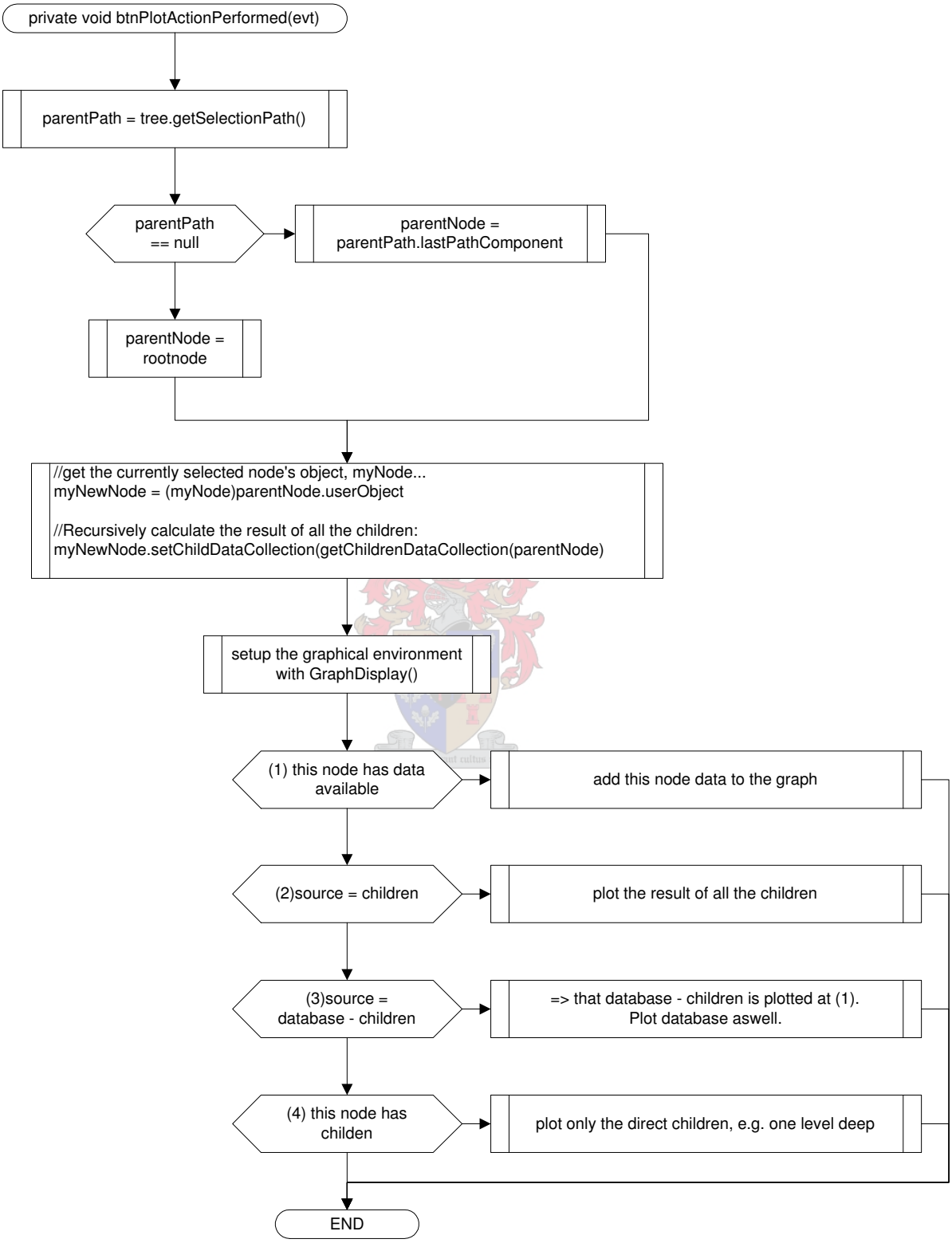


Figure 5.9: *Flow diagram of plotting functionality*

children and data of the node. Implementing statistical indicators and the bootstrap [8] paired samples analysis method in each node can further expand the capability of the software. Incorporating the work of Fourie [32] could also prove extremely useful.

This package is also in line with the performance criteria as set out in the Municipal Act 2000 for municipalities [74], whereby an interface is provided to track targets and monitor the systems performance. The software package can easily be used and integrated to aid other performance criteria applications such as water losses.

Chapter 6

Mobile Remote Check Meter Design

6.1 Introduction

In Section 4.4 it is seen that quantifying electricity theft is difficult and in the information chain, a gap exists here. Cloete [14] achieved excellent results using previously installed conventional meters to check the new pre-paid consumers and hence applied the check meter principle successfully.

Populating a total LV reticulation network with new check meters is a costly exercise. By isolating different areas within the reticulation network and installing check meters, losses can be determined based on the Theory of Constraints. In order to reduce the technical losses the check meter is placed at the LV side of the transformer, instead of the MV side, which eliminates the technical losses due to transformer inefficiencies as well as the need for CT's and VT's. Applying the ideas in Chapter 4 using a check meter, it is now possible to manage losses in a specific area. From the simulations done by [20] and in Section 8.2, it seems feasible to implement this strategy and the results generated justified looking deeper into the implementation of a remote check meter.

With new advancements in communications and measurement technologies, a decision was made to research the design of a low cost, highly configurable, weatherproof, low maintenance automatic remote check meter that is also easy to implement.

6.2 System specification, selection and overview

For the purpose of checking the energy flow into an area the system should consist of the following:

Checkmeter A three phase meter

Data Warehouse Some database server

A data connection A link of some kind to transfer the data from the check-meter to the data warehouse.

For the purpose of the study, this check-meter solution should comply with the following specifcations:

3ϕ **Y** Measure three phase power on a LV reticulation network, which implies $V_{LN} = 230V$, $\pm 10\ \%$, with an added $10\ \%$ for contingency the design should at least be capable of handling $V_{LN} \approx 280V$.

800kVA Should be able to measure a current of up to $I_\phi = 1160A$

Weatherproof The device should be able to operate in an all-weather environment.

Low Cost As with all research, the cost associated with the device should be as low as possible. This cost should include maintenance and running costs.

Accuracy For this specific function, namely of comparing its measurements against other measurements monitored at various arbitrary intervals, an accuracy within 1% to 2% should be sufficient. This accuracy is in line with a Class 2 meter specification.

Remote Operation The meter should be able to operate without any form of human interaction for the lifetime of the meter.

Flexibility The solution should form a platform where researchers can use and change devices in software as required.

Ease of use The system should be easy to install

With these specifications a comparison was made with other available products. Table 6.1 shows a comparison of available products against the criteria as set out in the previous paragraph [1].

Comparing the available products with the requirements that are necessary revealed that it would be worthwhile designing a new system and thereby take advantage of newly developed components and reducing the total system cost[2]. With the design of such a meter in combination with a data software package as described in Chapter 5 a flexible platform for future in-house research will be developed. To ensure a fully integrated platform and solution, an overview of the proposed system is showed in Fig. 6.1.

[1] Prices determined August 2004

[2] The costs of a potential meter at this stage does not include R&D or manufacturing costs.

Criterea	Actaris SL7000[4, 3]	Elite Pro[16]	EnerMax[79]
Voltage Range	57-230V	600V	380V
Current Range	5A or CT	CT dep	100A or CT
Weather Proof	No info	No info	Indoor appl.
Cost	Not available	R10793	R8798
Accuracy	Class 1	1%	Class 1
Remote Comms.	via RS232 RS 485 Infra Red DLMS-Cosem	RS232	RS232 or RS485
Modem included	No	No	Yes
Data logging	Energy and tariff registers	128k 25000 readings	160k 300days at 30 min
Flexibility	Proprietary products and software req.	Easy data imports e.g. excel SQL	Exports data to flat files
Installation	Fair	Very Easy	Fair
Data Solution	Eclipse Not available	ELog R1628	E-Man LU R 2000

Table 6.1: *Overview of available technologies*

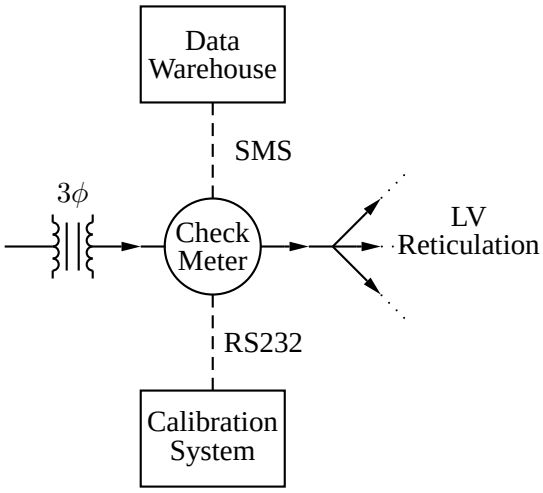


Figure 6.1: *System Overview of Check Meter*

6.3 Hardware Design

The check-meter consists of several subsystems: an energy measurement component, a controller unit, a power supply and a telemetry system. The design and construction is discussed in this section and the schematics, printed circuit board layouts and component list can be found in Appendix L. The total cost is roughly estimated at $R\ 4\ 700$ excluding the current transformers.

6.3.1 System Overview

The topology chosen is based on the ADE7758 energy measurement chip from Analog Devices and a micro controller, MSP430F149 from Texas Instruments as discussed in Section 6.3.2. Both are low power, low cost chips with almost no additional external active components required. The ADE7758 is capable of measuring active energy with true RMS up to 14kHz with a error of less than 0.1%[7]. To simplify the design a Traco power supply was chosen. The telemetry is done with a GSM[3] modem. The WISMO M2106B Integra modem is compatible with other vendor's modems, for example Falcom, which makes sense for production purposes. The system layout of the hardware is presented in Fig. 6.2.

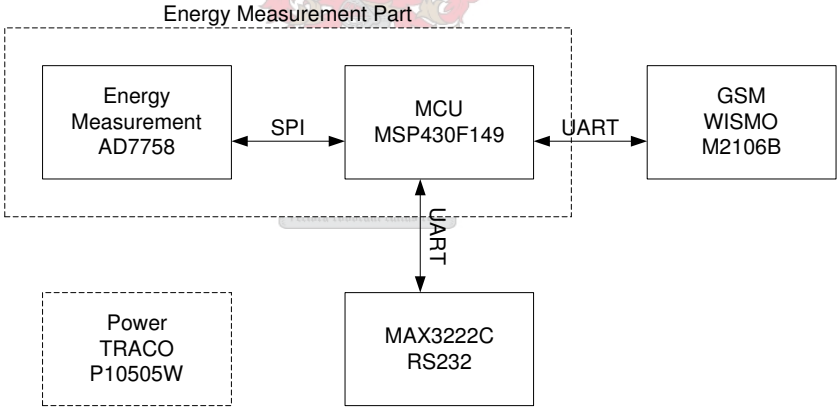


Figure 6.2: *Component Overview of Check-Meter*

6.3.2 Energy Measurement

DSP and ADE7758 evalutation

For the energy measurement two options were evaluated: a DSP based and pre-programmed energy measurement chip based solution.

Using a DSP, the energy measurement and controller unit can be implemented and combined using software. However, a DSP requires an amount of external components, for example

[3]Global System for Mobile Communications

level shifters, and requires commercial compilers. A DSP based design will require extensive software development, since the energy measurement system has to be built up from scratch in the software. The DSP would have been more flexible in terms of calibration methods, such as using a lookup table instead of two-point calibration to increase accuracy.

Analog devices released a new energy measurement chip, the ADE7758. The ADE7758 can be interfaced via a SPI bus to a Micro Controller Unit (MCU). This MCU should then handle the interfacing to different components. The ADE7758 does not require any additional active components due to the differential inputs which use a resistor divider network for voltage conditioning.

A quick overview of the costs in terms of the proposed components is shown in Table 6.2. The costs are based on a six channel measurement system. Prices available from the various companies [5, 91] on 17 August 2004.

Component	DSP 1	DSP 2	ADE7758
TMS320F240	$13.00	$13.00	
AD620(x6)	$18.84		
AD7715		$10.30	
ADE7758			$6.88
MSP430F1491			$5.60
Total	$31.84	$23.30	$12.48

Table 6.2: *Evaluation of different energy measurement strategies*

The following factors influenced the selection of the ADE7758 instead of a DSP:

- Accuracy
- Lower power consumption
- Lower cost (see Table 6.2)
- Calculates all the desired energy values required for the application
- Easy interfacing possible using SPI bus
- No level shifting required, hence no additional active components
- Simple and quick calibration process

ADE7758

Inside the ADE7758, the data acquisition is done by differential input analog to digital converters. These inputs require an input signal smaller than $500\ mV_{peak}$. Simple voltage divider

networks for both the current and voltage inputs are used with a capacitor added over the input to act as an anti-aliasing filter. For the current sensors an additional $100\ nF$ capacitor is added to increase the phase lead.

In order to protect the input stage of the ADE7758, two diodes are placed parallel, each in the opposite direction, to the inputs of the converters. This is to ensure that the inputs are not driven extensively past their desired input range of $\pm\ 500\ mV_{peak}$. Metal Oxide Varistors capable of $275\ V_{RMS}$ are placed parallel to the voltage measurement inputs, prior to the voltage dividing network. This is to ensure that spikes are damped and the device is operated within limits. The voltage measuring probes are fitted with $500\ mA$ fuses to protect the device.

Two LEDs were connected to the chip to help with calibration. The chip is driven by a $10\ MHz$ crystal. The SPI bus voltage levels are $5\ V$ TTL. A dual voltage buffer is used to interface with the $3.3\ V$ logic of the micro controller.

The circuit diagram of the energy measurement system is shown in Fig. 6.3.

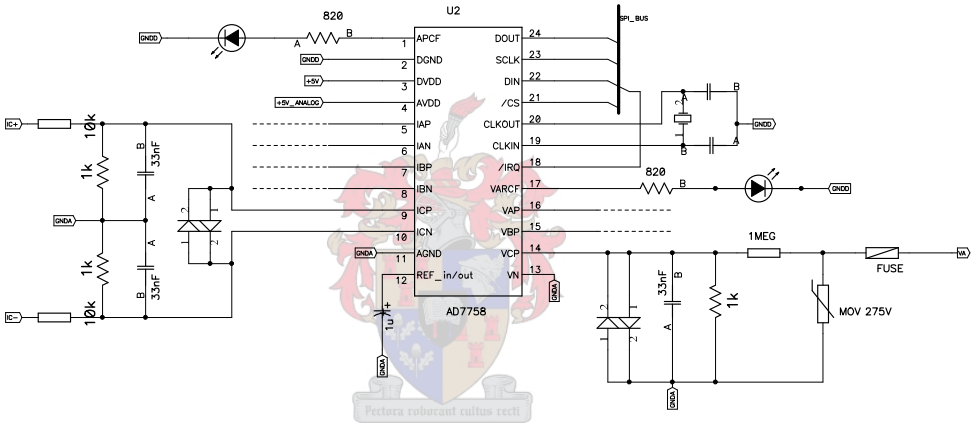


Figure 6.3: *Energy Measurement Configuration*

Signal Conditioning

Voltage The ADE7758 has software selectable input ranges for example $\pm 500\ mV_{peak}$, $\pm 250\ mV_{peak}$ and $\pm 125\ mV_{peak}$. The nominal measurement voltage should be around $V_{LN} = 230\ V_{RMS} = 325\ V_{peak}$. To keep things simple, it was decided to design the hardware for a nominal measurement voltage of between $\pm 250\ mV_{peak}$ and $\pm 500\ mV_{peak}$ input, which allows for measurement of both higher and lower voltages. A simple resistor divider network is used, as shown in Fig. L.3.

For $R_2 = 1\ k\Omega$ is chosen. R_1 is then calculated:

$$R_1 = R_2 \frac{V_{in} - V_{out}}{V_{out}} \tag{6.1}$$

$$= 1000 \frac{325 - 0.375}{0.375} \tag{6.2}$$

$$\approx 870\ k\Omega \tag{6.3}$$

In order to keep the values simple, a resistor value of $R_1 = 1\ M\Omega$ was chosen. This implies that with $V_{LN} = 327\ V_{peak}$, the voltage at the ADE7758 will measure $325\ mV_{peak}$, which is in the acceptable range of operation.

Since the voltage over R_2 is negligibly small, the selected resistor R_1 should be able to handle all voltage that is put over the terminals. The selected Vishay HTS 68 resistor can handle a voltage of $2\ kV$ over its terminals.

When measuring[4] the capacitance of the $1\ M\Omega$ resistor is $0.02\ pF$ and with $C_2 = \frac{R_2}{R_1}C_1$ results in $0.2\ pF$, which is negligibly small.

Current The current sensing range and capability are conventionally associated with the CT ratios. The selection of the sensing device is, therefore, important for the design of the check-meter. For this specific application a clip-on CT or probe should be sufficient since no DC offset should be present. The selected technology should be able to allow for a wide variety of current specifications. The device should preferably be able to operate in an all weather conditions. Cost should also be considered. The various options are considered and summarized in Table 6.3 for $I_{nominal} = 1000\ A$.

Criteria	Conventional CT	Typical LEM	Current Probes
Clamp On	Wide range	No	Yes
Maximum Current [A]	w	1200	10-1000
Safety	Fair	Good	Excellent
Voltage Output	No	Yes	Yes
External Power	No	Yes	No
Cost	R1800*	R1805**	1900*
Model		LT 505-S	

Table 6.3: *Evaluation of different current measurement solutions.* **June/August 2004,* ***September 2003*

For current sensing, LEM current probes were used with voltage output instead of conventional CT's. The main advantage is that the voltage output reduces the need for a burden resistor. For safe operation, when clamping the current sensor over the live conductor, this implementation also reduces the risk of electric shock when no burden resistor is connected to a conventional CT. For this application it is only required to measure the AC component, due to the connection at the transformer. The LEM, however, has a $3\ ^o$ phase shift. This is corrected by putting a $100\ nF$ capacitor parallel to the conditioning network.

[4]@ $75\ kHz$, tested 23/08/2004

The current inputs on the ADE7758 can also be selected for inputs ranging similarly to the voltage inputs. For the prototype the laboratory current probes were used mainly as a consequence of availability. The specific current probes, LEM M2, measure a nominal input current of $20\ A$ and a conversion ratio of $output = \frac{5\ V}{20\ A} = 250\ \frac{mV}{A}$.The other available probes similar to these provide an $output = 10\ \frac{mV}{A}$ and $1\ \frac{mV}{A}$. This equates to a maximum output of $V_{out} = 4.242\ V_{peak}$ when the $200\ A$ LEM probes are used. Considering that the ranges are software selectable, the a resistive network with $R_2 = 1\ k\Omega$ and $R_1 = 10\ k\Omega$ is used. The capacitance difference between these two resistors is relatively small. With a $V_{out} = 4.242\ V_{peak}$ the input voltage into the ADE7758 would be $V_{in} = 385\ mV_{peak}$.

Anti-aliasing Filter In order to prevent aliasing a capacitor is placed parallel to R_2. The ADE7758 samples at

$$f_{sample} = f_{CLKIN}/128 = 10.10^6/128 = 78.1\ kSPS$$

[7]. This means that the anti-aliasing filter should reject frequencies higher than $44\ kHz$. However, the evaluation board design [6] suggests a cut-off frequency of $4.8\ kHz$, which requires that the RC network should consist of a $33\ nF$ capacitor parallel to $R_2 = 1\ k\Omega$.

Over voltage protection

The check-meter should be able to operate safely at $V_{LN} \approx 280V$ (see Section 6.2). To ensure safe operation and protection against spikes and short over voltage bursts, two measures are taken to protect the circuitry:

1. The voltage channels are fitted with MOVs and the input leads are fitted with fuses.
2. In order to protect the Analog to Digital converters of the ADE7758, two diodes are placed parallel to each other, in opposite directions, to ensure the voltage may never exceed $\pm 0.7\ V_{peak}$.

The Metal Oxide Varistors (MOV) are placed across the input channels of the check-meter to ensure that energy in spikes is dissipated and the voltage is clamped. This should cause an increase in current flowing into the device. If the current exceeds $500\ mA$, the fuses placed in series with the voltage measurement channels should blow. From the data sheets, the fuses will blow when a constant voltage of $V_{rep} = 580.0V$ is placed over the lines due to the current flowing through the MOVs. This value implies that measurements will not be affected by the voltage clamping in the operating range, but will instead protect the device against over voltages and spikes.

The energy measurement and conditioning circuits should be able to handle the constant voltage at $V_{LN} = 580\ V_{RMS}$ but the power supply will not operate in this region, since it is specified at $V_{LN} = 264\ V_{RMS}$ ans can handle $1\ kV$ surges.

The fuses are placed into the probes used for the clipping onto the measurement points.

The selected MOVs have proved [5] to protect the circuit during testing.

6.3.3 Micro Controller

Since most of the calculations are done by the ADE7758 only a micro controller is required. Various controllers are on the market, but it was decided that the MSP430F149 will be more than sufficient for the purpose of interfacing the energy measurement chip with the communications components. The micro controller was chosen due to the following features:

- Low power
- 2 available UARTS
- SPI interface capability on one of the UARTs
- Ample capacity for future expansions
- Low cost
- In-system programming
- Free compilers available
- Good support on internet forums [72]
- Availability

Clock System

The clock frequency selected for the MCU is $f_{xtal} = 7.372\ MHz$. This crystal, XTAL, frequency is the highest available multiple of 9600, which increase the reliability of inter device communications.

XTAL1 is configured to run, instead of the internal clock of $32\ kHz$[87].

The following code is implemented to start the clock correctly:

```
unsigned int j;
BCSCTL1 |= XTS;                       // ACLK = LFXT1 = HF XTAL

do {
  IFG1 &= ~OFIFG;                     // Clear OSCFault flag
  for (j = 0xFF; j > 0; j--);         // Time for flag to set
} while ((IFG1 & OFIFG) == OFIFG);

BCSCTL2 |= SELM1+SELM0;               // MCLK = LFXT1 (safe)
```

[5] An over voltage was generated on 2004/08/03 which caused one of the MOVs to blow. The MOV was replaced and the system was working as normal again. Unfortunately at that stage of testing the fuses were not in place.

SPI Interface

The first UART is configured as an SPI bus to communicate with the ADE7758. The micro controller operates on $3.3\,V$ logic and hence, has to interface via a $3.3\,V$ to $5\,V$ bus transceiver. The transceivers, SN74LVC4245A from Texas Instruments, are able to allow signals to travel in both directions [88]. Two transceivers were used and hard wired to allow for up and down conversion of the various control and data lines.

MAX3222C

The second UART is shared between the GSM modem and normal RS232 communications. The RS232 is used for both debugging and calibration purposes. The RS232 must be operated via an electrically isolated link to prevent damage to devices due to different ground potentials.

For communications with a PC, a RS232 line driver is used. Since two different devices, the GSM modem and the PC, operate on the same UART-bus from the MCU, a line driver capable of going into high a impedance state is required, to prevent the possibility of the different devices driving each other. The RS232 line driver is only required when calibrating the device. For the rest of the time the device is switched off. While the line driver is switched on the GSM modem can be shut down and is put in a high impedance state.

6.3.4 Mobile Communications

The GSM modem is a Wismo Integra M2106B from Wavecom. This unit was selected due to availability and the compatibility between other vendors. It also offers good value for money for this specific application. The modem is only required to send SMSs[6] to the server. Therefore, a GSM instead of GPRS solution was selected. An easy AT command interface was used to communicate with the modem. In standby mode the unit consumes a low amount of power, and since only SMSs are sent, the transmission power used is minimal.

6.3.5 Power

To provide power to the system, a TRACO TML10505 module is used. This module is capable of supplying $2\,A$ at $5\,V$. Two capacitors are used to reduce the ripple on the $5\,V$ output. This is ample current to supply the various devices, including the GSM modem [96] at the full power of $1\,A$.

A 3.3V voltage regulator [78] for ST Microelectronics was used to supply the 3.3V circuit with power. This regulator is rated for a maximum of 1.5A.

---

[6]"Short for Short Message Service. Similar to paging. SMS is a service for sending short text messages to mobile phones" [94]

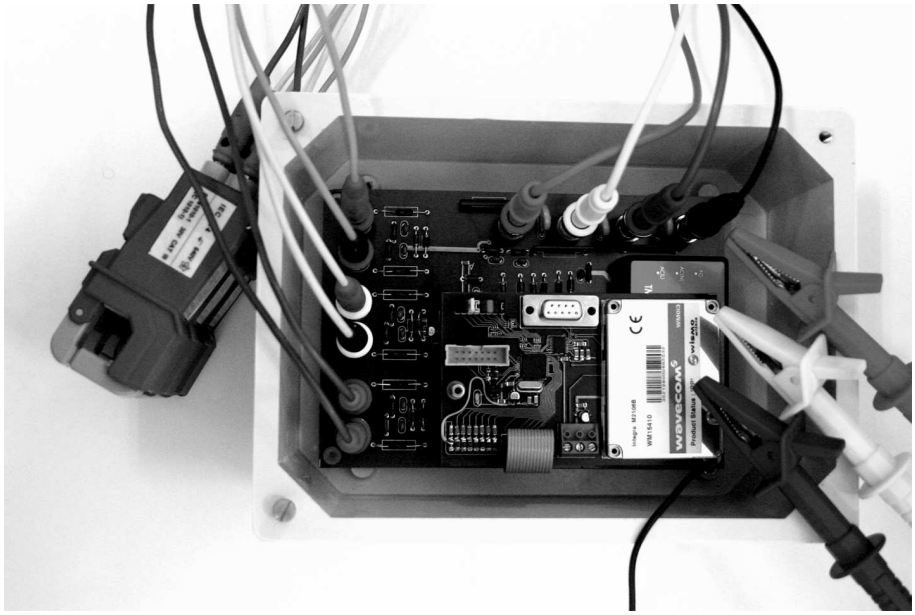


Figure 6.4: *Check Meter Prototype*

6.3.6 Construction

In order for the check-meter to function in a real-world environment, some thought must be given to the physical implementation of a possible final prototype, although not yet tested in the field.

The prototype is constructed on two stacked PCB's. The unit is placed in a weatherproof container to allow for all-weather installation. See Fig. 6.4. Because of its proposed use in rural areas, the container can be pole mounted as well.

Enclosure

The check meter is constructed using an ABB J2-S enclosure with dimensions of $200 \times 145 \times 80\ mm$. This console is specifically selected for its ability to be mounted on a pole, in an all-weather environment. An opening should be made at the bottom of the console and closed off with some cladding to ensure that water does not leak or spatter into the console in extreme weather. The exact cladding is not yet decided on.

Printed Circuit Boards

The check meter consists of two separate Printed Circuit Boards (PCB), which are assembled in two layers on top of each other. The constructed PCB's are shown in Fig. 6.5.

These should be mounted on a mounting plate supplied for this Voltex console.

Figure 6.5: *Layers PCB stack*

Probes

The probes are connected to the board via $4\ mm$ safety sockets, allowing for $32\ A$ and $8.2\ kV$ isolation without flash-over. The probes selected for the assembly are fused with $500\ mA$ ($50\ kA$, $1\ 000\ V$) and rated for $600\ V$ [66].

6.3.7 Connection and Installation

Before any connections are made, the box should be mounted onto the pole or safely fixed near the transformer or connection points. The leads coming out of the check-meter should hang down, to prevent water leakage into the meter. Depending on the intended use, it is suggested to strap the check-meter onto the pole using $10mm$ zip-ties or with brackets, to allow for easy removal.

To connect the device electrically to the network the following steps are suggested:

1. Connect the current probes to the check-meter.
2. Clamp each current probe over the conductor.
3. Connect the neutral voltage probe to the check-meter and connect it to the network. Take care to connect it to the neutral and not any of the other cables. This is important to ensure no damage to the device and to ensure that the correct measurement is made.
4. Connect each phase separately, firstly by connecting the probe to the check meter and then to the power grid. While connecting the probe to the transformer or the line, ensure that no physical contact is made by the user to the clamps of the probe to reduce the risk of electric shock.
5. After completing the electrical connections, ensure that the leads at the bottom of the console are wrapped with some form of cladding.
6. Ensure that the LEDs are flickering with in the correct sequence.
7. Close the lid of the check-meter and ensure its position by tightening the screws provided with the enclosure.

6.4 Check Meter Software Design

Each of the various components, except for the power supply as shown in Fig. 6.2, are coordinated by the MCU using software. The different component software is discussed in this section. Firstly, the design philosophy will be discussed. The state-event machine implementation is explained further in the section hereafter. The section following the state-event explanation will concentrate on some of the higher level components, such as calibration and GSM software.

6.4.1 Check Meter Software Design Philosophy

The check-meter is designed to run and operate in remote areas and should have minimal human contact for operation. The check-meter is an integral part of the data collection process and reliable data is required from this device. The software must, therefore, be extremely stable and reliable.

For this purpose the design was based on a state-event driven system. For this approach the following is required:

- No infinite loops may be present, except for one main loop in the software.
- The total state-event cycle time should be fast enough to allow for reliable information capture from the fastest device whose data can possibly be lost.
- The state-event machine should run in a stable environment.

The software is further broken up into different layers and parts. These are run as drivers which service the periphery and interface with other drivers. Each driver is programmed as a state-event machine. Events are either triggered by devices or calls to the drivers. The calls set specific flags which tell the state-event machine to start a task or process. The various levels, and their subsystems are outlined as follows:

GlobalHardware Ensures that the basic hardware is setup correctly for the MCU to ensure correct instruction execution.

- `main.c` The main application coordinating and implementing the system.
- `clock.h` Sets the correct clock for the system.

Data This layer provides interfacing with the internal periphery available on the MCU.

- `timer.h` Provides reliable timing functionality.
- `LEDs.h` Allows LEDs to be used for debugging purposes.
- `DIPswitch.h` Allows the user to provide input for the software by means of DIP switch selection.
- `spi.h` Provides a communication via an SPI interface with external components on serial port 1.
- `uart.h` Provides communication via the UART interface to a computer using serial port 2 on the MCU.
- `flash.h` Allows storage on the internal flash module of the MCU.

Protocols Communications to the external periphery

- `ad7758.h` Provides the interface to address the registers in the ADE7758 energy measurement chip.
- `rs232.h` Allows communications with the calibration software on the PC via the RS232 port.
- `atCommand.h` Interfaces with the GSM modem's AT command set.

Applications Provides functionality to the system

- `initMeter.h` Responsible for correct setup of all the global functions of the system.
- `energy.h` This header is responsible for measuring and storing the energy data.
- `onLine.h` Provides the calibration and real-time monitoring routines when connecting to a PC.
- `gsmMode.h` Allows the check-meter to communicate to a remote server using the available GSM modem.

When using this state-event approach the various processes can run concurrently. Each process is allocated the execution of one state. The execution sequence is done in the `main.c` code. Depending on the current mode of operation either the `onLine.h` or `gsmMode.h` drivers are executed. The following code snippet is implemented to do the process coordination:

```
for (;;) {                  // State Event Machine
    //Main Loop Heartbeat
    LEDs_toggleLED6();

    //MCU internal periphery
    LEDs_driver();
    DIPs_driver();
    SPI_driver();
    UART_driver();
    FLASH_driver();

    //External Hardware Communications
    AD7758_driver();

    //Application Drivers
    switch (mode) {   //mode is determined by the DIP switch/jumper setting
       case MODE_RS232:
          RS232_driver();
          OnLine_driver();
          break;
       case MODE_GSM:
          Energy_Driver();
          AT_driver();
          GSM_driver();
          break;
    }
 }
```

6.4.2 The State Event Machine

Depending on the function of the driver a state-event machine is implemented. Although `LEDs.h`, `dipSwitch.h` and `timer.h` performs some updating function the functionality is such that no state-event machine is required. For the drivers requiring a state-event functionality the following general guidelines were used to ensure stability in the system:

- Each driver has its own variable storing the state of the driver.
- Each driver should have a set of defined states which is unique to the driver
- The state may only change in the state-event driver.
- Each state should have at least one state-change code.

The state-event machine is programmed using the `switch` and `case` statements provided in C. This approach is preferred above the lookup table approach to increase execution efficiency and aid the understanding of the state-event programming for other programmers. This approach also has an advantage in that typing and logical errors are converted into syntax errors. Debugging the state-event machine using this approach proved to be fast and effective.

The structure of a state-event machine can be broken up into the following:

Pre-amble The code starts with an explanation of the driver, with which part and at what level it interfaces with other parts in the system. This includes all the declarations in terms of definitions and variable declarations for the machine to operate in. For example:

```
/*********************************************************************
*  ExampleDriver (ED)
*
*  Comments are entered here to explain the basic functionality to
*     the users and programmers.
*********************************************************************/

// Any includes may be specified here, which is specific to the driver, e.g.
#include <io.h>
#include <signal.h>

// State Event Declarations
//-------------------------------------------------------------------
//States
#define ED_stInit             0x000
#define ED_stIdle             0x001
#define ED_stFirstState       0x002
  |
  |
#define ED_stLastState        0x0FFFF


//Variables
int ED_state = ED_stInit;           //The current state within a the ED driver
```

```
//Flags
#define ED_InitFlag           BIT0
#define ED_NewCommand         BIT1
 |
 |
#define ED_LastFlag           BITF
int ED_flags = 0x00;               //Flags to allow the driver to keep track of user input

unsigned char ED_SomeVariable = 0x0;
   |
   |
char *ED_AnotherVariable = "";
```

User Calls The second part consists of code which can be called by other users or programs within the system. This code is responsible for setting the flags and variables so that the driver has all the required information to execute the request successfully. These calls are required to send the execution status, so that the user knows if it has to re-send the call to the driver.

```
// ED_SomeUserCall
//
// The user requests this function call to be executed.
// Returns:
//    0 if successful que placement
//    -1 if a character is already placed for a write itteration
//
//-------------------------------------------------------------------------------
int ED_SomeUserCall(unsigned char someVariable) {
   if ((ED_flags & (ED_NewCommand)) == 0x00) {
      //Do some initialization
      ED_flags |= ED_NewCommand; //Set the flag to let the driver know an action is requested
      return 0;
   } else {
      return -1;  //If action is already requested and not completed.
   }
}
```

Actions The third section allows for code which is called by the driver, which is normally called within the states. The code appearing in this section can theoretically be executed in the state, but depending on the implementation it might be more appropriate to implement it as a auxiliary function or action.

```
//  ED_init
//
// Intialize the ED driver.
//
//-------------------------------------------------------------------------------
void ED_init(void) {
   // Initialization code is entered here...
   ED_flags |= ED_InitFlag;  //And the flag is set, indicating a successful initialization
}
```

Driver The driver part is called by a higher level application or code. Normally this is called in `main.c`. It consists of a case statement responsible for selecting the correct state. The

driver is only allowed to execute one state at a time and should then exit. The format of the case statement implements this automatically, but should however be kept in mind when programming. To keep code maintainable the states should only change within the case statement. This is done by setting `ED_state=stNewState`, prior to exiting the case statement with `break`.

```
// ED_driver
//
// The state event machine implementation is done here.
// In the main loop this code is called by ED_driver();
//
//-----------------------------------------------------------------------------
void ED_driver(void) {

   switch (ED_state) {

      case ED_stInit:
         ED_init();
         ED_state = ED_stIdle;   //Change the state
         break;

      case ED_stIdle:
         if ((ED_flags & ED_NewCommand) == 0x0) {
            ED_state = ED_stFirstState;
         }
         break;

      case ED_stFirstState:
            //Does some thing, and wait if required.
            //If all is complete and the requirements is satisfied the state may be changed.
            // Remember that even if the requirements are not satisfied, the state must exit,
            // but the state does not need to change.
            ED_state = ED_stLastState;
         break;
      case ED_stLastState:
            //With the last state executed, the state event machine's state should
            //  return to idle
            ED_state = ED_stIdle;
         break;
   }
}
```

6.4.3 The Initialization Process

Prior to making any measurements, the system is required to run through an initialization process. All the different components, either in hardware or software, are initialized. The strategy behind the initialization and software design is to ensure that a stable environment exists for the state-event machines to operate in.

The process is started in `main.c`, and follows the following process:

1. The watchdog timer is disabled, to ensure that no unexpected resets occur.
2. The clock is setup to use the external crystal in XTAL1.

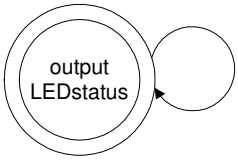


Figure 6.6: *State diagram of LED driver*

3. The `LEDs.h`, `dipSwitch.h` and `timer.h` code is initialized.

4. The mode is determined according to the DIP switch settings on DIP switch 7. With the dip switch closed the check meter will run in RS232 mode. With an open DIP switch the GSM mode is selected.

With all the basic functionality in place the state-event machine is started and each driver is then responsible for initializing all its own functions. On the application level however, a special driver is called, namely `initMeter.h`. This is called within `onLine.h` and `gsmMode.h`. The reason for this, is that this driver is only required for initialization and would after initialization take up unnecessary system time. The following code, shows how `initMeter.h` is called in `onLine.h`:

```
case OnLine_stInitStart:
   if (INIT_init() == 0) {
      OnLine_state = OnLine_stInitSystem;
   }
   break;
case OnLine_stInitSystem:
   if (INIT_isReady() != 0) {
      INIT_driver();
   } else {
      TIMER_startTimer(tmrWDTOnLine);
      OnLine_state = OnLine_stIdle;
   }
   break;
```

A similar implementation is done in `gsmMode.h`. The initialization state machine will be discussed in Section 6.4.4.

6.4.4 Software Sub-systems

This section will briefly discuss and show the implementations of each of the implemented software sub-systems of the check-meter.

LED driver

The LED driver consists of one state. This state ensures that the LED status, which is changed by the calls to the driver is always updated, as shown in Fig. 6.6.

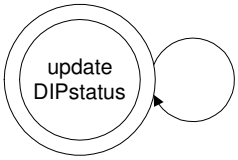


Figure 6.7: *State diagram of DIP switch driver*

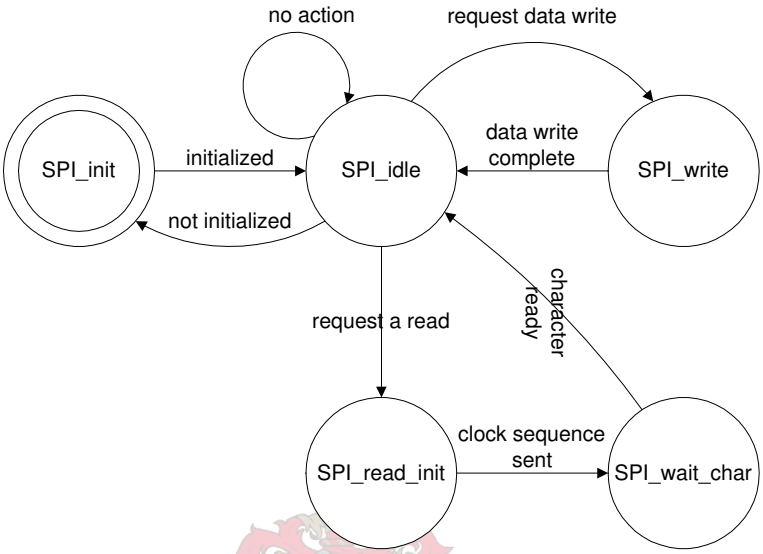


Figure 6.8: *State diagram of SPI driver*

DIP switch driver

The DIP switch driver updates the `DIPstate` register, as shown in Fig. 6.7. When other drivers test for the current DIP switch status they access the `DIPstate` variable. In future this could also be handy when implementing debouncing for keypads or buttons routines.

SPI driver

The SPI driver ensures that the data layer is correctly setup and communications can take place between any SPI device on the bus. The setup of all the registers is done in the `SP_init` state. The state diagram is shown in Fig. 6.8 and consists of an initialization part, a character write part and a character read part.

UART driver

The UART driver is used to communicate with the PC and with the GSM module. It also forms the data layer to these devices. This driver is also responsible for switching between these devices using the power off function of the MAXC3222C[89].

Incoming data is polled using the `UART_isCharReady()` function of the driver. The

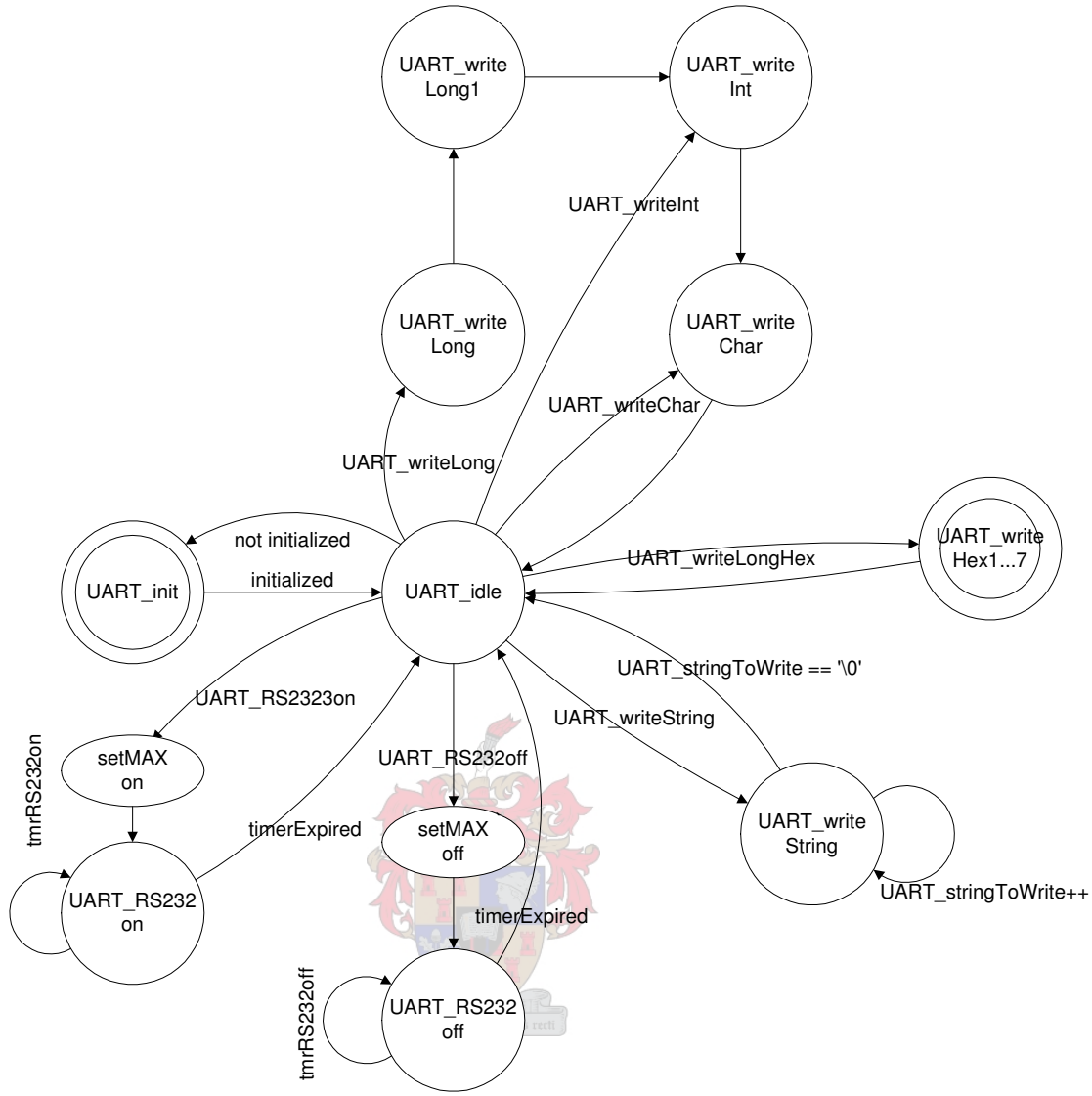


Figure 6.9: *State diagram of UART driver*

driver can send data in 8bit, 16bit and 32bit mode and also includes a process to send a 32bit character in hex format. Strings are sent using a single state which iterates through the string until the null character is found.

The state machine is shown in Fig. 6.9.

FLASH driver

The flash driver interfaces with the internal flash memory of the MCU. During the initialization the correct clock is setup. After each state the BUSY register is polled to see if the flash unit is free for a new command. A pointer is used to write to the memory. The read operation is a straightforward read from the correct address and is accessed by users of this driver with

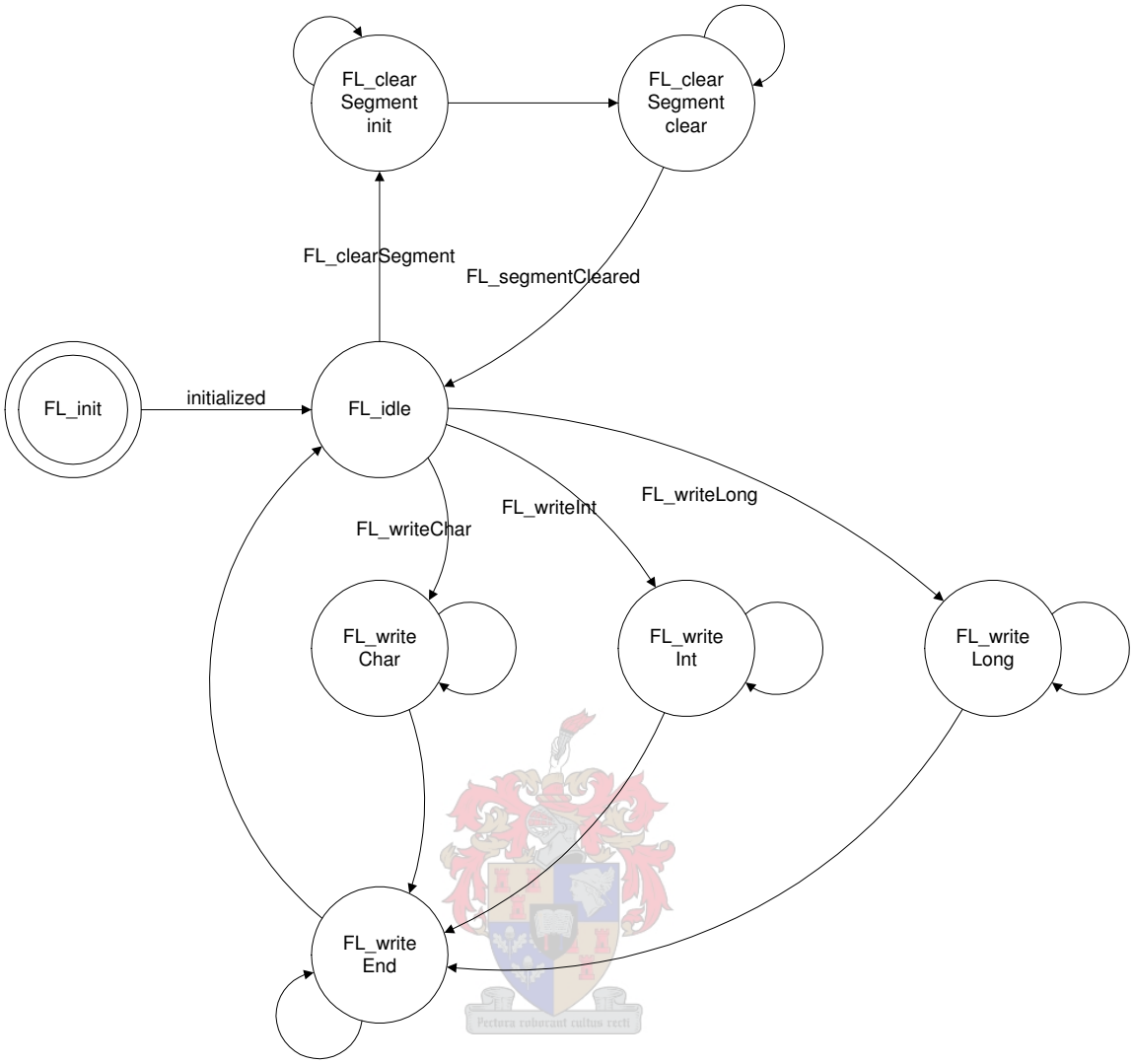


Figure 6.10: *State diagram for FLASH driver*

`FLASH_readChar()`, `FLASH_readInt()` and `FLASH_readLong()`. All the storage registers are defined in this driver.

The various states are shown in Fig. 6.10.

AD7758 driver

The AD7758 driver is responsible for handling all the communications via the SPI driver to the ADE7758 energy measurement chip. This driver contains all the register codes and handles all the various data types that might be sent and received to and from the device. The driver is also responsible for handling the CS signal. All the data sent or received is preceded by a register byte. Depending on the register byte the ADE7758 will expect 1 to 3 bytes of data. A read operation is indicated by setting the most significant bit with a 0 and for a write it should be set to 1.

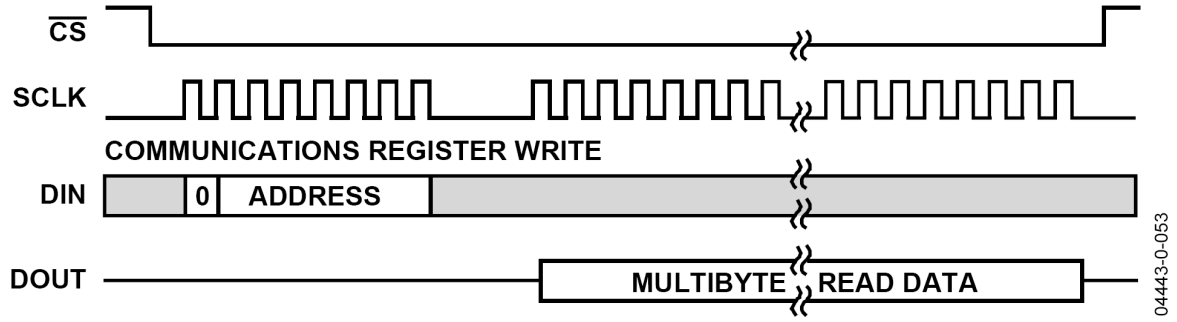


Figure 6.11: *Reading Data from the ADE7758 via SPI, taken from [7]*

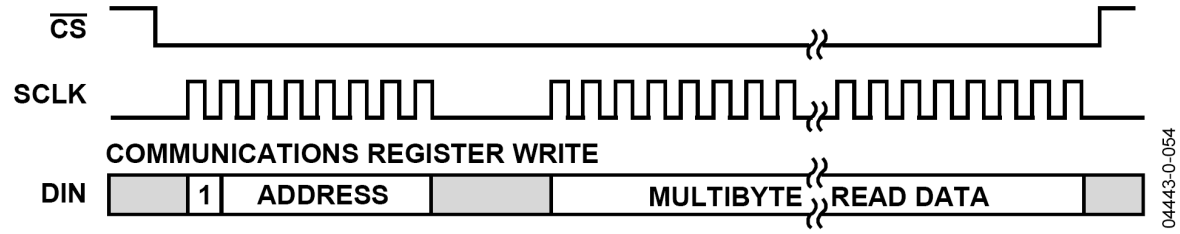


Figure 6.12: *Writing Data from the ADE7758 via SPI, taken from [7]*

Fig. 6.11 and Fig. 6.12 shows the timing requirements as set out in the ADE7758 data sheet [7].

The states are shown in Fig. 6.13.

For the read sequence the register requested by the AD7758 driver is written out to the SPI driver and then the state machine waits for the register to be clocked out. This is followed by the read states. The multiple read states consists of the following states:

1. Read a byte from the SPI driver.
2. Wait and continues to read the next byte until the read operation is complete.

The write operation is fairly similar except that the when a call is received by the AD7758 driver to write data to the AD7758, the register is included in the byte array and can be written out.

RS232 driver

The RS232 driver is responsible for encoding and decoding data to and from the PC. The driver is not responsible for error checking on the data link. Table 6.4 shows the implemented structure of the protocol as implemented in Appendix R.4.2.

The RS232 driver consists of a receive and send component. This is implemented using the following code:

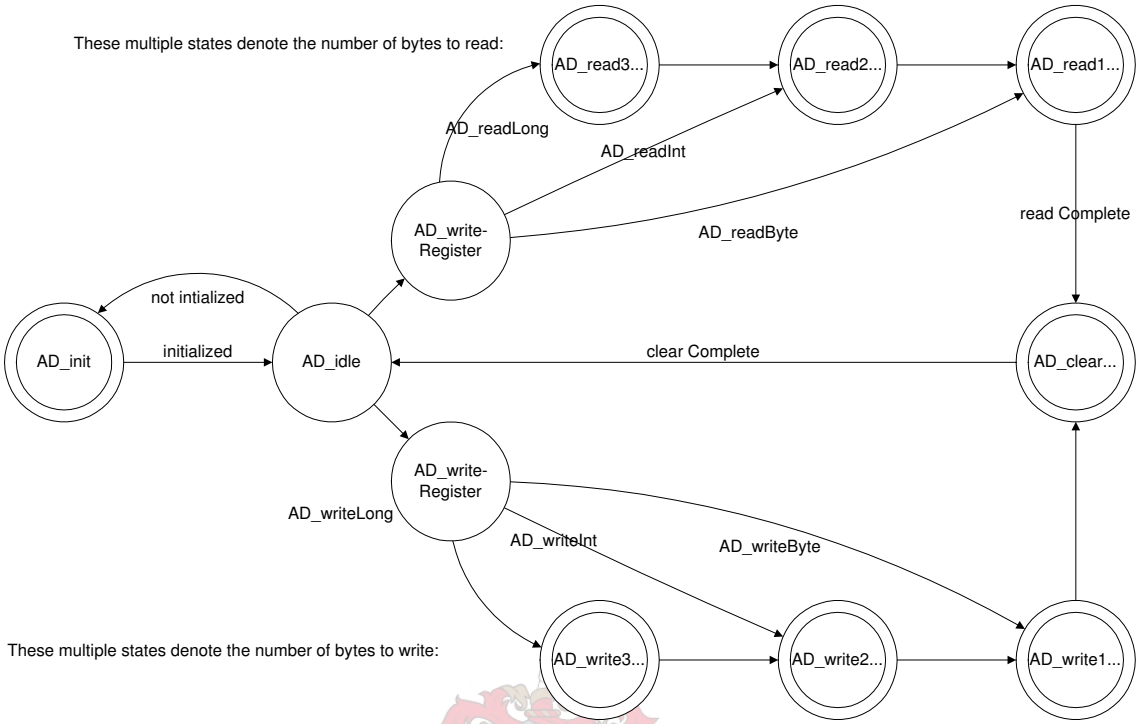


Figure 6.13: *State diagram of AD7758 driver*

Byte	Meaning
0	Start Character
1	Length
2	Device
3	Device Register
4	Data[1]
5	Data[2]
⋮	⋮
N	Data[Length-4]
N+1	Last Character

Table 6.4: *Protocol elements*

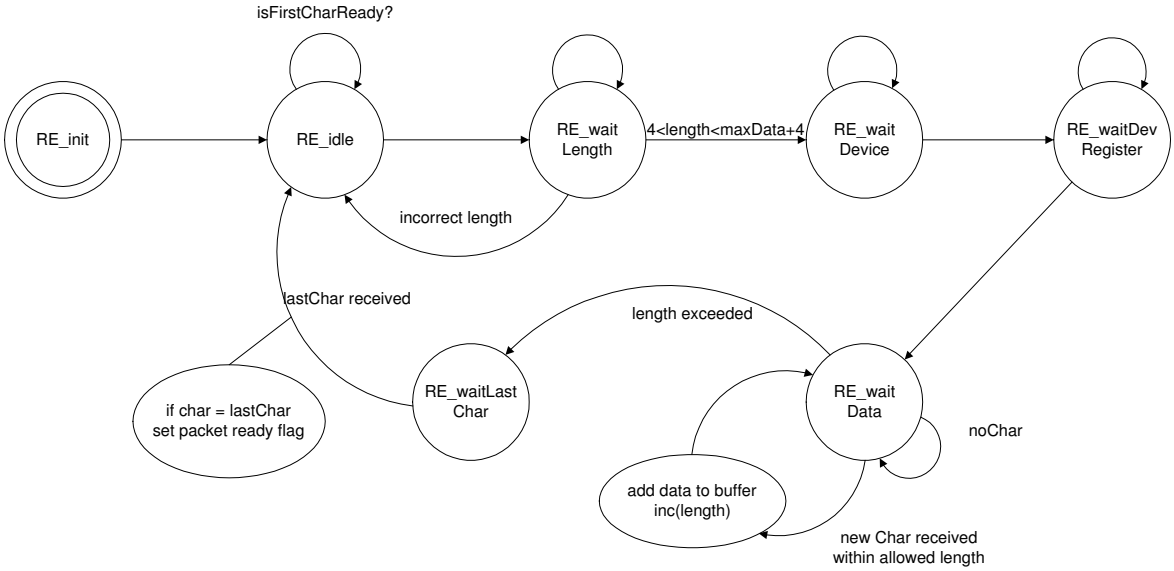


Figure 6.14: *State diagram of receiving RS232 driver*

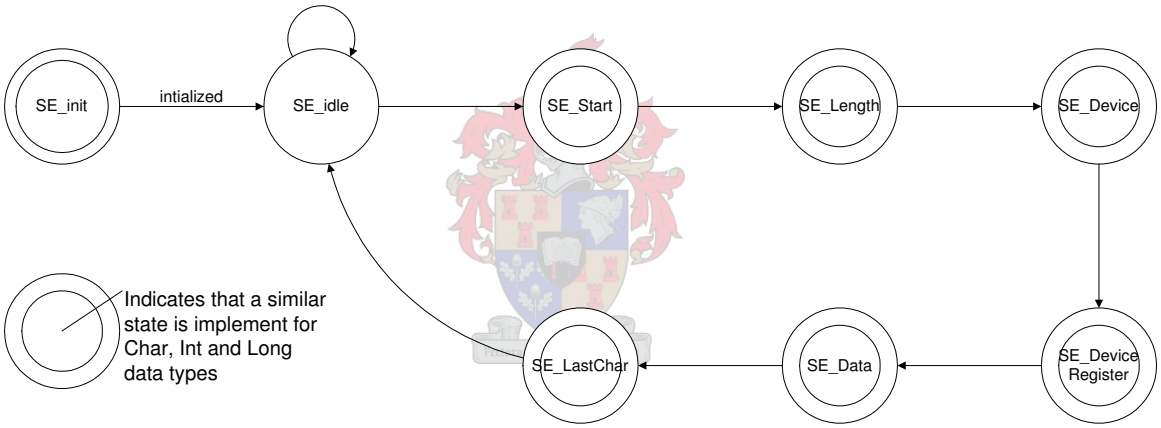


Figure 6.15: *State diagram of sending RS232 driver*

```
void RS232_driver(void) {
   if (UART_isRS232on() == 0) {
      RS232_receiveStateDriver();
      RS232_sendStateDriver();
   }
}
```

The receive state machine is showed in Fig. 6.14.

The sending state machine is showed in Fig. 6.15.

AT command set driver

The AT command set is supplied by Wavecom [93] and complies with ETSI specifications [26, 28, 27, 29, 30], ITU-T recommendations [42] and 3GPP technical specifications [1].

The AT command interface is a text interface that allows easy communications to the GSM

module. Each command is started with the text `\n\rAT` and submitted using the newline and carriage return characters, `\n\r`. The driver is responsible for checking the communications to and from the module and does some error checking to ensure that the commands are sent correctly.

This driver consists of a send and a receive part and can only operate when the RS232 driver chip is switched off. This is implemented using the following code:

```
void AT_driver(void) {
   if (UART_isRS232off() == 0) {
      AT_receiveStateDriver();
      AT_sendStateDriver();
   }
}
```

The send state machine is shown in Fig. 6.16. Simple error correction is handled here. The possibility to extend this exists, but for this application only checking for `OK` and `Error` is required. Using these two strings provides ample error detection. If an `Error` is detected, the AT command is repeated.

In the AT driver the sending of the Energy Report, via SMS, is also implemented. All the energy registers are sent using hex representation. Each register consists of 8 alpha-numeric characters. The following text gives an overview of how the text messages in the SMS are constructed, where W denotes the real power, VAR the reactive power and VA the active power measured per phase A, B or C:

```
WAAAAAAAWBBBBBBBWCCCCCCC:VARAAAAAVARBBBBBVARCCCCC:VAAAAAAAVABBBBBBVACCCCCC:
```

The receive state machine is used to capture the AT commands received from the GSM module. Only limited functionality is required and decoding of the various messages are done by other drivers or part of drivers. The state diagram is showed in Fig. 6.17.

Initialization driver

The initialization driver runs in a single sequence. The flow of this state machine is shown in Fig. 6.18. This driver is initiated upon startup of the application layer, in both `GSMmode.h` and `onLine.h`. This is done to ensure that both applications have all the resources setup is exactly the same way.

Energy measurement driver

The energy driver is responsible for giving other drivers the current energy readings. With a cycle time of $T_{cycle} = 100\ ms$ the registers are read from the AD7758 driver and stored. Other drivers can access these registers directly from the memory. Each multi state in Fig. 6.19 consists of a request to the ADE7758 for a specific register and an accumulator update for all three phases, namely A, B and C.

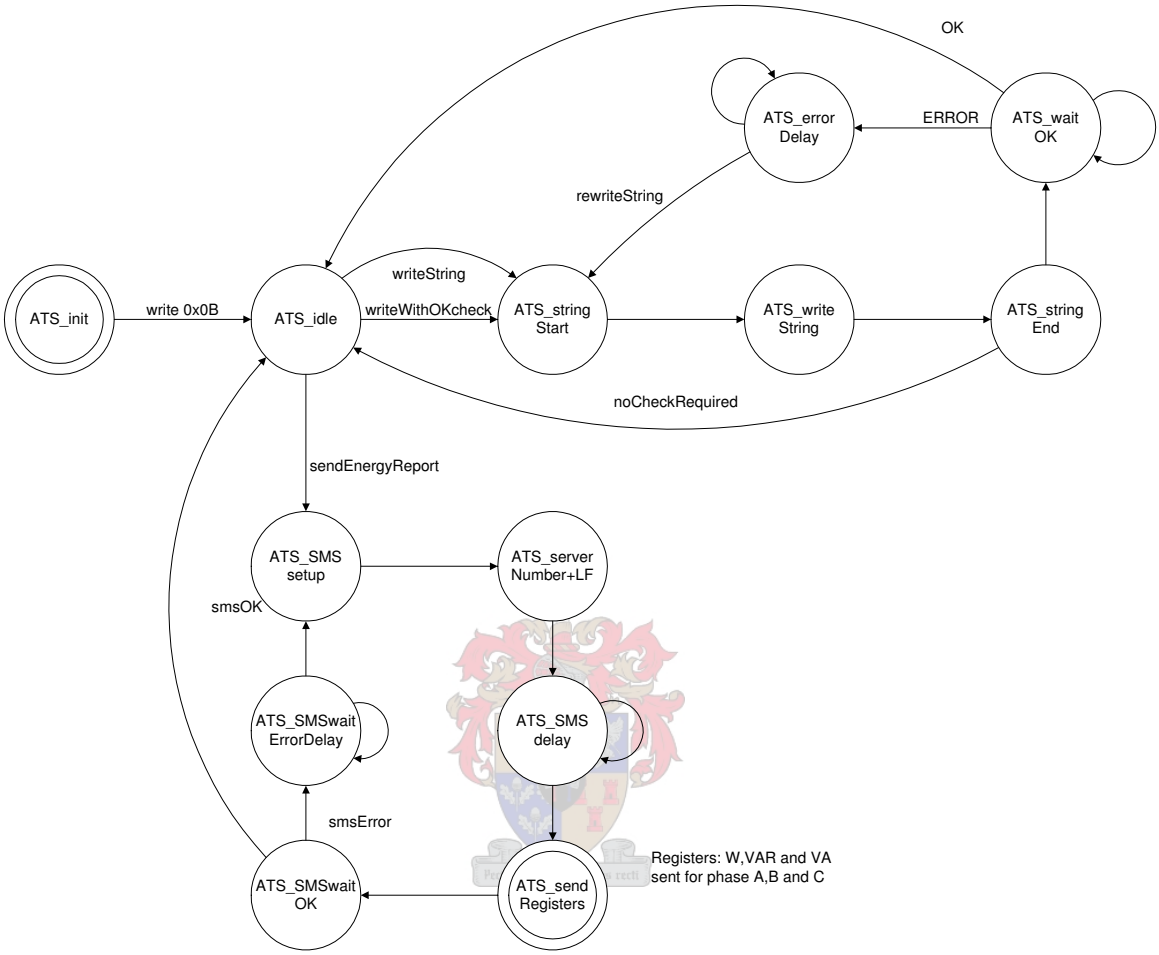


Figure 6.16: *State diagram for sending part of AT driver*

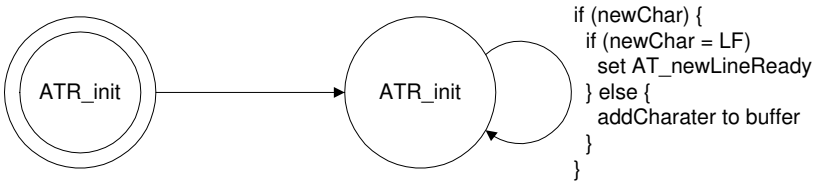


Figure 6.17: *State diagram for reception part of AT driver*

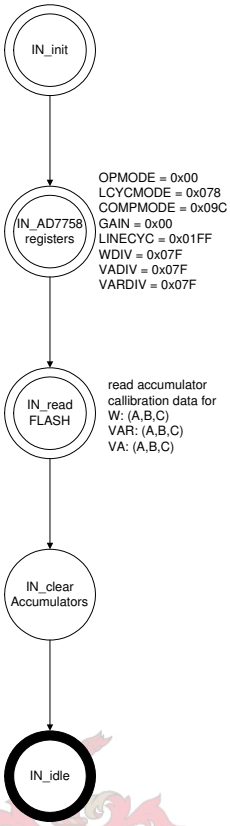


Figure 6.18: *State diagram of initialization process*

The strategy for measuring the energy is by reading the energy accumulators from the ADE7758. The ADE7758 is set up to reset the accumulator each time the registers are read. These register values are then added to the interim energy accumulators in the micro controller. If these are larger than the dividing factor, the values are added to the final accumulators. The whole process is illustrated using the following code for the real power calculation of phase A:

```
energy_W_a = energy_W_a + signedIntToLong(AD7758_getInt());
if ((energy_W_a >= MCU_WDIVA) | (energy_W_a <= -MCU_WDIVA)) {     //Scaling
   tmpLong = energy_W_a/MCU_WDIVA;
   energy_W_A = energy_W_A + tmpLong;           //Add new unit to final accumulator
   energy_W_a = energy_W_a - tmpLong*MCU_WDIVA; //Correction for rounding errors
}
```

Online mode

Since the `onLine.h` is responsible for handling the calibration routines, this driver is by far the largest driver of them all. Interaction takes place using the RS232 port. PC software is designed to interface with the check-meter and is discussed in Section 6.5. The various steps of the calibration process are discussed in Section 6.7.

A state diagram is presented in Fig. 6.20.

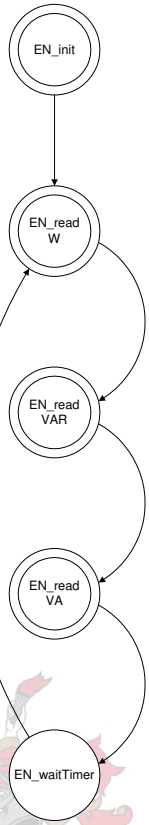


Figure 6.19: *State diagram of energy driver*

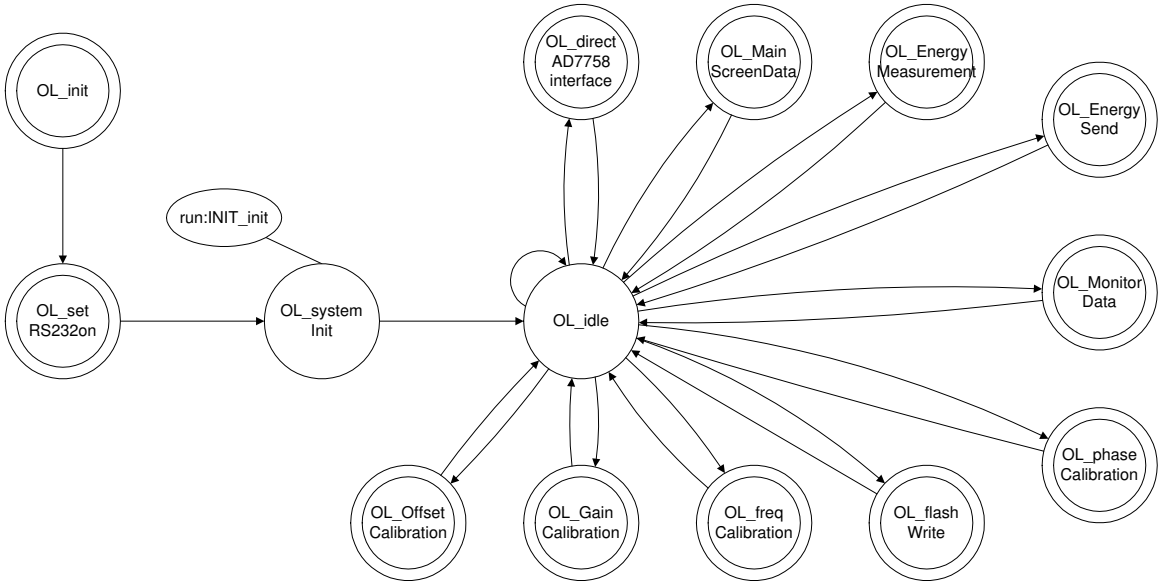


Figure 6.20: *State diagram of online driver*

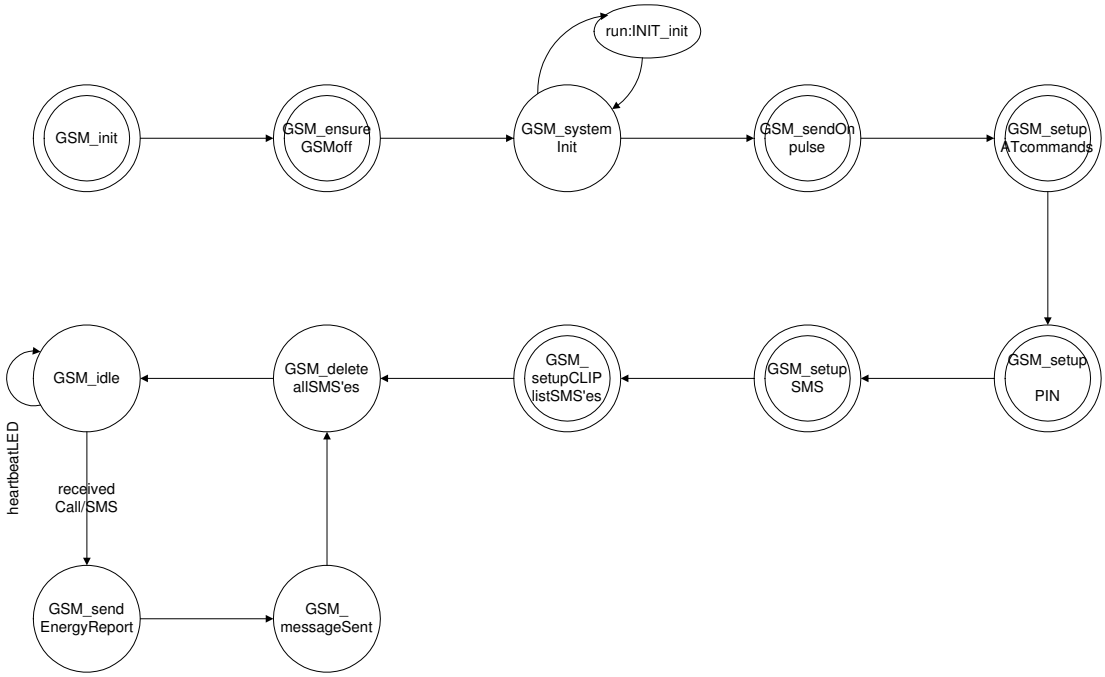


Figure 6.21: *State diagram of GSM mode driver*

GSM driver

The GSM driver uses the AT command driver extensively in combination with the UART driver. The UART driver is used to ensure that bus contentions do not occur, especially during startup.

An energy report is sent only when a mobile terminating call or message is received. In other words, when a message of some sort is received by the device an energy report is sent. It was decided to exclusively keep the logic at this level, to ensure that no “run-away” SMSs would be sent. The information frequency is set at the server. The client can only react to messages from the server, as can be seen in Fig. 6.21. The SMS server is responsible for capturing and initiating communications as discussed in Section 6.6.

6.5 Calibration PC Software

The calibration software was designed on the Java platform [83] in the Netbeans environment [59]. Since the data mining application in Chapter 5 was developed in Java, future integration and re-use of already written classes would be seamless.

The software is implemented using an event driven model based on user and communications input from the serial port. This section discuss the software while the calibration process will be discussed in Section 6.7 in more detail.

6.5.1 Communications

The software implements the serial communication using the `peg.io.serialInterface`(see Apendix Q.5.5) class using the Communications API [84] from Java to access the serial port. The `serialInterface` class is designed to aid and ease the use of the communications API.

The protocol used for communications is implemented in `peg.io.protocol` (see Appendix Q.5.4) and is designed to comply with the protocol used in Section 6.4.4.

6.5.2 Graphical User Interface

The user interface is done using Java's Swing API. The menu allows access to various tasks. The menu layout was established as follows:

File Allows the user to save the current setup of the energy measurement software.

Calibration Some general commands to aid the calibration process.

- **Calibration LEDs ON/OFF** Sets the LED's connected to the ADE7758 to be toggled on or off.
- **Write to Flash** Ensures the current calibration details are written to the Flash on the MCU.
- **Read Flash** Reads all the data on the flash and updates all the register in the MCU and ADE7758.

MCU This command requests that the micro controller sends a collection of important registers to the GUI.

Windows This menu provides access to additional windows for setup for specific tasks, as can be seen in Fig. 6.22. It is divided into a monitor and a calibration part. The following windows are available:

- **Energy Measurement** Allows the user to verify the calibration procedure.
- **Measurement** Allows the user to monitor voltage, current and the status of the energy registers on the ADE7758.
- **Other Measurements** This window shows the user information about temperature, frequency and the communications link status.
- **Registers** Various register are displayed here to provide feedback for the embedded software programmer.
- **Offset Calibration** A two-point gain and offset calibration is done on the RMS voltage and current.

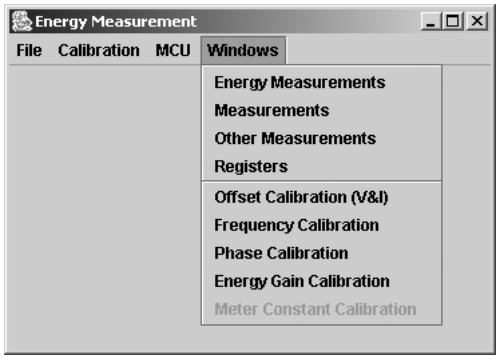


Figure 6.22: *Available windows in graphical user interface*

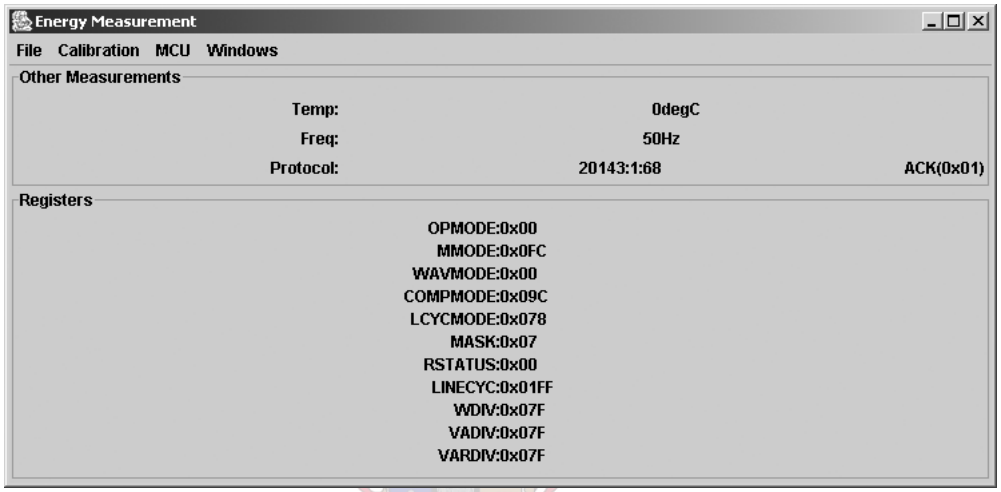


Figure 6.23: *Register and Other measurements*

Frequency Calibration The frequency is calibrated using this window.

Phase Calibration Slight phase errors introduced by the probes or current transformers is corrected for using the calibration window.

Energy Gain Calibration Depending on the current transformer or probe ratios the energy accumulators are calibrated.

Measurement Screens

The energy measurements screens are used to give feedback to the operator when calibrating to verify the calibration.

The *register* screen gives an overview of the important registers in the ADE7758, while the *other measurements screen* shows the current communication activities, line frequency and temperature. These screens are shown in Fig. 6.23. These screens are especially monitored when the unit is switched on for the first time to evaluate the correct initialization of the check-meter unit. All the registers can be dumped using the MCU — DUMP ALL REGISTERS command.

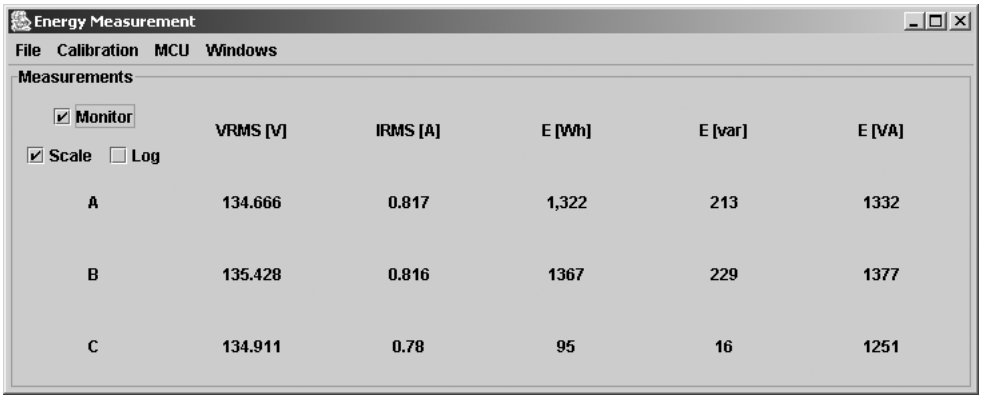


Figure 6.24: *Measurement screen*

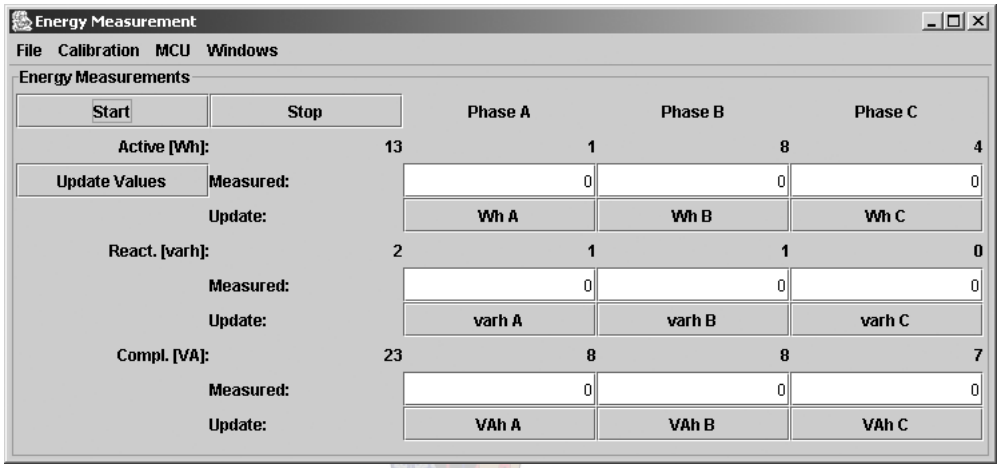


Figure 6.25: *Energy Measurement screen*

In order to verify the correct calibration values the *measurement* and *energy measurement* screens are used.

The measurement screen, as shown in Fig. 6.24, allows the user to activate the continuous update or monitor function to provide instantaneous feedback. The scale can be set to alternate between the actual register values of the ADE7758 and the stored calibrated values. With the log function enabled, the measured values are dumped to `c:\log.csv` and can be imported into a spreadsheet or database.

The energy measurement screen, see Fig. 6.25, presents the current micro controller accumulators, for all the phases and energy registers. The user can fine tune using the edit boxes on the screen. The calibration values are updated using the *Update Values* buttons. The accumulators and `energy.h` subsystem are started and stopped using the *Start* and *Stop* buttons provided on the screen.

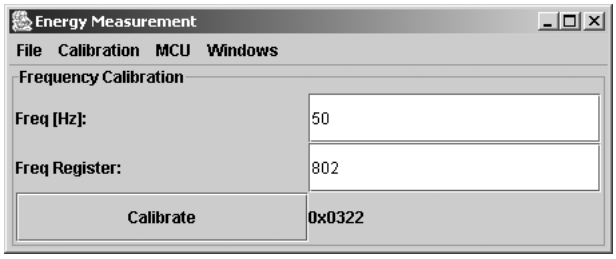


Figure 6.26: *Frequency Calibration screen*

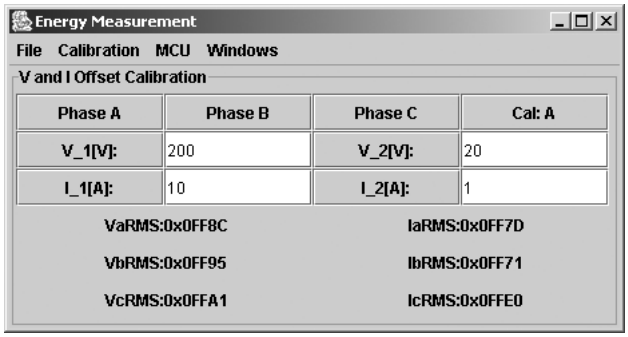


Figure 6.27: *Voltage and Current Offset Calibration screen*

Calibration screens

The first calibration screen is used for frequency calibration. This screen is fairly simple to use. The frequency value is entered in the appropriate text box and the *Calibrate* button is pressed to initiate the calibration process.

The Voltage and Current offset calibration screen is shown in Fig. 6.27. The top row buttons allow for selection of the current phase in the calibration process. The values are entered at two points in the text boxes. In order for the user to measure the value on the ADE7758, the button on the left of the text is clicked. This will start the measurement procedure for the specific value and currently selected phase. The system is calibrated for the specific phase using the top right button with the leading *Cal:* text.

To correct phase errors in the measurement path, the phase calibration screen is used, as shown in Fig. 6.28. For each phase, two power measurements at different power factors are measured. The previous measurements can be loaded using the *Update Form* button. The calibration data is calculated and sent to the check-meter by pressing the *Calibration* button.

To allow for various ranges of current transformers and probes the energy accumulators are calibrated using the Energy Accumulator Screen as shown in Fig. 6.29. For a first iteration the registers should be reset, using the *Reset* button. The current power flowing through the measuring point is entered and submitted using the *Cal Phase* buttons for the particular phase. Preferably this calibration should be done using a power factor of $pf \approx 0.7$. For fine tuning of these gain parameters the energy measurement screen can be used, as seen in Fig. 6.25.

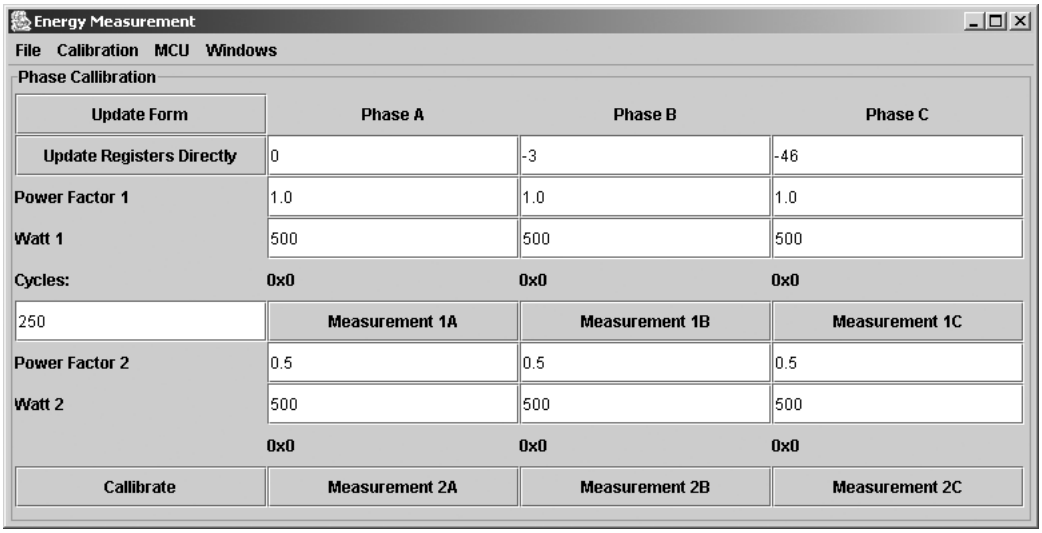


Figure 6.28: *Phase Calibration screen*

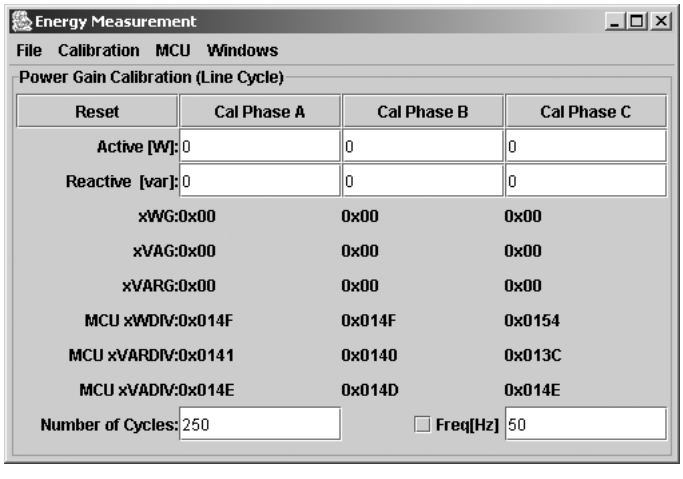


Figure 6.29: *Energy Accumulator calibration*

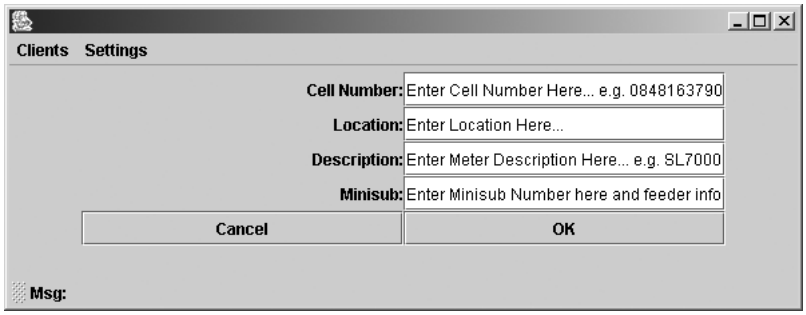


Figure 6.30: *Adding Clients screen*

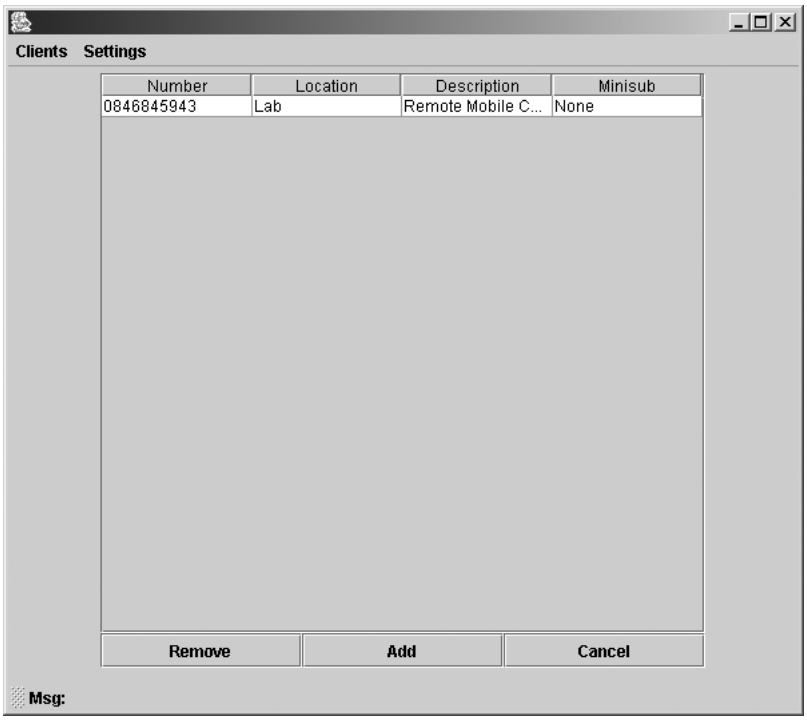


Figure 6.31: *View Clients screen*

6.6 PC Server Software

The PC server software allows the operator to send requests to the check-meter and to process reports received from the various check-meters and to store the data in a data warehouse.

A program was developed to allow users to easily set and modify server settings. Clients or check-meters can be added and viewed as shown in Fig. 6.30 and Fig. 6.31. A link to an SQL database can be set up using the Database settings screen in Fig. 6.32. The server settings allow a user to specify the numbers required for the GSM modem, the time the daily update should occur and the ability to force an update. The server settings are shown in Fig. 6.33.

The PC server is a simple eternal loop thread, as shown in Fig. 6.34. The thread interfaces to a GSM modem using the `peg.io.serialInterface`(see Apendix Q.5.5) class and an AT command set [93] class. Received SMSs are processed and stored in the database using the

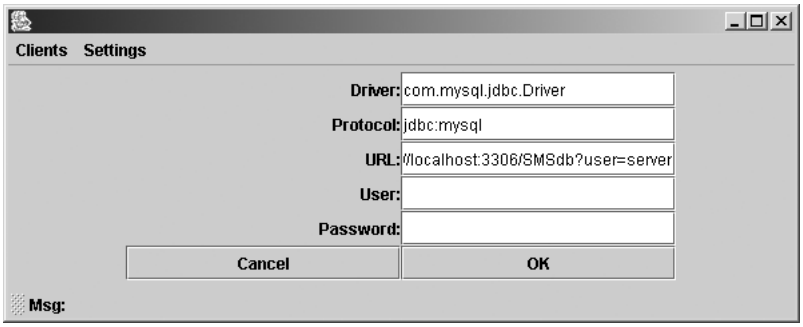


Figure 6.32: *Database Settings Screen*

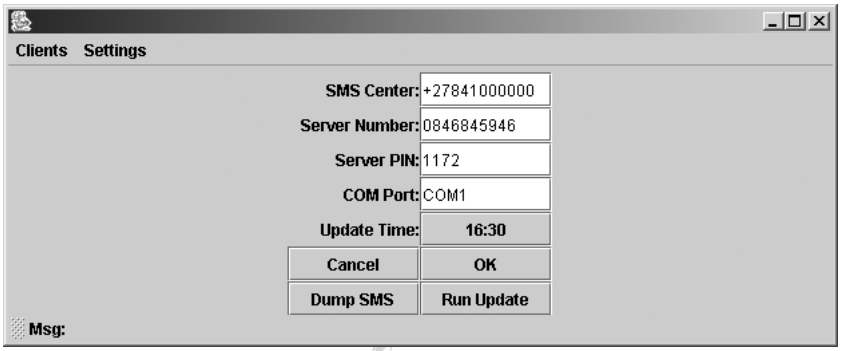
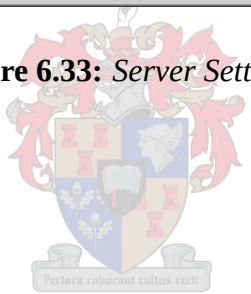


Figure 6.33: *Server Settings screen*



`peg.sql.DBConnect` class.

6.7 Calibration

Although it seems as if the ADE7758 requires an extensive calibration procedure, it can be broken down into four simple calibration steps. The calculations for the various parameters required by the ADE7758 and micro controller software, are done by the calibration software in the `energyMeasure.energyCalibration` class. When measurements from a user are required, it implies that an accurate measurement from an already calibrated device is used. It is important to note that the calibration accuracy is dependent on the accuracy of the device used to calibrate it against.

In order to understand the calibration process, the functional block diagram in Fig. 6.35 is presented to show how the ADE7758 calculates the various registers and how calibration registers fit into the equations.

The calibration process consists of the following steps:

1. The frequency is required to supply the other steps with correct timing data.
2. Offset calibration is used to increase the voltage and current accuracy.

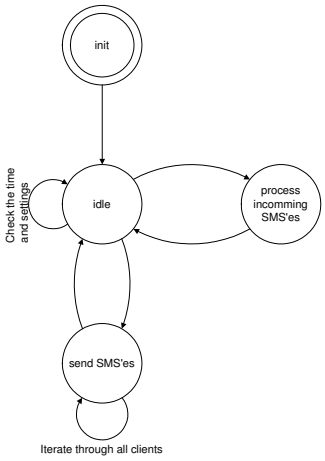


Figure 6.34: *State diagram of PC server program*

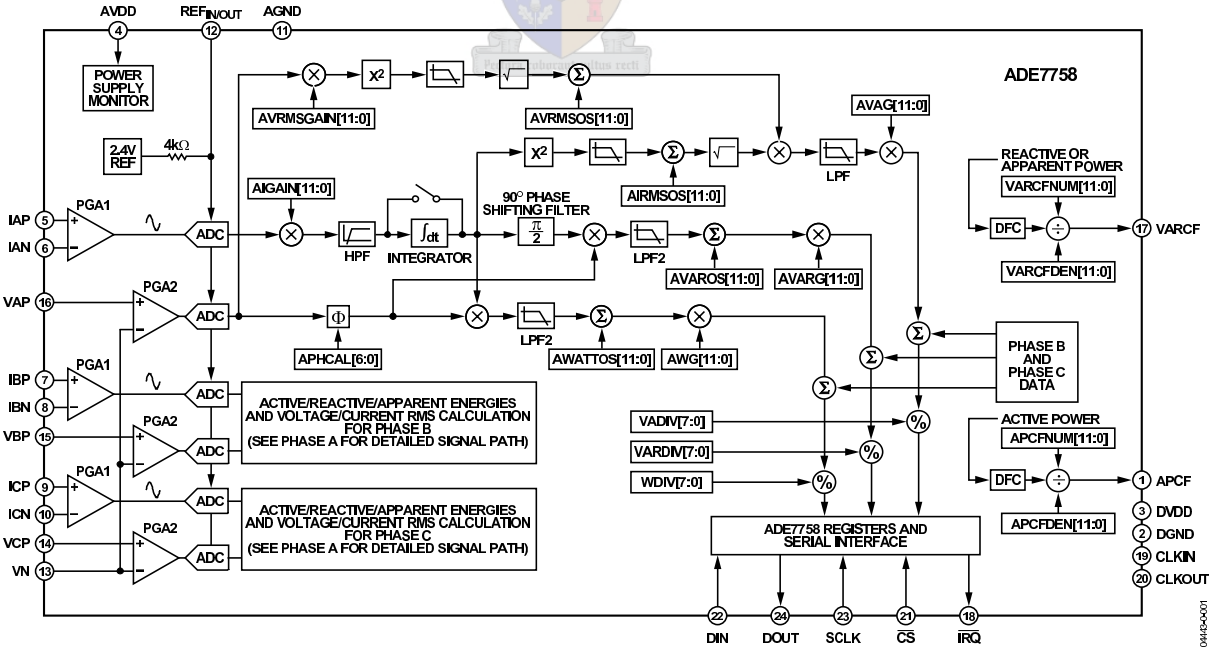


Figure 6.35: *ADE7758 Functional Block diagram [7]*

3. The difference in phase angle added to the current measurement path is calibrated to calculate the power factor and real power component accurately.

4. The energy accumulators are calibrated to include the additional gain of the current transformers and probes various ratios.

Each process can be broken down further into the following steps:

Screen selection The user is required to select the correct screen for calibration

Measurements Setup the equipment and measure the required values from the available equipment and input the values into the screens.

Acquire the ADE7758 registers The register values of the measurements on the ADE7758 must be retrieved and stored in the program.

Calculation and parameter storage Once all the measurements have been taken the parameters are calculated and stored on the ADE7758.

6.7.1 Frequency Calibration

The frequency calibration is done using a single point and calculating the LSB gain. This gain is only used for calibration purposes and is therefore, only stored in the application. The frequency calibration is done on Phase A. This, however, can be set in the software by changing the FREQSEL mnemonic of the MMODE register of the ADE7758.

When the frequency calibration process is initiated by the user, a request is sent to the check-meter to retrieve the current frequency value of the FREQ register. This is sent back to the calibration software and with the known frequency from the user input the gain is calculated. The process is outlined in Fig 6.36.

6.7.2 Voltage and Current Offset Calibration

This voltage and current offset calibration is not required for the active or reactive energy measurement, but is done firstly to give the user a good idea if the system is working and to determine the accuracy of the analog to digital converters of the ADE7758. Secondly, this calibration process is used by the apparent power calculation, which is not that important for this thesis's application. All measurements are done in RMS unless stated otherwise.

The LSB_{gain} calculation is only used in the calibration software, but the offset is sent to the ADE7758. The gain registers in the ADE7758 are not used to prevent any quantization errors, since fixed point number representation is used within the ADE7758, and the total gain is rather

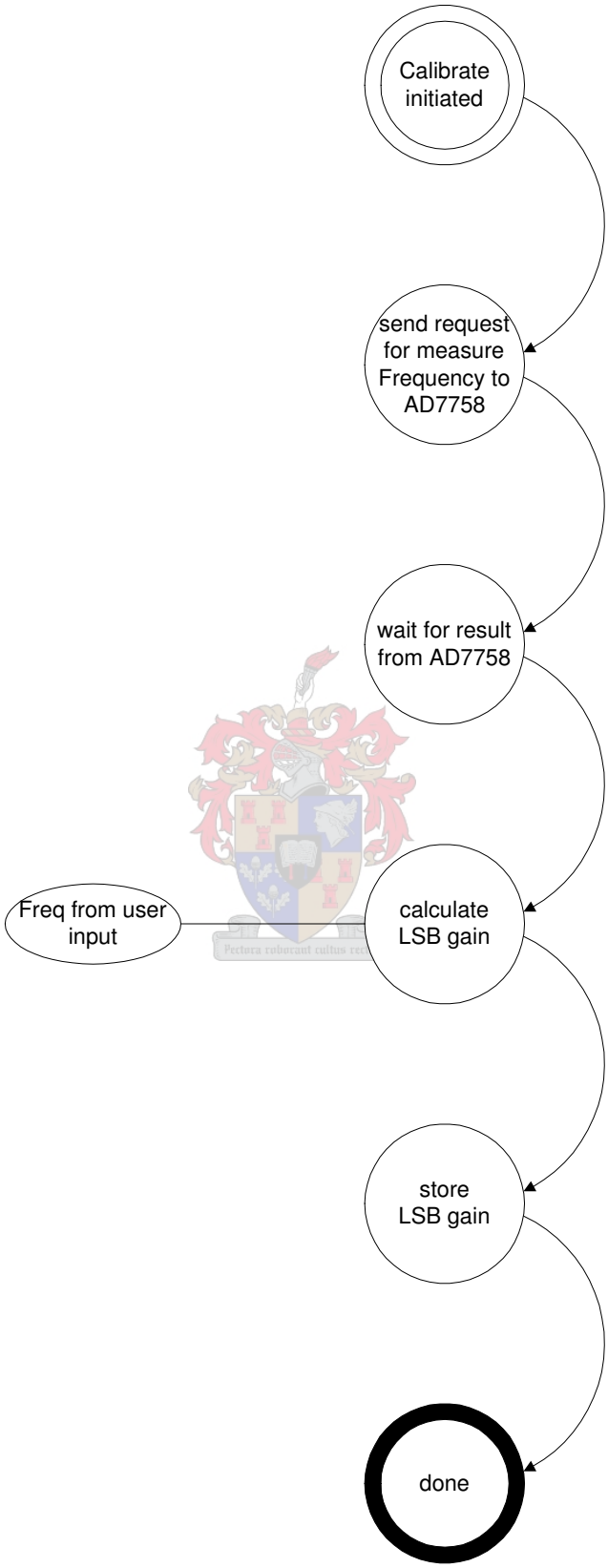


Figure 6.36: *Frequency Calibration process*

adjusted at the end by setting the gain for the energy accumulators. The voltages and currents are calculated in the ADE7758 using (6.4) and (6.5) according to [7].

$$V_{RMS} = V_{RMS_0} + 64 \times V_{RMSOS} \tag{6.4}$$

$$I_{RMS} = \sqrt{I^2_{RMS_0} - 128^2 \times I_{RMSOS}} \tag{6.5}$$

Using the calibrated V_{RMS} and I_{RMS} received from the ADE7758 the calibration software calculates the scaled values by multiplying the V_{RMS} and I_{RMS} with the LSB_{gain}.

To calibrate the voltage and current, two values are required in order to calculate the offset and the gain. After the two measurements for both current and voltage are taken by entering the values in the GUI, the calibration can be initiated (see Section 6.5.2 and Fig. 6.27).

When the measurement process is initiated by the user, the entered value is stored and the current measurement is requested from the ADE7758. The returned result is also stored. After all four values are gathered, the user can calibrate the selected phase using the CAL: button.

The voltage offset is then calculated in the application:

$$V_{offset} = \frac{V_1 V_{2_{ADC}} - V_2 V_{1_{ADC}})}{64(V_2 - V_1)}$$

The current offset in the diagram is found after the square root of the measurement is taken and therefore:

$$I_{offset} = \frac{I_1^2 I_{2_{ADC}}^2 - I_2^2 I_{1_{ADC}}^2}{128^2(I_2^2 - I_1^2)}$$

The LSB gain is calculated using the first value from the user input and is divided by the measured ADC value for current and voltage and stores it:

$$LSB_{gain} = \frac{M_1}{M_{1_{ADC}}}$$

where M indicates the measurement as either V or I.

The process is shown in Fig. 6.37.

6.7.3 Phase Calibration

CTs and current probes phase errors can be expected to vary from CT to CT [7] and probe to probe[48, 49, 47]. The phase error has a direct impact on the active energy calculations and therefore this is an important parameter to incorporate into the calculation path.

Hardware considerations

The ADE7758's phase correction can compensate for -2.72^o to 1.36^o at $50\ Hz$ using the xPHCAL registers. In some cases the lagging phase error could be as much as 10^o [47]. This lag should be corrected for realizing a phase lead by placing a capacitor parallel to current input stage of the ADE7758, as shown in Fig. 6.38.

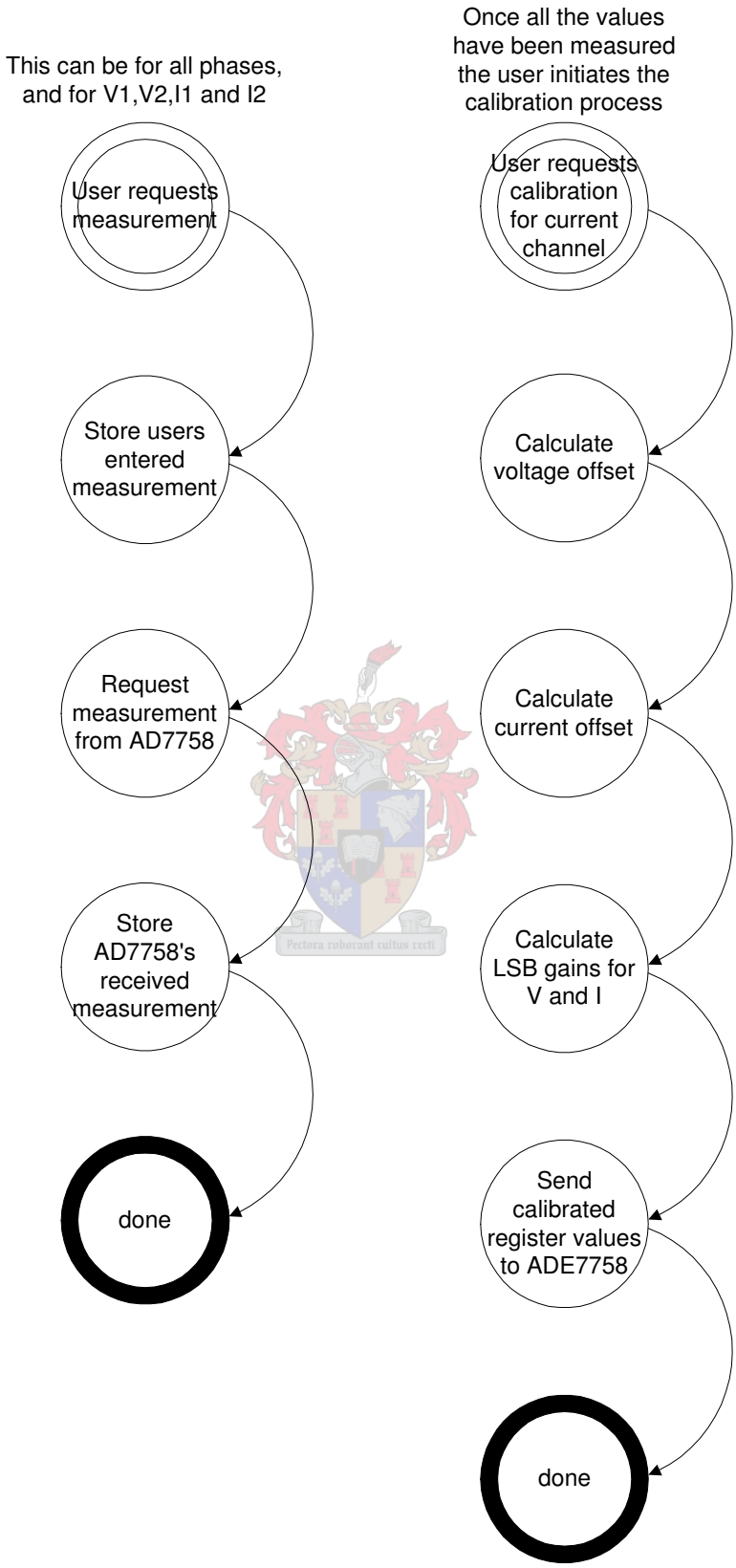


Figure 6.37: *Voltage and Current Offset Calibration process*

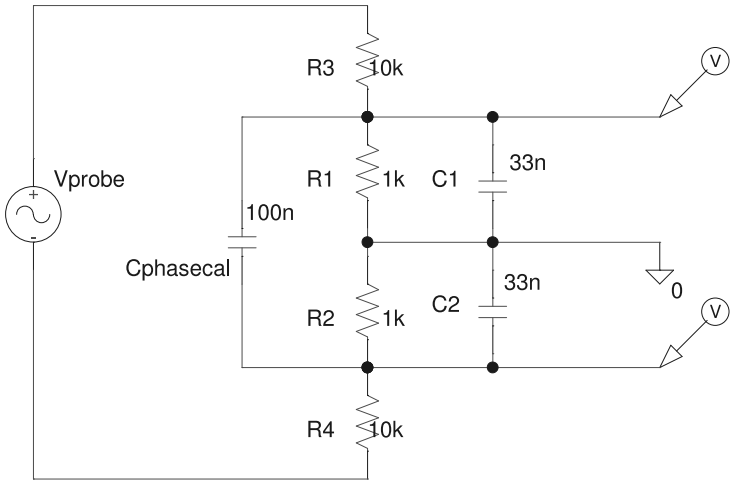


Figure 6.38: *Phase Calibration Hardware Implementation*

The required lead capacitor can roughly be calculated using the RC constant as follows:

$$Phase_{lead} = 3.6\,^{o} \tag{6.6}$$

$$t_{lead} = \frac{Phase_{lead}}{360^{o} f_{line}} \tag{6.7}$$

$$= 200\ \mu s \tag{6.8}$$

$$C = \frac{t_{lead}}{R} \tag{6.9}$$

$$= 100\ nF \tag{6.10}$$

where $Phase_{lead}$ is the necessary phase lead required, t_{lead} the lead time required, f_{line} the line frequency at $50\ Hz$ and C the required capacitance to realize the phase lead.

Software implementation

In the graphical user interface the phase calibration screen is used. This screen allows the user to enter two power measurements at two different power factors. Each of these measurements can be read from the ADE7758 using the appropriate button. Once all the measurements are taken the calibration calculations can be done and the registers of the ADE7758 can be updated.

The phase error is calculated in `calPhaseOffset` using the following equations. For the procedure the following values are required for each measurement:

pf_x the power factor of the measurement

W_x the measured power

$W_{x_{ADE7758}}$ the power measured by the ADE7758.

With this calculation the first measurement should have a power factor as close to $pf \approx 1$

as possible and the second measurement should be near $pf \approx 0.7$.

$$Error = \frac{\frac{W_1}{W_{1_{ADE7758}}} - \frac{W_2}{W_{2_{ADE7758}}}}{\frac{W_2}{W_{2_{ADE7758}}}} \tag{6.11}$$

$$\Delta pf = pf_2 - pf_1 \tag{6.12}$$

$$phaseError = \arcsin \frac{Error}{\sin \Delta pf} \tag{6.13}$$

If the phase error is negative

$$phCal = \frac{phaseError}{(2.4 \times 10^{-6} \times 360 \times 50)} \tag{6.14}$$

otherwise

$$phCal = \frac{phaseError}{(4.8 \times 10^{-6} \times 360 \times 50)} \tag{6.15}$$

The phase calibration register, $phCal$, is updated in the ADE7758. To further refine the values the register can be updated directly.

6.7.4 Gain Calibration

The final gain calculation stage is implemented in the Energy Measurement Driver. The code for calculating these registers is explained in Section 6.4.4. This step is a one point LSB_{gain} calibration. The meter is expected to send values in Watt Hours. Each LSB should represent one Whr, $VARhr$ and $VAhr$. The calibration values are entered into the calibration software with a $0.7 < pf < 0.76$ to have an active and reactive energy component. This measurement should be taken at nominal current and voltage as specified for the meter and current probes. Once all the measurements are taken the calibration constants can be calculated and the registers updated.

This measurement is done over a 250 line cycle period. The energy measured in the ADE7758 is then used with the externally measured power values.

The energy division registers are calculated as follows:

$$t_{totalCycle} = 250/(2 \times f_{line}) \tag{6.16}$$

$$LSB_{gain} = \frac{energy_{ADE7758} \times 3600}{t_{totalCycle} \times power} \tag{6.17}$$

where the power already takes the power factor into account.

The registers are now updated into the micro controller.

6.8 Calibration and Test Results

The calibration was done using a Voltec PM3300 Universal Power Analyser as reference. The system was calibrated using a three phase variac, with three paired $900\ W$ loads and a $25\mu F$ per phase capacitor bank to provide for a leading power factor.

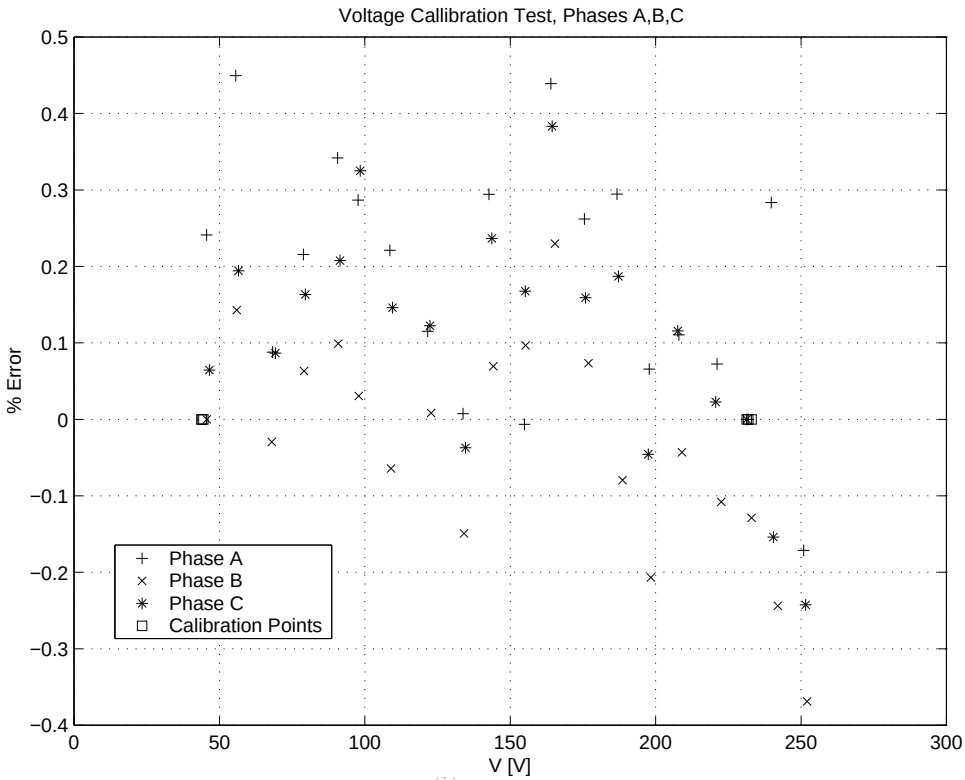


Figure 6.39: *Voltage Calibration results*

6.8.1 Voltage Calibration

The voltage calibration was done using the points in Table 6.5:

Measurement	Phase A	Phase B	Phase C
V_1	231.3 V	233.0 V	231.9 V
V_2	43.85 V	43.94 V	44.49 V
I_1	4.107 A	4.144 A	4.000 A
I_2	399.0 mA	404.0 mV	428.9 mA

Table 6.5: *Voltage and current offset calibration*

The results from the voltage and current calibration experiments are shown in Fig. 6.39 and Fig. 6.40.

Around the nominal point an error of $0.25\ \%$ is seen for the voltage and $0.3\ \%$ for the current calibration. This matches the typical expected error as provided in the data sheets [7].

6.8.2 Phase and Gain Calibration

The phase and gain calibration is done using the resistor and capacitor bank to absorb the required energy.

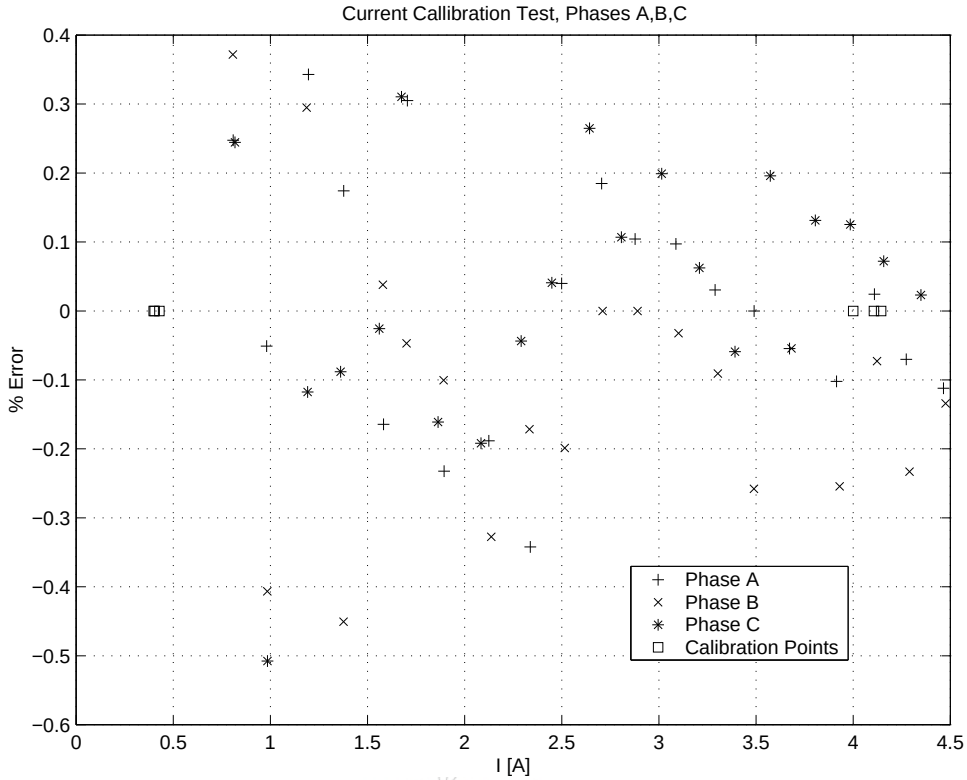


Figure 6.40: *Current Calibration results*

Prior to the gain calibration the phase calibration is done by setting the cycles for the calibration process to $1\ 000$. Two measurements for each phase are made and are shown in Table 6.6

Energy Register	Phase A	Phase B	Phase C
PF_1	0.982	0.986	0.984
W_1	350.2	380.6	373.4
$W_{1_{ADE7758}}$	0x0149	0x0161	0x015C
PF_2	0.715	0.728	0.734
W_2	351.8	375.0	376.4
$W_{2_{ADE7758}}$	0x0144	0x0161	0x0156

Table 6.6: *Values used for phase calibration*

The gain calibration procedure is done at $V_{nom} = 223\ V_{RMS}$ and $I_{nom} = 2.15\ A_{RMS}$ and fine tuned by allowing a $16\ min$ energy accumulation.

The results when testing the phase and gain calibrated check-meter yielded the results in Table 6.7.

Due to quantization and test synchronization errors caused by the radio link the percentage

Power Factor	Voltage [V]	Current [A]	Active E. [Whr]	Reactive E. [VARhr]	Apparent E. [VAhr]
0.7	223	2.15	1.30 %	0.48	-0.08 %
0.984	223	1.66	2.27 %	-8.47	0.09 %
0.984	225	1.66	1.49 %	-9.19	0.04 %
0.987	225	3.56	1.39 %	-11.99	-0.74 %
0.987	225	3.56	1.39 %	-12.58	-0.54 %
0.942	225	3.73	0.39 %	6.67	-0.29 %
0.003	225	1.84	0 %	-2.27	1.46 %
0.939	189	3.16	0.79 %	6.67	-0.63 %
0.939	189	3.16	1.47 %	-12.65	-0.91 %

Table 6.7: *Percentage Error per accumulator*

error decreases as time increases as can be seen in Fig. 6.41. Fig. 6.42 supplements Fig. 6.41 by showing only the active power component and the error associated with this accumulator. These results show that there is room for improvement with these measurements. For the purpose of this thesis, however, the results attained are good enough. Incorporating the energy accumulators offset register can be investigated to improve the measurements.

6.9 Conclusions and Recommendations

The check-meter was developed and tested according to the requirements.

The calibration tests gave good results for the voltage and current measurements, but the energy accumulator's results are only fair, and in this area there is room for improvement. An additional active and reactive energy offset can be considered to further increase measurement accuracy.

The construction, assembly and implementation requires some refinement, and should be tested for weather-proofing prior to any field tests.

The check meter is designed in such a way that it can easily be expanded or manipulated for other applications. Possible applications could include peak and sag or dip detection, which can be used for flicker and quality of service measurements. A general data logging capability is also a possibility. An auto memory dump should also be considered when power is removed from the meter. However timing and energy management should then quickly be applied to ensure the integrity of the measurements. The possibility of sending an off status SMS can also be investigated.

Combined with the developed data mining application this device can prove helpful for field analysis and energy measurements in remote areas.

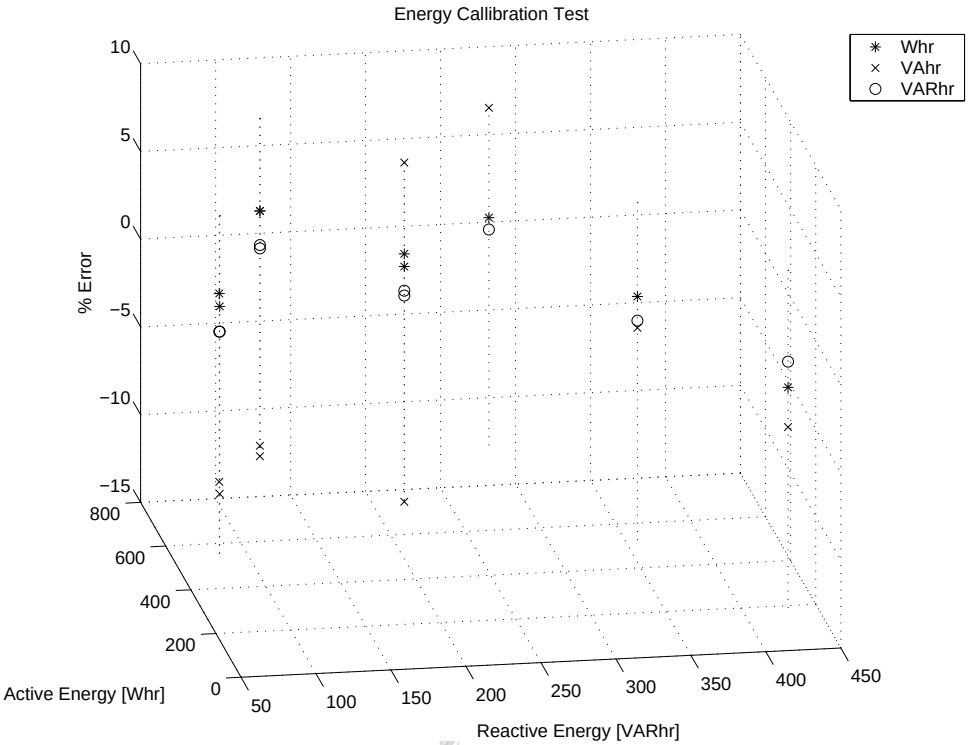


Figure 6.41: *Gain Calibration with Phase correction results*

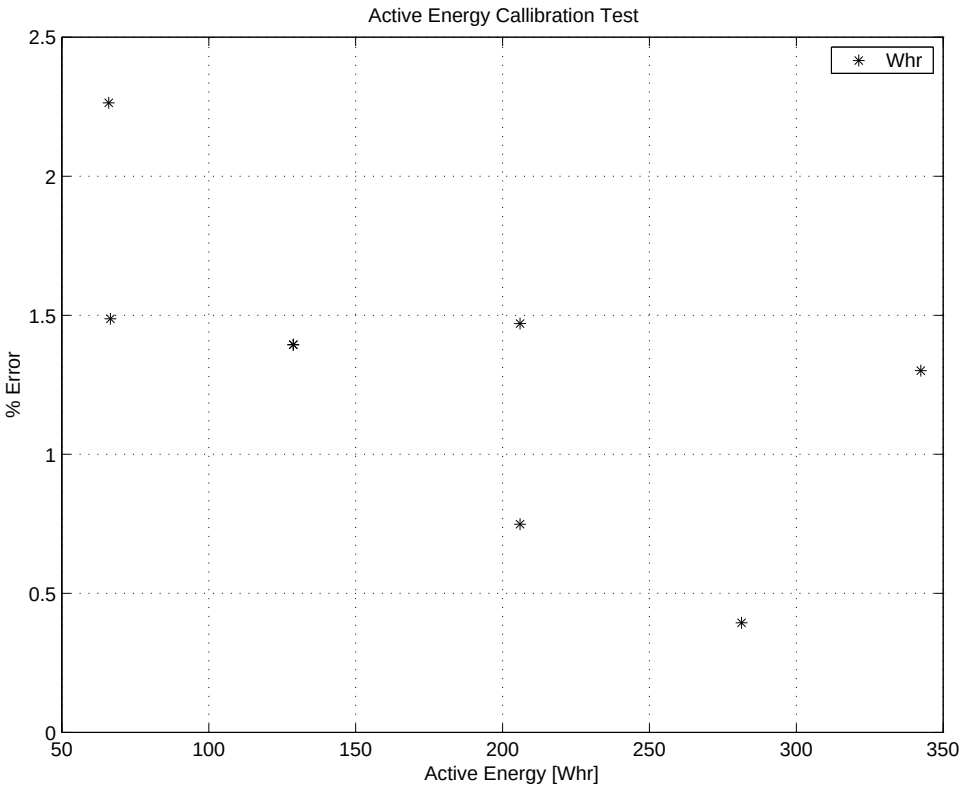


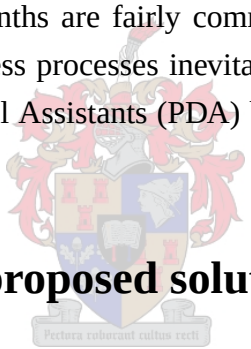
Figure 6.42: *Active Energy Calibration results*

Chapter 7

Data Collection Software Development

7.1 Introduction

Detection of electricity theft based on previously developed data mining solution depends on the quality of the data. Conventional meters are read by meter readers roughly every month, but periods longer than three months are fairly common. Longer measurement periods and a number of possible erroneousness processes inevitably have an effect on data quality. This chapter presents a Personal Digital Assistants (PDA) based solution, whereby data in the field is captured accurately.



7.2 Motivation and proposed solution

To bill consumers for their electricity usage means that their electricity meters must be read frequently and accurately. This is typically done by meter reading personnel who walk along the route and write the required values into a notebook. The readings are then entered into the municipality's debtor system by clerks in the back office.

The data collection involves two human processes. Each of these processes are prone to error. By using a hand held terminal this can be reduced to one process. The amount of errors can further be reduced by incorporating simple checks and intelligence into the hand held terminal before validating the data on the system. Reducing errors not only saves money in terms of incorrect bills and the client service associated with incorrect bills, but time is also saved by reducing the number of meter reading iterations per bill.

The idea to use hand held terminals is well known [54, 22, 95]. Models are currently available on the market, but these are expensive and bulky[95]. The recent advance in PDA technology and the affordable pricing of these units make it economically viable to implement a PDA based meter reading solution.

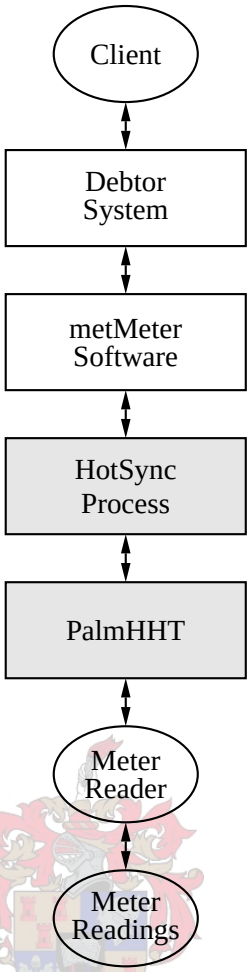


Figure 7.1: *Process Flow and System Overview*

7.3 Functionality of System Components and Platform

An overview of the proposed system is given in Fig. 7.1 with the system development for the thesis shaded in gray. The human processes are indicated in the ellipse areas.

The system comprises of the following components:

PDA The palm is responsible for collecting the data in the field.

PDA Software The software used for capturing and storing the data on the PDA.

Conduit The software necessary for uploading and downloading the meter readings out of and into a specified database of the metMeter Software.

7.3.1 Personal Digital Assistants

The PDA, introduced in 1993 [37] by Apple, is a small device able to provide mobile computing capabilities. Various products are available and this market is changing at a rapid pace with new

models being released daily. Because of this development it was decided to use a Palm OS based product, since Palm has been successful in this area since 1996 [61] and has provided an economical product. The development is tested on the Zire 21 and Zire 71 (Fig. 7.2) series.

Figure 7.2: *The Palm Zire 71 PDA*

7.3.2 Palm Software

The software development platform chosen for the project is Java[1] [86]. Java software can be ported to other platforms without major changes, since these devices are developing rapidly. Code can be written easily in a reusable manner. To run the software on the palm the Java code is compiled into a JAD file by the Java compiler. This file is then converted with the converter tool [86] into the Palm PRC file. The “palmHHT.prc” and “MIDP.prc” files are uploaded to the palm. The application is started by clicking on the “palmHHT” icon.

The meter reading software operating on the palm has a simple interface to even allow users with basic skills to enter data into the system.

The simple screen is displayed to allow the user to enter data as shown on Fig. 7.3. The following fields are displayed to aid the users in the entering the correct data:

Owner The owner of the premises or meter

Address The physical address of the meter

Meter ID The meter number, for example if more than one meter is located at the same address.

Meter Type Several types of meters can be read with this system, for example W-Water, E-Electricity and K-kVA or maximum demand meters[2].

---

[1] J2ME MIDP for Palm OS

[2] The meter type is indicated in the square brackets on the screen

Figure 7.3: *Palm software screenshot for user input*

Account Number The owner's account number.

Meter Number The serial number of the meter.

Messages Warning messages or special instructions are displayed in these fields, for example, "Fetch the keys at the neighbor's".

Depending on the specific setup[3] the following input fields are available:

Reading The reading displayed on the meter must be entered here. If the value is out by $\pm 10\%$ of the average consumption, the user will be requested to re-enter the reading.

Messsage This field allows the meter reader to indicate wether there is a reason for an incorrect or complete lack of a reading. For example if the meter is broken or locked.

New Message If a message different to the messages stored on the palm is required, the meter reader can create a new message and store it on the device.

Users can skip forward and backward between records with the BACK and NEXT button. The data is validated by pressing the OK button.

7.3.3 HotSync Conduit

Synchronization of the data on the palm is done with Palm's HotSync process. This software is supplied by Palm with the palm device for the Palm Desktop Environment. Each different program or application is required to have a conduit. The conduits knows the data structure of

[3]The setup is contained in the ".prc" file and generated for each user

the application on the palm, namely where and how the data should be stored or received on the PC. Fig. 7.4 shows the HotSync process screen.

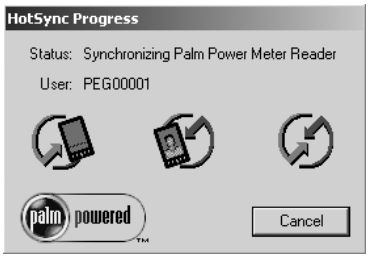


Figure 7.4: *A screen shot of the HotSync process*

The conduit was developed using Palm's JSync Java Conduit Development Kit [63].

When synchronizing, the conduit uses the specific palm unit's username to determine which route should be downloaded and uploaded. The completed route and messages are then downloaded and stored in flat files, ready for post processing by the metMeter[4] software. Next the new route, prepared by the metMeter software is uploaded and the messages for the specific route are updated.

7.4 Software Design and Implementation

The software of the system is broken up into two sections namely the Palm Software and the HotSync Conduit, which are discussed in the following subsections. Fig. 7.5 is shown to help the reader understand how the palm software fits into the rest of the system.

7.4.1 Palm Software

Program Flow

To program the hand held device the Mobile Independent Device Profile (MIDP) is used. This profile provides a common platform for programming mobile devices in Java to allows cross-platform portability. In the Java or MIDP environment an object is created which extends the `midlet` object. This object is first constructed in the MIDP environment and stays resident until the application is removed from the Palm[5]. On the Palm the program data and state are preserved when the power is switched off and program execution continues after the power is switched on later.

[4]This is the back-office software which interfaces with the debtor system and is used with permission from MetGovIS

[5]Source code can be found in Appendix Q.3

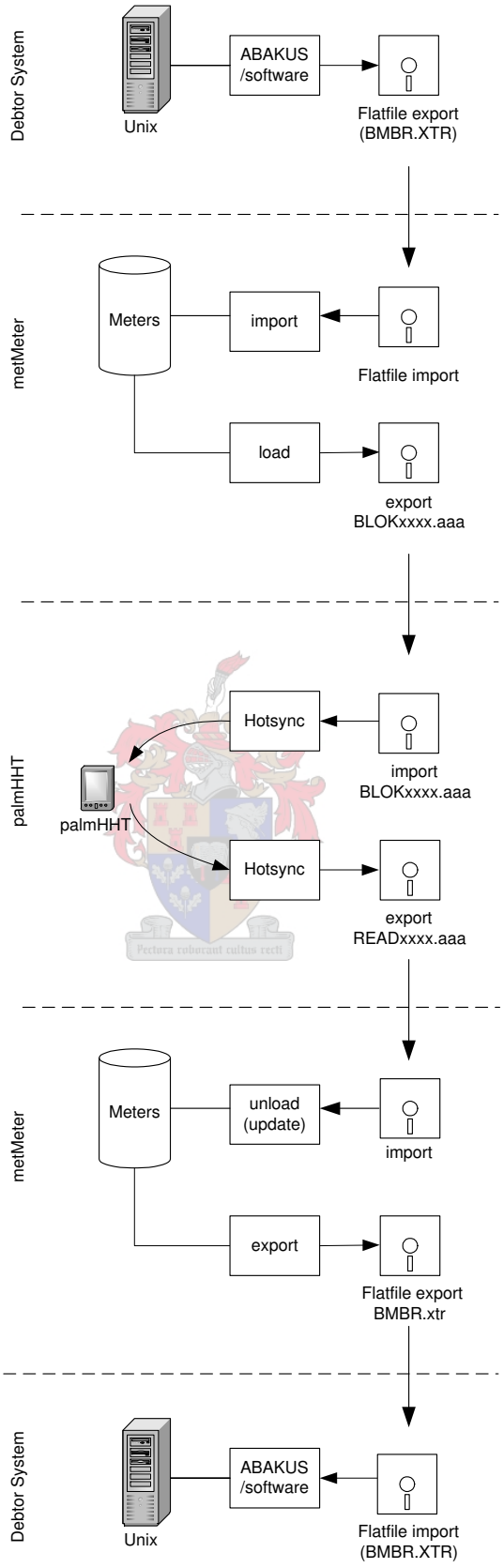


Figure 7.5: *The hand-held system within the total system*

As mentioned above, the program or object is initialized with a constructor `palmHHT()`. The constructor is responsible for ensuring that valid records are in place for the route, before the program can continue as shown in Fig. 7.6. If the route is not correctly uploaded a synchronization process should be initiated by the user. The constructor also sets up the first screen to be displayed (See Fig. 7.3).

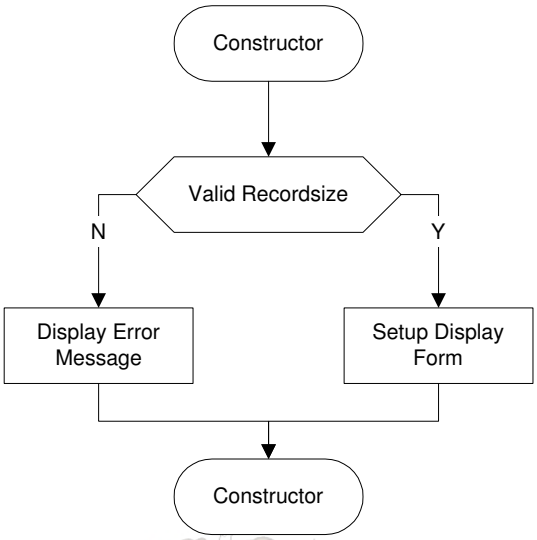


Figure 7.6: *Flow Diagram of Palm Software Constructor*

Each time the program is called or started by MIDP the `startApp()` method is called. The `startApp()` method activates the first screen and invokes the command listener.

The objects on the form or screen can be updated by the user. Once one of the buttons is ticked with the stylus, the `commandAction(...)` method is called and the events serviced. Fig. 7.7 shows the flow diagram of the implemented code.

Data structures

All data in the data records are stored as strings. This makes importing and exporting to and from the data structure simple. The fact that Strings are used does however require additional processing on the palm.

The data structures are stored in a resizable array on the palm. The data structure is accessed with the MIDP `RecordStore` object. The `RecordManager` object is created to encapsulates the `javax.microedition.rms.recordStore` for easy access to the stored data. This method uses the enumeration[6] method of the `RecordStore` to access the data. This is the only object allowed to directly access the records store.

In the palmHHT software the messages are stored in the `MessageInfo` object. The `HeaderInfo` and `MeterInfo` object are used to store all the route specific data. The

[6] A way to put the records in a sequence

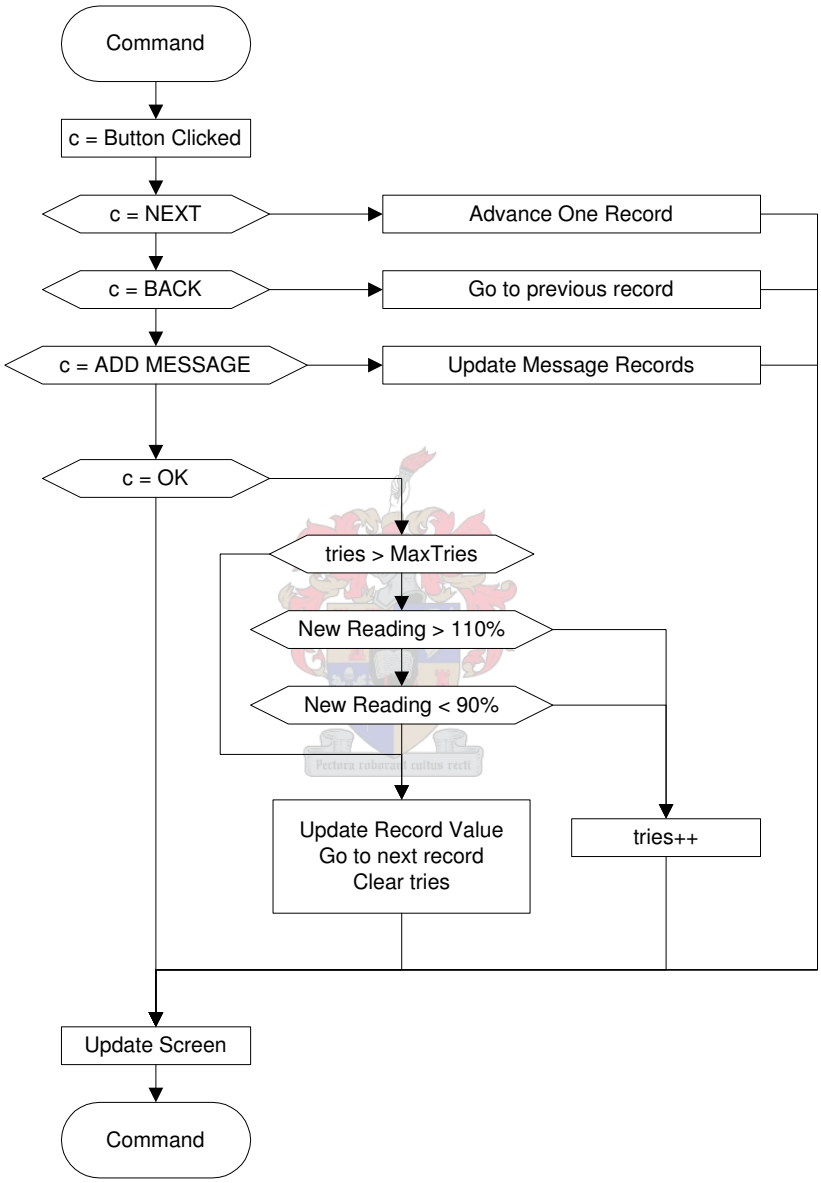


Figure 7.7: *Flow Diagram of Command Action Event*

`MessageRecords` and `RouteRecords` inherits methods to access the `RecordStore` from the `RecordManager` class. Fig. 7.8 illustrates the object components and inheritance of the data objects.

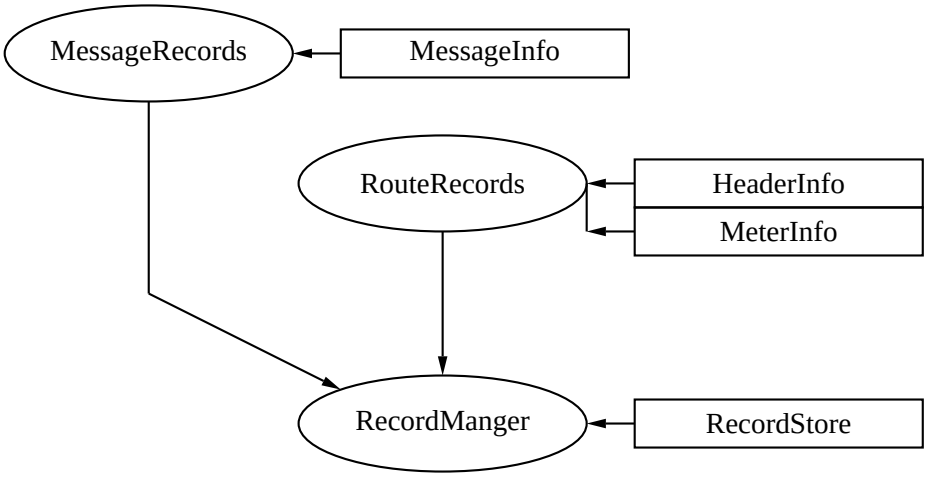


Figure 7.8: *PalmHHT software Data Structure*

The complete source code for the Palm Software can be found in Appendix Q.3.

7.4.2 HotSync Conduit

Program Flow

The HotSync code implements the Conduit abstract class which initiates the synchronization process by calling the `open(...)` method in `HHTCond`. An overview of the program flow is given in Fig. 7.9 and can be referenced in Appendix Q.4.

The setup is managed by the `green.HHTSync.HHTSyncGUI` object and creates a XML file containing the setup on the specific machine.

Prior to loading any data, the correct path, where the data files are stored and retrieved, is loaded from the setup file. First the palm User is retrieved from the palm to ensure the correct configuration file is read for the palm[7]. This file contains the header information which can now be uploaded onto the palm.

The data of the route completed prior to this synchronization process is now downloaded into a flat file. The route number is used to identify this block of data and the flat file is saved as “READxxxx.dat”, where xxxx is the route number.

With the old data stored the new data can be uploaded. This route number is used from the header information for the new block of data. The data from the “BLOKxxxx.DAT”, where xxxx is the new route number, is uploaded onto the palm.

[7]This is an “username.PCT” file

The last process downloads all the messages from the old route in a file “RTMSxxxx.DAT” and uploads this route’s messages from “MESSxxxx.DAT” where xxxx is the route number of both the old and new route.

Supporting methods are also included in the HHTCond and can be seen in the source code in App. Q.4.

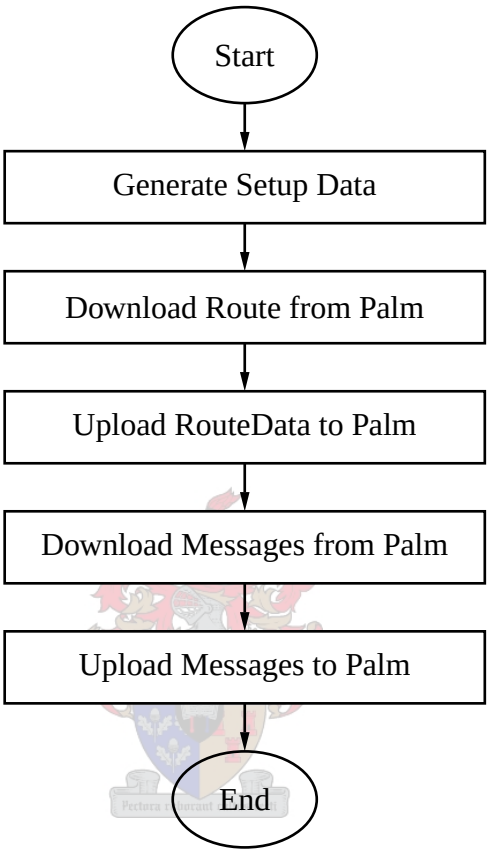


Figure 7.9: *HotSync software proces flow*

Data Structures

The data structure on the palm is accessed in the conduit by using the `StringRecord` which inherits the `palm.conduit.AbstractRecord`. Records on the palm are accessed using data IO streams. Each record in the `StringRecord` class is encoded or decoded by using the `setString()` and `getString()` after the record’s index is specified by `writeRec()` and `readRecByIndex()` of the `SyncManager()` class.

Two different record stores or sets are used, namely the `MessageRecords` and the `RouteRecords`. The `MessageInfo` records are stored sequentially. The `RouteRecords` contains a `HeaderInfo` part. Each `MeterInfo` is added sequentially after the `HeaderInfo`.

The data required for import is contained in a flat file generated by the metMeter software from the Debtor System. The format of these files have a fixed width and are accessed

by the `flatFile` class. The import of data by the `flatFile` class is done by defining a `dataStructure` array with the width of each field for example,

```
private int[] dataStructure = {12,3,14,4,1,18,2,8,8,8,3,4,8,2,8,1,6,1,3,3,3,8};.
```

This structure is then referenced to retrieve the number of fields and length of each field per line. Each line is then iterated and added to the route on the Palm to form the Route.

7.5 Installation

In order to get all the different components together an installation procedure is required. It was found that a solution from `http://www.handx.net` is most suitable for the application. The installation procedure is programmed in a XML file, which can be found in Appendix P.1.1. It was found that the JSync software is designed with Java 2 Standard Edition v 1.3. To be compliant with code created for the conduit the J2SE v 1.4 is required. In the installation process the Java 1.3. virtual machine is replaced with Java 1.4.

7.6 Testing and Implementation

Functionality of the system is tested according to the functional description and specification. The software is also tested according to the Palm Powered Solution Test [62]. The results of the tests can be found in Appendix O.1.

The system is currently undergoing tests at Theewaterskloof Municipality[8]. The software is provided in a single file compiled by pInstaller [38]. A training session was presented in Caledon to empower the Meter Reading Personnel. See Appendix N. The installation instructions can be found in Appendix M. The documented source code for this chapter is included in Appendix Q.3, Q.4, Q.5.2, Q.7.3 and Q.8.3.

7.7 Conclusion

By using the PalmHHT software with the combination of the metMeter post- and preprocessing software, the data collection of meter readings is optimized by reducing the number of human processes and consequently increasing accuracy. By increasing the accuracy and data quality, billing can be done more accurately. Also, using cleaner data should yield better results when investigating electricity theft. Both consumer and utility will gain from this investment.

[8] October 2004

Chapter 8

Simulations and Measurement results

8.1 Introduction

The testing and simulation of the data mining software and the testing of the total integrated system is discussed in this chapter.

The first part of this chapter focuses on the idea of a mobile remote check-meter in the field and uses data from Mooiwater in Franschoek to simulate the feasibility of such a system.

The second part integrates the complete system and shows how measurements can be made.

8.2 Simulation Functionality

8.2.1 Introduction

In order to determine the feasibility of implementing a mobile remote check-meter and to illustrate the capability of the developed package in Chapter 5, a simulation function was added, since no check-meter was available.

The simulation capability allows a user to simulate conditions on the network by introducing a random process in combination with real data. This required a histogram function to be included in each node as well as a random function generator [71] (see Section 5.4.3).

The calculations are iterated through and the values are calculated using the `toDataCollection` method. After each iteration these values are placed in the histogram. The iterations are started by pressing the CALC button and setting the number of iterations required for the simulation. The SHOW HISTOGRAM button is used to plot the histogram.

With this functionality [20] the capability of the software was illustrated and it simulated the technical losses in the network, according to Fourie's [33] method, in order to determine possible outputs of a check-meter for electricity theft detection, prior to the construction and implementation of such a meter.

8.2.2 Methodology

Variables

With the consumption data ($E_{consumed}$) available from a test site it is possible to simulate data for the required variables, for example the check-meter (E_{in}), technical losses (E_{tl}) and non-technical losses (E_{nontl}). The technical losses are calculated as described below:

- **Check meter data** With no implemented check-meter, the first step is to simulate a check-meter (E_{in}), by using available field data from consumers and by simulating the losses in the network. The data for the modeled check-meters consist of technical losses (E_{losses}) and metered energy usage ($E_{consumed}$) as in (5.1), where E_{losses} consist of the technical (E_{tl}) and non-technical losses (E_{nontl}).

  For the calculation of the check-meter reading it is assumed that all these meters are fully functional and no meter is omitted from the database. Therefore, $E_{consumed}$ is known and E_{nontl} is assumed to be zero in (8.1).

  $$E_{in} = E_{tl} + E_{consumed} + E_{nontl} \tag{8.1}$$

- **Technical losses** E_{tl} for a low voltage reticulation network, in general, is the algebraic sum of the loss contributions of all the consumers' respective phase resistances and the vectorial sum in the neutral resistance [33]. The main factor contributing to losses in the LV reticulation network is due to feeder losses (I^2R) [33]. Depending on the load and usage patterns of consumers, the losses will vary. E_{tl} is therefore approached with a Beta probability density function as shown by [33]. Although other methods, such as as Heunis [40] suggested, are available, the method implemented in using this [33] was found practical. The mean or expected value (μ) and the standard deviation (σ) is calculated for the losses in the area using circuit breaker sizes as well as the number of consumer and cable ratings [2]. These are combined in order to calculate the total technical losses for the area.

- **Non-technical losses** Electricity theft is modeled by removing specific or random readings from the initial data set and thereby altering $E_{consumed}$ and E_{nontl} in a controlled fashion.

Thresholds

To determine whether action should be taken the feasibility of such action should be assessed. This is done by including the initial capital put into a check-meter and by calculating the losses in monetary terms for a year period. The threshold, in terms of how many losses are measured by the check-meter, are calculated to aid the decision process in terms of action planning required for determining a suitable method for eradicating the illegal connections in the area.

Simulation and verification

A simulation is done with a number of expected illegal users to verify the expected electricity theft in the area. This is done to verify that the simulation and data package will assist an electricity distributor to determine the extent of the electricity theft in the area. For the simulation a time period, t, in months, is chosen and a number of n iterations are done. This is then evaluated for each of the different configurations, by comparing the results from the experiment against the case where no electricity theft is present in the simulation.

The different nodes of data sources are linked together with the tree structure as described in Fig. 5.1. Each node can be altered as seen in Fig. 5.2, to extract actual data from a data source or generate random data formed by a PDF. The mathematical method of connecting to the tree can also be specified, for example add, multiply, subtract or divide in the node.

The following is setup (XML as in Appendix P.2.2) and used in the Data Minor package.

Check-Meter The check-meter is modeled using all the samples stored in the database. The technical losses are added using a Gausian distributed random variable as found in Section 8.2.3.

Consumers The consumer database is changed by randomly selecting a number of consumers and excluding them from the accumulated energy consumption as measured over the total period under investigation.

Loss calculation The losses can now be determined by subtracting the measured value from the check-meter from the changed consumer consumption of the area.

8.2.3 Modeling

Site data

Stellenbosch Municipality granted access to their consumer data for the purpose of this study. The site under investigation is Mooiwater at Franschoek. See Appendix K.1 for more information. This data was used to extract consumer data and validate the simulations. Data for October 2003 to December 2003 was used to create the histogram in Fig. 8.1. The data for the parameter calculation is grouped together per consumer, for example each consumer's consumption is added together for the sample to give an indication of the consumption diversity.

From the sample a user profile was calculated in Matlab®.

```
============================
data3monthGroupedByUser.txt
============================
From load current: Calculations
----------------------------
```

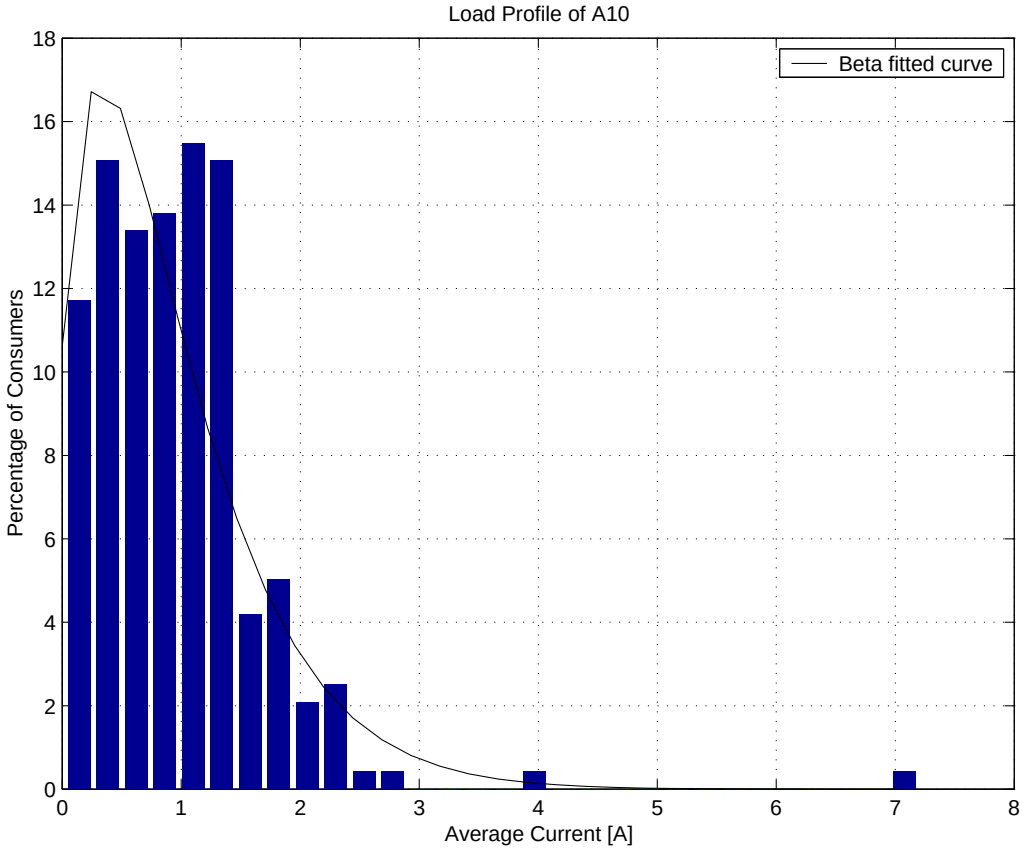


Figure 8.1: *Histogram of usage per consumer for Mini sub A10*

```
Connections   = 239[]
I RMS         = 1.215[A]
I Average     = 0.97938[A]
I VARiance    = 0.52544[A^2]
Average Power = 53836.5741[W per month]
Total Load    = 38762.3333[kWhr per month]
/Consumer     = 162.1855[kWhr]
alpha         = 1.6871
beta          = 32.7658

From beta: Calculations
----------------------------
I RMS      = 1.1114[A]
I Average  = 0.97938[A]
I VARiance = 0.52544[A^2]
```

where, from [33], the following is used:

$$\alpha = \frac{\mu\left(c\mu - \mu^2 - \sigma^2\right)}{c\sigma^2} \tag{8.2}$$

$$\beta = \frac{(c-\mu)\left(c\mu - \mu^2 - \sigma^2\right)}{c\sigma^2} \tag{8.3}$$

and

$$\mu = I_{Average} \tag{8.4}$$

$$\sigma^2 = I_{VARiance} \tag{8.5}$$

$$c = I_{max} \tag{8.6}$$

$I_{max} = 20\ A$ is used to scale the data and is normally an indication of the circuit breaker size.

Loss parameter calculations

Fourie [33] developed a spreadsheet solution (see Appendix K.3) to aid the estimation of the losses in the network base on Beta probability density functions used for E_{tl} simulation. From the spreadsheet the Beta attributes is taken and placed in the nodes in the software package of Chapter 5, to get a feel of the probability spread of the losses. Since a large amount of random functions are generated the histogram shown in Fig. 8.2, as expected, tends to have a very narrow and small standard deviation and approaches a Gausian probability density function with $\mu = 0.31\ \%$ and $\sigma = 0.03\ \%$. This simulation is done with $3\ 000$ iterations and the saved XML file is shown Appendix P.2.1.

These results were matched with simulations done in Retic Master 2004 and are available in Appendix K.

Threshold calculations and feasibility

The area under investigation includes 239 consumers. From the loss calculations, for this area and consumer base, the mean and standard deviation expressed as an percentage are $\mu \approx 0.31\ \%$ and $\sigma \approx 0.03\ \%$ of the total consumption.

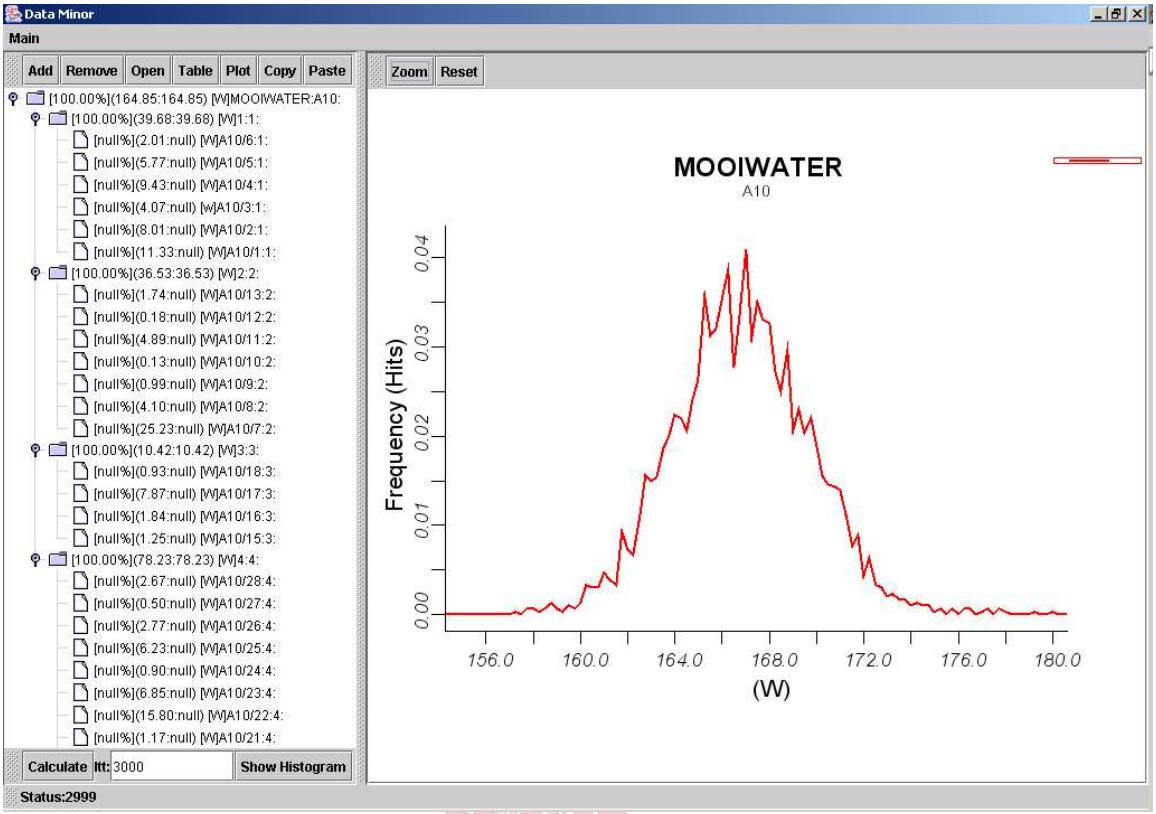


Figure 8.2: *Screen shot showing results of total technical losses simulation*

The cost of a mobile remote check-meter is estimated at $R\ 10\ 000$, since this device is mobile and can be moved to different areas. It is expected that this device can be moved four times a year. This suggests a cost per area of $R\ 2\ 500$. The selling price of electricity in this area is, on average, $R\ 0.33/kWhr$.

The cost per sweep is estimated at $R\ 27$ per consumer[1], which implies a total cost of $R\ 1\ 053$ per sweep. This includes the correction of any perpetrators. The total efficiency in terms of recurrence rate of illegal connections is assumed to be $10\ \%$.

Another option is to implement a DC reticulation network as proposed by Molepo [57]. This results in a cost of $R\ 700$ per consumer, but will require an extra rectifier circuit. Some other costs are estimated as shown in Table 8.1.

Network Simulation Results

The data from the Mooiwater mini substation MS A10 was used in the experiments.

For the simulation,the number of iterations $n = 500$ was selected. These were stored in a histogram for analysis. The output was also printed to the standard output buffer which allows easy data export to Matlab® or other software packages, if required.

[1]Cost of sweep received from Stellenbosch Municipality as an indicator

Threshold calculation			
Site data:			
Total Consumers	239		17
Total monthly consumption	38762	kWh	162.19
Loss percentage	0.31%		2757.15
Yearly Checkmeter measurement	466590	kWh	
Yearly User Consumption	465148	kWh	
Cost per kWh	R 0.33		
Check meter:			
Total Meter Cost	R 10,000.00		
Labour per shift	R 400.00		
Rotations per year	4		
Implementation cost	R 4,100.00		
Proposed solutions			
		Sweep solution	**DC Reticulation**
Assumed Efficiency		90.00%	100.00%
Overhead Costs		R 0.00	R 5,000.00
Cost per concumer		R 27.00	R 700.00
Total Cost		R 6,453.00	R 172,300.00
Including check meter		R 10,553.00	R 176,400.00
In terms of energy[kWh]		31979	534545
Energy including effieciency[kWh]		35177	534545
Percentage Threshold in terms of			
Payback period*		**Sweep solution**	**DC Reticulation**
1		7.54%	114.56%
2		3.77%	57.28%
3		2.51%	38.19%
4		1.88%	28.64%
5		1.51%	22.91%
*Payback period assumed with no devaluation of currency			

Table 8.1: *Spreadsheet for Threshold calculation*

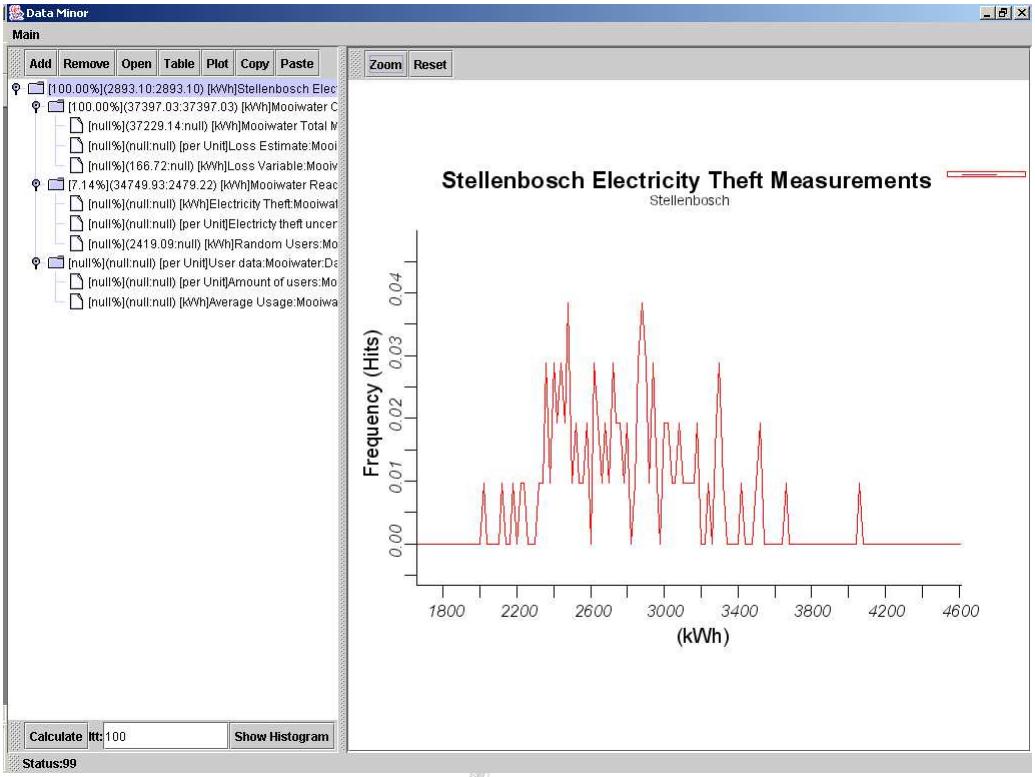


Figure 8.3: *Results from simulation with 17 illegal connections*

From the feasibility and threshold calculation it is shown that a minimum of $7.54\ \%$ losses should be measured by the check-meter. On average each user contributes 0.4% to the total energy consumption. This implies that roughly 19 consumers should actively benefit from illegal connections.

The results of the simulation with only 17 users, as in Appendix P.2.2, is seen in Fig. 8.3 and match the expected results.

8.3 Practical Laboratory Experiments

All the systems and subsystems are required to be tested and evaluated.

8.3.1 Setup

A laboratory setup was done to test the system as shown in Fig. 8.4.

The total interconnection of the system is shown in Fig. 8.5.

From the supply of the laboratory bench the Voltec Power Analyser is placed to confirm the measurements made by the check-meter and to indicate if power is switched on.

The check meter, as discussed in Chapter 6, is placed after $30\ A$ circuit breakers. These are used to isolate the meter from the power grid. A distribution board was made (see Fig. 8.6) to

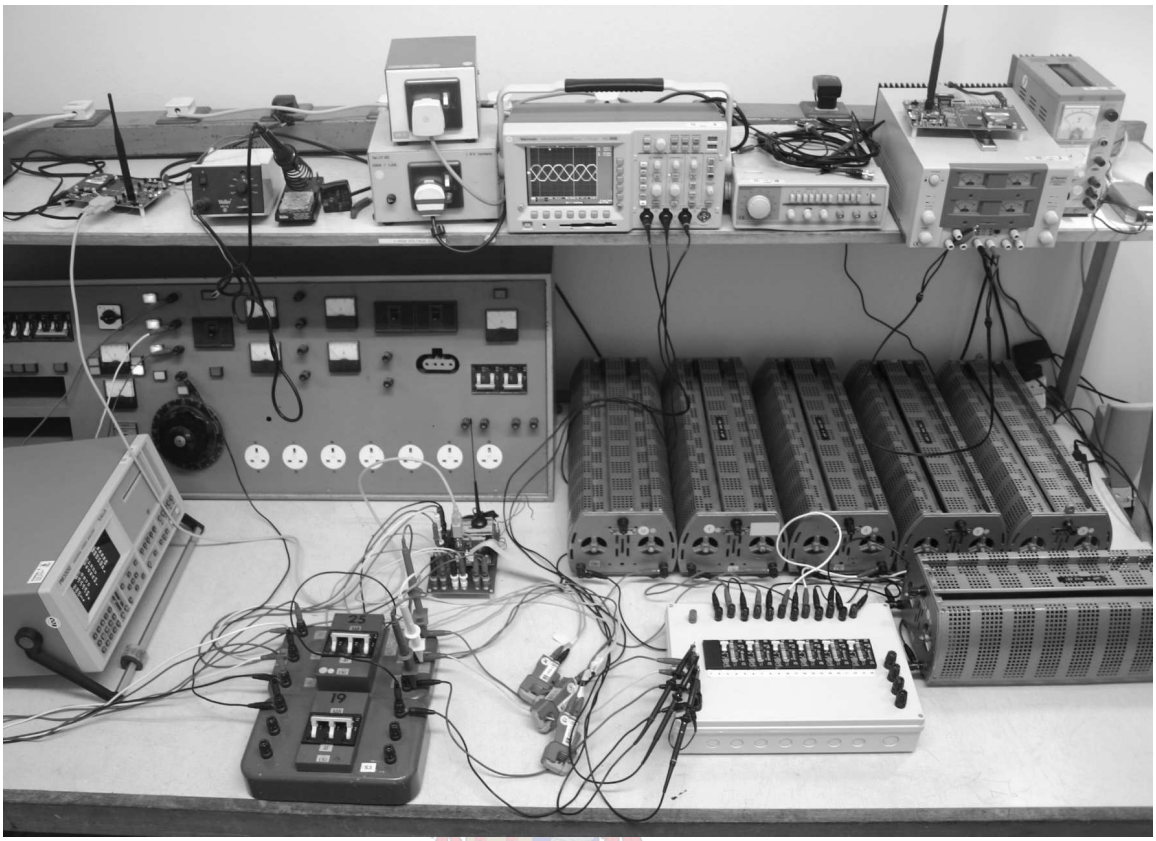


Figure 8.4: *Setup in laboratory*

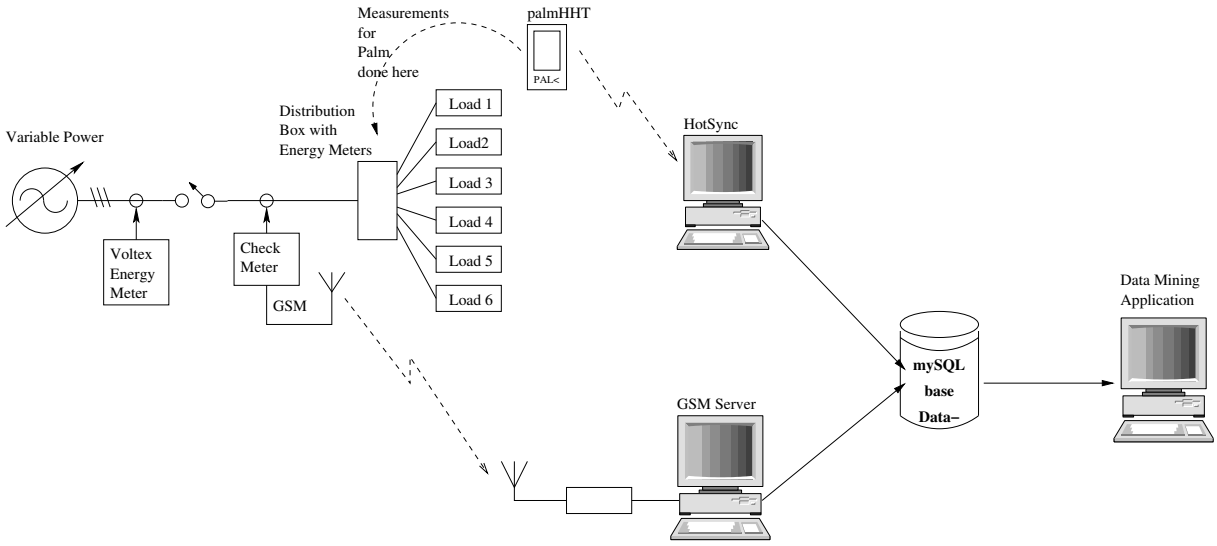


Figure 8.5: *Overview of test setup*

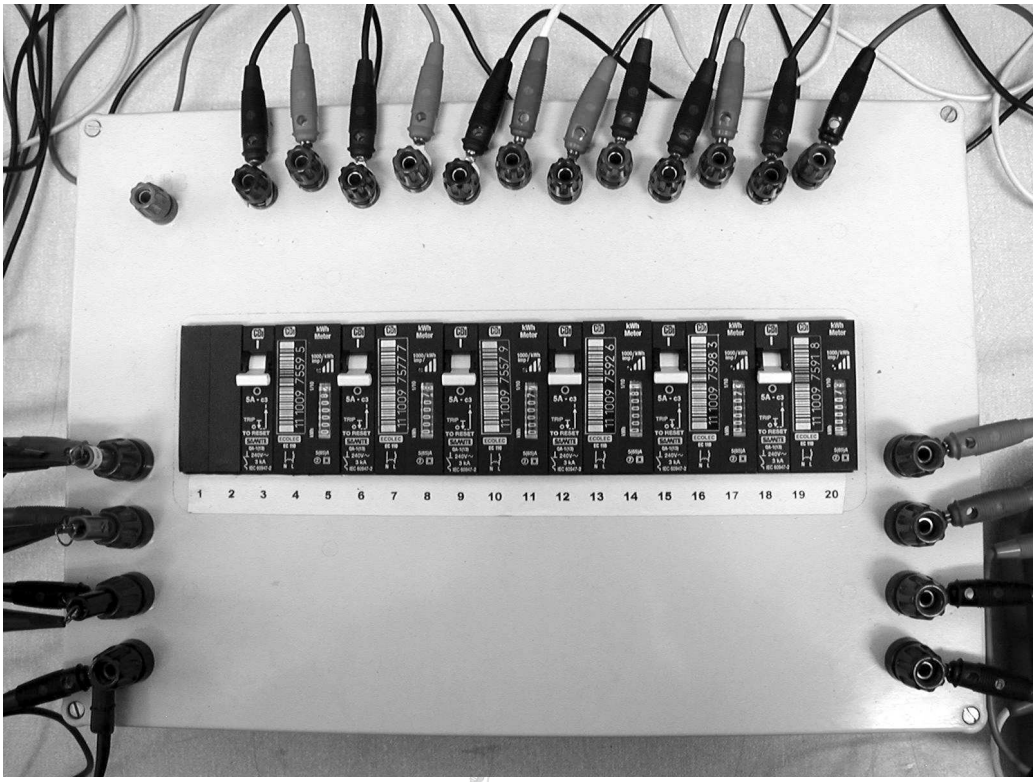


Figure 8.6: *Distribution board with energy measurement units*

allow individual load measurements to be taken. Six loads, capable of $I_{max} = 4\ A$, $R = 136\Omega$, are connected to the distribution board.

With the check-meter running the GSM server system is started. At preset daily intervals the server requests an update from the check-meter. Since the tests are conducted over an 8-hour period and requests are done on an hourly basis, the trigger times were advanced by an hour each time a request was made. In some instances during the tests the requests were made randomly. The data was captured on the database.

The load's individual measurements were made by using the palm's data collection software as described in Chapter 7. The data from the hand held is dumped into the database.

The results are presented using the data mining software developed in Chapter 5 and this completes the total information path.

8.3.2 Results

The results of the test are as expected. The total load measured by the individual meters and the check-meter match fairly closely. Some minor discrepancies were detected, but these can be attributed to the fact that the accuracy of the individual measurement units in the distribution board can only measure accurately up to $0.1\ kWhr$.

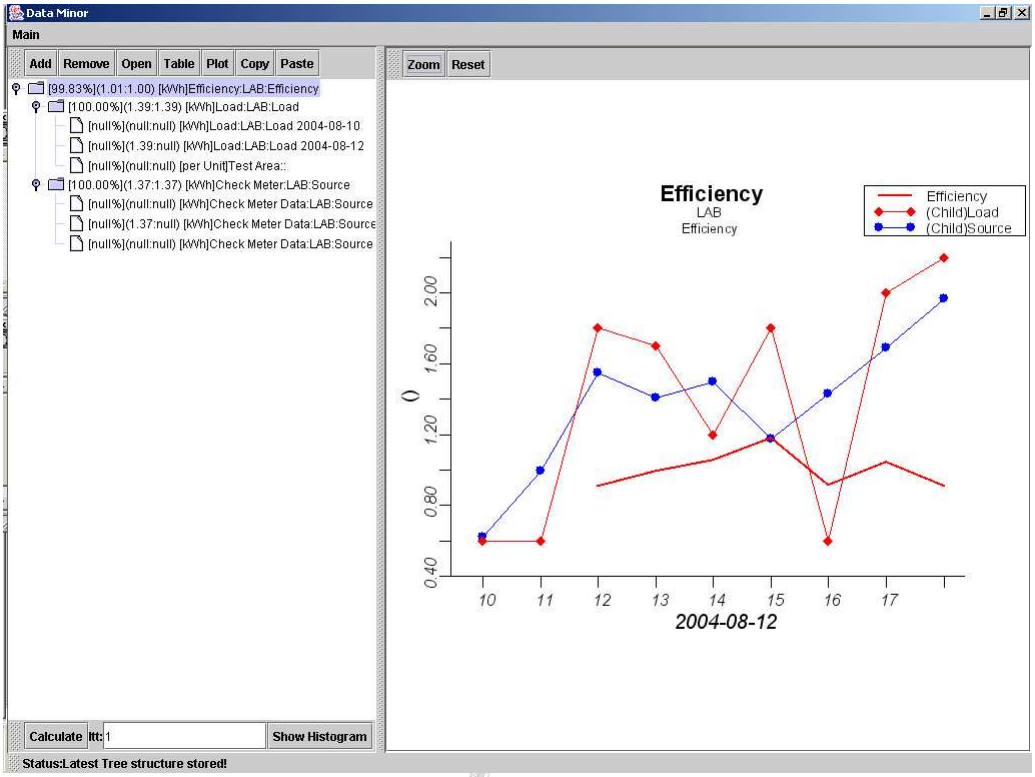


Figure 8.7: *All loads measured*

All metered loads connected

The first measurement set, as shown in Fig. 8.7, is done with the total load included. With a 3 sample moving average it can be seen that the average is around unity, as shown in brackets in text at the top node of the tree, indicating no electricity theft or major losses.

One load omitted from data set

The second set shown is done with one user disconnected from the database. This results in a $84\ \%$ efficiency and can be seen in Fig. 8.8.

8.4 Conclusions

The feasibility study done indicates that if appropriate eradication or action is taken, electricity theft can be addressed. The simulations helped with the modeling of the network and gave insight into the various aspects involved.

The test in the laboratory showed that the platform presented can be integrated successfully. The information path is demonstrated from measurement, data transmission and storage through to presentation and processing.

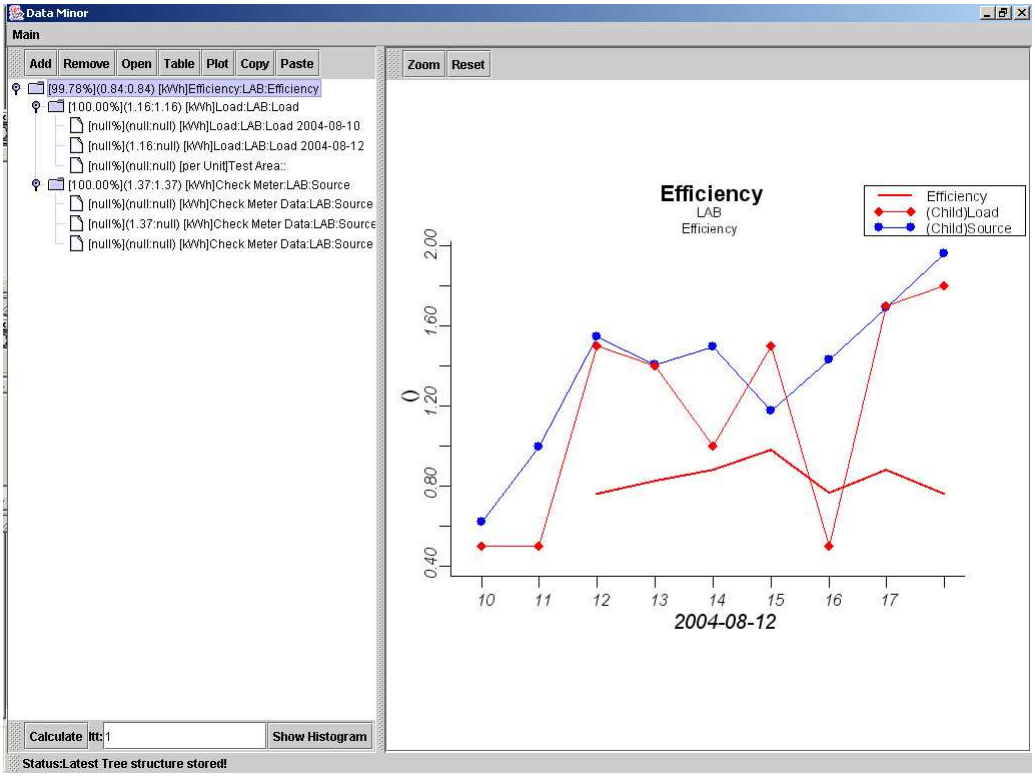


Figure 8.8: *One load omitted from data set*

Chapter 9

Conclusions

9.1 Overview

In South Africa, the electricity distribution industry is exposed to more unmonitored users each year due to the drive of the rural electrification projects and the installation of the pre-paid electricity meter since the early 1990s. With an extensive amount of infrastructure already in place the emphasis is on finding an economical as well as viable solution to reduce or eliminate electricity theft.

For this project, the following possible solutions were investigated:

- By generating a pulse on the network, Time Domain Pulse Reflectometry technology was developed.
- Based on the Theory of Constraints, information technology systems were developed to aid the management of electricity theft.

For these two methods, systems were developed for functioning in a low voltage reticulation environment. The devices developed were tested in a laboratory environment in order to evaluate their functionality.

9.2 Evaluations and Feasibility

The criteria for evaluation is fairly straightforward: how does the method or solution impact the bottom-line of the distribution network?

According to this criterion the following can be concluded from the research presented in this thesis:

- The reflections from the Time Domain Pulse Reflectometry are too complex to decode using known signal processing tools. This is due to the amount of reflections, distortions and signal attenuation. Therefore, a high capital layout would be required to acquire an

acceptable degree of accuracy. With the current technology level, a high level of human interaction is expected. Combined with high per unit costs and high density, the running costs would be extremely high. This solution does not seem viable in terms of electricity theft detection.

- Electricity theft must be managed using a performance driven system. There is no magic wand for this problem. Applying the Theory of Constraints in combination with the developed Data Mining platform, could have a payback period of less than a year for areas where electricity theft is expected to be more than 8%. This data driven approach adds value to already implemented investments by adding a minimal amount of additional infrastructure to provide an extremely powerful tool for electricity theft detection.

9.3 Recommendations and Future Research

9.3.1 Time Domain Pulse Reflectometry

Although Time Domain Pulse Reflectometry is not suitable for electricity theft detection the inverted pulse topology developed can be used for on-line real-time fault finding on cables. For further work this technology could be investigated for applications on medium voltage lines to provide a solution for cable theft detection.

9.3.2 Theory of Constraints and Data Driven Solution

The system developed in the thesis was tested in a controlled laboratory environment. These developments and this technology should be tested in the industry as soon as possible to determine its practical impact.

The Theory of Constraints proves to be an extremely helpful tool to address the problem of electricity theft at management level. During the research it was impossible to monitor this management strategy effectively and it should prove to be an interesting study to apply and monitor this strategy at a few local municipalities.

The developed Data Mining platform could be expanded to including learning algorithms on the macro, micro and individual scale by further developing the node capabilities. Features such as exception reporting and statistical analysis on the data of nodes can be included, such as the methods developed by Fourie [32], in a next release. Developing additional data collection software for input via PCs to assist performance management in other areas where no mobile electronic Data Collection applications, such as the PDA solution in Chapter 7, exist.

Due to the modular design of the check-meter, the systems feature set can be expanded to include the following:

- The design of single phase metering system.

- Implementation of a real time data logging system.
- Additional memory placed on the module can extend the real time data logging capabilities.
- The use of the $2.4\,GHz$ ISM band for inter-area data communications can be investigated.
- With the implementation of inter-area communications, the system can be used to perform pre-paid or conventional metering functions.
- Along with the metering functions the appropriate back office applications should be developed to support the system in the field.

9.4 Conclusions

In the field of electricity theft no extensive academic research has been done for the South African environment. A study was undertaken to establish the know-how for addressing the problem of electricity theft. During this process, gaps in the information chain were identified.

For the purpose of the thesis, tools and systems were developed from scratch to aid in the detection of electricity theft and to provide a platform for research into trends and other phenomena concerning electricity usage. This provides the ideal tool for electricity theft detection in a performance driven management culture. The evaluation of these new tools showed that a solution exists for determining the extend of electricity theft in an area by using a minimal amount of additional infrastructure and therefore, adding value to current investments.

Bibliography

[1] 3GPP, *ETSI GSM 07.05: Digital cellular telecommunications system (Phase 2); Use of DTE-DCE interface for Short Message Service (SMS) and Cell Broadcast Service (CBS)*. 3GPP, V6.9.2 edition, April 2001.

[2] ABERDARE, *ABC - Low voltage aerial bundle conductor*. Aberdare Power Cables, 2003. www.aberdare.co.za.

[3] ACTARIS, *ECLIPSE Prepayment Electricity Resource Management Software*. Actaris SAS, ZI de Chasseneuil Avenue des Temps Modenes 86361 Chasseneuil du Poitou cedex France.

[4] ACTARIS, *SL7000 Smart Comercial and Industrial Electricity Meter*. Actaris SAS, ZI de Chasseneuil Avenue des Temps Modenes 86361 Chasseneuil du Poitou cedex France.

[5] ANALOG DEVICE, "Analog Devices." `www.analog.com`. 2004.

[6] ANALOG DEVICES, *Evaluation Board Documentation ADE7758 Energy Metering IC*. Analog Devices, One Thenology Way, P.O.Box 9106, Norwood, MA 02062-9106, U.S.A., Rev.prb edition, August 2003.

[7] ANALOG DEVICES, *Poly phase multifunction Energy Metering IC with per Phase information, ADE7758*. Analog Devices, One Thenology Way, P.O.Box 9106, Norwood, MA 02062-9106, U.S.A., Rev.0 edition, February 2004.

[8] BRADLEY, T., *An introduction to the Bootstrap*. Chapman and Hall, Inc. 29 West 35th Street New York, NY 10001-2299: Chapman and Hall, 1993.

[9] BURTON-HOULE, T., "The Theory of Constraint and its thinking processes." `http://www.goldratt.com`. 2001.

[10] CABENA, P. *et al.*, *Discovering data mining from concept to implementation*. Prentice Hall, Inc., 1997. p. 12, p.43.

[11] CAMPBELL, K., "The good, the bad and the ugly." `http://www.sarpa.co.za/documents/document_list_all.asp`. 1999.

[12] CARTE BLANCHE, “Dangerous Sparks - 17 March 2002.” http://www.mnet.co.za/CarteBlanche/default.asp. 2002.

[13] CHAPMAN, T. and TERBLANCH, W., “History and Current Status of the Eskom dealings with non-payment in Soweto.” tech. rep., ESKOM, 2003.

[14] CLOETE, D. E., “Check meter principle in respect of tampering.” http://www.sarpa.co.za/documents/document_list4.asp?id=132. 2002.

[15] DAVIDSON, I. E., “Evaluation and effective management of non-technical losses in power networks.” *The Transactions of the SA Institute of Electrical Engineers*, September 2003.

[16] DENT INSTRUMENTS, *ELITEpro Recording Poly Phase Power Meter*. Dent Instruments, 65 NW Franklin Ave. Bend, OR 97701-2906 USA.

[17] DEPARTMENT OF MINERALS AND ENERGY, “BACKGROUND ON FREE BASIC ELECTRICITY (FBE).” http://www.dme.gov.za. 2004.

[18] DE VILLIERS AND MOORE CONSULTING ENGINEERS, “Mooiwater Housing Project - LV Reticulation Service and Connections.” CAD Drawing R3530/LV and SC1, Inframax DMS Western Cape, P.O.Box 472, Durbanville, 7551, November 2001.

[19] DICK, A. J. and MACEY, R., “Revenue protection in a competitive supply environment.” *Metering Tariffs for Energy Supply*, May 1999, No. 462, p. 229.

[20] DOORDUIN, W. A. *et al.*, “Feasibility Study of Electricity Theft detection using Mobile Remote Check Meters.” *AFRICON 2004*, January 2004.

[21] DOORDUIN, W. A. *et al.*, “On-line Time Domain Pulse Reflectometry in a LV reticulation network.” *SAUPEC 2004*, January 2004.

[22] DOUCETTE, S. and HAYWARD, R., “Handhelds Enhance Operation Excelence Program.” http://www.mt-online/articles/0504keyspan.cfm?pf=1. 2003.

[23] EPIC and DONALD, W. D., “Scientific Graphical Toolkit.” http://www.epic.noaa.gov/java/sgt/. October 2003.

[24] ESKOM, “Non payment of services.” http://www.eskom.co.za. January 2003.

[25] ESKOM, “Current Prepaid Related Projects.” http://www.eskom.co.za/electrification/current_projects.htm. 2004.

[26] ETSI, *ETSI GSM 03.38: Digital cellular telecommunications system (Phase 2): Alphabets and language-specific information*. ETSI.

[27] ETSI, *ETSI GSM 03.40: Digital cellular telecommunications system (Phase 2); Technical implementation of the Short Message Service (SMS) Point-to-point (PP)*. ETSI.

[28] ETSI, *ETSI GSM 04.80: Digital cellular telecommunications system (Phase 2): Mobile radio interface layer 3, Supplementary service specification, Formats and coding*. ETSI.

[29] ETSI, *ETSI GSM 07.05: Digital cellular telecommunications system (Phase 2); Use of DTE-DCE interface for Short Message Service (SMS) and Cell Broadcast Service (CBS)*. ETSI.

[30] ETSI, *ETSI GSM 07.07: Digital cellular telecommunications system (Phase 2); AT command set for GSM Mobile Equipment*. ETSI.

[31] FOCUSED PERFORMANCE, "The TOC thinking process." `http://www.focusesperformance.com`. Unkown.

[32] FOURIE, J. W. and CALMEYER, J. E., "A Statistical Method to Minimize Electrical Energy Losses in a Local Electricity Distribution Network." in *Africon*, vol. 02, pp. 667–673, IEEE, 2004.

[33] FOURIE, R. J., "Evaluation of losses in LV feeders using the Beta Load." Master's thesis, Stellenbosch Unversity, 1998.

[34] GIOVINAZZO, P., "The Fatal Currenty." `http://www.physics.ohio-state.edu/~p616/safety/fatal_current.html`. February 1987.

[35] GLOVER, D. J. and SARMA, M., *Power System Analysis and design*. Second edition. PWS Publishing Company, 1994.

[36] GOLDRATT, E. and COX, J., *The Goal: A Process of Ongoing Improvement*. Avraham Y. Goldratt Institute, 1992.

[37] HANDANGO.COM, "History of the Personal Digital Assistant." `http://www.handango.com/PDAHistory.jsp?siteId=1`. 2004.

[38] HANDX, "handX pInstaller." `http://www.handx.net/devtool/pInstaller/`. 2004.

[39] HAUSE, H. A. and MELCHER, J. R., *Electromagnetic Fields and Energy*. Pretice-Hall International, Inc., 1898.

[40] HEUNIS, S. W. and HERMAN, R., "Estimation of the annual resistive loss on LV residential feeders." *PMAPS, Naples*, September 2002.

[41] INFINEON, *CoolMOS Power Transistor SPW20N60S5*. Infineon Technologies AG, November 2002. www.infineon.com.

[42] ITU-T, *ITU-T Recommendation V.25 ter: Serial asynchrinous automatic dialing and control*. ITU-T.

[43] JAMIESON, D., "Conversion to pre-payment." tech. rep., Thembisa Ekurhuleni, June 2003.

[44] JAMIESON, D., "Saga of protective structures." tech. rep., Thembisa Ekurhuleni, June 2003.

[45] KRISHNA RAO, M. V. and MILLER, S. H., "Revenue improvement from intelligent metering systems.." *Metering and tariffs for Energy Supply*, May 1999, No. 462, p. 218. Conference Publication No. 462 IEE 1999.

[46] KU, Y. H., *Transient Circuit Analysis*. D. van Nostrand Company, inc., 1961.

[47] LEM, *Current Probe Model PR1200 MACV*. LEM.

[48] LEM, *Current Probe Model PR201 ACV*. LEM.

[49] LEM, *Current Probe Model PR221 ACV*. LEM.

[50] LOUW, D., "Legal Matters and RPAAU Proceedings." tech. rep., SARPA, 2002.

[51] MENZIES, T. and HU, Y., "Data Mining for Very Busy People." *IEEE Computer Society Journal - Computer*, November 2003, Vol. 36, No. 11, pp. 22–28.

[52] MICHAEL T. GOODRICH, R. T., *Data Structures and Algorithms in Java*. 2 edition. John Wiley & Sons, Inc., 2001.

[53] MICHIE, D. M., "Free basic electricity." tech. rep., Association of Municipal Electricity Undertakings, June 2001.

[54] MISETE, E., "New HandHelds Improve Worker Productivity." `http://www.integratedsolutionsmag.com/`. May 2003. Articles/2003_05/030503.htm.

[55] MOHAMED, Y., "Nie deur gure weer gekeer." *Die Burger*, August 2004, p. 6. 10 August 2004.

[56] MOHAN, N. *et al.*, *Power Electronics*. second edition. John Wiley & Sons, Inc., 1989.

[57] MOLEPA, S. A., "A multilevel inverter for DC Reticulation." Master's thesis, Stellenbosch Unversity, April 2003.

[58] NEAMEN, D. A., *Electronic Circuit Analysis and Design*. McGraw-Hill Companies, Inc., 1996.

[59] NETBEANS.ORG, "Netbeans downloads." http://www.netbeans.info/downloads/. 2004.

[60] OPENSHAW, C., "Report remote monitorig pre paid meters." ESKOM minutes, 2003.

[61] PALM, "Company Background." http://www.palmone.com/us/company/pr/background.html. 2004.

[62] PALM, "Compatibility Testing." http://www.palmos.com/dev/core/compatiblity/test.html. 2004.

[63] PALM, "Download Conduit Developer Kit (CDK)." http://www.palmos.com/dev/dl/dl_sdks/dl_cdk/. 2004.

[64] PALM, "palmOne Support: HotSync Technology Support Index." http://www.palmone.com/us/support/hotsync.html#network. 2004.

[65] POZAR, D., *Microwave Engineering*. Second edition. John Wiley and sons, inc., 1998.

[66] RS, "3 Phase Test Leads, 600V CATIII, RS #212-988." http://www.rssouthafrica.com/. 2004.

[67] SABS, "NRS048:1 - Electricity Supply - Quality of Supply." tech. rep., National Electricity Regulator, 1996.

[68] SABS, "NRS048:2 - Electricity Supply - Quality of Supply." tech. rep., National Electricity Regulator, 1996.

[69] SCHWARZ, A., "www.mikrocontroller.net - MSPGCC (english version)." http://www.mikrocontroller.net/mspgcc.en.htm. December 2003.

[70] SHINGAI, S., "The role of metering in revenue protection." *Metering and tariffs for Energy Supply*, May 1999, No. 462, p. 223. Conference publication IEE 1999.

[71] SIEGRIST, K., "Virtual Laboratories in Probability and Statistics." http://www.math.uah.edu/stat/objects/index.xml. 2004.

[72] SOURCE FORGE, "Mspgcc-users – GCC for MSP430 - http://mspgcc.sf.net." https://lists.sourceforge.net/lists/listinfo/mspgcc-users. 2004.

[73] SOUTH AFRICAN GOVERNMENT, "Electricity Act, Law 41 of 1987 article 27 (2,3)." 1987.

[74] SOUTH AFRICAN GOVERNMENT, “Municipal Systems Act 2000.” http://www.gov.za. December 2000.

[75] SOUTH AFRICAN GOVERNMENT, “OCCUPATIONAL HEALTH AND SAFETY ACT.” http://www.info.gov.za/gazette/acts/1993/a85-93.htm. 2003.

[76] STEHMANN, T., “Development and Optimsation of a Solid-State Pulsed Power Supply for a $C0_2$ TEA laser.” Master’s thesis, Stellenbosch University, April 2003.

[77] STELLENBOSCH MUNICIPALITY, “Stellenbosch Muncipal.” http://www.stellenbosch.org.za. 2004.

[78] ST MICROELECTRONICS, *1.5A Low Drop Fixed and adjustable positive voltage regulators*. ST Microelectronics, May 2000.

[79] STRIKE TECNOLOGIES, “ENERMAX.” http://www.strike.co.za/1024x768/content/Products/Enermax.htm. 2004.

[80] STROBER, K., “Revenue Protection - Proactively Changing Attitudes.” tech. rep., Cape Town, 2003.

[81] SUN, “javax.swing (Java 2 Platform SE v.1.4.2).” http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/package-summary.html. 2003.

[82] SUN, “JTree (Java 2 Platform SE v.1.4.2).” http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JTree.html. 2003.

[83] SUN, “Java 2 Platform, Standard Edition v 1.4 Overview.” http://java.sun.com/j2se/1.4/. June 2004.

[84] SUN, “Java Communications API.” http://java.sun.com/products/javacomm/. 2004.

[85] SUN, “JDBC Technology.” http://java.sun.com/products/jdbc/index.jsp. 2004.

[86] SUN, “MIDP for Palm OS.” http://java.sun.com/products/midp4palm/. 2004.

[87] TEXAS INSTRUMENTS, *Configuring the MSP430x11x(1)’s Basic Clock Module*. Texas Instruments Inc., Slaa082 edition, November 1999. www.ti.com.

[88] TEXAS INSTRUMENTS, *Octal bus transceiver and 3.3-V to 5-V shifter with 3-state outputs*. Texas Instruments Inc., Sca5375g edition, March 1999. www.ti.com.

[89] TEXAS INSTRUMENTS, *MAX3222 3-V to 5.5-V Multichannel RS-232 Line Driver/Receiver with 15-kV ESD protection*. Texas Instruments Inc., Slls408g edition, January 2000. www.ti.com.

[90] TEXAS INSTRUMENTS, *Single 9-A High Speed Low-Side MOSFET driver with enable*. Texas Instruments Inc., September 2002. www.ti.com.

[91] TEXAS INSTRUMENTS, "Texas Instruments." www.ti.com. 2004. www.ti.com.

[92] VAN DER MERWE, J., "Experiments on electromagnetic fields and waves." Master's thesis, Stellenbosch Unversity, September 1995.

[93] WAVECOM, *AT Commands Interface Guide*. Wavecom, 003 edition, December 2003.

[94] WEBOPEDIA.COM, "A Word Definition From the Webopedia Computer Dictionary." http://www.webopedia.com. 2004.

[95] WILLIAMS, D. F., "Meter readers make the switch to PDA." http://www.istart.co.nz/index. December 2003. /HM20/PC0/PV22447/EX24706/CS24777.

[96] ZENOU, B., *WMOi3 user's guide*. Wavecom, 1.4 edition, January 2001.

Appendix A

Additional Background Information

A.1 Eskom's Notice of Unlawfull Tampering

ESKOM

Aksie teen Dr,Mnr,Me .....
Action against Dr,Mr,Ms

in terme van Eskom se Voorsieningsvoorwaardes vir klein toevoere
in terms of Eskom's Conditions of Supply for small supplies

Met verwysing na bogenoemde dokument word u in kennis gestel dat u wederregtelik met Eskom se apparaat of toerusting gepeuter het en dat u aanspreeklik gehou word vir die koste verbonde aan die herstel van die toevoer en enige verliese deur Eskom gely. U het u blootgestel aan vervolging ingevolge die Elektrisiteitswet, nommer 41 van 1987, soos gewysig en die toevoer na u installasie is gestaak en die meter verwyder.

With reference to the abovementioned document you are notified that you have unlawfully tampered with Eskom's apparatus or equipment and that you are held responsible for the cost of restoring the supply and of any losses suffered by Eskom. You have exposed yourself to prosecution in terms of the Electricity Act, number 41 of 1987, as amended and the supply to your installation has been discontinued and the meter removed.

Installasie adres: _____ Installation address

_____

_____

Erfnommer _____ Erf number

Meternommer _____ Meter number

Eenhede (kWh) op meter _____ Units (kWh) on meter

Beskrywing van skade aan installasie / Description of damage to installation :

_____

	Naam / Name	Handtekening/Signature	Datum / Date
Eskom verteenwoordigers Eskom representatives	.....	.....	.....
	.....	.....	.....
Klant / Customer	.....	.....	.....

Nota aan klant:
Aansoek om heraansluiting kan by Eskom se Klantedienskantoor te Bellville ingedien word. Hierdie vorm sowel as 'n geldige identiteitsdokument moet deur applikant getoon word voor heraansluitingsfooi bereken kan word.

Heraansluitingsfooi is in kontant betaalbaar by indiening van aansoekvorm.

Note to customer:
Application for reconnection can be made at Eskom's Customer Service Centre in Bellville. This form, as well as a valid identity document must be produced by the applicant before a reconnection fee can be calculated.

The reconnection fee is payable in cash on submission of the application form.

VIR KANTOOR GEBRUIK / FOR OFFICE USE

Vorige remiediërende aksies
Previous remedial actions

Huidige aksie op PPS/CMS geplaas
Current action entered on PPS/CMS — Datum / Date ..... — Deur wie ingepons / Punched by .....

Heraansoek ingedien
Re-application submitted — Datum / Date ..... — Deur wie ontvang / Received by .....

PPS Geskiedenis Kode
PPS History Code ....

Bedrag betaalbaar
Amount payable R.....

WKV212

A.2 Electricity Theft Methods

This section is taken from [45] and placed to provide the reader with a reference as to what has been found in the field:

Pilferage through tapping of lines

- Direct tapping of lines without meter
- Tapping supply from behind the meter board by joining one end of a wire to incoming phase wire and the other ens to the main switch and bypassing the meter.
- Pilfering energy from cutouts where they are connected before the meter.
- Breaking the neutral connection and incorporating a switch in its circuit with efficient earth connection.

Pilferage through damage to meter mechanism without tampering terminal cover seals and meter seals

- Drilling small hole in the meter cover and inserting a fine needle to stop the rotation of the meter disc and covering it up with paint to avoid being noticed.
- Creating gaps in meter glass and inserting a tongue cleaner to obstruct the rotation of the meter disc.
- Creating fine gaps in between meter top and bottom covers and inserting thin bits of tongue cleaner or polymeric leaves or celluloid films to obstruct the rotation of the meter disc.
- Subjecting the meter to violent shock to render it totally inoperative without forming a dent on the surface.
- Breaking the meter glass and fixing another glass in its place with adhesive compound to have accessibility to recording arrangement.
- Keeping the meter in an inclined position to slow down the meter speeder keeping it horizontal to bring the meter speed to a stop.
- Burning the meter either by creating a loose connection at the terminal or by setting fire to it by pouring petrol or other inflammable material or creating a short circuit.

Pilferage through tampering terminal cover seals

- Reversing the connections between line (incoming) and load (outgoing) in the terminal block. If one phase is reversed in a 3 phase service, the meter runs in a forward direction whenever three phase load is switched on but records only $\frac{1}{3}^{rd}$ of the consumption.
- By keeping open the potential links in the meter terminal block for such meters which are not provided with internal potential links, so that the meter will not work due to non-availability of supply to potential coil.
- By providing a shunt wire to the current coil terminal in the meter terminal thus creating parallel path and partial recording of energy.
- Removing the phase and neutral wires from the meter terminal block and joining them externally so as to stop running of the meter.
- By reversing incoming phase and neutral wires in the terminal block and providing a switch with earth connection.

Pilferage through tampering meter cover seals

- Stopping the meter disc.
- Cutting potential wire, resulting in non-rotation of the meter disc.
- By providing a shunt wire to the current coil terminals inside the meter, thus creating parallel path and partial recording of energy.
- Reducing maximum demand in case of Trivector meters.
- Adjusting the consumption recorded at will.
- Providing parallel paths to current coil elements in 'R' phase and 'B' phase of Trivector meters by connecting up extra wires with switches fixed in parallel paths to switch off or on the parallel paths in order to reduce the flow of normal current in the mtering ciruit.

Pilferage through tampering seals of metering equipment(CTs & PTs)

- Providing parallel paths through CT secondary terminals.
- Increasing the CT ratio to a higher range by changing the CT secondary connections thus resulting in recording of lower consumption.
- Meddling of PT secondary terminal wires including opening/reversing the connections.
- Blowing or removing PT primary fuses from inside the metering cubicle.
- Short circuit current transformers on secondary side.

- Change the current transformer ratio on the primary side.

Pilferage through wrong readings and desk top readings

- Meters are not read properly and ofter desk readings are entered.
- Often consumers are not available at their premises to enable a meter reader to record meter reading.
- Some consumers bypass the meter but when the meter reader comes to check the reading they revert it back to the original connection.

Appendix B

Transmission Line Theory

Some of the theory behind time domain pulse reflectometry will be explained here. This is included as a summary of Transmission Line Theory and is based on the derivations of Ku and Hause & Melcher.

B.1 Telegraphist Equation

Consider the transmission line in Fig. B.1 and the derivations in [46], which provide the necessary equations for the transmission line model. Let

R = resistance per unit length of the line,

L = inductance per unit length of the line,

G = conductance per unit length of the line,

C = capacitance per unit length of the line and

x = the distance measured from the sending end of the transmission line.

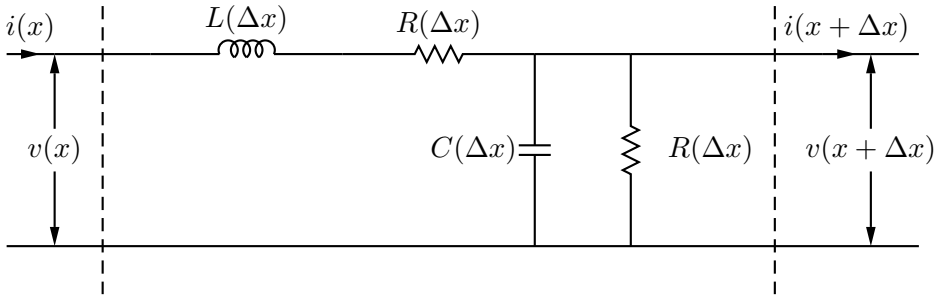


Figure B.1: *Section of Transmission Line*

The voltage drop and decrease in current in length Δx is given by (B.1) and (B.2)

$$-\frac{\partial v}{\partial x}\Delta x = (Ri + L\frac{\partial i}{\partial t})\Delta x \tag{B.1}$$

$$-\frac{\partial i}{\partial x}\Delta x = (Gi + C\frac{\partial v}{\partial t})\Delta x \tag{B.2}$$

This voltage and current functions are in the time domain where

$$v = v(t, x) \tag{B.3}$$

$$i = i(t, x) \tag{B.4}$$

The part of interes is in the transient response of the transmission line and by using the Laplace transformation with respect to t and dividing both (B.1) and (B.2) by Δx, (B.5) and (B.6) is obtained.

$$-\frac{\partial V(s, x)}{\partial x} = (R + Ls)I(s, x) \tag{B.5}$$

$$-\frac{\partial I(s, x)}{\partial x} = (G + Cs)V(s, x) \tag{B.6}$$

By differentiating (B.5) with respect to x and substituting the result in (B.6) yields (B.7):

$$\frac{\partial^2 V(s, x)}{\partial x^2} = \gamma^2 V(s, x) \tag{B.7}$$

Similarly, by differentiating (B.6) with respect to x and substituting the result in (B.5), (B.8) is obtained

$$\frac{\partial^2 I(s, x)}{\partial x^2} = \gamma^2 I(s, x) \tag{B.8}$$

where

$$\gamma = \sqrt{(R + Ls)(G + Cs)} = \sqrt{ZY} \tag{B.9}$$

and

$$Z = (R + Ls) \tag{B.10}$$

$$Y = (G + Cs) \tag{B.11}$$

General solutions for (B.7) and (B.8) are

$$V(s, x) = V^+ e^{-\gamma x} + V^- e^{\gamma x} \tag{B.12}$$

$$I(s, x) = \frac{1}{Z_0}(V^+ e^{-\gamma x} - V^- e^{\gamma x}) \tag{B.13}$$

where

$$Z_0 = \sqrt{\frac{(R + Ls)}{(G + Cs)}} = \sqrt{\frac{Z}{Y}} \tag{B.14}$$

and $V^+(s, x)$ and $V^-(s, x)$ are used to denote the signal traveling in the $+$ and $-$ direction of the transmission line[39, ch 14.3, (14.3.2), p.641], depending on our chosen reference point.

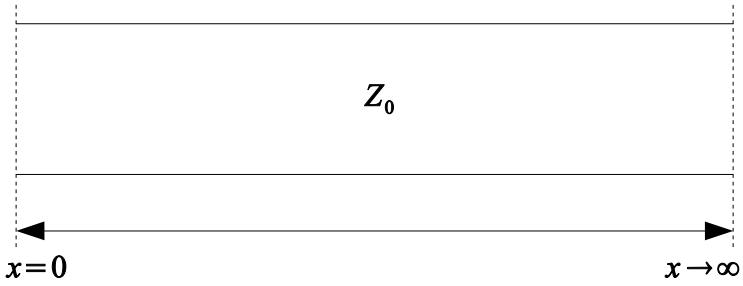


Figure B.2: *One Way Infinite Line*

B.2 Propagation in Transmission Lines

B.2.1 Infinite Lossless Line

Consider the infinite transmission line, Fig. B.2, without the lossy components and therefore $R = 0$ and $G = 0$ with the end of the transmission line at $x \to \infty$. Using boundary conditions and (B.12) it follows from [46, ch 10-4, p.291] that

$$\lim_{x\to\infty} V(s,x) = 0 \Rightarrow V^- = 0 \tag{B.15}$$

$$V(s,0) = V^+; \tag{B.16}$$

and results in

$$V(s,x) = V(s,0)e^{-\gamma x} \tag{B.17}$$

$$= V(s,0)e^{-\sqrt{LC}sx} \tag{B.18}$$

which, with the use of the inverse Laplace transform results in

$$v(t,x) = v_s\left(t-\sqrt{LC}x\right) \tag{B.19}$$

$$= v_s\left(t-\frac{x}{c}\right) \tag{B.20}$$

where the voltage at the sending end $v_s(t)$ is the inverse transform of $V(s,0)$ and $c = \frac{1}{\sqrt{LC}}$. When considering the infinite lossless line in Fig. B.3 in both directions, the reference point is moved to the middle of the transmission line and setup the initial conditions from here. The following initial conditions [39, ch 14.4, p.647] are substituted

$$V(s,0) = V(s) \tag{B.21}$$

$$I(s,0) = I(s) \tag{B.22}$$

in (B.12) and (B.13) in order to solve for V^+ and V^- and obtain

$$V^+ = \frac{1}{2}\left(V(s)+Z_0I(s)\right) \tag{B.23}$$

$$V^- = \frac{1}{2}\left(V(s)-Z_0I(s)\right) \tag{B.24}$$

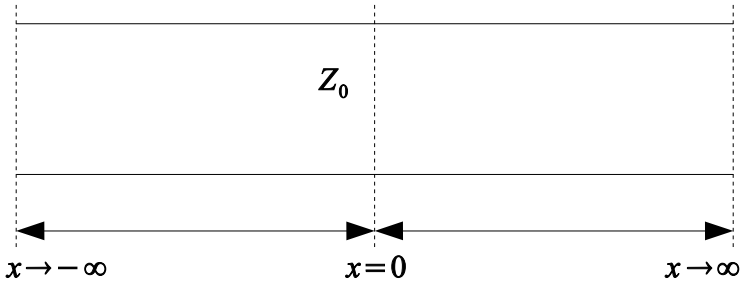


Figure B.3: *Both Way Infinite Line*

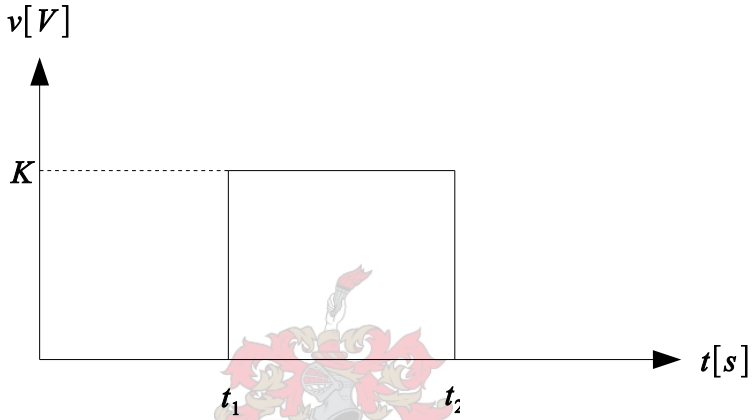


Figure B.4: *Rectangular Pulse*

B.2.2 Infinite Line with Distortion

For an infinite line, Fig. B.2, (B.18) is used and rewritten from (B.9) to obtain

$$\gamma = \frac{1}{c}\sqrt{\left[(s+\rho)^2 - \sigma^2\right]} \tag{B.25}$$

where

$$\rho = \frac{R}{2L} + \frac{G}{2C} \tag{B.26}$$

$$\sigma = \frac{R}{2L} - \frac{G}{2C} \tag{B.27}$$

Substituting (B.25) in(B.18) gives

$$V(s,x) = V(s)e^{-\frac{1}{c}\sqrt{\left[(s+\rho)^2-\sigma^2\right]}x} \tag{B.28}$$

For a rectangular pulse in Fig. B.4

$$V(s) = \frac{K}{s}\left(e^{-t_1 s} - e^{-t_2 s}\right) \tag{B.29}$$

(B.29) is used to obtain the inverse transform of (B.28) to realize (B.30)

$$\begin{aligned} v(t,x) = & Ke^{-\frac{\rho x}{c}}\left[u\left(t-t_1-\frac{x}{c}\right)-u\left(t-t_2-\frac{x}{c}\right)\right] \\ & +K\frac{\sigma x}{c}\left\{\left[\int_{\frac{x}{c}}^{t-t_1} e^{-\rho\tau}\frac{I_1(\rho z)}{z}d\tau\right]u\left(t-t_1-\frac{x}{c}\right)\right. \\ & \left.-\left[\int_{\frac{x}{c}}^{t-t_2} e^{-\rho\tau}\frac{I_1(\rho z)}{z}d\tau\right]u\left(t-t_2-\frac{x}{c}\right)\right\} \end{aligned} \tag{B.30}$$

where $z=\sqrt{t^2-\frac{x^2}{c^2}}$ and

$$I_1(\sigma z)=\frac{\sigma z}{2}\left[1+\frac{(\sigma z)^2}{2(4)}+\frac{(\sigma z)^4}{2(4)(4)(6)}+\cdots\right] \tag{B.31}$$

is the modified Bessel function of the first kind.

B.2.3 Distortionless Line

Heavyside first discovered the distortion less line, which is characterised by

$$\alpha=\frac{R}{L}=\frac{G}{C} \tag{B.32}$$

From the line with distortion (B.25) is taken and by using (B.32) the following relationship is shown:

$$\rho = \alpha=\frac{R}{L}=\frac{G}{C} \tag{B.33}$$

$$\beta = \frac{1}{c}=\sqrt{LC} \tag{B.34}$$

$$\alpha\beta = \frac{R}{Z_0} \tag{B.35}$$

For an infinite line propagation equation is expressed as

$$v(t,x)=e^{-\alpha\beta x}v_s(t-\beta x) \tag{B.36}$$

From (B.36) it can be seen that the $e^{-\alpha\beta x}$ term shows the attenuation factor, A.

B.3 Nominal and Equivalent Networks

For short transmission lines it is found convenient to replace the distributed constants with lumped constants in nominal T and Π networks as shown in Fig. B.5 and Fig. B.6. With

$$Z = (R+Ls)d \tag{B.37}$$

$$Y = (G+Cs)d \tag{B.38}$$

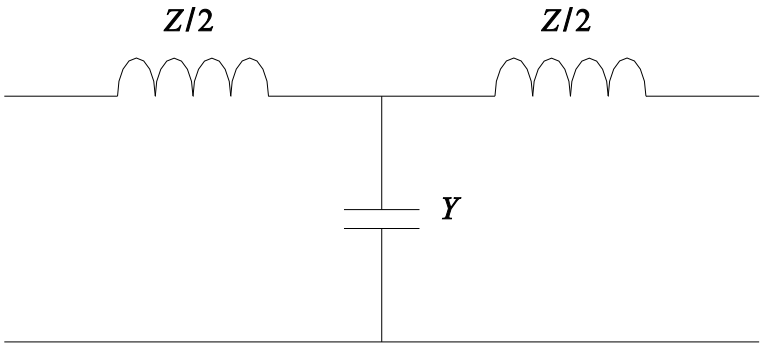


Figure B.5: *Nominal T Network*

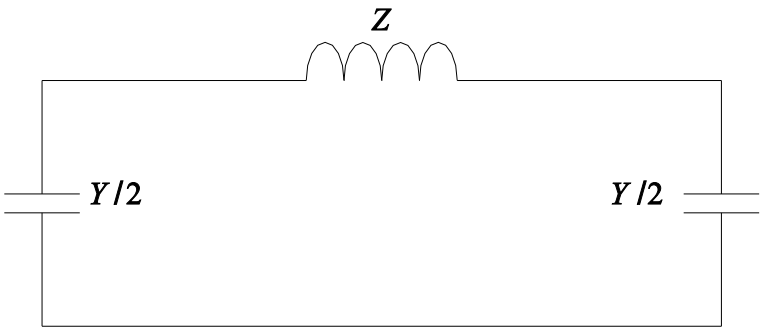


Figure B.6: *Nominal Π Network*

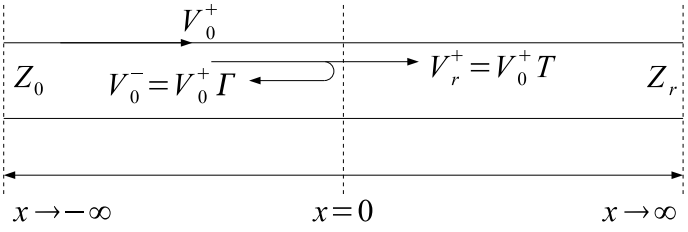


Figure B.7: *Reflection and Transmission Coefficient*

the following can be shown[46, ch.10-3, p.288]

$$I_r = I_s\left(1+\frac{ZY}{2}\right) - V_s Y \tag{B.39}$$

$$V_r = V_s\left(1+\frac{ZY}{2}\right) - I_s\left(2+\frac{ZY}{2}\right)\frac{Z}{2} \tag{B.40}$$

where s and r denotes the sending and receiving side of the transmission line. This allows the modeling of the transmission line with conventional circuit theory.

B.4 Reflection Coefficients

B.4.1 One line and Load Topology

The reflection coefficient is the main focus of this study. Reflections in the transmission line occur due to mismatching of the network elements due to impedance differences. The idea of TDR is to exploit this phenomena and use the reflections received from network elements to determine if additional illegal lines or loads are added to the reticulation network. The additional lines and loads are handled together, where Z_r can be replaced with Z_L. The extra transmission line, Fig. B.7, is considered with boundary conditions[39, ch 14-4, p.647]

$$V(s,0)_0 = Z_r I(s,0)_0 \tag{B.41}$$

$$V_0^+ + V_0^- = \frac{Z_r}{Z_0}\left(V_0^+ - V_0^-\right) \tag{B.42}$$

from which

$$V_0^- = \Gamma V_0^+ \tag{B.43}$$

can derived with

$$\Gamma = \frac{Z_r - Z_0}{Z_r - Z_0} \tag{B.44}$$

Pozar [65, ch 2.3, p.71] shows that the incident signal is transmitted onto the second line with a voltage amplitude given by a transmission coefficient multiplied by the incident signal amplitude. For $x < 0$

$$V(s,x)_0 = V_0^+\left(e^{-\gamma x} + \Gamma e^{\gamma x}\right) \tag{B.45}$$

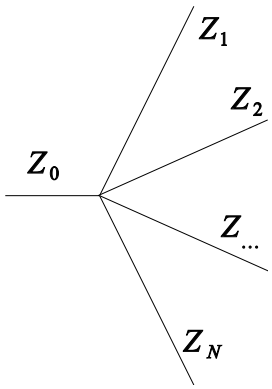


Figure B.8: *Split Pole Configuration*

V_0^+ is the voltage of the incident voltage signal on the feed line. It is assumed that a infinite long line in the positive direction and therefore do not expect any reflections and thus an absence of a new V_r^+ and can then be written for $x > 0$ as

$$V(s,x)_r = V_0^+ T e^{-\gamma x} \tag{B.46}$$

therefore at $x = 0$, $V(s,0)_r$ and $V(s,0)_0$ is equated to obtianed

$$T = 1 + \Gamma \tag{B.47}$$

B.4.2 Split Pole Configuration

Consider the split pole configuration as given in Fig. B.8.
The following conditions hold [35, ch.12.3, p.491]

$$V_0^+ + V_0^- = V_1^- = V_2^- = \cdots = V_N^- \tag{B.48}$$

$$I_0^+ + I_0^- = I_1^- + I_2^- + \cdots + I_N^- \tag{B.49}$$

which, by using $I_0^+ = \frac{V_0^+}{Z_0}$, $I_0^- = \frac{-V_0^+}{Z_0}$, $I_1^- = \frac{V_1^-}{Z_1}$, $\cdots$, $I_N^- = \frac{V_N^-}{Z_N}$, in (B.49), (B.50) is obtained

$$\Gamma_N = (Z_0||Z_1||Z_2||\cdots||Z_N)\left(\frac{1}{Z_0} - \sum_{k=1}^{N}\frac{1}{Z_n}\right) \tag{B.50}$$

and by using (B.48)

$$T_N = 1 + \Gamma_N \tag{B.51}$$

is obtained.
From Fig. B.9 it can be seen that for the thesis the reflections from a split pole with more than 4 connections can result in a large reflected signal, which will result in a small signal propagating further. For a configuration with four, 4-connection split poles, the reflected signal

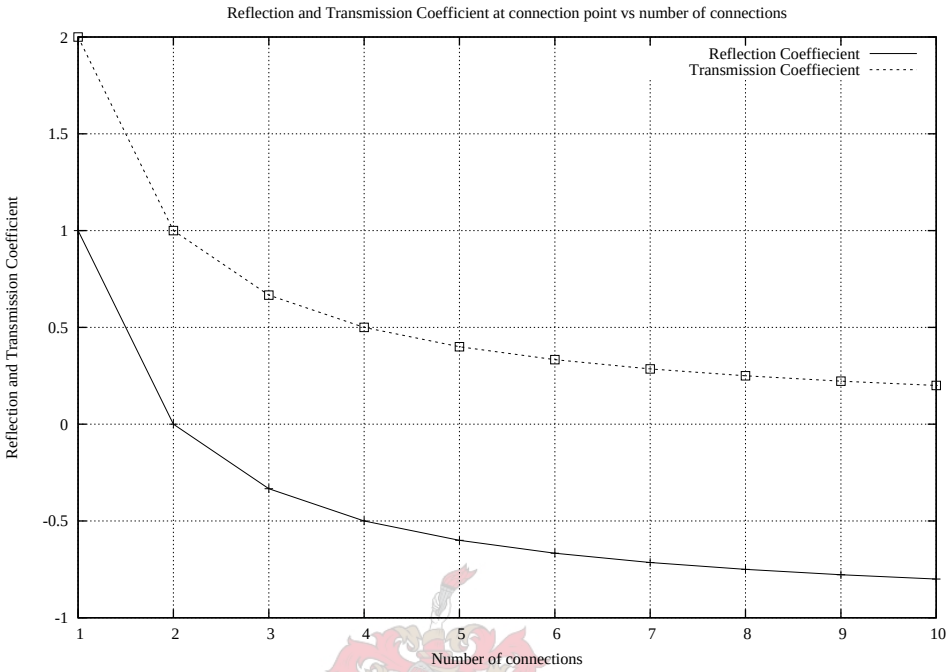


Figure B.9: *Results from Split Pole simulation*

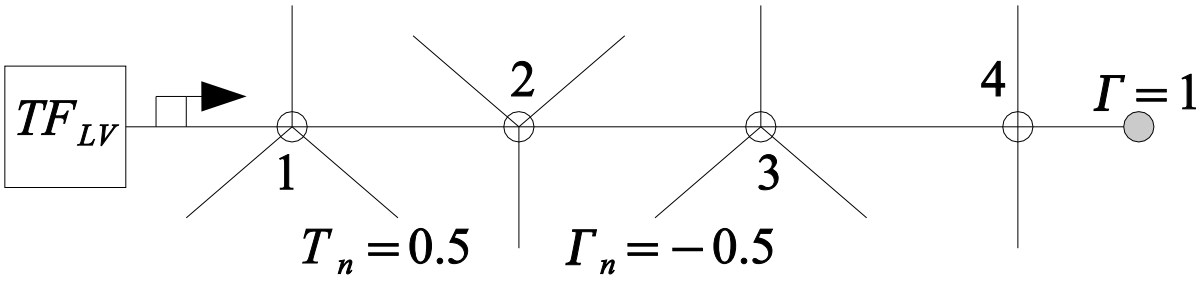


Figure B.10: *Example of Four Split Poles*

from an open line connected to the last split pole, as shown in Fig. B.10, will have a attenuation factor[1] of $1:2^8$ when the signal is measured at the low voltage transformer. For a standard 220V AC peak pulse it would mean a return voltage of roughly 1V, which is fairly small.

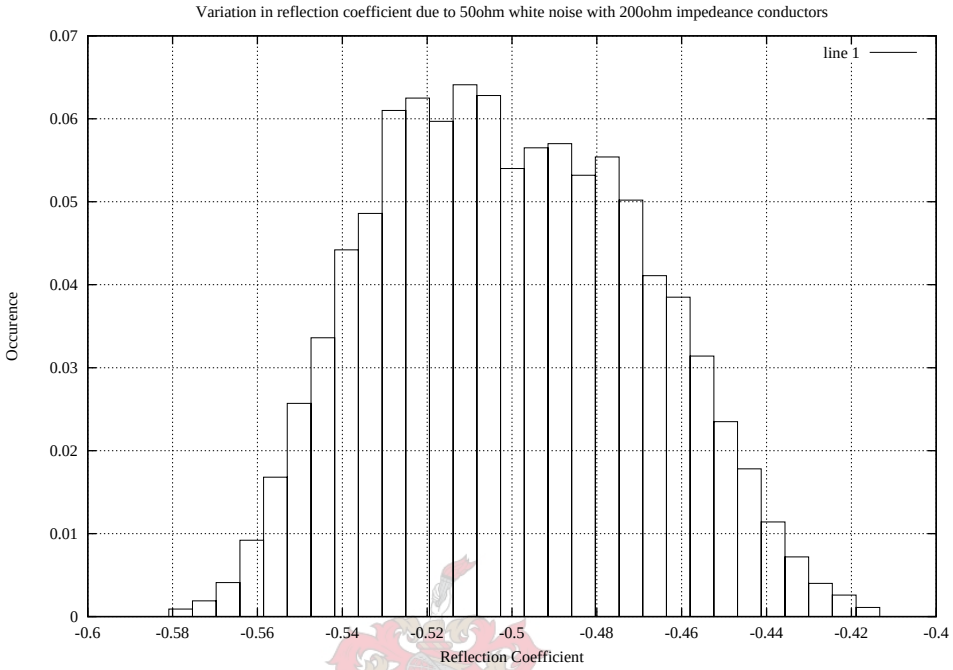


Figure B.11: *Reflection Coefficient Variation*

Power lines are not designed to transfer signals at high frequencies, therefore it can be expected that the impedance tolerances are low. Considering a split pole configuration with four connections with 12.5% tolerance we get the distribution as shown in Fig. B.11 when simulated with 10000 samples.

B.4.3 Scattering Matrix

It is of interest to represent the reflection coefficients. As indicated in (B.44) these can be calculated for one transmission line. It is required to retrieve and model many reflections. The use of the scattering matrix[65, ch 4.3, p.196] can prove helpfull. The scattering matrix is defined as

$$\begin{bmatrix} V_1^- \\ V_2^- \\ \vdots \\ V_N^- \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1N} \\ S_{21} & & & \\ \vdots & & & \vdots \\ S_{N1} & & \cdots & S_{NN} \end{bmatrix} \begin{bmatrix} V_1^+ \\ V_2^+ \\ \vdots \\ V_N^+ \end{bmatrix} \tag{B.52}$$

[1] Excluding normal signal attenuation due to line losses.

for a N-port network and can also be written as

$$[V^-] = [S][V^+] \tag{B.53}$$

$[S]$ can be calculated with

$$S_{ij} = \left.\frac{V_i^-}{V_j^+}\right|_{V_k^+, k \neq j} \tag{B.54}$$

For the configuration in Fig. B.8 it is found that with values $Z_0 = Z_1 = Z_2 = \cdots = Z_N$.

$$[S] = \begin{bmatrix} \Gamma & T & \cdots & T \\ T & \Gamma & & \\ \vdots & & & \vdots \\ T & & \cdots & \Gamma \end{bmatrix} \tag{B.55}$$

Pozar[65, ch.4.3, p.203] shows that the reference plain of the scattering matrix can be shifted by using a reference matrix and results in

$$[S'] = \begin{bmatrix} e^{-\gamma x_1} & & & 0 \\ & e^{-\gamma x_2} & & \\ & & & \vdots \\ 0 & & \cdots & e^{-\gamma x_N} \end{bmatrix} [S] \begin{bmatrix} e^{\gamma x_1} & & & 0 \\ & e^{\gamma x_2} & & \\ & & & \vdots \\ 0 & & \cdots & e^{\gamma x_N} \end{bmatrix} \tag{B.56}$$

where

$$[V'^-] = [S'][V'^+] \tag{B.57}$$

and x is the distance measured from where the reflection occurs.

B.4.4 Signal Flow Graphs

Pozar[65, ch.4.5, p.213] presents us with a method of graphically representing the scattering matrix. For completeness these rules and methods are summarized below:

Mapping A typical two port network is shown in Fig. B.12 and the equivalent signal flow graph in Fig. B.13.

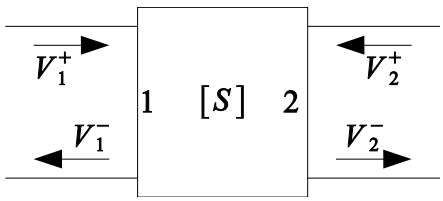


Figure B.12: *Two Port with Scattering Matrix Representation*

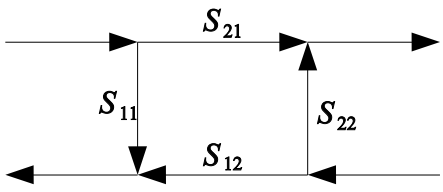


Figure B.13: *Two Port Signal Flow Representation*

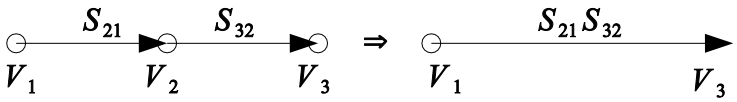


Figure B.14: *Series Rule*

Series Rule Two branches whose common node has only one incoming and one outgoing signal, can be combined to form a single branch as in Fig. B.14.

Parallel Rule Two branches from one common node to another may be combined as shown in Fig. B.15.

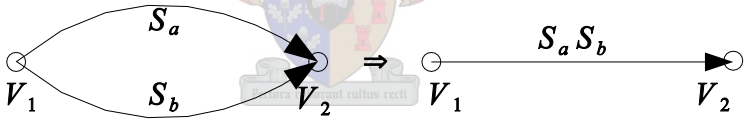


Figure B.15: *Parallel Rule*

Self-loop Rule When a branch begins and ends at the same node it can be eliminated as shown in Fig. B.16.

Splitting Rule A node may be split into two separate nodes as long as the resulting flow graph contains, once and only once, each combination of separate (not self loops) input and output branches that connect to the original node as shown in Fig. B.17.

B.5 Parameter Calculation

For simulation purposes it is required to calculate the cable parameters from the measured signals. The model in spice requires five parameters, e.g. L, C, G, R and distance of transmission line, ℓ. From (B.36) it can be seen that with the transmission line parameters can

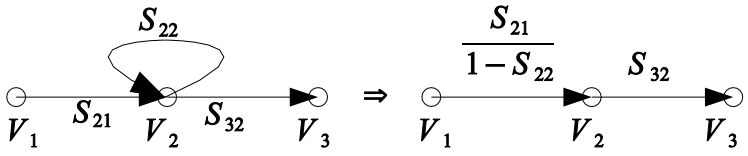


Figure B.16: *Self-loop Rule*

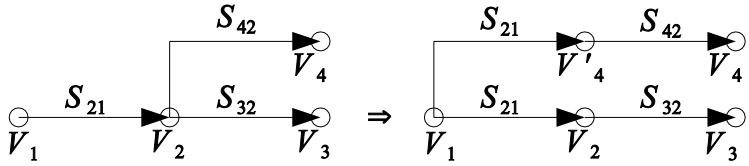


Figure B.17: *Splitting Rule*

be calculated by using the, attenuation A, line length ℓ, propagation time τ over the distance and surge impedance Z_0 and the distance of the transmission line. Since only four parameters are available a distortion less line is assumed and using the relations given in Section B.2.3 can be used to derived R, L, G and C:

$$A = \frac{|V_{in}|}{|V_{out}|} \tag{B.58}$$

$$A = e^{-\alpha\beta\ell} \tag{B.59}$$

$$\Rightarrow \alpha\beta = -\frac{\ln A}{\ell} \tag{B.60}$$

$$\beta = \frac{1}{v} = \frac{\tau}{\ell} \tag{B.61}$$

$$R = \alpha\beta Z_0 \tag{B.62}$$

$$L = Z_0\beta \tag{B.63}$$

$$C = \frac{\beta^2}{L} \tag{B.64}$$

$$G = \frac{RC}{L} \tag{B.65}$$

These parameters can now be put into the spice model and used for simulations.

B.6 Conclusion

Equations for the propagation, distortion and attenuation of signals in a transmission line with distributed parameters were derived and shown. It can be seen that the solutions, regarding the infinite line with distortion, are not easy to solve analytically. The lossless and distortion less solutions can be used to model the transmission lines.

Techniques giving us a better insight into transmission line reflections is presented in the form of signal flow graphs and the scattering matrix. These pose helpful when analizing the results of experiments.

Appendix C

Parallel Two Wire Parameters

L	$\frac{\mu}{\pi}\cosh^{-1}\left(\frac{D}{2a}\right)$
R	$\frac{R_s}{\pi a}$
C	$\frac{\pi\epsilon^{'}}{\cosh^{-1}\left(\frac{D}{2a}\right)}$
G	$\frac{\pi\omega\epsilon^{''}}{\cosh^{-1}\left(\frac{D}{2a}\right)}$

Table C.1: *Transmission Line Parameters for Parallel Two Wire configuration[65, Table 2.1, p.62]*

where

μ is the permeability,

R_s is the surface resistance

$\epsilon^{'}$ is the real part and $\epsilon^{''}$ is the imaginary part of the permeativity constant, where $\epsilon = \epsilon^{'} - i\epsilon^{''}$ and can be interpreted as $\epsilon = \epsilon^{'} - i\tan\delta$ [65, ch 2.2 ,p.61],

$\epsilon^{''}$ is $\tan\delta$,

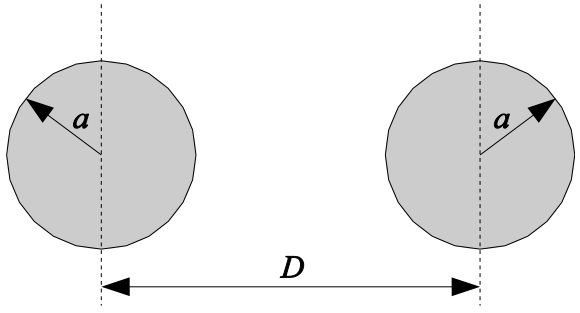


Figure C.1: *Parallel Two Wire Configuration*

ω is the applied frequency

L is the inductance per unit length,

R is the resistance per unit length,

C is the capacitance per unit length and

G is the conductivity per unit length.

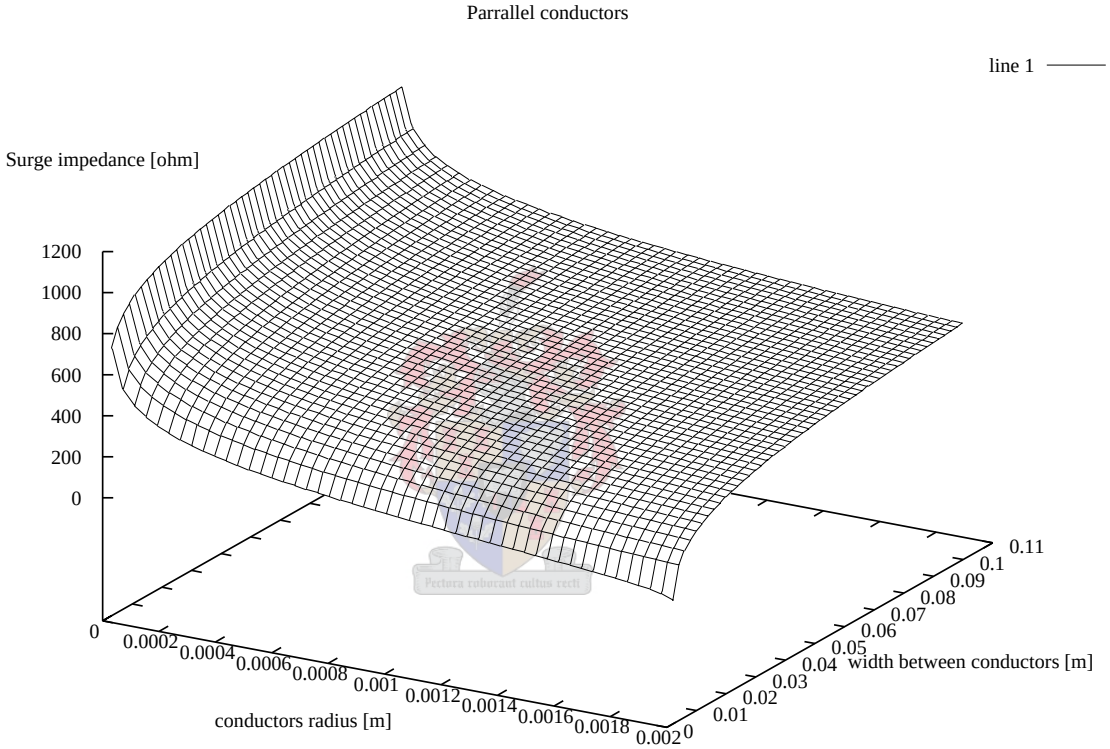


Figure C.2: *Parallel Two Wire Impedance*

Fig.C.2 shows the surge impedance of the parallel two wire configuration in terms of width between the wires and the radius of the conductors.

Appendix D

Pulse Generators

In order to implement Time Domain Reflectometry a suitable pulse generator is needed. Different Pulse generator technologies are discussed here.

D.1 Signal and Pulse Generators

Two power electronic device topologies, namely the half-bridge and parallel-switch are shown in Figs.D.2 and D.3. In our research some initial experiments were conducted using a few variations of these as well as a standard signal generator and glitch generator.

D.1.1 Signal Generator

A standard signal generator was implemented to start with the initial experiments as shown in Fig.D.1. With the signal generator it is easy to quickly get started with some experiments and we know that the impedance Z_0 is roughly 50Ω.

D.1.2 Glitch Generator

[92] uses a glitch generator in experiments to generate a pulse with a $-3dB$ pulse width of $40ns$. This device is extremely useful when a short pulse is required with a low repetition rate.

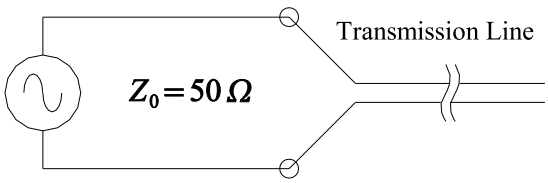


Figure D.1: *Standard Laboratory Signal Generator*

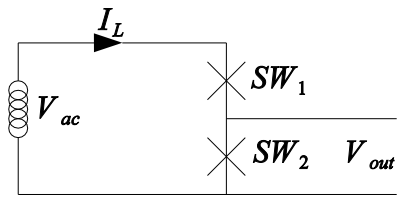


Figure D.2: *Half-Bridge Topology*

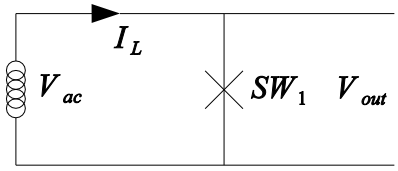


Figure D.3: *Parallel Switch*

The main part of this circuit is used in the parallel-switch design and discussed briefly in Section 3.4.2, page 18, and shown in Fig.3.5.

D.1.3 Half-Bridge

For higher voltage applications a half bridge switch (Fig.D.2) is provided in the laboratory. To get a feel for the power electronics it was decided to do a few experiments with this topology. The drawback of this configuration is that all the power delivered to the load must run through both SW_1 and SW_2 which, depending on the load on the network, can be around 25-200kVA[1] per transformer.

D.1.4 Parallel-Switch

The main advantage of the parallel-switch is that only during pulse generation time power is dissipated in the device. Therefore the power rating on the device is calculated depending on its position in the system.

D.1.5 Other Topologies

MPC The Magnetic Pulse Compression[76] unit can provide quick high voltage and energy pulses. Due to the mixed signal and possible harsh industrial environment the construction of the non-linear inductors was found too complex and therefore not used in the experiments.

[1]Typical values gathered from ESKOM. See Appendix E

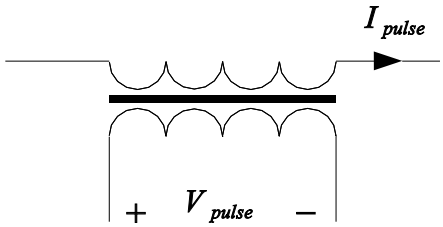


Figure D.4: *Current Transformer Topology*

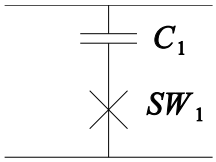


Figure D.5: *Capacitor coupled pulse generator*

Induction coupled pulse generator The use of an current transformer to induce current into the circuit transformer was investigated by sending a pulse through windings around small circular ferrite as shown in Fig.D.4. The results from these experiments were not significant enough and terminated.

Capacitor Coupled Pulse Generator The capacitor coupled topology as shown in Fig.D.5 was also considered and did not yield good enough results.

D.2 Experiments and Results

The experiments done in this seciton is for verification of the theory and for evaluation of the functionality of the various pulse generators. This section only covers the the following test phases namely, small signal experiments with a signal- and a glitch generator[92, ch.24].

D.2.1 Small Signal Signal-Generator Experiments

These experiment are variations of the one described by [92, ch.24]. The signal generator is used to generate the desired pulses. Reflection coefficients are easily controlled due to the fact that the signal generator and cables are matched at 50Ω. The typical setup is illustrated in Fig.D.6. We assume that the test setup does not have any reflection coefficients and therefore we will only consider the cable and devices under test in the explanation.

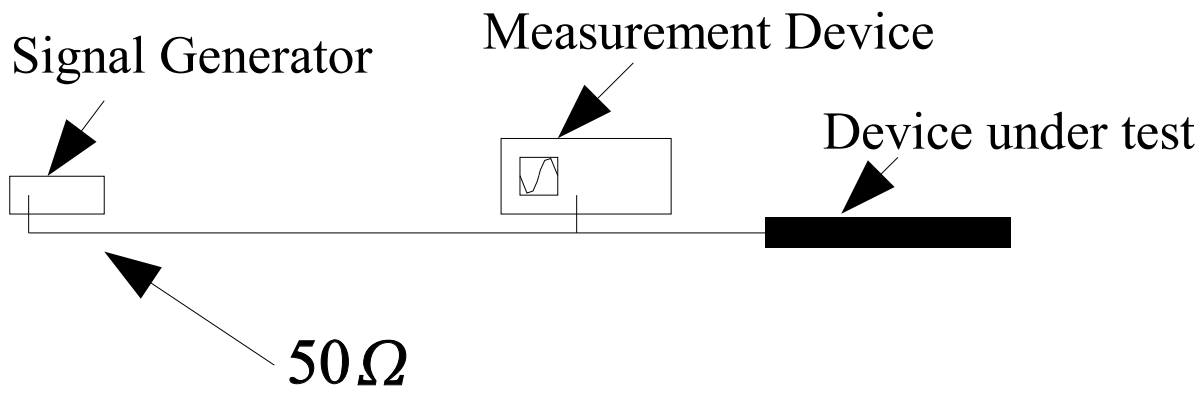


Figure D.6: *Default Test Configuration*

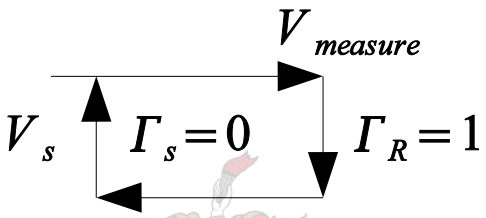


Figure D.7: *Signal Flow Graph of Open Circuit at Measurement Device*

Coaxial Line Reflection Coefficient Experiment

The idea behind the test is to get a idea of how reflection coefficients effect the circuit and give a feel for signal propagation in transmission lines.

Open Circuit Test From (B.44), with $Z_0 = 50\Omega$ and letting $Z_r \to \infty$ we get $\Gamma = 1$. Fig.D.7 shows the signal flow graph of the experiment. It can be seen from the results in Fig.D.10 that the reflected signal V^- adds to the forward going signal V^+.

Matched Circuit With a terminator of 50Ω added on the coaxial cable at the measuring point no reflections occur and the cable is matched. The signal flow graph is shown in Fig. D.8. Due to the fact that no reflection occurs at the terminated cable end, we only measure V^+ as seen in Fig. D.10.

24.6m Cable with Open End Consider a 24.6m 50Ω coaxial cable with an open circuit and repeat the experiment. A signal flow graph for the setup is shown in Fig. D.9. We can see from the results in Fig. D.10 that the pulse take some time to travel. Unfortunately the pulse width is

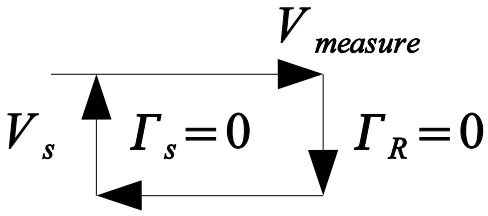


Figure D.8: *Signal Flow Graph of Matched Cable*

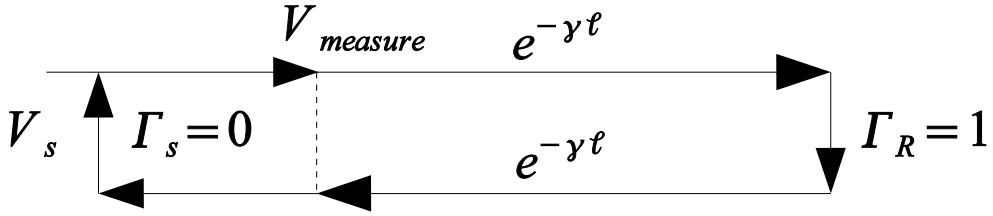


Figure D.9: *Signal Flow Graph for 24.6m Cable with Open Circuit*

too wide. The result is that the last portion from V^+ and V^- overlaps and therefore the small peak. From the rising edge of the first edge to the second edge is the total time for the pulse to return. From these it can be calculated that the speed in the coaxial cable with

$$t = 250\ ns \tag{D.1}$$

$$\ell = 2 \times 24.6\ m \tag{D.2}$$

$$v_{coax} = 196.8\ Mms^{-1} \tag{D.3}$$

This value is confirmed by [92, ch.24] with $v \approx 200\ Mms^{-1}$.

Results The results of the above experiments are shown in Fig. D.10. From the results we can confirm that the $V(0) = V^+ + V^- = V^+ + \Gamma V^-$.

Pulse Delay Experiment

The setup of this experiment is similar to the default experiment setup, except that the extra channel of the oscilloscope is used to measure the pulse at the end of the cable. The cables are matched and therefore only V^+ is measured. The signal flow graph for the two experiments are exactly the same as, shown in Fig.D.11, except that the delay and length of the different types of cables.

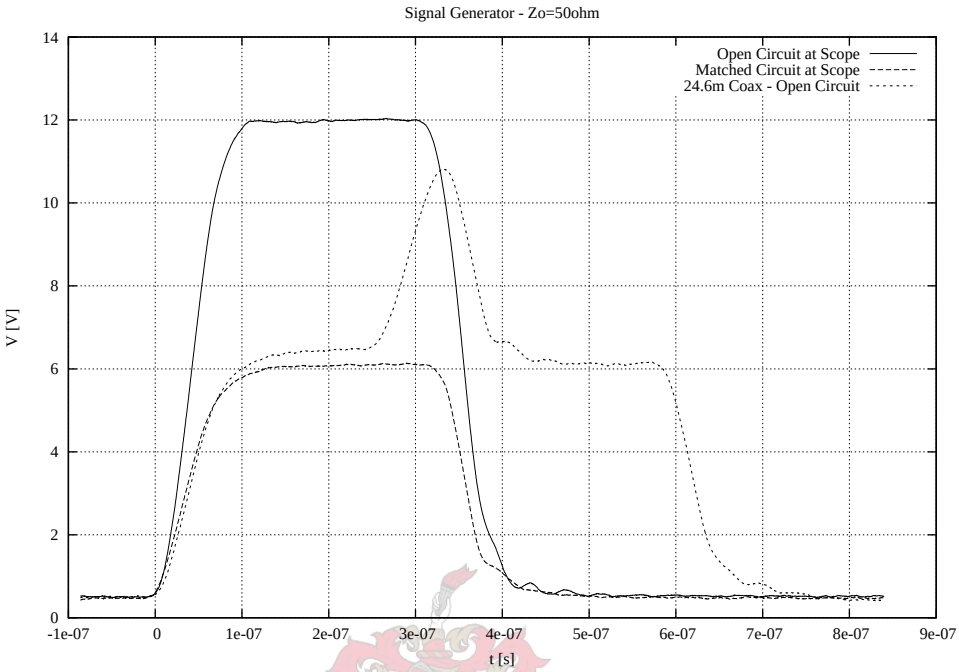


Figure D.10: *Results from Initial Experiment*

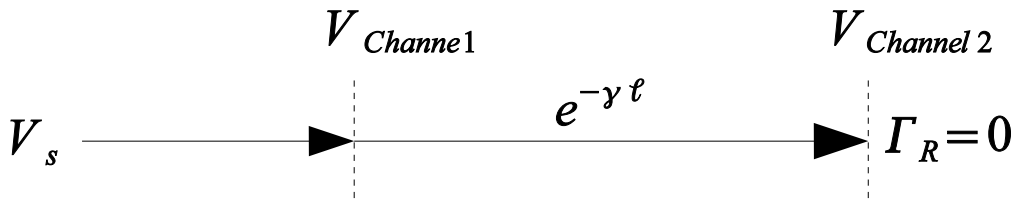


Figure D.11: *Signal Flow Graph for Delay due to Cable Length*

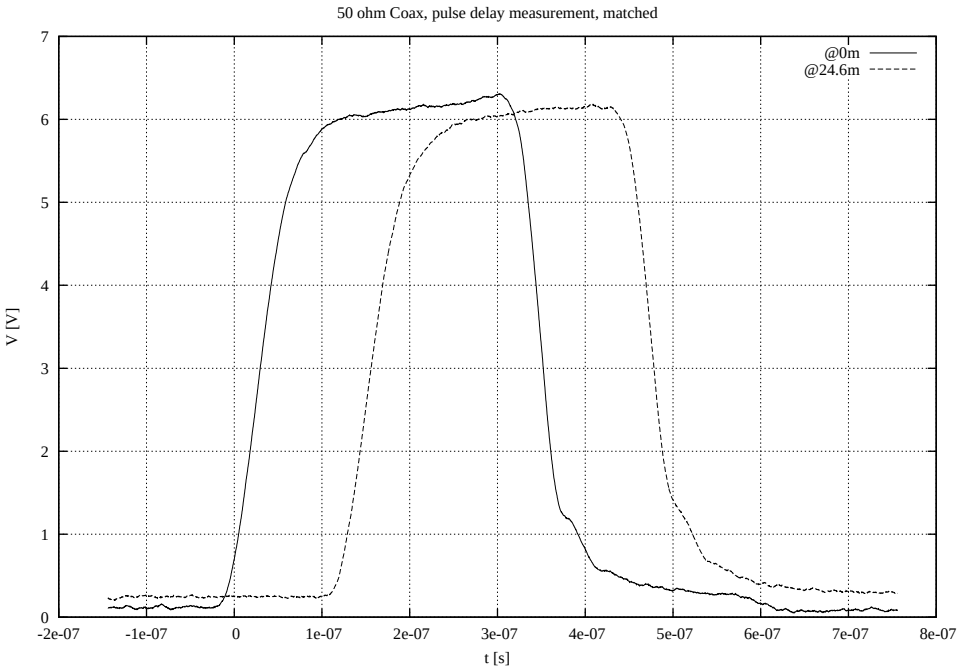


Figure D.12: *Delay Measurement in* 50Ω *Coaxial Cable*

Coaxial Cable A 24.6m coaxial cable is used for measurement. The results is shown in Fig.D.12 confirms the speed in a coax, where $t = 123.944ns$ and $\ell = 24.6m$ as $v_{coax} = 198.477Mms^{-1} \approx 200Mms^{-1}$.

Parallel Wire The $2.5mm^2$ parallel wire configuration is used. The wire is spaced roughly 5mm-15mm from each other. The isolation on the conductors were roughly 1mm. To match the cable a variable resistor was used. The results are shown in Fig.D.13. From the results the delay $t = 246.154ns$ over a distance of $\ell = 50m$ a speed of $v_{pw} = 203.125Mms^{-1} \approx 200Mms^{-1}$ is calculated.

Remarks The results shows clearly that pulses move along transmission lines as predicted by theory. The parallel wire and coaxial cable roughly has the same propagation speed.

Cable Junctions

For this experiment, the 24.6m coaxial cable is connected to the initial configuration shown in Fig.D.6. The cable is terminated with N-number coaxial cables, all having the same surge impedance. The results of the experiment is shown in Fig.D.14. The practical measure reflection coefficients are shown in Fig.D.15 against the calculated values as determined in (B.50). A summary of the results are given in the Table D.1.

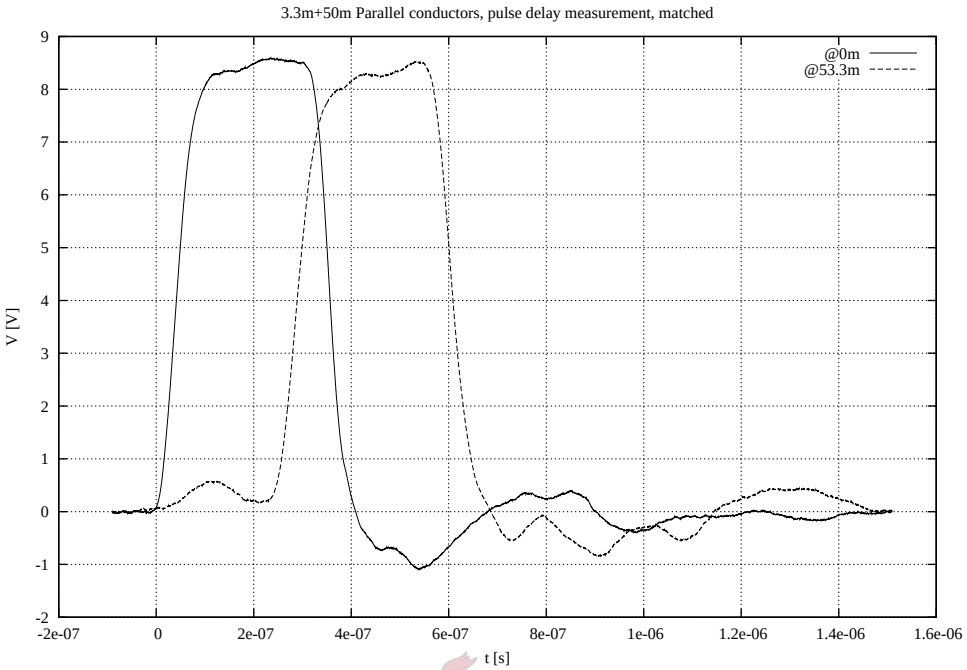


Figure D.13: *Delay Measurement in* $2.5mm^2$ *Parallel Wire*

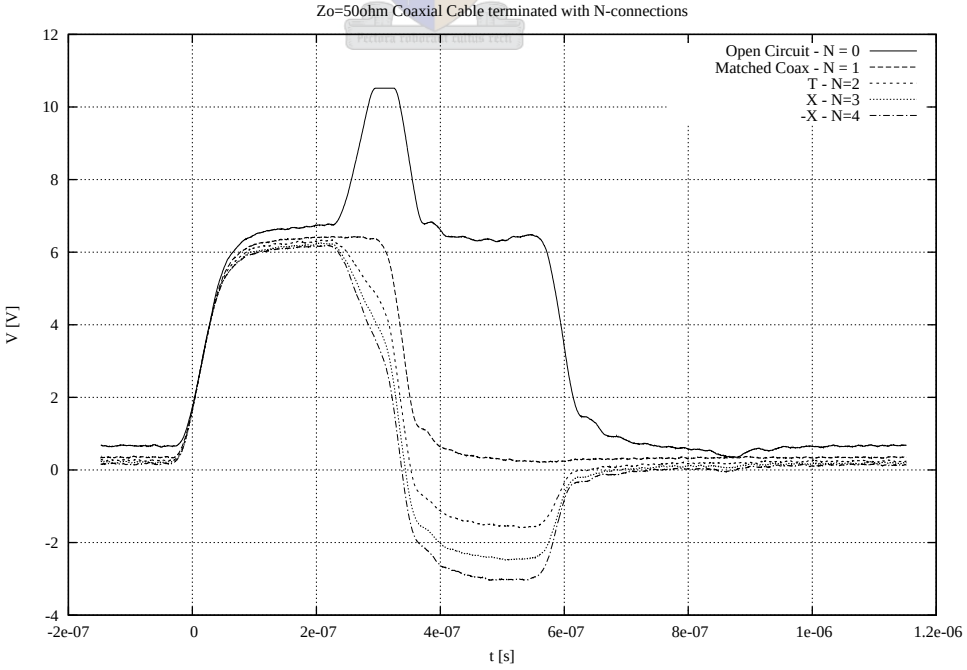


Figure D.14: *Measure Signals from N-conductor Split pole Configuration*

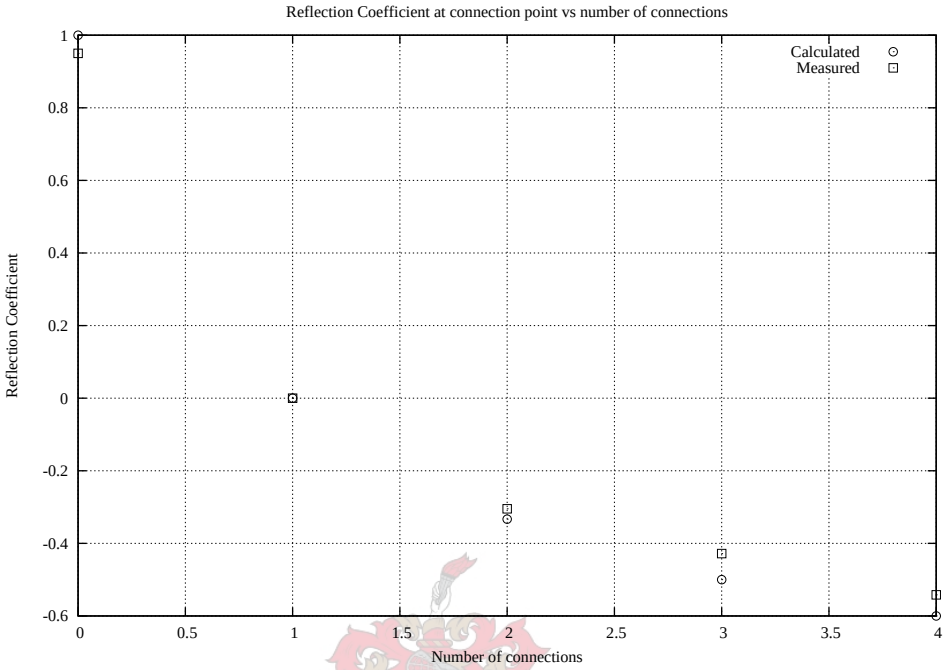


Figure D.15: *Theoretical vs Measured N-conductor Split Pole configuration*

N	$\Gamma_{measured}$	$\Gamma_{calculated}$
0	0.987	1
1	0	0
2	-0.308	-0.333
3	-0.428	-0.5
4	-0.542	-0.6

Table D.1: *Reflection Coefficient for N-conductor Split Pole configuration*

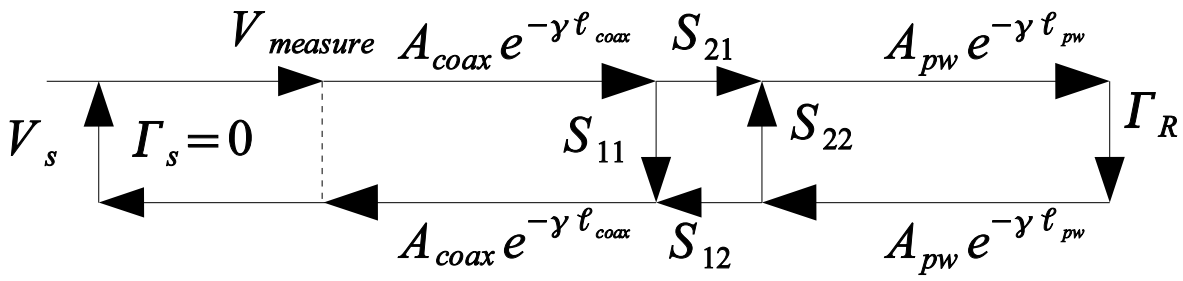


Figure D.16: *Signal Flow Graph of standard coaxial cable and parallel wire transmission lines*

D.2.2 Small Signal Glitch-Generator Experiments

To create a finer resolution and longer delay between pulses a glitch generator as proposed by [92, ch.24] was built and used for the mixed cable experiment.

D.2.3 Mixed Cable Experiment

For this experiment a 50Ω coaxial cable and the parallel wire transmission line is used and combined with the setup as shown in Fig. D.6. The parallel wire for this case was closely spaced next to each other.
First the open circuit reflection coefficient of the coaxial cable is measured. The next step is to extend the line with the 50m parallel wire. Two reflections is measured. For the second experiment with the open connection parallel wire fitted to the coaxial cable, four reflections are measured.
The signal flow graph for both experiments is shown in Fig.D.16. From the results in Fig.D.17 the different parameters for the scattering matrix where the junction occur in Fig. D.16 is calculated.

Coaxial Cable with Open Connection The 50Ω coaxial cable is considered. This is a well known standard and reference for the rest of the system. The coaxial cable is terminated with an open connection, which gives a $\Gamma = 1$. From here the attenuation A_{coax} is calculated:

$$V_{in} \approx 2.35V \tag{D.4}$$

$$V_{return} \approx 1.7V \tag{D.5}$$

where

$$V_{return} = V_{in} A^{+}_{coax} \Gamma A^{-}_{coax} \tag{D.6}$$

$$\Rightarrow A_{coax} = \sqrt{\frac{V_{return}}{V_{in}\Gamma}} \tag{D.7}$$

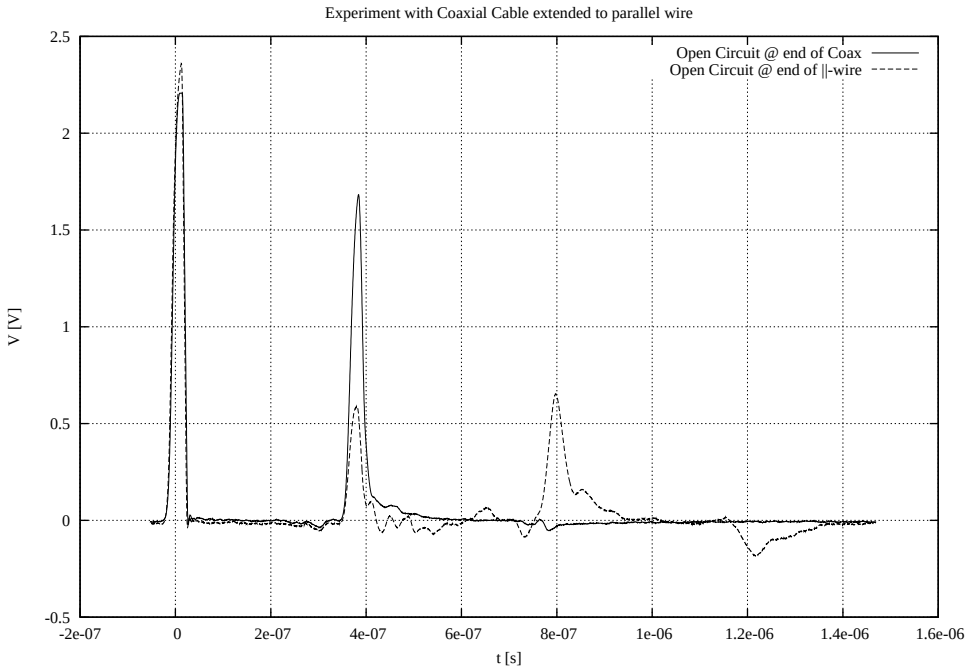


Figure D.17: *Reflections from Coaxial cable parallel wire experiment*

$$\approx \quad 0.850 \tag{D.8}$$

Parallel Cable with Open Connection Now consider the case where the parallel wire is connected. The first reflection with

$$V^{+} = V_{return_{coax_{open}}} \quad = \quad 1.7V \tag{D.9}$$

$$V^{-} = V_{return_{coax_{parallelwire}}} \quad = \quad 0.6 \tag{D.10}$$

is measured and hence the reflection coefficient is calculated as

$$\Gamma = S_{11} = 0.352941 \tag{D.11}$$

The transmission coefficient in the forward direction is calculated

$$T = S_{21} = 1 + S_{11} = 1.352941 \tag{D.12}$$

From (D.11) and (B.44) the surge impedance for the parallel wire is calculated

$$Z_0 = 104.545\Omega \tag{D.13}$$

Now S_{22} and S_{12} is calculated

$$S_{22} \quad = \quad -0.352932 \tag{D.14}$$

$$S_{12} \quad = \quad 0.647068 \tag{D.15}$$

The attenuation in the parallel wire is calculated. Due to the open circuit at the end it can be calculated with

$$\Gamma = 1 \tag{D.16}$$

$$\tag{D.17}$$

that

$$V^{+} = 2.35V \tag{D.18}$$

$$V^{-} = 0.65V \tag{D.19}$$

The total signal flow path for the first reflection from the open circuit is expressed as

$$V_{measured} = V_{sent} A_{coax} S_{21} A_{pw} \Gamma A_{pw} S_{12} A_{coax} \tag{D.20}$$

$$\Rightarrow A_{pw} = 0.661286 \tag{D.21}$$

With these known the next reflection coefficient is calculated

$$V_{measured} = V_{sent} A_{coax} S_{21} A_{pw} \Gamma A_{pw} S_{22} A_{pw} \Gamma A_{pw} S_{12} A_{coax} \tag{D.22}$$

$$\approx -0.1V \tag{D.23}$$

which is close enough to $-0.18V$ for a first iteration, although roughly 44% accurate, which can be attributed to possible distorion in the cable. When considering the time delay between the last two delays we see that these are roughly the same, $400ns \approx 420ns$. Which in turn give a propagation speed of $250Mms^{-1}$.

Discussion A difference in the propagation speed and the surge impedance for the parallel wire was seen when comparing the results of previous experiments. This can be attributed to the fact that for this experiment the cables were placed closer and precisely next to each other.

D.2.4 Conclusion

The experiments yielded good results. From the measured data the transmission line theory was verified. The fact that reflection coefficients at the split poles are large, as well as the relative high attenuation in the parallel wires is of great concern. The distortion levels in the parallel wire are larger than expected.

Appendix E

ESKOM Grabouw Field Visit - Nicky Schneider

A field trip was organized at ESKOM distribution - Grabouw to investigate the practical aspect of the project. Good insight was gained in how a Low Voltage Reticulation network is designed. The main network components were seen. Typical layouts and tampering cases were showed.

Weekday	Day	From-To	Location
Friday	04/06/2003	9:00-12:00	ESKOM - Grabouw
Project:	LV Electricity Theft		
Title:	LV Reticulation Field visit		

Participants			Notifications		
Company	Name	Location	Company	Name	Location
US-PEG	Riaan Doorduin	Stellenbosch	US-PEG	Prof. HdT Mouton	Stellenbosch
ESKOM	Nicky Schneider	Grabouw			

E.1 Contact Details

Tel.: (021) 859 4163

E-mail.: nicky.schneider@eskom.co.za

E.2 Network Components

The main power of an area is fed from the 3ϕ 11kV Δ network. A VT-CT combination device measures the power flow into this area. From this device 11kv:400V transformers are installed to supply the client side Υnetwork. Typical transformers seen ranged from 25kVA to 200kVA. This is illustrated in Fig.E.1.

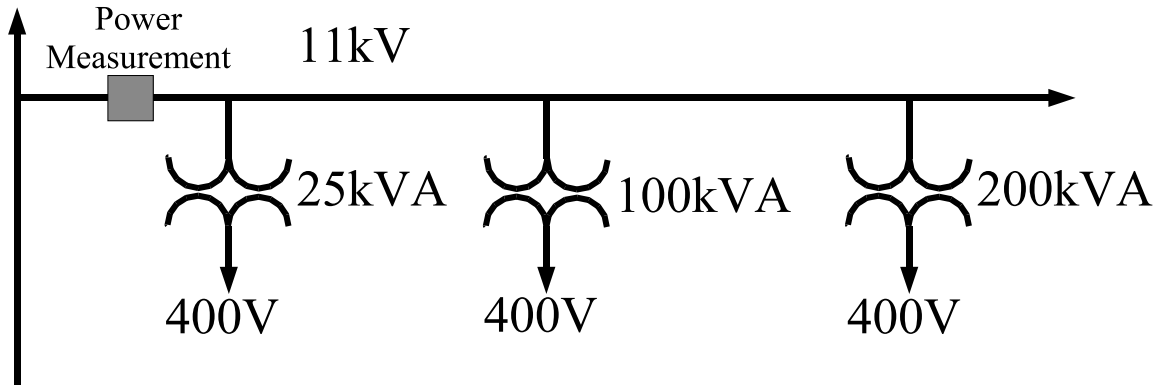


Figure E.1: *Medium Voltage distribution*

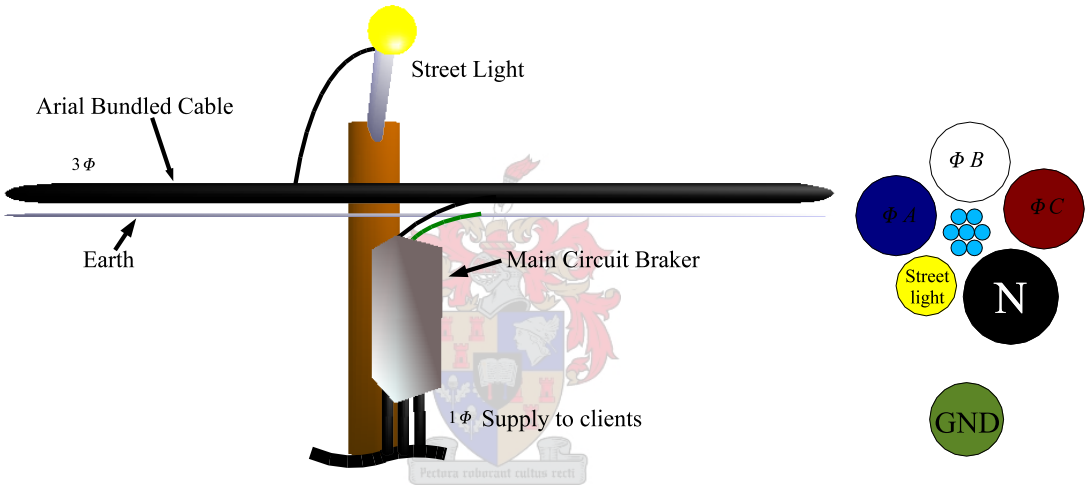


Figure E.2: *Cable configuration on overhead distribution poles and front view of ABC*

From the transformer the network is fed with ABC's[1]. These consist of a steel supporting cable(light blue), a neutral return path(black), street light feeder(yellow) and the 3ϕ's, blue, white and red. On the poles of the overhead network a earth cable(green) is also included. A typical pole and cable configuration is shown in Fig.E.2 with the MCB[2] isolating the single phase running to the clients houses. Up to nine houses can be connected to one of these MCB's.

[1]Aerial Bundled Conductors

[2]Main Circuit Breaker

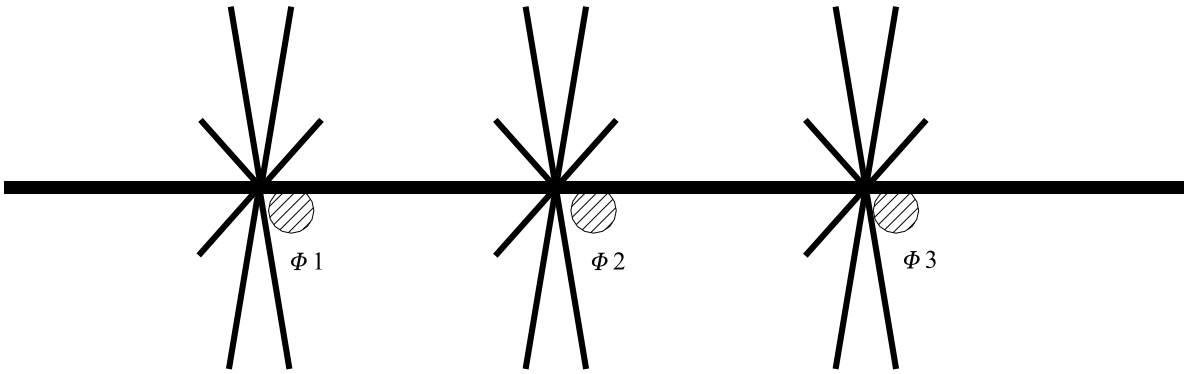


Figure E.3: *3 phase load distribution*

E.3 Network Layout

The LV network is laid out to optimize both cable usage and balance the load per phase. In Fig.E.3 it is shown how each third distribution pole use a separate phase, which optimize load-phase balancing.
The LV transmission lines are laid out similarly to a tree or root like structure. A typical layout is shown in Fig.E.5 and Fig.E.4. It can be seen that the ABC provide the cable to the nearest pole with a MCB. From the MCB a feeder cable provides power to the client.

E.4 Electricity Theft and Tampering

In Grabouw, Electricity theft and tampering is not common practice, although it was seen a few times. Tampering with the meters is reduced by changing to a new meter system and providing electricity to more houses, which reduce the need for private extensions of the network. However, some isolated cases were seen. Typically tampering occurs inside the huts. Flex and thin cables run either over and along the roads or through trees as shown in Fig.E.6. Typical flex, thin cable and $5mm^2$ wiring is used. This means that a higher than normal reflection coefficient could be expected due to a high probability of mismatching with an TDR solution.

E.5 General and Conclusions

The exposure to the a practical network gave incredible insight in both technical and social understanding of this particular environment. Network components and layout was discussed as well as typical tampering methods. The large distances between transformers and substations or ESKOM buildings will pose a problem setting up a possible measurement and

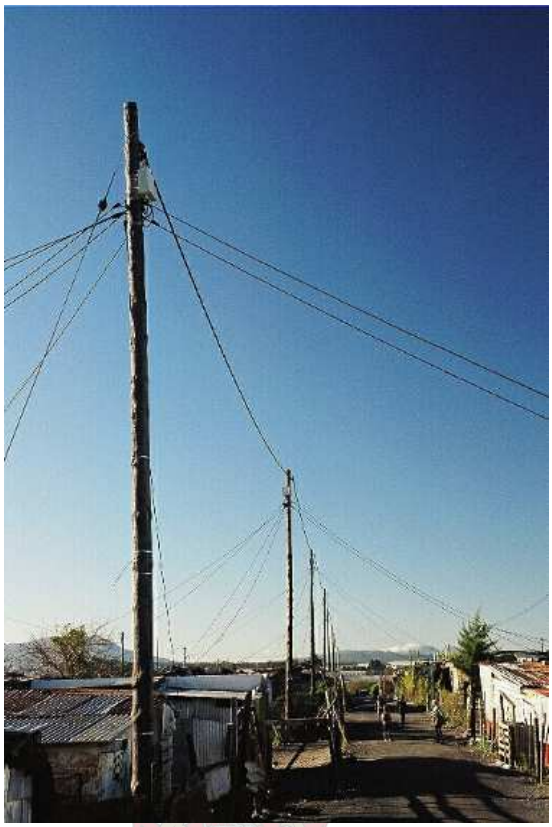


Figure E.4: *Reticulation Example*

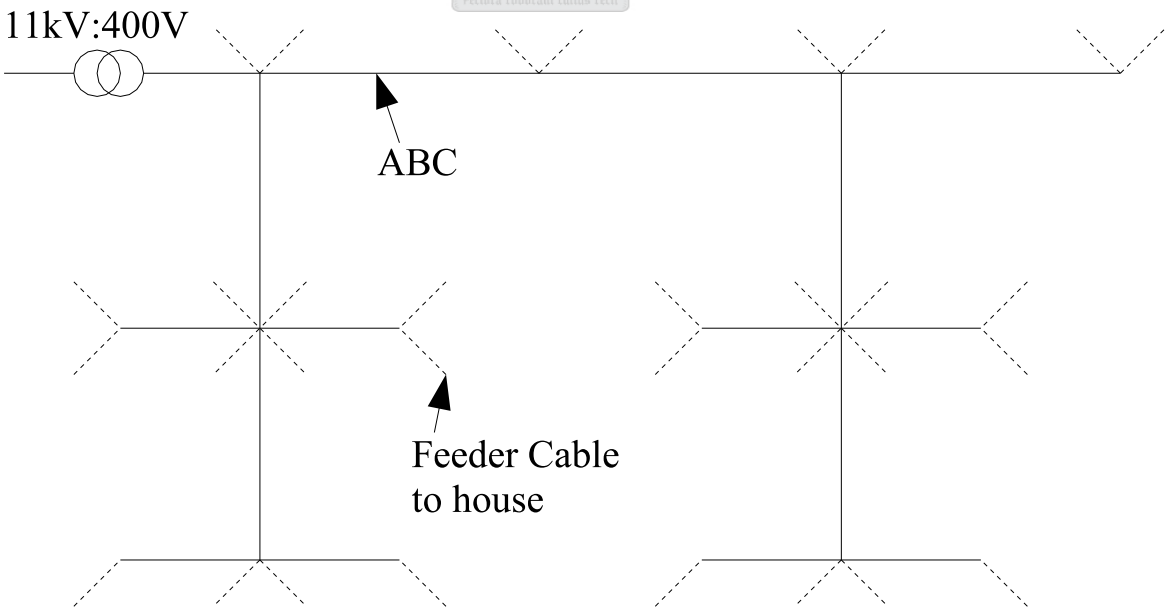


Figure E.5: *Low Voltage Reticulation Example*

Figure E.6: *Examples of Electricity Theft*

detection scheme. The length and size of the reticulation network may have a larger attenuation effect on the pulses generated with TDR than expected and must be investigated.

I would like to thank Nicky Schneider and Robert at ESKOM, Grabouw whom have made this valuable visit possible.

Appendix F

ESKOM Brackenfell - Andre Truter

Andre Truter is Security Manager at ESKOM Brackenfell. He works closely with SAP and all Electricity Theft cases in the Western Cape is handled by his department.

Weekday	Day	From-To	Location
Friday	30/07/2003	10:00-11:00	ESKOM - Brackenfell

Project: LV Electricity Theft

Title: Impact on Security

Participants			Notifications		
Company	Name	Location	Company	Name	Location
US-PEG	Riaan Doorduin	Stellenbosch	US-PEG	Prof. HdT Mouton	Stellenbosch
ESKOM	Andre Truter	Brackenfell	US-PEG	Dr. JH Beukes	Stellenbosch

F.1 Contact Details

Tel.: (021) 980 3533

E-mail.: andre.truter@eskom.co.za

F.2 Project Discussion

The overview of the current research was discussed up until 2003/07/30 and a presentation was given.

F.3 How does Eskom experience electricity theft and tampering?

F.3.1 Cable Theft

There are many ways to steal electricity and in broader terms this also includes the theft of cables.

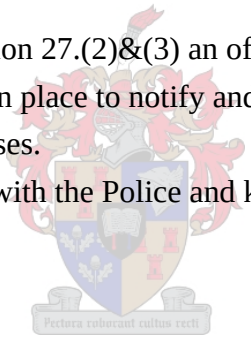
The most common practice is the theft of cables, especially the high voltage cables, which leads to many deaths. Effort is spent on educating the communities about electricity by means of the direct approach e.g. showing pictures of people shocked to death. The results of this initiative is still debatable.
Another approach is to remove the incentive of stealing cable, by applying strict regulation to the depots buying copper and cable. This problem is not limited to ESKOM and therefor a forum was established to initiate actions and ideas around the topic.

F.3.2 Electricity Theft

Cases of electricity theft and tampering on the low voltage network are known. Meters are either tampered with or bypassed. This happens on the split poles or ABC, on the airdacs and inside the huts, wherever the most appropriate place may be for the offender.

F.4 There is the law and then there is justice

According to Act 41 of 1987 Section 27.(2)&(3) an offender can be prosecuted for theft. ESKOM also has documentation in place to notify and offender of such activities and can disconnect the power to the premises.
Andre and his team work closely with the Police and keep trends of perpetrators.



F.5 Suggestions

Andre main concerns and suggestions are:

1. People must be sensitized
2. Mechanisms and systems must be automated.
3. Manpower is becoming a problem
4. Confronting perpetrator are extremely dangerous
5. Upfront vending as means of sales should be considered. - Units are prepaid for by the vendor to reduce ESKOM’s responsibility in terms of revenue. This will however imply that the vendor can sell the electricity at a higher price.
6. A cashless vending principle should be considered.

F.6 Contacts

The following people can be contacted for further information:

Jan Scholz	(021) 980-3122
	Power line communication and availability
	of data points on HV lines
Rens Bindeman	(021) 959-5352
	(082) 850-5318
	rens@pn.co.za
	SARPA - chairperson

F.7 Conclusion

Regarding a solution relevant to my study, LV Electricity theft prevention, the focus should be on automation, thus reducing the human factor as much as possible. If the study's solution can include cashless vending it would be an added benefit.

Appendix G

ESKOM Bellville - Monde Moletsane

Monde Moletswane is the Regional Revenue Protection Manager of the Western Cape.

Weekday	Day	From-To	Location
Friday	04/07/2003	13:00-14:00	ESKOM - Bellville
Project:	LV Electricity Theft		
Title:	Revenue Protection		

Participants			Notifications		
Company	Name	Location	Company	Name	Location
US-PEG	Riaan Doorduin	Stellenbosch	US-PEG	Prof. HdT Mouton	Stellenbosch
ESKOM	Monde Moletswane	Bellville	US-PEG	Dr. JH Beukes	Stellenbosch

G.1 Contact Details

Tel +27 21 915 2799
Fax +27 21 915 2928
E-mail monde.moletsane@eskom.co.za

G.2 Project Discussion

A brief idea of my current work was presented and discussed.

G.3 Current Situation

Cable theft is the main concern and is the main contributer to losses in the Western Cape region.
Regarding Electricity Theft and tampering, getting access to property for inspections and follow ups is a problem.

The customers experience ESKOM personnel as police and therefor cooperation is not good. Physical threats are common. There is expectation of permanency which results in demands in terms of electricity. Some consumers view it as a basic right.
Another problem is the illegal distribution of electricity. This involves paying customers selling from their electricity to neighbors via extension leads. This pose a safety threat. Education is used as a means of opposing the problem.
Losses due to electricity theft in the Western Cape roughly amounts to 10% (R6 million per annum) of the total energy distributed.

G.4 Availability and Geographical Correlation

Most of the problem does not occur in the rural areas, but in settled urban areas, where the community have more criminal tendencies. Post and prepaid metering are both targeted. Regarding this issue, Monde, found that people in rural areas are generally willing to pay. And that the losses, in terms of Electricity theft, here are really small compared to urban areas.
As mentioned in my presentation I pose the question that providing electricity to illegal users, would then reduce the problem. It was agreed upon that the opposite is also true, that providing electricity to more users makes the network more vulnerable to tampering and electricity theft.

G.5 Processes

Meter audits are done on a door to door basis. This is the safest way of knowing what is happening in the field. To refine and narrow the audits exception reports are drawn from the available data. These reports are only as accurate as data is maintained in the system, and it does occur that meters with a zero reading was removed by ESKOM personel and not updated in the system.
Energy balances also indicate areas of high revenue losses where losses are calculated on the sold units vs. used units.

G.6 Prepaid Metering

Revenue protection had its origin in the 90s and grew hand in hand with the evolution of prepaid metering. Due to the fact that these meters are out of sight and that the meter is not visited once a month, it is more susceptible to tampering.
Prepaid metering also introduced ESKOM to a new type of customers.

G.7 Postpaid Metering

Monde explained that the current focus is on postpaid meters, since these meters are also tampered with. Although postpaid meters are less in number than their prepaid counterparts, postpaid meters account for more revenue.

G.8 Basic Free Electricity

Government indicated that, as with water, people have a right to energy. Monde argues that there are other resources which they can turn to and if this grant will help the poor is still highly debatable. In the same light it was found that the perception exists that electricity is an expensive form of energy and therefor people use electricity only for lighting purposes. I guess that education plays an important role here.
From the 1st of July people can apply for 50kWh per month at the local authority. Government budgeted R300 million for this year and R1.7 billion for next year. The shortfall here is that no profile exist for deciding on who should get this subsidy.

G.9 Conclusion

Concerns such as accessibility to premises and physical threats must be addressed. Human error must be minimized in terms of data collection. Postpaid meters must also be included in the analysis of electricity theft prevention and a possible solution must be applicable in both pre- and postpaid environments.

Appendix H

ESKOM TSI - Dr. Nielsen

Dr. Shawn Nielson is affiliated with both ESKOM TSI and the University of Pretoria. His current research involves detecting water in HV insulators.

Weekday	Day	From-To	Location
Thursday	14/08/2003	10:45-12:00	ESKOM-TSI - Cleveland

Project: LV Electricity Theft

Title: TDR in a LV Reticulation Network

Participants			Notifications		
Company	Name	Location	Company	Name	Location
US-PEG	Riaan Doorduin	Stellenbosch	US-PEG	Prof. HdT Mouton	Stellenbosch
ESKOM	Dr. Shawn Nielsen	Cleveland	US-PEG	Dr. JH Beukes	Stellenbosch

H.1 Contact Details

Tel +27 12 629 3569

Fax +27 12 629 5454

Cell +27 83 617 4820

E-mail shawn.nielsen@eskom.co.za

H.2 Project Introduction

A brief overview of the project was given, with the specific focus on TDR. Dr. Nielson was impressed by the parallel switch solution i.t.o. generating an in line pulse and the sharp dV/dt's measured using this method.

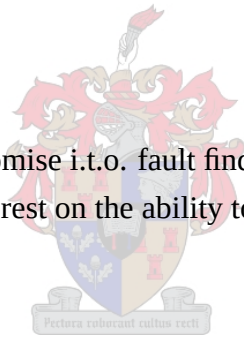
H.3 Tips & Tricks

Dr. Nielson proposed a directional bridge circuit to only measure pulses sent from the pulse genrator. This solution can propably be used to illiminate reflections from the pulse generator

itself.
By changing the pulse width information could be extracted using FDR (Frequency Domain Reflectiometry). The longer the line the higher the pulse frequency content must be. Standing waves would then be used to extract information from the network.
The main problem with TDR in a LV network would be to decode or interpret the signals received. Therefore the network must be modeled extremely accurate and methods should be investigated on how to estimated and determine connections. Since we are measuring in one dimension, GIS and other data can be used to deterimine the propability of certain phenomina measured by the system.
Temperature would not have a great impact on measurements since it would be spread evenly over a distance.
Dr. Neilsen was extremely positive about the current progress in the project. The possibility of other spins offs from the results of the study could be interesting. Detection of small faults could be made possible. However the relaxation of capacitive loads could pose a problem and must be investigated.

H.4 Conclusion

The study thus far shows great promise i.t.o. fault finding on cables. The success for detecting tampering in a LV network would rest on the ability to interpret the recieved signals.



Appendix I

Ekurhuleni Metropolitan Municipality - Dave Jamieson

Dave initiated the installation of a AMR (Automatic Meter Reading) system in the Thembisa area. Although expensive, the system have proved itself since 1996.

Weekday	Day	From-To	Location
Friday	15/08/2003	11:00-13:30	Thembisa
Project:	LV Electricity Theft		
Title:	AMR in a LV Reticulation Network		

Participants			Notifications		
Company	Name	Location	Company	Name	Location
US-PEG	Riaan Doorduin	Stellenbosch	US-PEG	Prof. HdT Mouton	Stellenbosch
EMM	Dave Jamieson	Thembisa	US-PEG	Dr. JH Beukes	Stellenbosch

I.1 Contact Details

Tel	+27 11 921 2477 / 926 2742
Fax	+27 11 975 4614 / 926 2804
Cell	+27 82 490 2627
E-mail	davej@ekurhuleni.com
Web	http://www.ekurhuleni.com

I.2 Project Introduction

Dave introduced me to the project at Ekurhuleni. He compiled a complete folder with presentations and articles.

I.3 Extract from Papers and Background

I.3.1 From ”Saga of protective structures”

Reference: [44]

Background

In January 1995 the Tembisa network with its 31500 consumers became the responsibility of Kempton Electricity with a bulk bad debt of R250 million. This debt was increasing with interest at prime and an alarming R3 million per month due to a non-payment campaign in the community. Networks were also overloaded due to a *self-help culture* in the form of illegal connections. An investment of over R100 million i.t.o. network renovation, electrification, meter replacement and the deployment of SCADA and telemetry was made to address this problem.

Why this system

Prepaid metering was introduced in the 1980s and was soon proved prone to electricity theft and fraud. The ingenuity of the domestic consumer was either totally overlooked or underestimated. Technical losses increased due to this and in 1996 more than 50% of all pre-payment meters in South Africa were bridged.

> Initially this loss represented a relatively small percentage of total energy sold, but with South Africa venturing into the ambitious electrification drive as it did in 1990 with less than 40% of all households being electrified at the time, it was realized that this sector would be growing cosiderably with resultant increasing losses to the industry. In 1995 the outstanding debt in this sector, country wide, amounted to a sum in excess of R2.0 billion.

Uniqueness of the system

System requirements were defined:

- Two way communications
- Meter and display separation (split meter)
- Telemetering, remote billing, cash collection and automated energy auditing
- Devices must have tamper detecting
- Long term cost effectiveness

System lay out

Up to 20 meters can be connected to concentrator in a mini sub. The substation communicates via a radio network to the control center. (Will be elaborated later in these minutes.)

Project implementation

Community involvement was considered to be of high priority. The first meters were installed in February 1997.

Protective structures

Tampering to the municipalities meters were common practice. This problem was addressed by experimenting with different protective structures. 10mm thick hydraulically operated steel structures were put over the metering kiosks and miniature substations at a cost of R24 million. With all these measures in place (including AMR) payment levels rose from ±R1million to R8.6million per month.

Latest developments

> The total cost for this investment in new technology(High-tech metering & protective structures) soon paid for itself not only from additional revenue received and reduce cost to suppliers (ESKOM) but also the savings in effective manpower utilization for maintenance and new projects as apposed to continually removing illegal connections and repairing vandalized equipment.

Regognition award

The Kempton Park/Tembisa Electricity Department together with Intelligent Metering Systems won the Residential Category of the prestigious 2002 ETA Award for promoting energy efficiency.

I.3.2 From "'Conversion to Pre-paid"'

Refference: [43]

Background

The removal of charges[1] was a strong motivation for customers to use to prepayment instead of conventional metering. Awareness and understanding of how the system worked was achieved

[1] R25 for the delivery of notice and R150 for discontinuing and restoring supply

by educating the community through a comprehensive market strategy.

The System

The system works in both post and pre-payed mode. A card is issued to each stand and is used to eliminate finger problem at the vending stations[2]. The metering system can communicate bi-directionally via various communication mediums to the control center. When a sale is made the new credit is loaded onto the stand's meter and can be used. To update the credit on the meter roughly takes 15 minutes. At the beginning of the month 50 units are loaded onto each consumers meter in order to assist those who cannot afford essential services.

Meter Advantages

- Both pre and post-payed functionality - without meter readers
- Remote management
- Tamper and bypass alarm
- Card eliminates finger problems at display terminal
- Credit is loaded automatic
- Monthly, weekly and daily records are stored per consumer
- SQL compliant database functionality implies easy integration
- Account details can be changed instantly
- Power can be switched on in case of an emergency
- Faulty equipment can easily be identified
- System priorities can be set in terms of maintenance tasks, e.g. Reconnection of customer, alarms and non functional meters
- Up to date information on customers display
- Maintenance reports can be printed daily
- Tampering alarms is monitored in the control center
- Display is operated via mains borne modem, therefore no additional communications cables are required
- Split meter principle i.t.o. meter and display (Accessible to Technicians)

[2]Ideal for the illiterate

Meter Disadvantages

- Expensive capital layout
- Not STS compliant
- Display does not function when electricity is cut off, due to mains modem
- Sole supplier of equipment

Conclusion

AMR has been found to be successful in terms of revenue protection in the Ekurhuleni Metropolitan Council.

I.3.3 From presentation ”Conversion to Prepayment”

Additional extracts from [43]

Levies vs. Payments

June 2002	30%
December 2002	51%
June 2003	59%

I.4 Technical Observations

The technical personal were introduced, as well as the equipment used in the system. A visit to the control room as well as printouts from the database was shown.
A technician showed how the system is implemented.
The vending system is very simple to operate and clerks are familiar with the process. People in general seem happy with the system.
Operation of the protective structures were shown. These are operated by hydraulic and electrical means. Mechanism to detect tampering are installed and in some cases these structures are connected to the metering system via optical fibre.
Hans said that in one case, a list of roughly 2500 were disconnected within 2 hours.
A substation was also shown.
The card used for customer identification in the system was shown. The magnetic strip only contains the account number. On the card, both the account number and stand number is shown. The stand number can be decoded as follows:

S72/	001/	0990	/0000
area	extension	stand	portion

I.5 Contacts

Further technical information can be requested at qurynl@ekurhuleni.com.

I.6 Conclusion

Ekuhurleni is currently reaping the benefit of a well thought out system. The load on human resources is lower than with the previous system. The combination of protective structures and AMR reduces tampering. The sole supplier situation is not healthy.

Appendix J

Stellenbosch Municipality - Kevin Bey-Liveld

Kevin is involved with the Municipalities' investigations into Automatic Meter Reading (AMR).

Weekday	Day	From-To	Location
Friday	22/08/2003	11:00-11:45	Stellenbosch
Project:	LV Electricity Theft		
Title:	Check Metering		

Participants			Notifications		
Company	Name	Location	Company	Name	Location
US-PEG	Riaan Doorduin	Stellenbosch	US-PEG	Prof. HdT Mouton	Stellenbosch
S.org	Kevin Bey-Liveld	Anglo Building	US-PEG	Dr. JH Beukes	Stellenbosch
S.org	Gerhard W.	Anglo Building			

J.1 Contact Details

Tel +27 21 808 8406

E-mail kevinb@stellenbosch.org

J.2 Project Overview

The project was briefly explained and how the involvement of Stellenbosch Municipality can help the project.

J.3 Discussion

Stellenbosch Municipality does not experience large quantities of electricity theft. However they are planning to start investigating AMR.
Their main concern are in the bulk electricity users. The total income per year from 400 users is R80 million per year. For the 12530 household meters the income per year is roughly R12 million. For the period from July to August R1.8 million per month was consumed by households.
The extend of possible theft in Stellenbosch is not determined yet. Proposals of using InfoPOD's (R150 per meter) to collect data was found an expensive solution.
It was suggested, that the focus of the LV electricity theft study must be changed to bulk electricity users, since the usage here justifies the extra costs incurred for equipment.
Actaris have currently been involved in the deployment of 300 AMR meters in the Blouberg area.

J.4 Joint Venture

The possibility of working together with the municipality was discussed. Stellenbosch Municipality have ten meters and modems to use as check meters. Involvement from the Power Electronics group could be beneficial if an arrangement between the two parties can be reached. The results from the venture can be used for research and a business plan, to motivate further AMR installations, can be drawn up.

J.5 Contacts

Gerhard V.	+27 21 808 8406	gerhardv@stellenbosch.org
Imran Mohammid	+27 83 306 2413	Actaris AMR

J.6 Action Items

Riaan Doorduin will discuss the proposal with Prof. Mouton. Depending on the outcome, the final proposal from the Power Electronics Group will be sent to Kevin. Kevin Bey-Liveld will discuss the proposal with Stellenbosch Municipality.

J.7 Conclusion

The possiblity of joining strengths could pose tremendous progress in terms of revenue protection for Stellenbosch Municipality and give the study a good practical insight into AMR

as well as real-world figures.

”What you can not measure, you do not know.” In general municipalities are not aware of how much electricity is stolen in various areas. The losses are then assumed to be technical losses in the network and are recovered by price increases. To develop a low cost remote check meter could speed up detection and identification of electricity theft.

Appendix K

Data Mining Loss Calculations

K.1 Site Layout

The site layouts is provided courtesy of Stellenbosch Muncipality. [77]. An aerial photograph from the GIS system is shown in Fig. K.1 and the design of the circuit is shown in Fig. K.2 [18].

Figure K.1: *Mooiwater Arial View*

Figure K.2: *Engineering Drawing of Site*

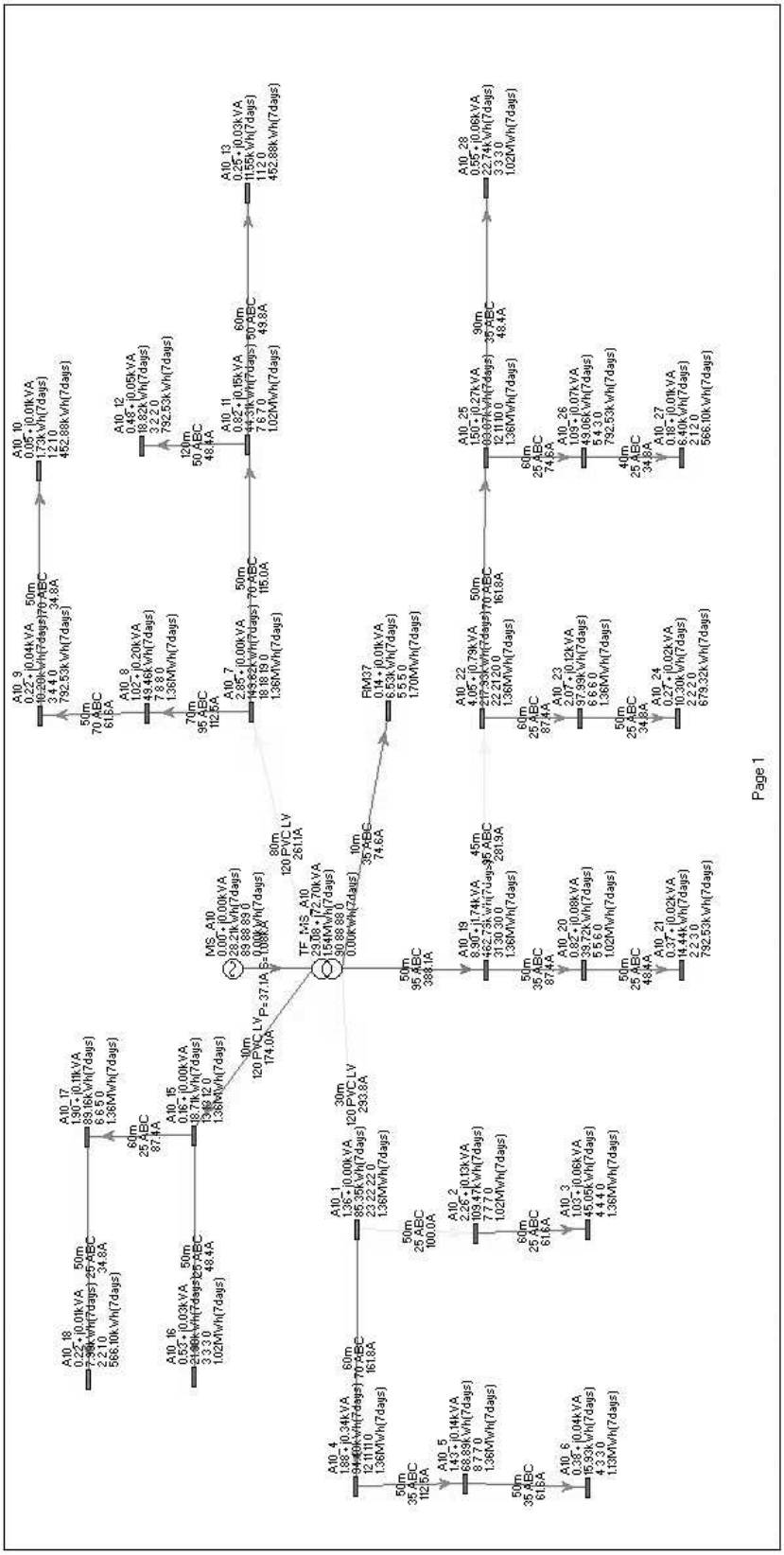


Figure K.3: *40A Standard user Retic-Master layout*

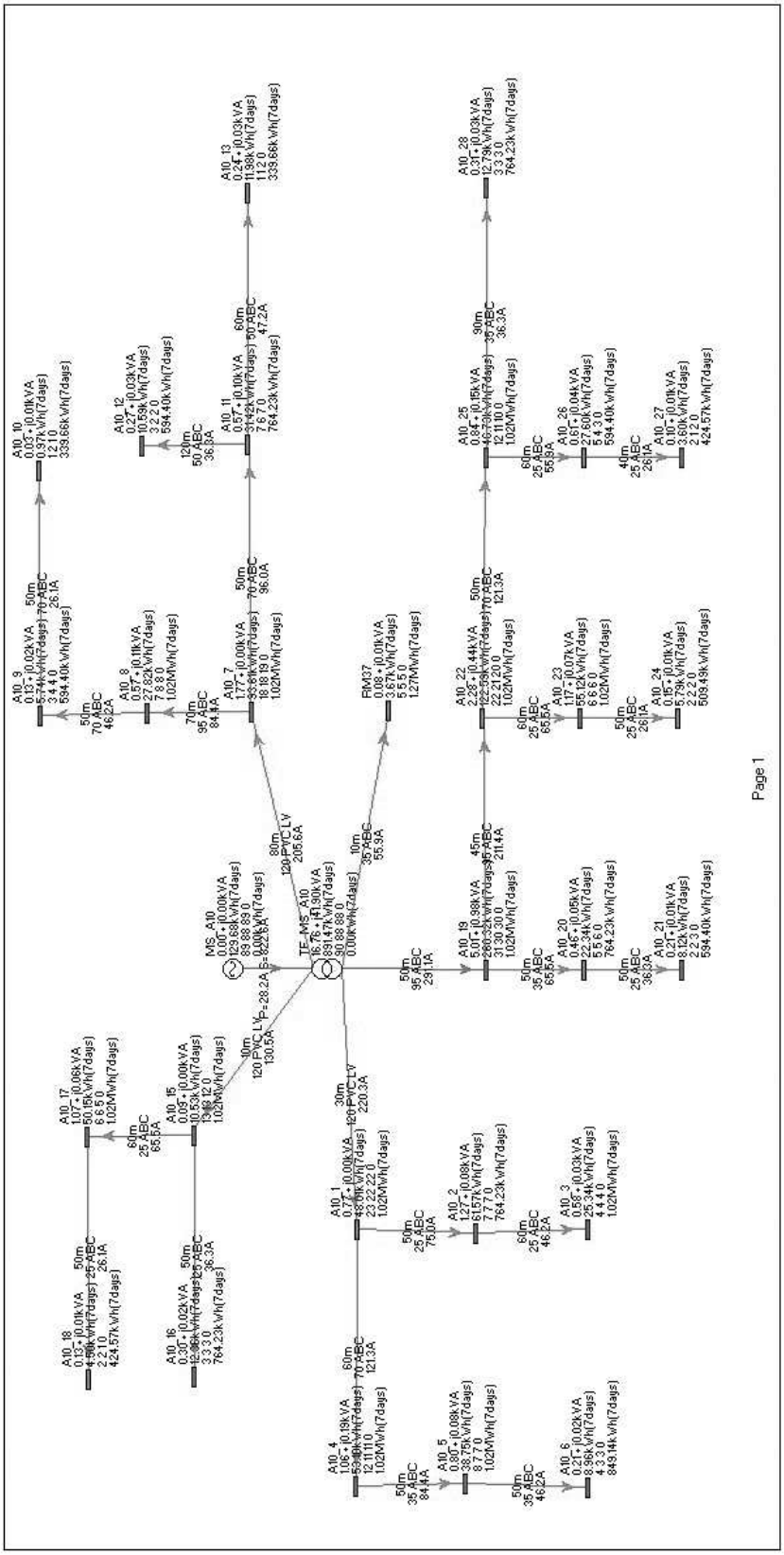


Figure K.4: *Retic-Master layout from data*

K.2 Reticmaster Simulations

K.2.1 Layouts

Two simulations were done using Retic Master. These graphical layout of the simulations are shown in Fig. K.1 and Fig. K.1.

K.2.2 Extracted Results

The following spreadsheets contains formatted results of the Retic Master simulations.

Standard consumer model

SORTED

From Node	Node Name	Total Load Loss Energy (kWh/7days)	Total Domestic Energy (kWh/7days)	% Total Losses (kWh)	Total Load (S,PF) (kVA)	Total Load Loss (S)	% Total Loss(VA)
-	MS_A10	3216.14	0	Err:503	813.96	0	0.00%
A10_1	A10_2	154.52	2377.6	6.50%	69.17	3.31	4.78%
A10_1	A10_4	179.23	3849.45	4.66%	107.75	3.72	3.45%
A10_11	A10_12	18.82	792.53	2.37%	26.97	0.48	1.79%
A10_11	A10_13	11.55	452.88	2.55%	27.85	0.25	0.90%
A10_15	A10_16	21.98	1018.97	2.16%	34.3	0.53	1.55%
A10_15	A10_17	97.15	1924.73	5.05%	57.77	2.12	3.68%
A10_17	A10_18	7.99	566.1	1.41%	20.6	0.22	1.07%
A10_19	A10_20	54.16	1811.51	2.99%	53.81	1.2	2.22%
A10_19	A10_22	487.48	7132.81	6.83%	182.31	9.81	5.38%
A10_2	A10_3	45.05	1358.63	3.32%	41.46	1.03	2.49%
A10_20	A10_21	14.44	792.53	1.82%	26.45	0.37	1.40%
A10_22	A10_23	108.29	2037.95	5.31%	56.83	2.35	4.14%
A10_22	A10_25	161.27	3736.23	4.32%	99.12	3.34	3.37%
A10_23	A10_24	10.3	679.32	1.52%	22.36	0.27	1.21%
A10_25	A10_26	55.46	1358.63	4.08%	39.64	1.27	3.21%
A10_25	A10_28	22.74	1018.97	2.23%	31.5	0.55	1.76%
A10_26	A10_27	6.4	566.1	1.13%	19.2	0.18	0.94%
A10_4	A10_5	84.82	2490.82	3.41%	71.3	1.81	2.54%
A10_5	A10_6	15.93	1132.19	1.41%	35.69	0.38	1.07%
A10_7	A10_11	74.68	2264.38	3.30%	77.51	1.57	2.02%
A10_7	A10_8	61.38	2604.04	2.36%	75.55	1.32	1.75%
A10_8	A10_9	11.93	1245.41	0.96%	39.6	0.28	0.72%
A10_9	A10_10	1.73	452.88	0.38%	17.38	0.05	0.29%
MS_A10	TF_MS_A10	3189.64	30116.31	10.59%	772.54	101.53	13.14%
TF_MS_A10	**A10_1**	419.09	7585.69	5.52%	205.11	8.37	4.08%
TF_MS_A10	**A10_15**	137.85	4302.33	3.20%	122.65	2.81	2.29%
TF_MS_A10	**A10_19**	801.86	10302.95	7.78%	263.88	20.06	7.60%
TF_MS_A10	**A10_7**	285.69	6227.06	4.59%	179.65	5.71	3.18%
Totals for TF_MS_A10		1644.49	28418.03	5.79%	771.29	36.96	4.79%

Consumer model based on site data

Sheet1

**********	**********	**********	**********	**********	**********	**********	**********
From Node	Node Name	Total Load Loss Energy (kWh7da	Total Domestic Energy (kWh7da	% Loss (kWh 7)	Total Loss (kVA)	Total Load (S,PF) (kVA)	
**********	**********	**********	**********			**********	
-	MS_A10	50.57kWh(7days)	0.00kWh(7days)			0	93.7
MS_A10	TF_MS_A10	50.22	4336.61	1.16%	1.37	92.09	
TF_MS_A10	A10_1	4.39	1092.3	0.40%	0.08	21.17	
TF_MS_A10	A10_7	12.61	896.67	1.41%	0.23	32.59	
TF_MS_A10	A10_15	1.47	619.52	0.24%	0.03	12.68	
TF_MS_A10	A10_19	10.38	1483.58	0.70%	0.19	27.99	
Totals		28.85	4092.07	**0.71%**	0.53	94.43	**0.57%**
Percentage Losses							
A10_1	A10_2	1.64kWh(7days)	342.36kWh(7days)			7.49	
A10_2	A10_3	0.48kWh(7days)	195.64kWh(7days)			4.64	
A10_1	A10_4	1.88kWh(7days)	554.30kWh(7days)			11.45	
A10_4	A10_5	0.90kWh(7days)	358.67kWh(7days)			7.79	
A10_5	A10_6	0.17kWh(7days)	163.03kWh(7days)			3.99	
A10_7	A10_8	0.65kWh(7days)	374.97kWh(7days)			8.1	
A10_8	A10_9	0.13kWh(7days)	179.33kWh(7days)			4.32	
A10_9	A10_10	0.02kWh(7days)	65.21kWh(7days)			1.91	
A10_7	A10_11	7.36kWh(7days)	326.06kWh(7days)			22.14	
A10_11	A10_12	0.20kWh(7days)	114.12kWh(7days)			2.98	
A10_11	A10_13	4.01kWh(7days)	65.21kWh(7days)			16.87	
A10_15	A10_16	0.24kWh(7days)	146.73kWh(7days)			3.68	
A10_15	A10_17	1.03kWh(7days)	277.15kWh(7days)			6.25	
A10_17	A10_18	0.09kWh(7days)	81.52kWh(7days)			2.28	
A10_19	A10_20	0.58kWh(7days)	260.85kWh(7days)			5.92	
A10_20	A10_21	0.16kWh(7days)	114.12kWh(7days)			2.98	
A10_19	A10_22	5.09kWh(7days)	1027.09kWh(7days)			19.9	
A10_22	A10_23	1.15kWh(7days)	293.45kWh(7days)			6.52	
A10_23	A10_24	0.11kWh(7days)	97.82kWh(7days)			2.63	
A10_22	A10_25	1.71kWh(7days)	538.00kWh(7days)			11.09	
A10_25	A10_26	0.59kWh(7days)	195.64kWh(7days)			4.62	
A10_26	A10_27	0.07kWh(7days)	81.52kWh(7days)			2.26	
A10_25	A10_28	0.25kWh(7days)	146.73kWh(7days)			3.65	
TF_MS_A10	RM37	0.07kWh(7days)	244.55kWh(7days)			5.64	

Page 1

K.3 Spreadsheets according to Fourie's Method

K.3.1 Standard 40A Consumers

Consumer constants			Load Current Parameter for Loss Calc	
Cb	40	1.7	Alpha=	1.700
Confidence	100%	4.5	Beta=	4.500
Vsupply[V]	230		E(Yi)=	2.74E-01
Period[hrs] (year)	168	1.7	E(Yi2)=	1.03E-01
Load Current		4.5	E(Yi3)=	4.64E-02
Mean (RMS) [A]	13.2765		E(Yi4)=	2.37E-02
Mean (AVE) [A]	10.97		E(Yi,Yk)=	7.52E-02
Variance (AVE)	44.2251600	50.09%	E(Yi2,Yk)=	2.82E-02
Consumption			E(Yi3,Yk)=	1.27E-02
Total ma	87		E(Yi,Yj,Yk)=	2.06E-02
Total mb	85		E(Yi2,Yj,Yk)=	7.73E-03
Total mc	85		E(Yi,Yj,Yk),Yl)	5.65E-03
Total m	257		E(Yi2,Tk2)=	1.06E-02
Max Cons.[W]	2364400	W		
Alpha	1.700			
Beta	4.500			
E(Yi)	0.27			
E(Yi2)	0.1			
E(Yi,Yk)	0.08	**From Total PDF**		
E(Ptot) (mu)	648301		From alpha,beta	From data
E(Ptot2)	420895112879	**Max Loss(Scaling)**		199561.46
E(Ptot)2	420293858561	**E[PtotLoss]**	16086.9381	16086.94
VAR(Ptot)=(STDdev)^2	601254318	**E[PtotLoss^2]**		277237749.63
STD DEV	24520	**E[PtotLoss]^2**		26985865.98
Alpha	507	**VAR**	142005.78	142005.78
Beta	1342	**Std DEV**		376.84
Consumption[kW]	771.29	**Alpha**		1675.40
Consumption[kWh]	129.58	**Beta**		19108.27
	0.01	**Total Loss**	17975.724	
%Loss (90%)	**4.91%**	**DEV_loss**	994.81%	Loss
E[%loss]	**2.48%**	**DEV_total**	48.85796%	

Section:	1	1	1	1	1	1	2	2	2	2	2	2	2
Subsection	A10/6	A10/5	A10/4	A10/3	A10/2	A10/1	A10/13	A10/12	A10/11	A10/10	A10/9	A10/8	A10/7
Type:	35	35	70	25	25	120	50	50	70	70	70	95	120
Res/Km@t2	0.58	0.58	0.3	0.81	0.81	0.19	0.41	0.41	0.3	0.3	0.3	0.23	0.19
Section length[m]	70	50	60	60	50	30	60	20	50	50	50	70	80
Rphase[ohm]	0.0405	0.0289	0.0178	0.0484	0.0404	0.0056	0.0246	0.0082	0.0149	0.0149	0.0149	0.0158	0.0148
Cumulative ma	4	8	12	4	7	23	4	3	10	1	3	7	21
Cumulative mb	3	7	11	4	7	22	4	2	9	2	4	8	21
Cumulative mc	3	7	11	4	7	22	4	2	9	1	4	8	21
Max Loss[Wh]	*2265.76*	*7537.12*	*11034.14*	*3718.66*	*9490.32*	*13302.24*	*1889.28*	*236.16*	*6248.88*	*166.32*	*997.92*	*4505.54*	*31328.64*
E(Yp)a=	1.10E+00	2.19E+00	3.29E+00	1.10E+00	1.92E+00	6.31E+00	1.10E+00	8.23E-01	2.74E+00	2.74E-01	8.23E-01	1.92E+00	5.76E+00
E(Yp)b=	8.23E-01	1.92E+00	3.02E+00	1.10E+00	1.92E+00	6.03E+00	1.10E+00	5.48E-01	2.47E+00	5.48E-01	1.10E+00	2.19E+00	5.76E+00
E(Yp)c=	8.23E-01	1.92E+00	3.02E+00	1.10E+00	1.92E+00	6.03E+00	1.10E+00	5.48E-01	2.47E+00	2.74E-01	1.10E+00	2.19E+00	5.76E+00
E(Yp2)a=	1.31E+00	5.03E+00	1.12E+01	1.31E+00	3.88E+00	4.04E+01	1.31E+00	7.60E-01	7.79E+00	1.03E-01	7.60E-01	3.88E+00	3.37E+01
E(Yp2)b=	7.60E-01	3.88E+00	9.40E+00	1.31E+00	3.88E+00	3.70E+01	1.31E+00	3.56E-01	6.34E+00	3.56E-01	1.31E+00	5.03E+00	3.37E+01
E(Yp2)c=	7.60E-01	3.88E+00	9.40E+00	1.31E+00	3.88E+00	3.70E+01	1.31E+00	3.56E-01	6.34E+00	1.03E-01	1.31E+00	5.03E+00	3.37E+01
E(Yp3)a=	1.70E+00	1.20E+01	3.89E+01	1.70E+00	8.21E+00	2.63E+02	1.70E+00	7.70E-01	2.29E+01	4.64E-02	7.70E-01	8.21E+00	2.01E+02
E(Yp3)b=	7.70E-01	8.21E+00	3.02E+01	1.70E+00	8.21E+00	2.31E+02	1.70E+00	2.62E-01	1.69E+01	2.62E-01	1.70E+00	1.20E+01	2.01E+02
E(Yp3)c=	7.70E-01	8.21E+00	3.02E+01	1.70E+00	8.21E+00	2.31E+02	1.70E+00	2.62E-01	1.69E+01	4.64E-02	1.70E+00	1.20E+01	2.01E+02
E(Yp4)a=	2.33E+00	2.99E+01	1.40E+02	2.33E+00	1.81E+01	1.74E+03	2.33E+00	8.45E-01	6.96E+01	2.37E-02	8.45E-01	1.81E+01	1.22E+03
E(Yp4)b=	8.45E-01	1.81E+01	1.00E+02	2.33E+00	1.81E+01	1.46E+03	2.33E+00	2.13E-01	4.66E+01	2.13E-01	2.33E+00	2.99E+01	1.22E+03
E(Yp4)c=	8.45E-01	1.81E+01	1.00E+02	2.33E+00	1.81E+01	1.46E+03	2.33E+00	2.13E-01	4.66E+01	2.37E-02	2.33E+00	2.99E+01	1.22E+03
E(Ploss)=	*206.13*	*622.89*	*883.15*	*330.97*	*788.45*	*1032.97*	*168.15*	*22.83*	*506.58*	*17.76*	*89.47*	*370.92*	*2437.81*
E(Ploss2)=	49227.24	414718.29	814307.59	123008.84	665583.41	1090647.58	31750.97	643.30	270407.06	450.97	9137.86	146615.66	6082445.11
E(Ploss)2=	42489.70	387990.59	779961.12	109539.43	621645.72	1067033.15	28274.25	521.30	256621.25	315.35	8005.51	137579.13	5942898.46
VAR(Ploss)=	6737.54	26727.70	34346.47	13469.42	43937.69	23614.42	3476.72	122.00	13785.81	135.61	1132.35	9036.52	139546.65
STD DEV =	*82.08*	*163.49*	*185.33*	*116.06*	*209.61*	*153.67*	*58.96*	*11.05*	*117.41*	*11.65*	*33.65*	*95.06*	*373.56*
Alpha(section)	5.64	13.23	20.81	7.32	12.89	41.60	7.32	3.76	17.02	1.97	6.35	13.89	39.20
Beta(section)	56	147	239	75	142	494	75	35	193	16	64	155	465
Loss[W]	794.853	1692.340	2039.216	1139.934	2163.358	1932.929	579.149	104.851	1254.316	105.884	327.920	989.510	4636.072
Total Power Loss[kW]	37.880												
Total Energy Loss[kWh]	6						Incl. Church of 15kVA						

3	3	3	3	4	4	4	4	4	4	4	4	4	4
A10/18	A10/17	A10/16	A10/15	A10/28	A10/27	A10/26	A10/25	A10/24	A10/23	A10/22	A10/21	A10/20	A10/19
25	25	25	120	35	25	25	70	25	25	95	25	35	95
0.81	0.81	0.81	0.19	0.58	0.81	0.81	0.3	0.81	0.81	0.23	0.81	0.58	0.23
50	60	50	10	90	40	60	50	50	60	45	50	50	50
0.0404	0.0484	0.0404	0.0019	0.0520	0.0323	0.0484	0.0149	0.0404	0.0484	0.0102	0.0404	0.0289	0.0113
2	6	3	13	3	1	4	11	2	6	21	2	5	30
2	6	3	13	3	2	3	10	2	6	20	2	5	29
1	5	3	12	3	2	3	10	2	6	20	3	6	30
645.6	*7592.26*	*1743.12*	*1429.68*	*2247.26*	*516.48*	*2711.52*	*7650.72*	*774.72*	*8366.98*	*2029.82*	*1162.08*	*4022.88*	*47767.36*
5.48E-01	1.65E+00	8.23E-01	3.56E+00	8.23E-01	2.74E-01	1.10E+00	3.02E+00	5.48E-01	1.65E+00	5.76E+00	5.48E-01	1.37E+00	8.23E+00
5.48E-01	1.65E+00	8.23E-01	3.56E+00	8.23E-01	5.48E-01	8.23E-01	2.74E+00	5.48E-01	1.65E+00	5.48E+00	5.48E-01	1.37E+00	7.95E+00
2.74E-01	1.37E+00	8.23E-01	3.29E+00	8.23E-01	5.48E-01	8.23E-01	2.74E+00	5.48E-01	1.65E+00	5.48E+00	8.23E-01	1.65E+00	8.23E+00
3.56E-01	2.87E+00	7.60E-01	1.31E+01	7.60E-01	1.03E-01	1.31E+00	9.40E+00	3.56E-01	2.87E+00	3.37E+01	3.56E-01	2.02E+00	6.85E+01
3.56E-01	2.87E+00	7.60E-01	1.31E+01	7.60E-01	3.56E-01	7.60E-01	7.79E+00	3.56E-01	2.87E+00	3.06E+01	3.56E-01	2.02E+00	6.40E+01
1.03E-01	2.02E+00	7.60E-01	1.12E+01	7.60E-01	3.56E-01	7.60E-01	7.79E+00	3.56E-01	2.87E+00	3.06E+01	7.60E-01	2.87E+00	6.85E+01
2.62E-01	5.29E+00	7.70E-01	4.92E+01	7.70E-01	4.64E-02	1.70E+00	3.02E+01	2.62E-01	5.29E+00	2.01E+02	2.62E-01	3.16E+00	5.77E+02
2.62E-01	5.29E+00	7.70E-01	4.92E+01	7.70E-01	2.62E-01	7.70E-01	2.29E+01	2.62E-01	5.29E+00	1.74E+02	2.62E-01	3.16E+00	5.22E+02
4.64E-02	3.16E+00	7.70E-01	3.89E+01	7.70E-01	2.62E-01	7.70E-01	2.29E+01	2.62E-01	5.29E+00	1.74E+02	7.70E-01	5.29E+00	5.77E+02
2.13E-01	1.02E+01	8.45E-01	1.90E+02	8.45E-01	2.37E-02	2.33E+00	1.00E+02	2.13E-01	1.02E+01	1.22E+03	2.13E-01	5.23E+00	4.92E+03
2.13E-01	1.02E+01	8.45E-01	1.90E+02	8.45E-01	2.13E-01	8.45E-01	6.96E+01	2.13E-01	1.02E+01	1.01E+03	2.13E-01	5.23E+00	4.31E+03
2.37E-02	5.23E+00	8.45E-01	1.40E+02	8.45E-01	2.13E-01	8.45E-01	6.96E+01	2.13E-01	1.02E+01	1.01E+03	8.45E-01	1.02E+01	4.92E+03
66.38	*643.60*	*163.17*	*113.70*	*210.36*	*53.11*	*246.68*	*615.91*	*79.66*	*706.13*	*1574.28*	*112.35*	*343.35*	*3680.18*
5801.67	451418.28	30949.02	13436.58	51439.94	3713.07	70502.30	397705.65	7849.92	539715.23	2538647.36	15576.52	129208.71	13768841.52
4406.58	414226.62	26624.83	12928.50	44252.76	2820.21	60852.93	379346.26	6345.47	498622.11	2478349.19	12622.46	117886.26	13543706.84
1395.09	37191.66	4324.19	508.08	7187.18	892.86	9649.37	18359.39	1504.45	41093.12	60298.17	2954.05	11322.45	225134.68
37.35	*192.85*	*65.76*	*22.54*	*84.78*	*29.88*	*98.23*	*135.50*	*38.79*	*202.71*	*245.56*	*54.35*	*106.41*	*474.48*
2.73	10.11	5.49	23.34	5.49	2.73	5.64	18.92	3.68	11.03	37.82	3.76	9.44	55.45
24	109	53	270	53	24	56	216	32	120	448	35	101	664
347.565	1943.045	634.310	252.715	817.764	278.052	951.230	1469.333	364.484	2058.495	3023.749	515.943	1065.842	6397.412

K.3.2 Consumer model from Data

Consumer constants			Load Current Parameter for Loss Ca	
Cb	20	2.16	Alpha=	2.165
Confidence	90%	36.79	Beta=	36.788
Vsupply[V]	230		E(Yi)=	5.56E-02
Period[hrs] (year)	720	1.7	E(Yi2)=	4.40E-03
Load Current		4.5	E(Yi3)=	4.48E-04
Mean (RMS) [A]	1.1114		E(Yi4)=	5.51E-05
Mean (AVE) [A]	0.98		E(**Yi**,Yk)=	3.09E-03
Variance (AVE)	.5254400	65.22%	E(Yi2,Yk)=	2.45E-04
Consumption			E(Yi3,Yk)=	2.49E-05
Total ma	87		E(Yi,Yj,Yk)=	1.72E-04
Total mb	85		E(Yi2,Yj,Yk)=	1.36E-05
Total mc	85		E(Yi,Yj,Yk),Yl)	9.54E-06
Total m	257		E(Yi2,Tk2)=	1.94E-05
Max Cons.[W]	1182200	W		
Alpha	1.687			
Beta	32.766			
E(Yi)	0.05			
E(Yi2)	0			
E(Yi,Yk)	0	**From Total PDF**		
E(Ptot) (mu)	57891		From alpha,beta	From data
E(Ptot2)	3358528971	**Max Loss(Scaling)**		49890.36
E(Ptot)2	3351385457	**E[PtotLoss]**	166.9370	166.94
VAR(Ptot)=(STDdev)^2	7143514	**E[PtotLoss^2]**		2976.15
STD DEV	2673	**E[PtotLoss]^2**		2879.52
Alpha	446	**VAR**	18.60	18.60
Beta	8664	**Std DEV**		4.31
Consumption[kW]	61.341	**Alpha**		1493.51
Consumption[kWh]	44165.244	**Beta**		444852.17
	3.68	**Total Loss**	172.487	
%Loss (90%) (kWh)	**0.36%**	**DEV_loss**	1966.35%	Loss
E[%loss] (VA)	**0.29%**	**DEV_total**	7.03028%	

Section:	1	1	1	1	1	1	2	2	2	2	2	2	2
Subsection	A10/6	A10/5	A10/4	A10/3	A10/2	A10/1	A10/13	A10/12	A10/11	A10/10	A10/9	A10/8	A10/7
Type:	35	35	70	25	25	120	50	50	70	70	70	95	120
Res/Km@t2	0.58	0.58	0.3	0.81	0.81	0.19	0.41	0.41	0.3	0.3	0.3	0.23	0.19
Section length[m]	70	50	60	60	50	30	60	20	50	50	50	70	80
Rphase[ohm]	0.0405	0.0289	0.0178	0.0484	0.0404	0.0056	0.0246	0.0082	0.0149	0.0149	0.0149	0.0158	0.0148
Cumulative ma	4	8	12	4	7	23	4	3	10	1	3	7	21
Cumulative mb	3	7	11	4	7	22	4	2	9	2	4	8	21
Cumulative mc	3	7	11	4	7	22	4	2	9	1	4	8	21
Max Loss[Wh]	*566.44*	*1884.28*	*2758.54*	*929.66*	*2372.58*	*3325.56*	*472.32*	*59.04*	*1562.22*	*41.58*	*249.48*	*1126.38*	*7832.16*
E(Yp)a=	2.22E-01	4.45E-01	6.67E-01	2.22E-01	3.89E-01	1.28E+00	2.22E-01	1.67E-01	5.56E-01	5.56E-02	1.67E-01	3.89E-01	1.17E+00
E(Yp)b=	1.67E-01	3.89E-01	6.11E-01	2.22E-01	3.89E-01	1.22E+00	2.22E-01	1.11E-01	5.00E-01	1.11E-01	2.22E-01	4.45E-01	1.17E+00
E(Yp)c=	1.67E-01	3.89E-01	6.11E-01	2.22E-01	3.89E-01	1.22E+00	2.22E-01	1.11E-01	5.00E-01	5.56E-02	2.22E-01	4.45E-01	1.17E+00
E(Yp2)a=	5.47E-02	2.08E-01	4.60E-01	5.47E-02	1.61E-01	1.66E+00	5.47E-02	3.17E-02	3.22E-01	4.40E-03	3.17E-02	1.61E-01	1.39E+00
E(Yp2)b=	3.17E-02	1.61E-01	3.88E-01	5.47E-02	1.61E-01	1.52E+00	5.47E-02	1.50E-02	2.62E-01	1.50E-02	5.47E-02	2.08E-01	1.39E+00
E(Yp2)c=	3.17E-02	1.61E-01	3.88E-01	5.47E-02	1.61E-01	1.52E+00	5.47E-02	1.50E-02	2.62E-01	4.40E-03	5.47E-02	2.08E-01	1.39E+00
E(Yp3)a=	1.47E-02	1.02E-01	3.29E-01	1.47E-02	7.00E-02	2.21E+00	1.47E-02	6.78E-03	1.94E-01	4.48E-04	6.78E-03	7.00E-02	1.69E+00
E(Yp3)b=	6.78E-03	7.00E-02	2.56E-01	1.47E-02	7.00E-02	1.93E+00	1.47E-02	2.36E-03	1.43E-01	2.36E-03	1.47E-02	1.02E-01	1.69E+00
E(Yp3)c=	6.78E-03	7.00E-02	2.56E-01	1.47E-02	7.00E-02	1.93E+00	1.47E-02	2.36E-03	1.43E-01	4.48E-04	1.47E-02	1.02E-01	1.69E+00
E(Yp4)a=	4.30E-03	5.27E-02	2.42E-01	4.30E-03	3.21E-02	2.97E+00	4.30E-03	1.60E-03	1.22E-01	5.51E-05	1.60E-03	3.21E-02	2.09E+00
E(Yp4)b=	1.60E-03	3.21E-02	1.74E-01	4.30E-03	3.21E-02	2.50E+00	4.30E-03	4.25E-04	8.18E-02	4.25E-04	4.30E-03	5.27E-02	2.09E+00
E(Yp4)c=	1.60E-03	3.21E-02	1.74E-01	4.30E-03	3.21E-02	2.50E+00	4.30E-03	4.25E-04	8.18E-02	5.51E-05	4.30E-03	5.27E-02	2.09E+00
E(Ploss)=	*2.17*	*6.49*	*9.16*	*3.48*	*8.22*	*10.66*	*1.77*	*0.24*	*5.26*	*0.19*	*0.94*	*3.86*	*25.17*
E(Ploss2)=	5.72	45.69	88.31	14.09	73.51	116.63	3.64	0.08	29.51	0.06	1.05	16.12	651.04
E(Ploss)2=	4.73	42.08	83.82	12.12	67.52	113.64	3.13	0.06	27.68	0.04	0.89	14.90	633.31
VAR(Ploss)=	0.99	3.62	4.50	1.97	5.99	2.99	0.51	0.02	1.83	0.02	0.16	1.22	17.72
STD DEV =	*0.99*	*1.90*	*2.12*	*1.40*	*2.45*	*1.73*	*0.71*	*0.14*	*1.35*	*0.15*	*0.41*	*1.10*	*4.21*
Alpha(section)	4.75	11.60	18.57	6.14	11.23	37.86	6.14	3.13	15.08	1.64	5.36	12.19	35.61
Beta(section)	1234	3357	5577	1633	3232	11772	1633	759	4462	357	1414	3545	11048
Loss[W]	3.507	9.012	11.956	5.356	11.468	12.928	2.721	0.426	7.052	0.388	1.485	5.325	30.689
Total Power Loss[kW]	0.219												
Total Energy Loss[kWh]	157.904							Incl. Church of 15kVA					

3	3	3	3	4	4	4	4	4	4	4	4	4	4
A10/18	A10/17	A10/16	A10/15	A10/28	A10/27	A10/26	A10/25	A10/24	A10/23	A10/22	A10/21	A10/20	A10/19
25	25	25	120	35	25	25	70	25	25	95	25	35	95
0.81	0.81	0.81	0.19	0.58	0.81	0.81	0.3	0.81	0.81	0.23	0.81	0.58	0.23
50	60	50	10	90	40	60	50	50	60	45	50	50	50
0.0404	0.0484	0.0404	0.0019	0.0520	0.0323	0.0484	0.0149	0.0404	0.0484	0.0102	0.0404	0.0289	0.0113
2	6	3	13	3	1	4	11	2	6	21	2	5	30
2	6	3	13	3	2	3	10	2	6	20	2	5	29
1	5	3	12	3	2	3	10	2	6	20	3	6	30
161.4	*1898.06*	*435.78*	*357.42*	*561.82*	*129.12*	*677.88*	*1912.68*	*193.68*	*2091.74*	*5052.46*	*290.52*	*1005.72*	*11941.84*
1.11E-01	3.33E-01	1.67E-01	7.22E-01	1.67E-01	5.56E-02	2.22E-01	6.11E-01	1.11E-01	3.33E-01	1.17E+00	1.11E-01	2.78E-01	1.67E+00
1.11E-01	3.33E-01	1.67E-01	7.22E-01	1.67E-01	1.11E-01	1.67E-01	5.56E-01	1.11E-01	3.33E-01	1.11E+00	1.11E-01	2.78E-01	1.61E+00
5.56E-02	2.78E-01	1.67E-01	6.67E-01	1.67E-01	1.11E-01	1.67E-01	5.56E-01	1.11E-01	3.33E-01	1.11E+00	1.67E-01	3.33E-01	1.67E+00
1.50E-02	1.19E-01	3.17E-02	5.39E-01	3.17E-02	4.40E-03	5.47E-02	3.88E-01	1.50E-02	1.19E-01	1.39E+00	1.50E-02	8.38E-02	2.82E+00
1.50E-02	1.19E-01	3.17E-02	5.39E-01	3.17E-02	1.50E-02	3.17E-02	3.22E-01	1.50E-02	1.19E-01	1.26E+00	1.50E-02	8.38E-02	2.64E+00
4.40E-03	8.38E-02	3.17E-02	4.60E-01	3.17E-02	1.50E-02	3.17E-02	3.22E-01	1.50E-02	1.19E-01	1.26E+00	3.17E-02	1.19E-01	2.82E+00
2.36E-03	4.53E-02	6.78E-03	4.15E-01	6.78E-03	4.48E-04	1.47E-02	2.56E-01	2.36E-03	4.53E-02	1.69E+00	2.36E-03	2.72E-02	4.83E+00
2.36E-03	4.53E-02	6.78E-03	4.15E-01	6.78E-03	2.36E-03	6.78E-03	1.94E-01	2.36E-03	4.53E-02	1.46E+00	2.36E-03	2.72E-02	4.37E+00
4.48E-04	2.72E-02	6.78E-03	3.29E-01	6.78E-03	2.36E-03	6.78E-03	1.94E-01	2.36E-03	4.53E-02	1.46E+00	6.78E-03	4.53E-02	4.83E+00
4.25E-04	1.83E-02	1.60E-03	3.29E-01	1.60E-03	5.51E-05	4.30E-03	1.74E-01	4.25E-04	1.83E-02	2.09E+00	4.25E-04	9.47E-03	8.40E+00
4.25E-04	1.83E-02	1.60E-03	3.29E-01	1.60E-03	4.25E-04	1.60E-03	1.22E-01	4.25E-04	1.83E-02	1.73E+00	4.25E-04	9.47E-03	7.35E+00
5.51E-05	9.47E-03	1.60E-03	2.42E-01	1.60E-03	4.25E-04	1.60E-03	1.22E-01	4.25E-04	1.83E-02	1.73E+00	1.60E-03	1.83E-02	8.40E+00
0.71	*6.73*	*1.73*	*1.18*	*2.23*	*0.57*	*2.60*	*6.39*	*0.85*	*7.38*	*16.25*	*1.19*	*3.59*	*37.93*
0.73	50.40	3.65	1.45	6.06	0.47	8.19	43.25	0.98	60.09	271.86	1.88	14.48	1467.23
0.50	45.24	2.98	1.39	4.96	0.32	6.77	40.83	0.73	54.39	264.19	1.43	12.90	1438.96
0.23	5.16	0.66	0.07	1.10	0.14	1.42	2.42	0.25	5.69	7.67	0.45	1.58	28.27
0.47	*2.27*	*0.81*	*0.26*	*1.05*	*0.38*	*1.19*	*1.56*	*0.50*	*2.39*	*2.77*	*0.67*	*1.26*	*5.32*
2.23	8.74	4.49	20.91	4.49	2.23	4.75	16.82	2.89	9.52	34.35	3.13	8.14	50.74
504	2458	1128	6326	1128	504	1234	5018	653	2690	10642	759	2271	15921
1.346	9.751	2.817	1.517	3.632	1.077	4.197	8.447	1.524	10.550	19.888	2.097	5.266	44.887

Appendix L

Check Meter Schematics and Components

To simplify design and in order to keep the design as modular as possible, especially with the idea to re-use this design for other applications it was decided to design two separate printed circuit boards, e.g. analog and digital.
The schematics and printer circuit boards are presented in the following pages.

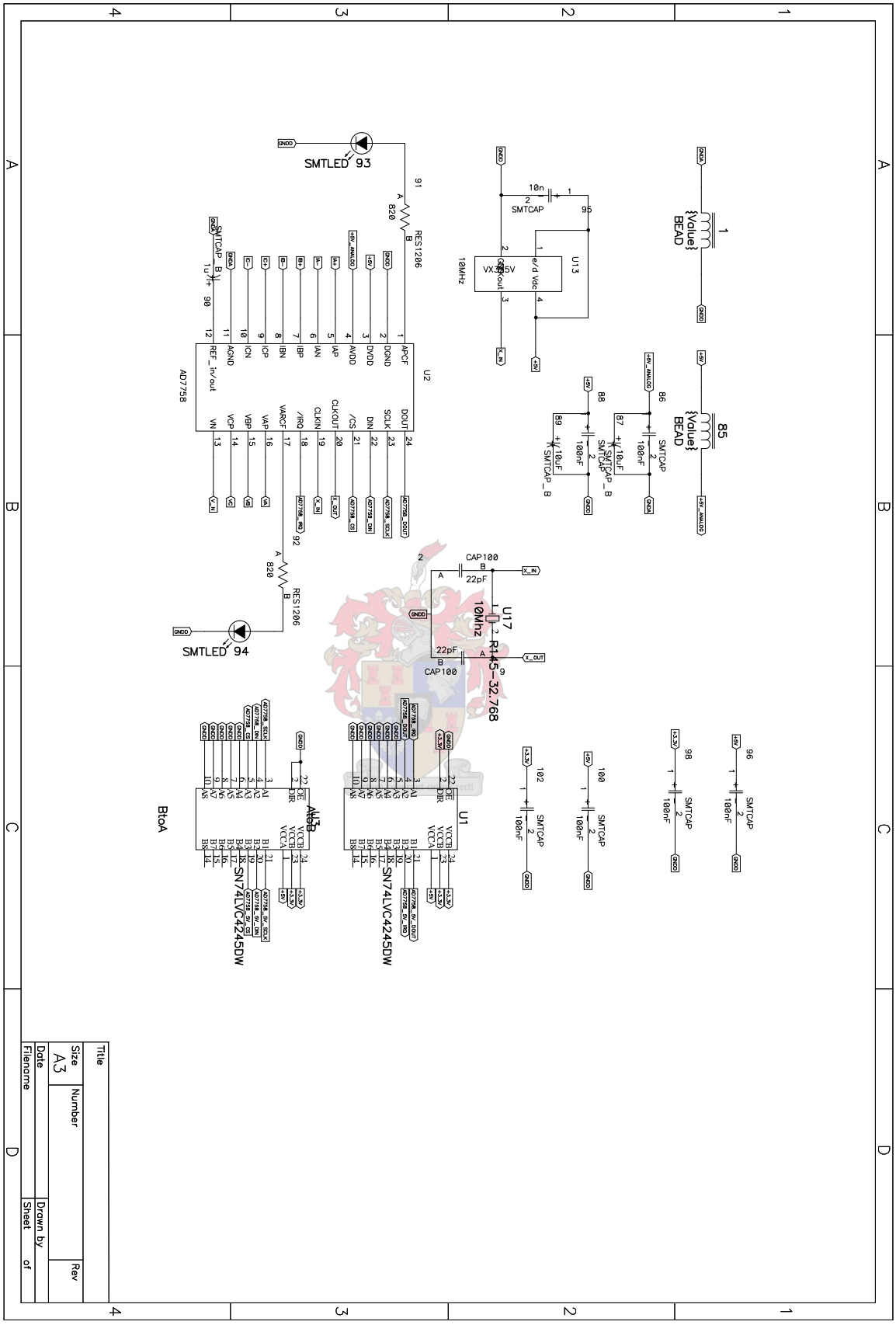


Figure L.1: *Schematic of Analog Circuitry: Energy Measurement*

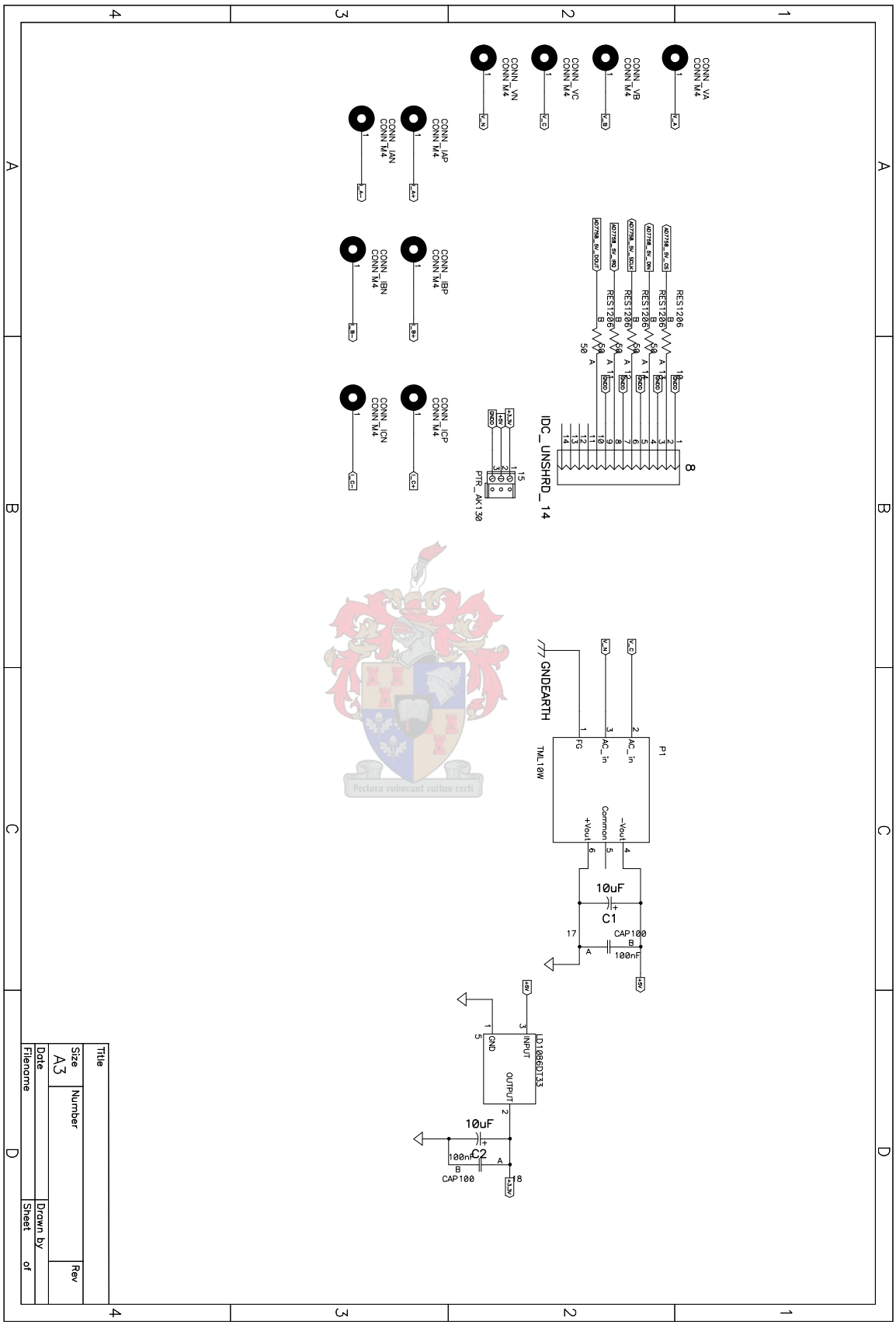


Figure L.2: *Schematic of Analog Circuitry: Power and Connectors*

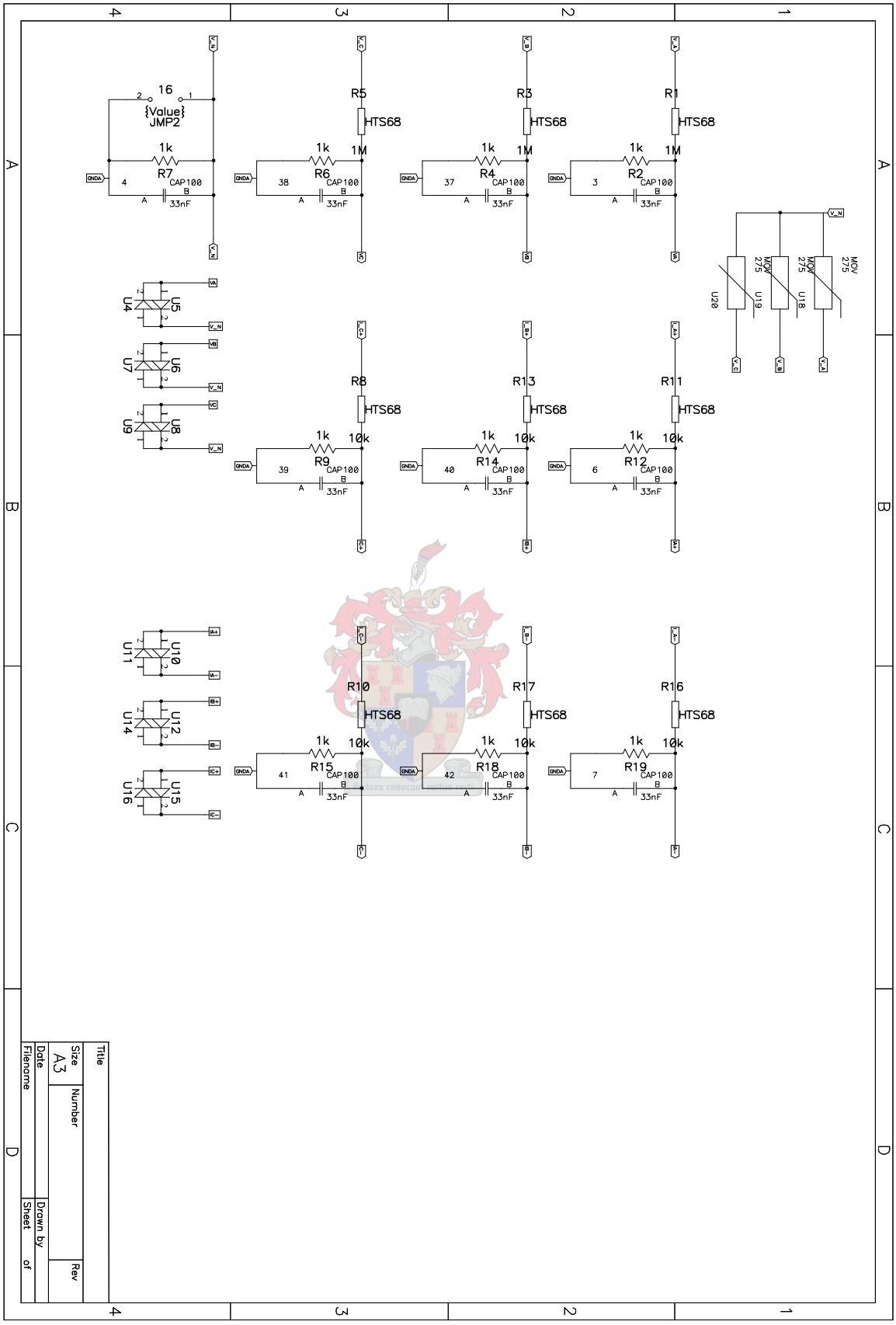


Figure L.3: *Schematic of Analog Circuitry: Signal Conditioning*

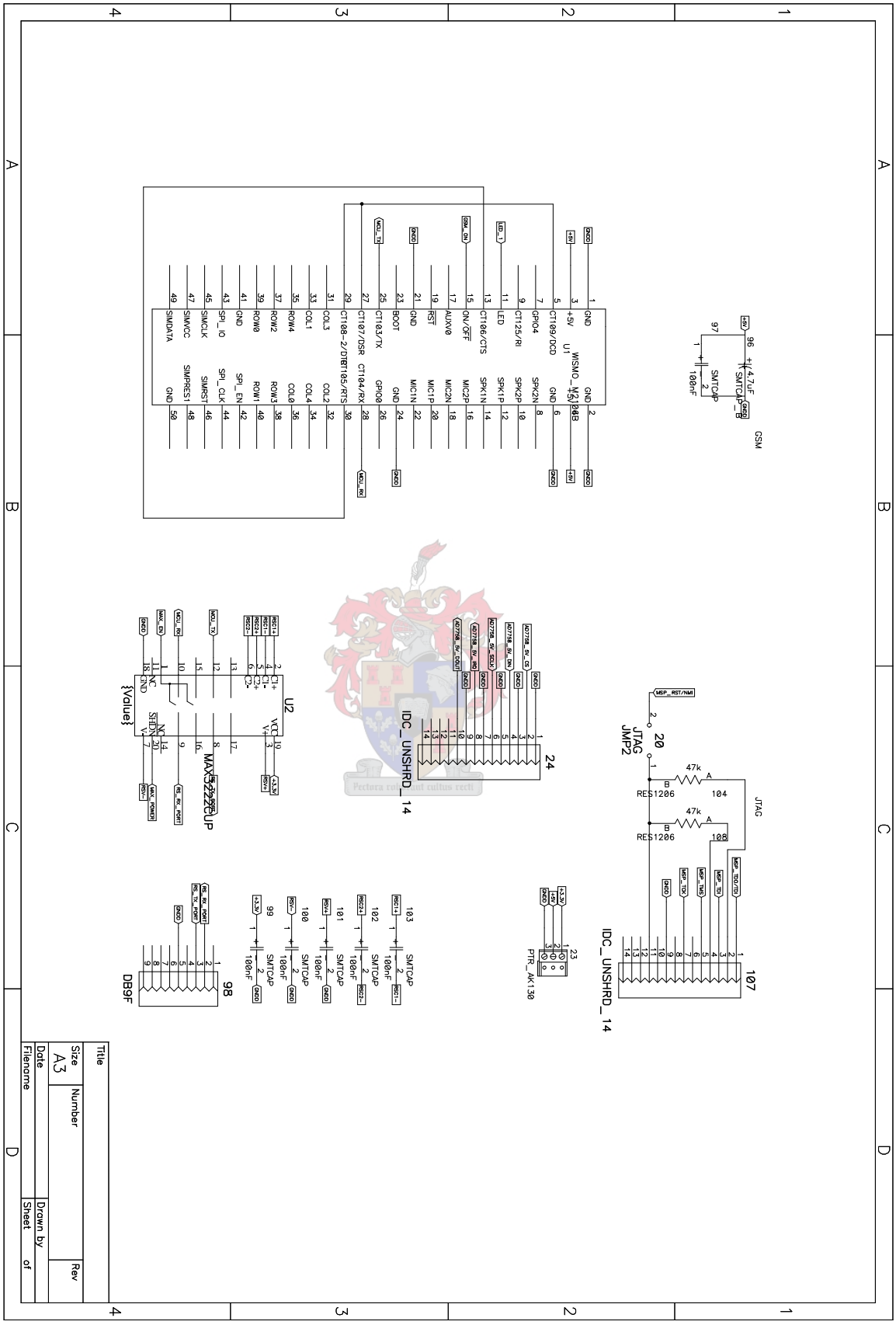


Figure L.4: *Schematic of Digital Circuitry: Communication Devices*

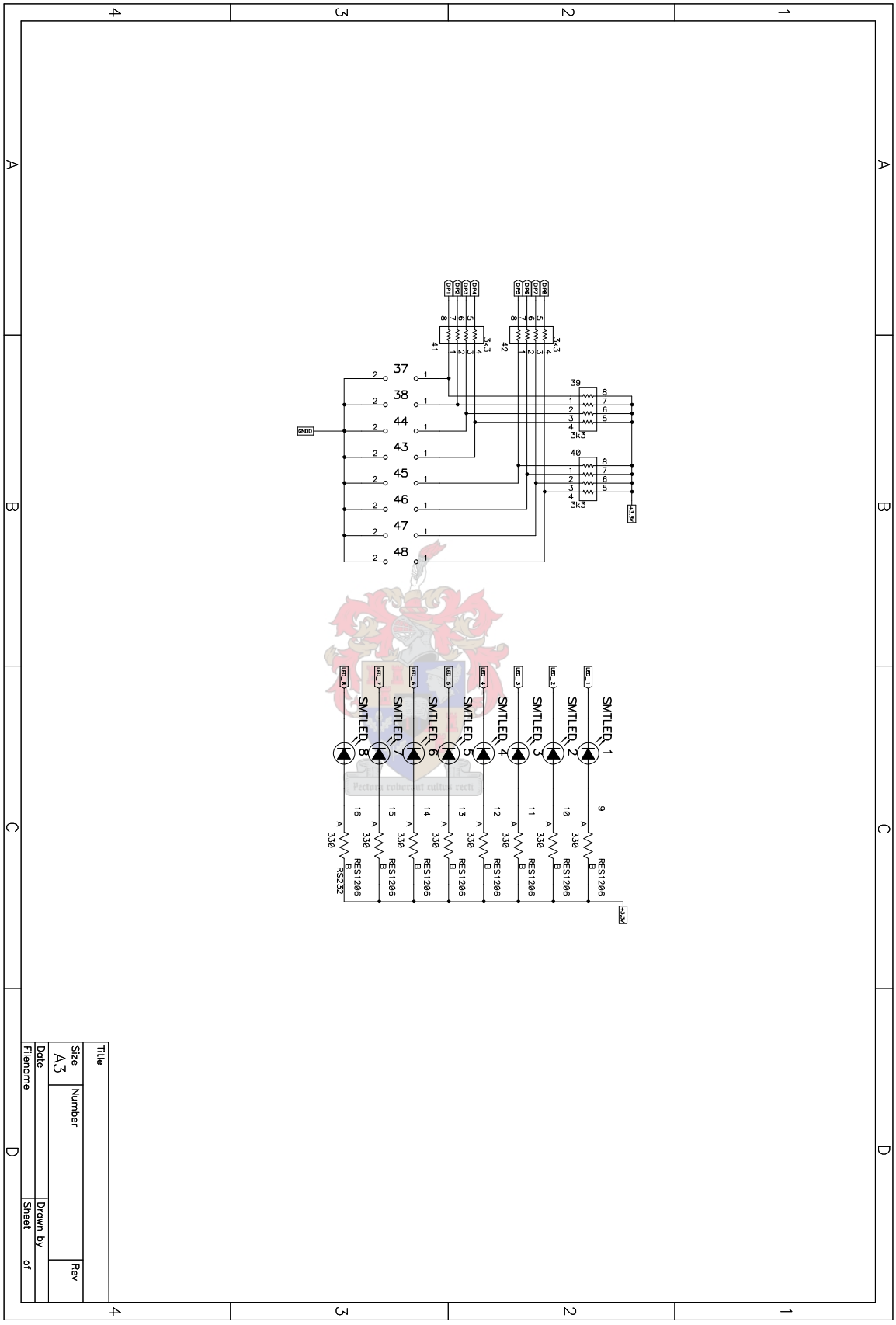


Figure L.5: *Schematic of Digital Circuitry: LED's and DIP Switches*

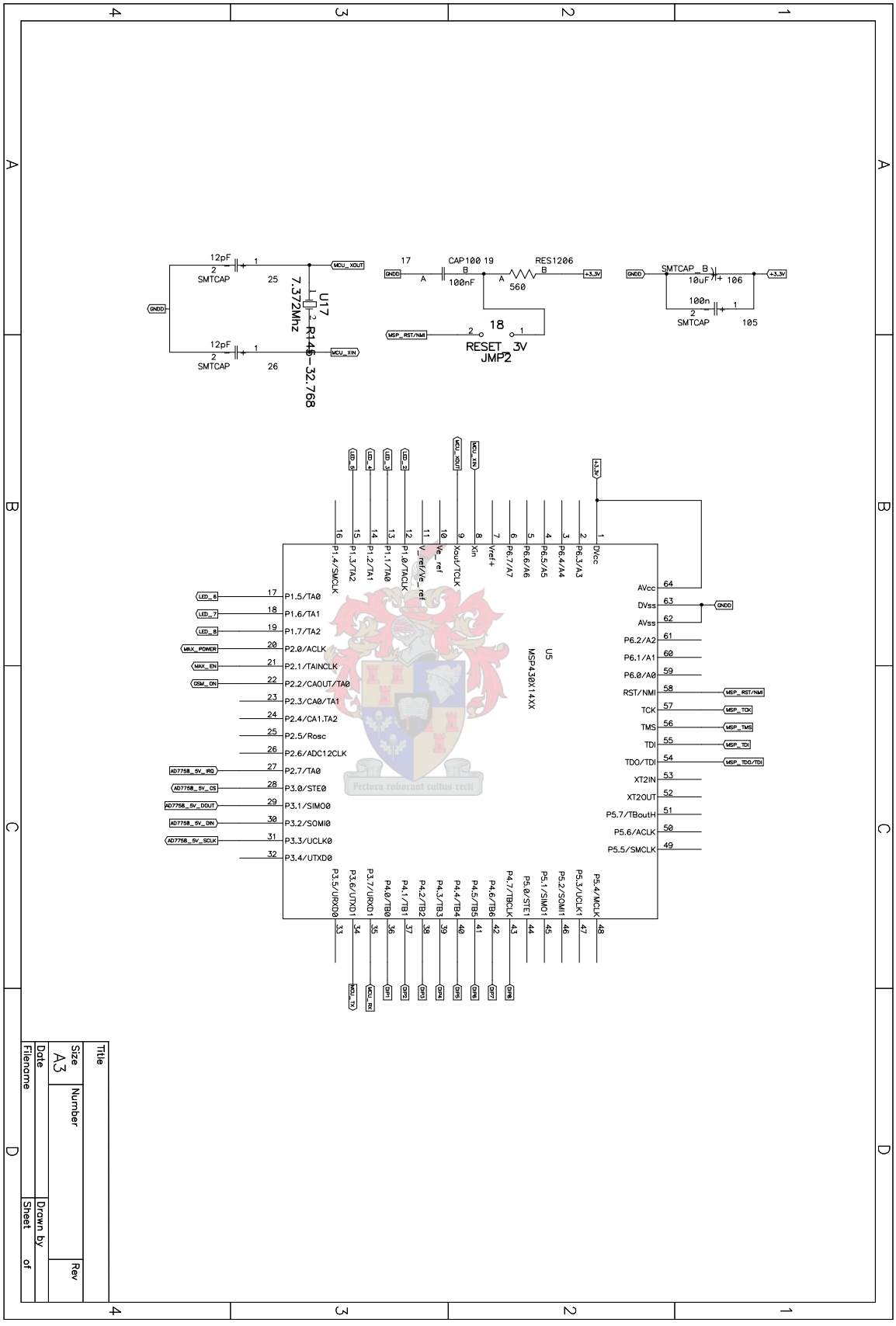


Figure L.6: *Schematic of Digital Circuitry: Micro Controller Unit*

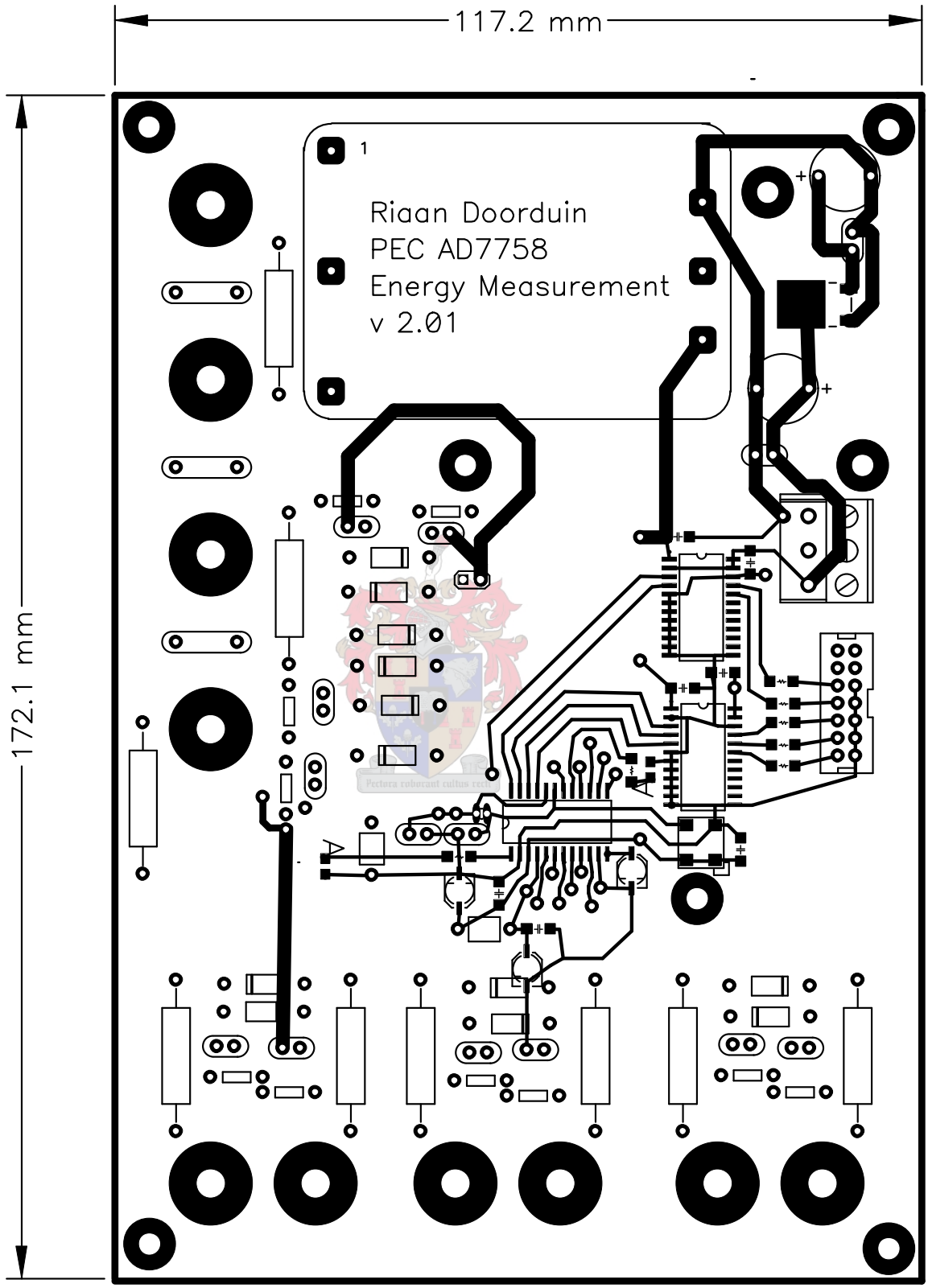


Figure L.7: *Top PCB printout of Analog Circuitry*

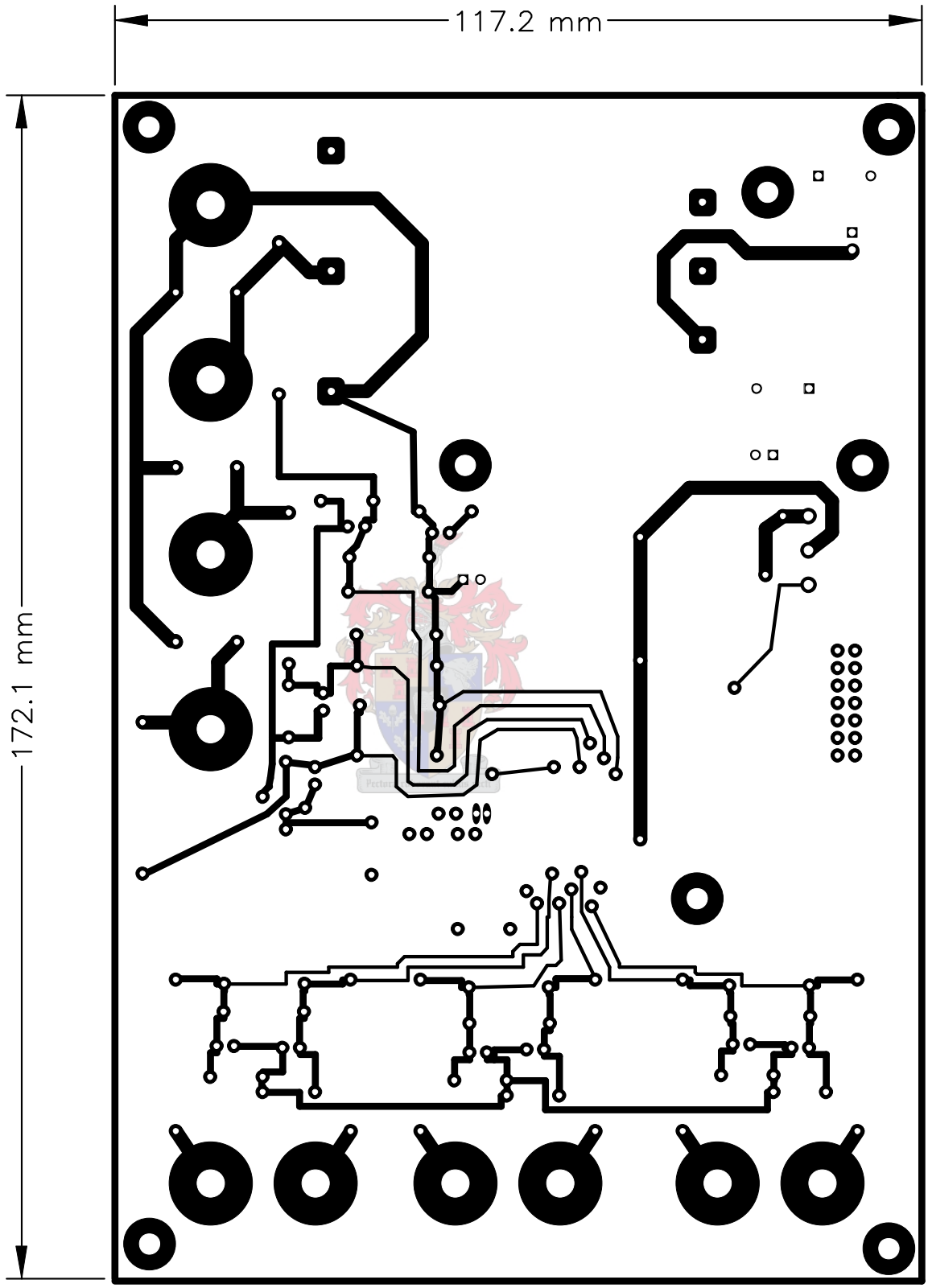


Figure L.8: *Bottom PCB printout of Analog Circuitry*

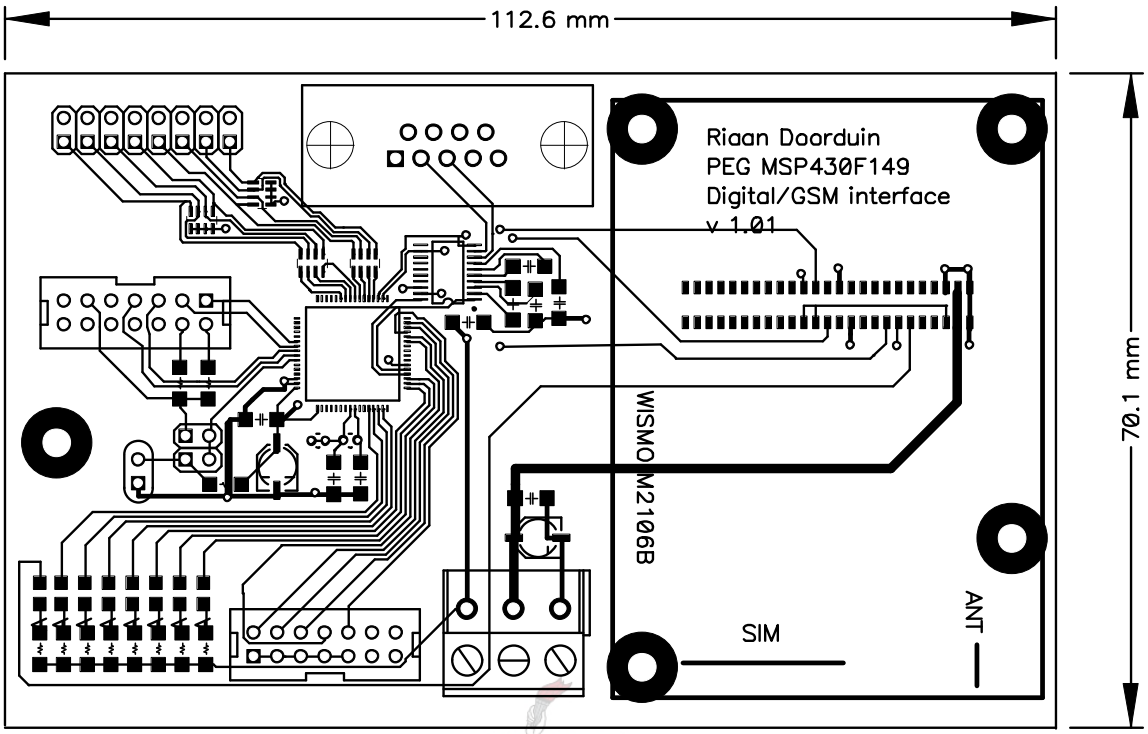


Figure L.9: *Top PCB printout of Digital Circuitry*

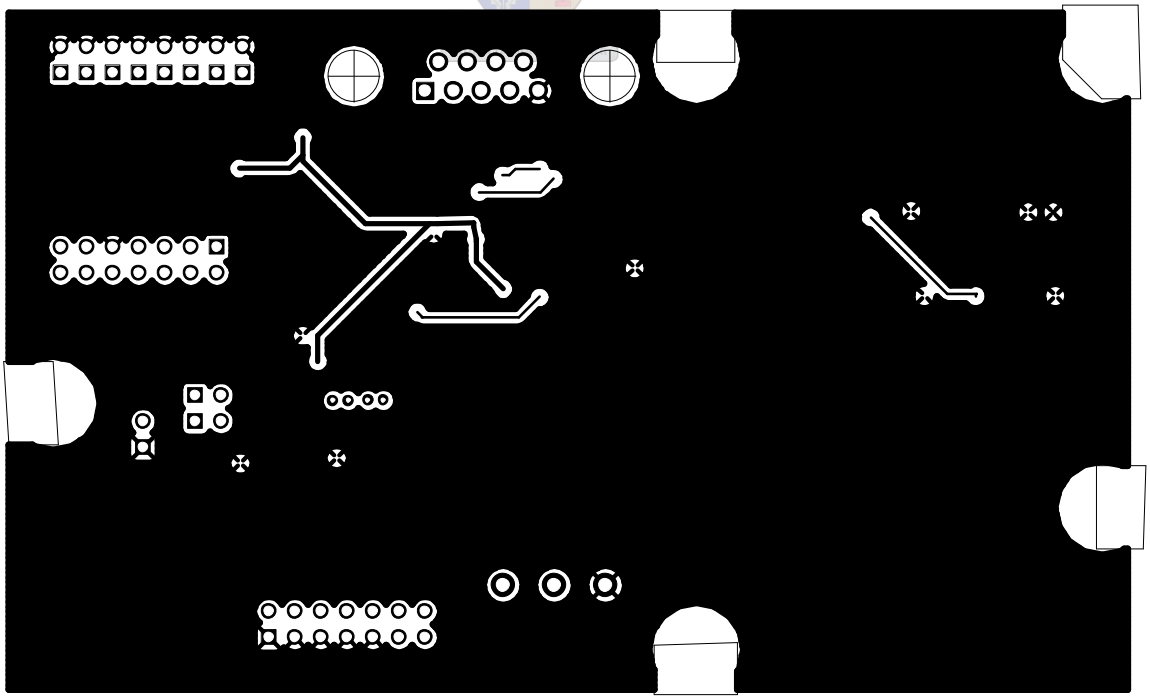


Figure L.10: *Bottom PCB printout of Digital Circuitry*

					Total System Cost:		R 9,311.33	
PCB	Ref	Description	Order Code	Manufacturer	Supplier	Quantity	Cost	Total Cost
		Box Std Slide Lid J2-S	V1ABB/YORK/J2-S	ABB	Voltex	1	R 62.59	R 62.59
		3 Phase Leads	212-988	RS	RS	1	R 512.90	R 512.90
		Spare Fuses	388-4132		RS	1	R 35.37	R 35.37
		Current Probes 1000A	SCL1000	Dent Instruments	SA elite Pro	3	R 1,608.54	R 4,825.62
		Neutral lead	206-753	RS	RS	1	R 87.73	R 87.73
	PCB1,2	Printed Circuit Boards		Northtech	Northtech	1	R 701.82	R 701.82
								R 0.00
EM	U1,3	3V to 5V buffer: SN74LVC4245DW	380-0500	Texas Instruments	RS	2	R 16.65	R 33.30
	U2	Poly Phase Energy Measurement	ADE7758ARW	Analog Devices	Avnet	1	R 65.50	R 65.50
	U4-12,14-16	Clipping Diodes	1N4007		Communica	12	R 0.09	R 1.08
	U17	10MHz Crystal	CRYSTAL			1	R 2.05	R 2.05
	U18-20	MOV 275V	238-621	BC	RS	3	R 15.68	R 47.04
	SMTLED	LEDs	SMTLED1206		Communica	2	R 0.67	R 1.34
	BEAD	EMI Leaded Filter	239-598		RS	2	R 1.49	R 2.98
								R 0.00
	CONN	Female Banana Connectors	404-200,216,250,244	RS	RS	10	R 36.51	R 365.10
	IDC14	Unshreaded IDC connector 14pin	482-121		RS	2	R 21.82	R 43.64
								R 0.00
	TML10W	Traco 10W 5V power supply	418-1917	RS	RS	1	R 498.02	R 498.02
	PTR_AK130	Power Connector Strip (3)	PTR_AK130		Communica	1	R 3.18	R 3.18
	PTR_STL130/24	Power Connector (3)	PTR_STL130/24		Communica	1	R 8.15	R 8.15
	LD1086DT33	3.3V voltage regulator	355-4768	ST	RS	1	R 22.22	R 22.22
								R 0.00
	86,88,96,98,100,102	SMT Cap 100nF	SMTCAP 100nF		Communica	6	R 1.24	R 7.44
	87,89	SMT Cap 10 uF	SMTCAP_B 10uF		Communica	2	R 3.07	R 6.14
	90	SMT Cap 1 uF	SMTCAP_B 1uF		Communica	1	R 3.50	R 3.50
	2,9	Cap 22pf	CAP 22 pF		Communica	2	R 0.40	R 0.80
	C2	Cap 10uF	CAP 10 uF		Communica	1	R 0.40	R 0.40
	18	Cap 100nF	CAP 100nF		Communica	1	R 0.40	R 0.40
	3,4,6,7,37-42	Cap 33nF	CAP 33nF		Communica	10	R 0.40	R 4.00
								R 0.00
								R 0.00
	91,92	RES1206, 820 ohm	RES1206 820ohm			2	R 0.30	R 0.60
	10-14	RES1206, 50 ohm	RES1206 50ohm			5	R 0.30	R 1.50
	R2,R4,R6,R7,R9,R12,R14,R15,R18,R19	1k ohm High Precision Resistor	165-769	Vishay	RS	10	R 13.38	R 133.80
	R1,R3,R5	1M ohm High Voltage Resistor	296-0544	Vishay	RS	3	R 76.34	R 229.02
	R8,R13,R11,R10,R17,R16	10k ohm High precision Resistor	166-728	Vishay	RS	6	R 13.63	R 81.78
								R 0.00
MCU	U1	GSM Modem	WISMO_M2106B	Wavecom	Trinity	1	R 1,140.00	R 1,140.00
	U2	MAX322CUP RS232 Driver		Texas Instruments	RS	1	R 44.00	R 44.00
	U3							R 0.00
	U4							R 0.00
	U5	MCU: MSP430x1479F	MSP430C1479F	Texas Instruments	EBV	1	R 75.22	R 75.22
	U17	7.372MHz Crystal	CRYSTAL			1	R 2.88	R 2.88
	39-41	3k3 Quad Resistor Package	241-9349		RS	2	R 0.94	R 1.88
	EM16,18,20,37,38,43-48	SIL Header, 40Way	203SSX1		Campus Elect	1	R 1.08	R 1.08
	SMTLED	LEDs	SMTLED1206		Communica	8	R 0.67	R 5.36
	U1_Conn	Connector for GSM Modem	WISMO_Connectors	Wavecom	Trinity	1	R 37.62	R 37.62
	IDC14	Unshreaded IDC connector 14pin	482-121		RS	2	R 21.82	R 43.64
	PTR_AK130	Power Connector Strip (3)	PTR_AK130		Communica	1	R 3.18	R 3.18
	PTR_STL130/24	Power Connector (3)	PTR_STL130/24		Communica	1	R 8.15	R 8.15
	DB9F	Female PCB Connector	DB9F		Communica	1	R 6.38	R 6.38
	ANT1	Wismo Antenna	WISMO_Antenna	Wavecom	Trinity	1	R 135.66	R 135.66
								R 0.00
	17,97,99-103,105	SMT Cap 100nF	SMTCAP 100nF		Communica	8	R 1.24	R 9.92
	25,26	SMT Cap 12pF	SMTCAP 12pF		Communica	2	R 1.24	R 2.48
	96	SMT Cap 4.7 uF	SMTCAP_B 10uF		Communica	1	R 3.07	R 3.07
	104,108	RES1206, 47k ohm	RES1206 47k ohm			2	R 0.35	R 0.70
	9-16	RES1206, 330 ohm	RES1206 330ohm			8	R 0.10	R 0.80
	9-16	RES1206, 560 ohm	RES1206 560ohm			1	R 0.30	R 0.30

Figure L.11: *Component List*

Appendix M

palmHHT Installation Notes and Users Guide

M.1 Package Contents

The Palm Meter Reader suite comprises the following :

- Setup File: palmHHT Setup.exe
- User Documentation: This file you are reading now.

Please take care when using the different programs bundled together and adhere to the various license agreements. **Please note that the current software and instalation procedure is still under testing.**

M.2 Pre-requisites

The system was only tested on Windows XP, although it should work on any machine able to run the Java 2 SE and Palm Desktop software.

M.2.1 Required Hardware

PC A PC workstation, with a Windows XP.

Palm A Palm with 4MB or more memory and with Palm OS 4 or greater.

M.2.2 Required Software

The following software needs to be installed on the PC prior to installation of the Palm and HotSync Software[1].

Palm Desktop The Palm desktop is supplied on the CD with the purchased palm.

Java The Java Runtime Environment must be installed, J2SE v 1.4 or later[83].

The Palm software must be installed with HotSync and Palm Desktop fully functional. A user must also be created for the purpose of synchronization. If required the network synchronization can be implemented. Please refer to the Palm HotSync documentation[64]. For the total solution to work metMeter must be installed and fully functional.

M.3 Installation

The following steps must now be executed:

1. Start the instalation program palmHHT Setup.exe.
2. Follow the instructions.
3. When the WinZip Program opens, do not “UnZip”, but use the “Run WinZip” option.
4. Click on “extract”.
5. Select “yes to all” for all the popups.
6. After install reboot if asked.
7. With the PC rebooted and ready for user interaction[2]. Select Click on the HotSync icon in the system tray.
8. Select “Custom...”.
9. Select “Palm Power Meter Reader”. (Fig. M.1)
10. Click on “Change”.
11. Ensure that “Default Data Path” is the same as the installed metMeter path. (Fig. M.2)
12. Connect the palm to the PC. Initiated a HotSync operation.

[1] Palm and HotSync is trademarks of the Palm Company

[2] After the user logged in...etc.

13. The palmHHT software should now be installed. Run the application.
14. Read the “License Agreement”. (Fig. M.3)
15. The Databases should now be generated when Fig. M.4 appears.
16. Synchronize a last time to upload the metMeter database.
17. The system is now ready for use.

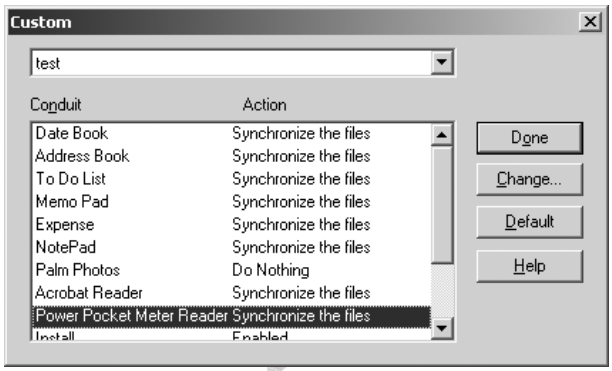


Figure M.1: *HotSync Custom Setup*

Figure M.2: *HotSync Palm Setup*

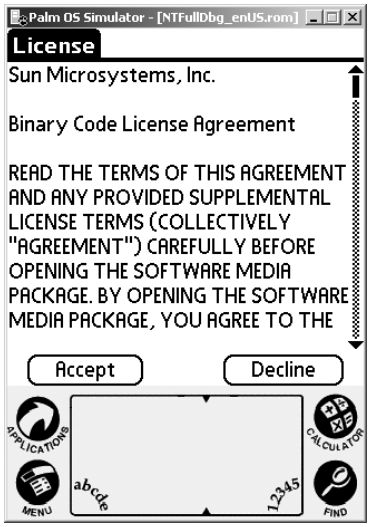


Figure M.3: *Java License Agreement for Palm MIDP*

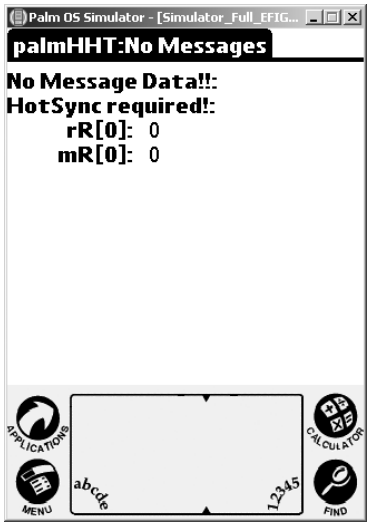


Figure M.4: *No Data uploaded for palmHHT*

Figure M.5: *First Screen of correctly setup palmHHT*

M.4 Training

For additional training or user related inquiries please refer to Appendix N.

Appendix N

Palm HHT training slides

Training was given to the Meter Readers at Theewaterskloof Municipality. The audience was a diverse group. The slides used for the training is presented in the next section.

metMeter – Palm HHT meter Reader

Training/Opleiding

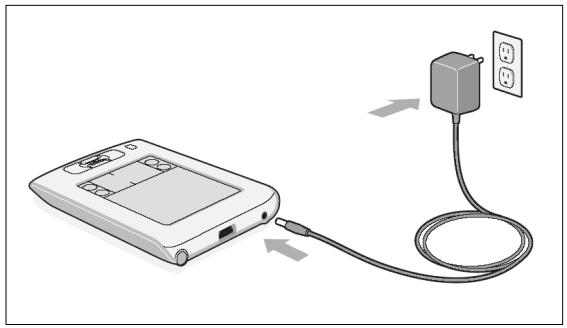
Riaan (W.A.) Doorduin
18/05/2004

Overview / Oorsig

- Palm Zire 21/71
  - Basics
  - Graffiti
- Hotsync
  - metMeter
  - palm
- palmHHT
  - 1, 2, 3...
  - Screen / Skerm
  - Datacapture / data versameling
  - Boodskappe
- ????
- Exercises / Oefeninge

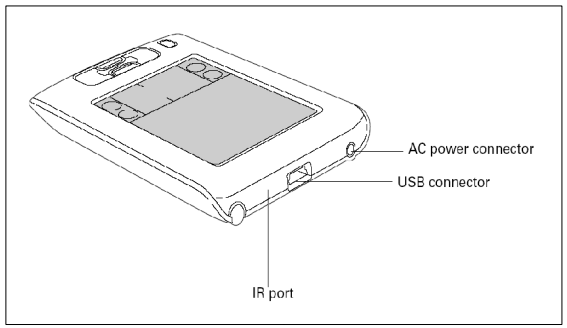
Palm - Basics

- Charge / herlaai (p.1)



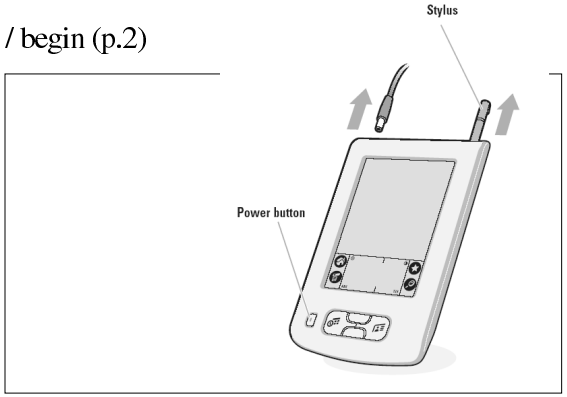
Palm - Basics

- Connectors / kabels



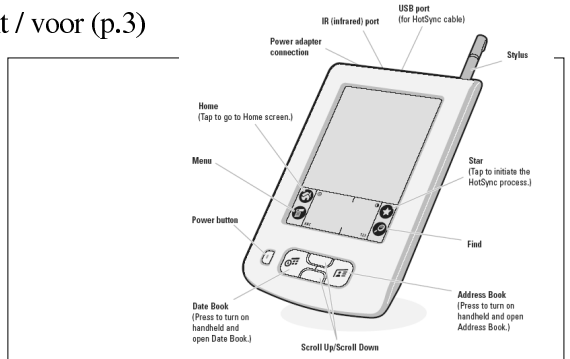
Palm – Basic

- Start / begin (p.2)



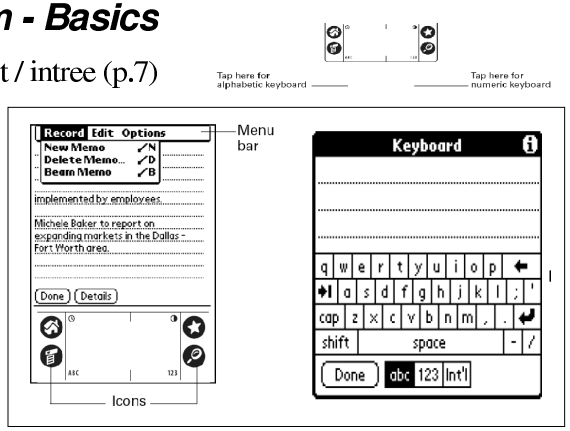
Palm - Basics

- Front / voor (p.3)



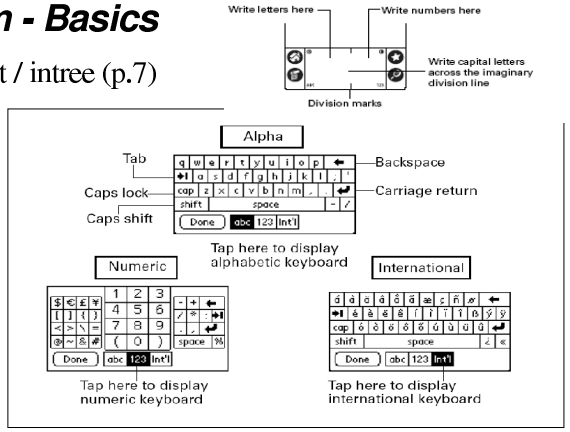
Palm - Basics

- Input / intree (p.7)

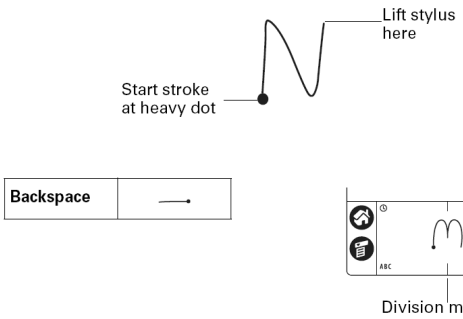


Palm - Basics

- Input / intree (p.7)



Palm - Graffiti



Palm - Graffiti

Draw letters on LEFT side of input area

Letter	Strokes	Letter	Strokes
A		B	
C		D	
E		F	
G		H	
I		J	
K		L	
M		N	

Palm - Graffiti

Draw letters on LEFT side of input area

Letter	Strokes	Letter	Strokes
O		P	
Q		R	
S		T	
U		V	
W		X	
Y		Z	

Palm - Graffiti

Draw numbers on RIGHT side of input area

Number	Strokes	Number	Strokes
0		1	
2		3	
4		5	
6		7	
8		9	

Palm - Graffiti

Draw these marks on LEFT side of input area			
Question ?		Tab	
Exclamation !			

Draw these marks on LEFT side of input area			
Mark	Stroke	Mark	Stroke
Period .		Ampersand &	
Comma ,		Carriage return	
Apostrophe '		At @	
Space		Straight quotes "	

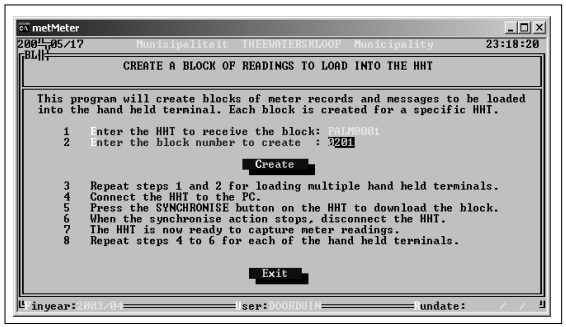
Palm - Graffiti

Draw these marks on RIGHT side of input area

Mark	Stroke	Mark	Stroke
Period .		Backslash \	
Comma ,		Slash /	
Tilde ~		Left Paren (	
Dash –		Right Paren)	
Plus +		Equal =	
Asterisk *			

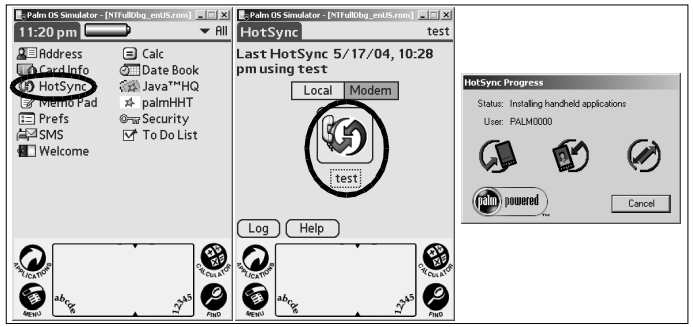
HotSync - metMeter

- Setup route / stel roete op

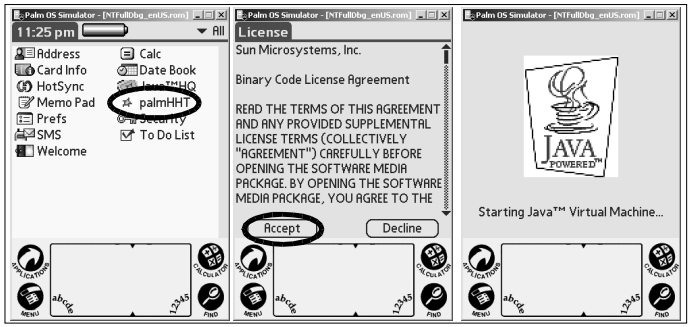


HotSync - palm

- Connect cable / koppel kabel

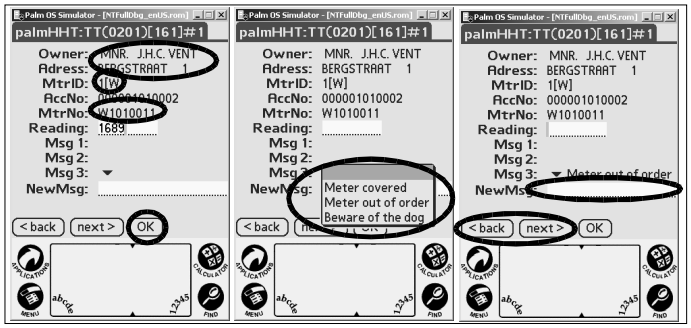


palmHHT - 1,2,3...

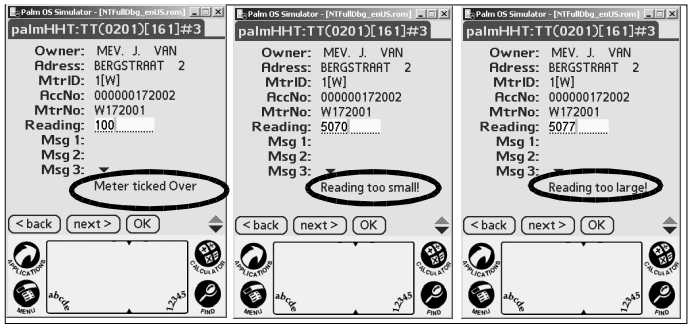


palmHHT - 1,2,3...

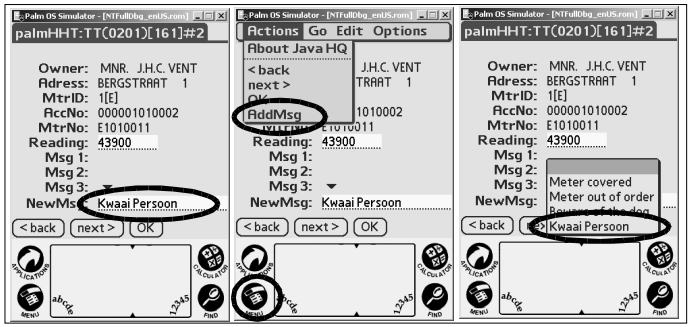
palmHHT – Screen / Skerm



palmHHT – Data capture / -versamel



palmHHT – Messages / Boodskappe



Questions / Vrae

Pause...

Exercises / Oefeninge

- Palm
  - Datum / Tyd
  - Memo Pad
- Hotsync
  - palm
- palmHHT
  - 1690
  - 43834
  - 5076
  - 223
  - “Meter onleesbaar”
  - “Beware of the dog”

The End / Einde

Thank you / Dankie

Appendix O

System Tests

O.1 Data Collection System

The tests cases and results applicable to the software can be found here:

Testcases

TESTLIST

Customer: Derick Retief
Project: Palm Handheld Meter Reader Software
Ref.: DR/HHT/P001/0001
Start: 24/06/2004
Completed: 26/06/2004

Total Tests: 49
Executed: 41
83.67%

		%	Tests
Positive	P	75.61%	31
Executed	E	2.44%	1
Negative	N	0.00%	0
Not applicable	N/A	21.95%	9

System	Subsystem	Test description	Results	Date	Complete	Results
PPST for 68k Apps 200	Palm OS	Default Setting for Palm OS Simulator	P	24/06/2004	24/06/2004	PalmOSSimulatorSetup
		Default Setting for Palm OS Emulator	P	24/06/2004	25/06/2004	Palm OS Emulator Setup
	Application Launch	Normal Launch and Exit	P	25/06/2004	25/06/2004	
		Unsported devices	E	25/06/2004	25/06/2004	Could not setup unsupported device
		Launching via Global Find	N/A			
		Launching from Memory Card	P	25/06/2004	25/06/2004	
		Handling the reset launch Code	P	25/06/2004	25/06/2004	
		Alarms	N/A			
	Alternet Sreen Config	320x480 16bit	P	25/06/2004	25/06/2004	
		240x320 8-bit	P	25/06/2004	25/06/2004	
		Additional Resolutions	P	25/06/2004	25/06/2004	
		Dynamic Input Area	N/A			
	User Interface Experie	Field Input and edit Menu	P	25/06/2004	25/06/2004	
		UI Objects	P	25/06/2004	25/06/2004	
		Modal dialog design should avoid data loss.	N/A			
		About Dialog	N/A			
	System Integration	Mask/Unmask	N/A			
		Obscuring the launcher Icon	P	25/06/2004	25/06/2004	
		Unique creator ID and database Names	P	25/06/2004	25/06/2004	Ppmr
		System Preferences	P	25/06/2004	26/06/2004	
		Application Visible in Launcher	P	26/06/2004	26/06/2004	
		Reserved Shortcuts	N/A			
		Application deletion	P	26/06/2004	26/06/2004	
		Backup Bit Set	P	26/06/2004	26/06/2004	
		Stress Test	P	26/06/2004	26/06/2004	Gremlins Test Setup
		Low Mem	N/A			Could not find MemHog
	Cobalt Compatibility	Palm OS Cobalt Simulator Compatibility	N/A			
HHTCond	GUI	Correct XML storage after Default Directory Change, e.g. Java 1.4 correctly installed.	P	26/06/2004	26/06/2004	
		When invoking Setup the correct Default Directory is displayed	P	26/06/2004	26/06/2004	
		GUI can be accessed via Conduit(HotSync) Custom Setup	P	26/06/2004	26/06/2004	
		After access via HotSync, when the GUI is closed, the HotSync process stays active	P	26/06/2004	26/06/2004	
	HotSync	Correct detection of data directory	P	26/06/2004	26/06/2004	
		Correct interpretation of PCT file	P	26/06/2004	26/06/2004	
		User correctly detected	P	26/06/2004	26/06/2004	
		Correct detection of databases	P	26/06/2004	26/06/2004	
		Upload successful	P	26/06/2004	26/06/2004	
		Download successful	P	26/06/2004	26/06/2004	
Zire 21	MIDP.prc	Java VM on Palm	P	26/06/2004	26/06/2004	
	PalmHHT.prc	Palm HHT loaded	P	26/06/2004	26/06/2004	
	Palm HHT & HotSync	1st execution correctly with no Database. Displays a zero entry into databases.				
		HotSync process now uploads data correctly				
		2nd execution shows new uploaded data.				
		HotSync process now downloads and uploads data correctly, e.g. Datafile changed				
		3rd invokation shows new database				
	PalmHHT	When a lower value than the previous value is entered for a meter reading, the error message ticked over is displayed				
		Within 10% minimum of average value added to previous reading				
		Within 10% maximum of average value added to previous reading				
		Correct display of empty and full reading	P	26/06/2004	26/06/2004	
MetMeter	Import/Export	Correct export to the Palm HHT and correct export to the metMeter system.	P	26/06/2004	26/06/2004	

Appendix P

XML Code

P.1 pInstaller

In order to package the Data Collection software, a program, pInstaller[38] used. This software requires an XML file to collect data to compile the installation package, which is supplied below.

P.1.1 xml: `settings.xml`

See accompanying PDF text for code.

P.2 Data Minor

The various tree structures used to simulate and visualize data is stored in XML files to reconstruct the environment.

P.2.1 xml: `Mooiwater_SimAllNode_ActualData.xml`

See accompanying PDF text for code.

P.2.2 xml: `CheckMeter.xml`

For this XML file there are various permutations possible. It is advised to switch the functionality of the various nodes as needed.
See accompanying PDF text for code.

P.2.3 xml: `LabTests100.xml`

See accompanying PDF text for code.

P.2.4 xml: `LabTests84.xml`

See accompanying PDF text for code.

Appendix Q

Java Code

The code are sorted according to the packages. The code were generated in Netbeans 3.5.1, and can be compiled with `javadoc` to generate documentation complient with the Java Doc specification. The original source code is shown here with the precompiled comments due to the document compiler only include public declared implementations. I found that [52] was a great help during the coding process.

Q.1 Package: `DataMinor`

Q.1.1 Object: `DataMinor.myTree`

See accompanying PDF text for code.

Q.2 Package: `energyMeasure`

Q.2.1 Object: `energyMeasure.clients`

See accompanying PDF text for code.

Q.2.2 Object: `energyMeasure.clientsInfo`

See accompanying PDF text for code.

Q.2.3 Object: `energyMeasure.energyCalibration`

See accompanying PDF text for code.

Q.2.4 Object: `energyMeasure.energyServer`

See accompanying PDF text for code.

Q.2.5 Object: `energyMeasure.energyServerGUI`

See accompanying PDF text for code.

Q.2.6 Object: `energyMeasure.energyServerSetup`

See accompanying PDF text for code.

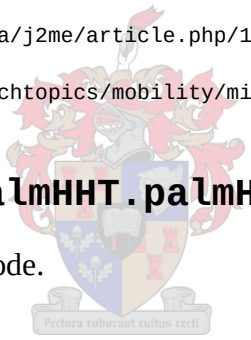
Q.2.7 Object: `energyMeasure.myEnergy`

See accompanying PDF text for code.

Q.3 Package: `green.palmHHT`

For coding the palm device the following refferences were used:

- `http://java.sun.com/j2me`
- `http://www.developer.com/java/j2me/article.php/1561591`
- `http://developers.sun.com/techtopics/mobility/midp/articles/databaserms/`



Q.3.1 Object: `green.palmHHT.palmHHT`

See accompanying PDF text for code.

Q.3.2 Object: `green.palmHHT.HeaderInfo`

See accompanying PDF text for code.

Q.3.3 Object: `green.palmHHT.MeterInfo`

See accompanying PDF text for code.

Q.3.4 Object: `green.palmHHT.MessageInfo`

See accompanying PDF text for code.

Q.3.5 Object: `green.palmHHT.RouteRecords`

See accompanying PDF text for code.

Q.3.6 Object: `green.palmHHT.MessageRecords`

See accompanying PDF text for code.

Q.3.7 Object: `green.palmHHT.RecordManager`

See accompanying PDF text for code.

Q.3.8 Object: `green.palmHHT.utils`

See accompanying PDF text for code.

Q.4 Package: `green.HHTSync`

Q.4.1 Object: `green.HHTSync.HHTCond`

See accompanying PDF text for code.

Q.4.2 Object: `green.HHTSync.HHTSyncGUI`

See accompanying PDF text for code.

Q.4.3 Object: `green.HHTSync.HHTConduitSetupData`

See accompanying PDF text for code.

Q.4.4 Object: `green.HHTSync.StringRecord`

See accompanying PDF text for code.

Q.5 Package: `peg.io`

Q.5.1 Object: `peg.io.byteBuffer`

See accompanying PDF text for code.

Q.5.2 Object: `peg.io.flatFile`

See accompanying PDF text for code.

Q.5.3 Object: `peg.io.gsmATinterface`

See accompanying PDF text for code.

Q.5.4 Object: `peg.io.protocol`

See accompanying PDF text for code.

Q.5.5 Object: `peg.io.serialInterface`

See accompanying PDF text for code.

Q.5.6 Object: `peg.io.stdInput`

See accompanying PDF text for code.

Q.6 Package: `peg.sql`

Q.6.1 Object: `peg.sql.DBConnect`

See accompanying PDF text for code.

Q.7 Package: `peg.swing`

Q.7.1 Object: `peg.swing.GraphDisplay`

See accompanying PDF text for code.

Q.7.2 Object: `peg.swing.myNode`

See accompanying PDF text for code.

Q.7.3 Object: `peg.swing.progressBar`

See accompanying PDF text for code.

Q.7.4 Object: `peg.swing.TableDisplay`

See accompanying PDF text for code.

Q.8 Package: `peg.utils`

Q.8.1 Object: `peg.utils.dataCollection`

See accompanying PDF text for code.

Q.8.2 Object: `peg.utils.dataPoint`

See accompanying PDF text for code.

Q.8.3 Object: `peg.utils.debug`

See accompanying PDF text for code.

Q.8.4 Object: `peg.utils.hystogram`

See accompanying PDF text for code.

Q.8.5 Object: `peg.utils.stringUtils`

See accompanying PDF text for code.

Q.8.6 Object: `peg.utils.TicToc`

See accompanying PDF text for code.

Appendix R

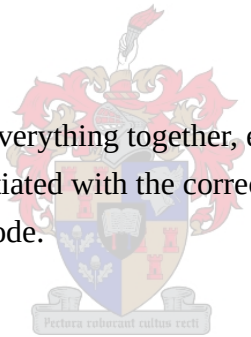
C Code

The code presented in this appendix is used to program the Check Meter as discussed in Chapter 6. The code was compiled with MSPGCC[69].

R.1 Global and Hardware setup software

R.1.1 main.c

This piece of code is what keeps everything together, ensuring all the correct drivers are running and the eternal loop is initiated with the correct program.
See accompanying PDF text for code.



R.2 MCU and System

R.2.1 memory.h

All the global variables used throughout the system is declared here.
See accompanying PDF text for code.

R.2.2 clock.h

Ensures that the clock is sets up correctly.
See accompanying PDF text for code.

R.2.3 utils.h

Handy procedures that can be used troughout the system is implemented here.
See accompanying PDF text for code.

R.3 MCU periphery

R.3.1 timer.h

See accompanying PDF text for code.

R.3.2 leds.h

See accompanying PDF text for code.

R.3.3 dipswitch.h

See accompanying PDF text for code.

R.3.4 spi.h

See accompanying PDF text for code.

R.3.5 uart.h

See accompanying PDF text for code.

R.3.6 flash.h

See accompanying PDF text for code.

R.4 Communications

R.4.1 ad7758.h

Responsible for correct communications to the ADE7758.
See accompanying PDF text for code.

R.4.2 rs232.h

Allows for communication to the callibration system.
See accompanying PDF text for code.

R.4.3 atCommand.h

Ensures communications to the GSM modem via AT command set.
See accompanying PDF text for code.

R.5 Applications

R.5.1 initMeter.h

See accompanying PDF text for code.

R.5.2 energy.h

See accompanying PDF text for code.

R.5.3 onLine.h

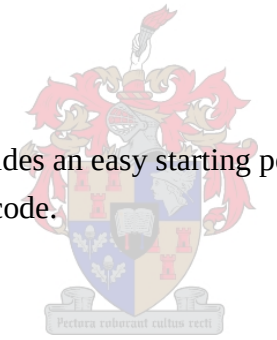
See accompanying PDF text for code.

R.5.4 gsmMode.h

See accompanying PDF text for code.

R.5.5 testStateEvent.h

Included for debugging and provides an easy starting point for testing of new routines.
See accompanying PDF text for code.



Electricity Theft Detection on a Low Voltage Reticulation Environment

Thesis presented in partial fulfillment of the requirements for the degree of
Master of Science in Electrical and Electronic Engineering
at the
University of Stellenbosch

RIAAN (W.A.) DOORDUIN

Supervisor: Prof. H. dT. Mouton

December, 2004

Declaration

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

.....
Riaan (W.A.) Doorduin
November 23, 2004

Abstract

Electricity theft in South Africa has become a major problem. This led to several developments from both industries and research institutes to counter these actions. Since equipment is already installed and major capital has been invested to provide electricity for a broad spectrum of consumers, the challenge is to find a low cost solution harnessing current investments and technology to detect electricity theft more accurately.

This thesis investigates into the electricity theft topic. Two different methods, Time Domain Pulse Reflectometry and a data driven platform based on the Theory of Constraints philosophy, were investigated to provide means to detect and determine the impact of illegal electricity usage. Both methods required detailed designs to conduct preliminary proof of concept tests in a laboratory environment.

These methods are evaluated against their economical viability, possible practical implications and applications. This thesis presents a practical approach to electricity theft detection and provides the basic tools for management of this ever-increasing problem.

Opsomming

Suid Afrika se elektrisiteit diefstal statistiek het die afgelope jare skrikwekkend gegroei. Dit het die industrie genoop om baie meer navorsing in die area te doen. Met reeds gevestigde toerusting en tegnologie om dié energie medium so effektief moontlik te versprei, is die uitdaging juis om ’n ekonomiese oplossing te vind om reeds beskikbare tegnologieë meer doeltreffend aan te wend.

Die doel van die tesis is om die gebied van elektrisiteit diefstal na te vors. Twee verskillende metodes is ondersoek, naamlik Tydgebied-pulse-reflektometrie en ’n informasie gebaseerde stelsel wat op die Randvoorwaarde Teorie gebaseer is, om effektief die omvang van elektrisiteit diefstal in ’n mikro, asook makro omgewing te bepaal. Die twee metodes is in ’n beheerde omgewing getoets sodat die konsepte wat ontwikkel is bewys kon word.

Die metodes is ge-evalueer in terme van die ekonomiese lewensvatbaarheid daarvan met inagneming van die praktiese implikasies. Die tesis bied bestuur die nodige kennis om elektrisiteit diefstal in die praktyk doeltreffend die hok mee te slaan.

Acknowledgements

Thank you to all who helped and encouraged me through my studies:

- My heavenly Father for His providence.
- My family for their encouragement.
- Prof. H. dT. Mouton for his guidance.
- Dr. H.J. Beukes for his insights and contributions.
- Dr. and Miss. D.E. Steinmann for the splendid time I had during my two year stay in Stellenbosch.
- ESKOM, the NRF and Thrip for providing funds for the project.
- Stellenbosch Municipality for providing test facilities and allowing the use of their data.
- Actaris for providing the data concerning the Mooiwater test site linked to the Stellenbosch Municipality.
- Prof. R. Herman and Mr. L. Swart for assistance with the site analysis.

Contents

List of Figures

List of Abreviations

ABC	Aerial Bundled Conductor
API	Application Program Interface
AC	Alternating Current
CT	Current Transformer
DC	Direct Current
DM	Data Mining
DSP	Digital Signal Processor
ET	Electricity Theft
GUI	Graphical User Interface
GSM	Global System for Mobile Communications
HHT	Hand Held Terminal
HV	High Voltage, e.g. $\pm 100\ kV$ to $400\ kV$
IDE	Integrated Desktop Environment
ISM	Industrial, Scientific and Measurement
LSB	Least Significant Byte
LV	Low Voltage, e.g. $\pm 230\ V_{LN}$
MCU	Micro Controller Unit
MIDP	Mobile Independent Device Profile
MOSFET	Metal-Oxide Semiconductor Field-Effect Transistor
MOV	Metal-Oxide Varistor
MV	Medium Voltage, e.g. $\pm 11\ kV$
PCB	Printed Circuit Board
PDA	Personal Digital Assistant
PDF	Probability Density Function
RMS	Root Mean Square
SCADA	Supervisory Control And Data Acquisition
SMS	Short Message Server
SPI	Serial Peripheral Interface
SQL	Structured Query Language

TDR	Time Domain Pulse Reflectometry
TOC	Theory of Constraints
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver-Transmitter
XTAL	Crystal Oscillator

List of Symbols

C_{DS}	Drain Source Capacitanse
f_{sample}	Sample Frequency
$I_{D_{max}}$	Continuous drain current
$i_{network}$	Network current
I_{ϕ}	Phase current
i_s	Source current
i_{Switch}	Switch current
$\Im(Z)$	Imaginary part of complex impedance
$L_{network}$	Network inductance
L_s	Source inductance
P_{tot}	Power dissipation
$R_{DS(on)}$	Drain source on resistance
$t_{d(off)}$	Turn-off delay time
$t_{d(on)}$	Turn-on delay time
t_f	Fall time
t_r	Rise time
$V_{DSS_{max}}$	Drain Source breakdown voltage
V_{GS}	Gate-Source voltage
V_{LL}	Line-to-Line voltage
V_{LN}	Line-to-Neutral voltage
v_s	Source voltage
Z_{Load}	Load modeled as an impedance

Chapter 1

Introduction

1.1 Background

Although various figures are available, the non-technical losses for Eskom are estimated at almost R250 million per annum [60] (2003). Compared to Eskom's annual profit of R2561 million in 2001 this can be estimated at roughly 10% of their annual profit. Electricity theft does not only concern losses in monetary terms, but poses an extremely dangerous health and safety risk [75].

Although electricity theft is mostly concentrated in urban areas and on bulk users (see Appendix J.3), Eskom was introduced to a new market with the introduction of prepaid systems and the electrification initiative in the early 1990s (see Appendix G.6). It can be deduced that more consumers had unmonitored access to electricity and hence the increase in non-technical losses.

1.2 Electricity Theft Defined

During the course of the study it was found that, in general, electricity theft is confused with cable theft (see Appendix F.3.1). This prompted the definition of electricity theft to be clearly defined before addressing it in the thesis.

The energy usage in an electricity reticulation system can consist of the energy used by consumers, technical losses and non-technical losses.

The energy consumed by the users is measured by the electricity distributor at a consumer measuring point, such as a conventional or a prepaid meter. The utility is responsible for delivering quality power up to that measuring point within strict specifications [67, 68].

The technical losses consist of transmission and transformer inefficiencies in the transmission system to the consumers in the transmission system [33].

Non-technical losses are best defined as [11]:

> ...non-technical is a euphemism encompassing poor management or human fail-

ing related reasons such as supplies that haven't been metered, meters that measure inaccurately or don't work properly, meters that are tampered with, bad or fraudulent accounting by vendors, illegal connections to the distribution system and other similar circumstances.

Electricity theft is then defined as the non-technical losses, encompassing both criminal activity and non-criminal actions and circumstances, in a reticulation system.

1.3 Scope of the Thesis

The goal of this thesis is to investigate methods of detecting electricity theft in Low Voltage (LV) networks. Only the case from the LV transformer up to and including the meter measurements will be included. The processing and back office will not be considered or discussed, for example revenue losses due to inadequate administration. The focus will be on the physical tapping of electricity and tampering with metering equipment.

From [15] only the following non-technical losses will be considered:

- Unauthorized line tapping and diversion
- Losses due to faulty meters and equipment
- Inadequate or faulty metering
- Inadequacies of meter reading
- Inaccurate estimation of non-metered supplies, for example public lighting, agricultural consumption and rail traction.

Two methods were suggested for study, namely Time Domain Pulse Reflectometry (TDR) and a Data Driven Solution based on the Theory of Constraints [36].

1.4 Methodology

To gain an understanding of the LV networks, Time Domain Pulse Reflectometry (TDR) was studied and evaluated. This approach looks into the unauthorized line tapping mentioned in the section above.

Goldratt's Theory of Constraints [36] is used as a tool to address the Check Meter and Data Mining study of the thesis, which encompasses the rest of the points mentioned in the previous section. Although designed for production processes it can be applied to electricity theft in the sense that it optimizes the detection process. Using all the available resources the problem area is identified and addressed in a cost-effective manner.

Each of the studied methods is evaluated according to the following criteria:

Capital Layout Cost The total cost of the system is evaluated.

Running Cost How expensive it will be to run the system, especially in terms of man-power.

Accuracy To what extent can the solution determine electricity theft and under what conditions are these results expected.

1.5 Outline of Thesis

The thesis consists of four different parts.

The first part, of which this chapter forms part, consists of some background and an introduction to the topic of electricity theft. Chapter 2 discusses some additional issues regarding electricity theft and provides the required statistics.

The second part, Chapter 3, discusses Time Domain Pulse Reflectometry as an electricity theft detection method.

The third part is built on the Theory of Constraints, discussed in Chapter 4, and is held together using the Data Mining package of Chapter 5. Two data capturing methods are investigated. Chapter 6 looks at the design and implementation of a Mobile Remote Check Meter and Chapter 7 developed software for deployment in a hand held device to capture meter readings. The simulations and results of the integrated data driven solution are documented in Chapter 8.

The thesis is finally summarized and concluded with Chapter 9, providing an overview of the work done and discusses possible future research opportunities.

Chapter 2

Background and Insights into Electricity Theft

> The reasons for fraud are complex and revolve undeniably around the prevalent social patterns. The question facing utilities is to determine cost justifiable strategies to overcome the losses. [70]

2.1 Introduction

Electricity theft directly affects the bottom-line of a utility and is directly associated with revenue protection (see Appendix G). The outstanding debt due to non-payment runs into billions per area, together with the millions of rands lost per year to electricity theft. This chapter is an overview of the current electricity theft situation in South Africa in terms of available statistics, the law, forms of electricity theft and provides a look at current solutions for addressing electricity theft.

2.2 Background and Statistics

Electricity theft has grown rapidly in the last ten to fifteen years since the introduction of prepaid meters (see Appendix G) and the rural electrification drive initiated in the early 1990s [25], whereby from 1994 to 1999 roughly 300 000 new connections were added to the national grid by Eskom.

Although electricity theft occurs in such small amounts relative to the total energy consumption of the utilities, the problem must be addressed firstly, to promote safety and secondly, to avoid any unreasonable prices. Combined with the non-payment levels, for example in Soweto which in 1995 had a $51\ \%$ payment level, the cost for faithful paying consumers increased, which certainly is unfair [19]. Consumers in Philippi, Cape Town, are reported to be removing these illegal connections in order to prevent this effect (see Fig. 2.1).

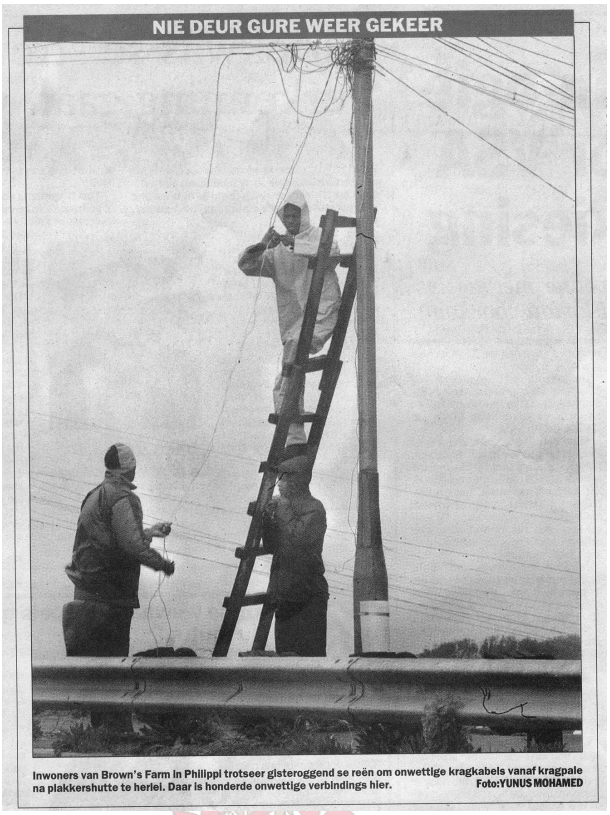


Figure 2.1: *Removal of illegal connections [55]*

As reported by Shingai [70] the problem in South Africa is a social one. As electricity is supplied to people from previously disadvantaged groups the assumption is that electricity should, like roads and health services, be provided free of charge. The local government is in the process of phasing in a $50\ kWh$ basic electricity grant [53, 17] to provide relief for the poor.

Roughly $38\ \%$ of electricity in South Africa is supplied to households by municipalities [24].

Openshaw reported that electricity theft cost Eskom more than R250 million per year in South Africa [60].

Shingai reports the following statistics [70]:

- The United States of America reports a $4\ \%$ distribution loss on total revenue.
- The United Kingdom reports revenue losses of £250 million per year.
- Pakistan's 30 transmission feeder's losses exceed $30\ \%$.
- In certain areas in South Africa up to $38\ \%$ non-technical losses are reported.
- Non-technical losses in India amount to $10\ \%$ per year.

Chapman [13] reports $36\ \%$ non-technical losses in Molofo, Soweto.

Region	R'mil	%
Northern	0.45	3
North-Eastern	0.5	4
Central	7.6	58
Eastern	1.7	13
North-Western	1.5	11
Southern	1.1	8
Western	0.3	2

Table 2.1: *Breakdown of revenue loss in 1998 [11]*

Municipality	2001	2002	2003
Stellenbosch	2.60 %	2.05 %	0.98 %
Matzikama	7.95 %	16.55 %	

Table 2.2: *Breakdown of electricity losses in distribution network*

In 1998 the non-technical loss statistics broken down into the various provinces are shown in Table 2.1 [11], and do not nearly add up to the R250 million mentioned earlier [60].

Information from the annual report of local municipalities was compared in Table 2.2. Stellenbosch and Khayelitsha (see Section 2.5.2) show that losses can be reduced to at least $2\ \%$.

2.3 The South African Law

2.3.1 Theft

According to the South African law, Law 41 of 1987 article 27 (2-3), states that if a person is found guilty of tapping electricity they will be liable to experience the penalties indicated in the following extract:

> (2) Iemand wat sonder 'n wettige reg (waarvan die bewyslas op hom rus) elektriese stroom uitneem, aftak of uitkeer of laat uitneem, aftak of uitkeer of sodanige stroom wat wederregtelik uitgeneem, afgetak of uitgekeer is, verbruik of gebruik, wetende dat dit wederregtelik uitgeneem, afgetak of uitgekeer is, is aan 'n misdryf skuldig en by skuldigbevinding strafbaar met die strawwe wat vir diefstal opgele kan word.
>
> (3) Iemand wat sonder 'n wettige reg (waarvan die bewyslas op hom rus) 'n apparaat vir die ontwikkeling, oorstuur of voorsiening van die elektrisiteit afsny of beskadig of hom daarmee bemoei, is aan 'n misdryf skuldig en by skuldigbevinding strafbaar met 'n boete van hoogstens die bedrag wat van tyd tot tyd by regulasie bepaal of met gevangenisstraf vir 'n tydperk van hoogstens 12 maande of met

daardie boete sowel as daardie gevangenisstraf. [73]

Eskom uses a form as seen in Appendix A.1 to give a written notice to users with unlawful connections as currently used by Andre Truter (see Appendix F.4). Without this completed document no legal action can be taken against the perpetrator.

Although this seems fairly straightforward, a judgment in the Cape High Court stands [50]:

Theft of electricity is not possible as it is not tangible.

2.3.2 Safety

Another important law and issue regarding electricity theft concerns the safety risk introduced due to illegal connections. The utilities and distributors are therefore in terms of the Occupational Health and Safety Act forced to remove any dangerous and illegal connections [75]. In eThekwini Electicity, Durban, 14 deaths have been reported, of which more than half are under the age of 10 [12], due to illegal connections.

When technicians try to remove illegal connections the, communities threaten the lives of the personnel removing illegal connections [11]. This clearly poses a problem.

2.4 Forms of non-technical losses

Although many methods exist to obtain illegal electricity, this section mentions a few obvious methods detected during this study. As an indication of the vast range of possibilities available and the ingenuity of the perpetrators a list from [45] with more methods, has been included in Appendix A.2.

1. Obvious tapping of electricity from a nearby electricity point, Fig. 2.1, Phillipi, Caledon.
2. Posing an extreme safety hazard is the distribution of electricity to individuals from another premises, Fig. 2.2, Grabouw.
3. Tampering with conventional magnetic disc meters, Fig. 2.3, Caledon, for example forcing the disc to rotate slower with a pin.

2.5 Available solutions and implementations

2.5.1 Sweeps

The simplest way to combat electricity theft is by using sweep teams to audit meters and check for illegal connections. Depending on the community this is sometimes extremely dangerous for the personnel [11].

Figure 2.2: *Secondary Distribution in Grabouw*

Figure 2.3: *Tampering with Disc at Caledon*

Depending on the resources available within the local distribution authority the auditing function can be outsourced to private companies.

2.5.2 Case Studies

Ekurhuleni

The subsection is based on a report from Jamieson [44, 43] as well as documentation received during a visit to Ekurhulni in August 2003 (See Appendix I).

History In 1995 the Tembisa reticulation network became the responsibility of Kempton Electricity. With the responsibility Kempton inherited 31 500 consumers and a bad debt exceeding $R\,250$ million and with an additional $R\,3$ million per month unpaid usage due to a non-payment campaign within the community. This, with a neglected network due to the ”self-help” culture required intense maintenance to comply with safety regulations. A plan was approved to upgrade the network and to add an automatic meter reading SCADA system which exceeded $R\,100$ million in cost.

Community buy in Due to the political instability at the time, the time being just after the first election in 1994, the municipality’s new government expected to get paid for services delivered. It was also a high priority to ensure the community’s involvement in the project. A survey was done to develop the market strategy and a marketing campaign was launched to help initiate the new metering system.

Metering implementation The first meters were installed in November 1996. Due to further unrest in the area the need to protect the meters with $10mm$ thick steel protective structures was required, costing $R\,24$ million.

Results With all structures and meters in place credit control was implemented and payment levels rose from $R\,0.9$ million to $R\,8.9$ million per month.

Khayelitsha

The progress seen in Khayelitsha is phenomenal. Within six years the 70% losses are down to 1.5% and this area boasts the lowest non-technical losses in the country [11]. This progress can be attributed to excellent management of the network.

Cape Town

In Keith Strober’s paper, Cape Town [80] sets out his practical approach to revenue protection. The following are a few main points from his report:

- He assigned a team of whom three members were master electricians. This enabled the team to enter into all possible installations.
- His approach is not to disconnect, but to encourage users to remain connected and pay the outstanding debt at an additional rate of $14\ \%$ added to their electricity purchases.
- In order to find perpetrators a printout of all the consumers who have not purchased electricity is used.
- $76\ \%$ of the tampering is found in low-cost areas.
- From the $10\ 000$ annual inspections done only 15% are apprehended.
- By approaching electricity theft in an understanding manner his efforts recovers nearly $R\ 3$ million per year.

2.5.3 DC Reticulation

Since electricity for domestic purposes is provided at $230\ V$ AC it is easy to tap from the distribution company's lines.

Hence, Molepo [57] proposed and developed a solution whereby the electricity is distributed using $750\ V$ to $1\ 000\ V$ DC. At the consumer's connection, a high voltage DC to $230\ V$ AC inverter is installed to allow the power to be used by the consumer. The proposed system can be implemented on the existing aerial bundled conductors used for rural electrification.

The solution's cost is estimated to be under $R\ 1\ 000$ per unit or consumer. The solution is almost fail-safe and would definitely reduce tampering due to the extremely high risk of electrocution when attempting to tamper [34]. This also indicates the one negative aspect of this solution: In the event of equipment failure innocent people may also be electrocuted.

2.6 Conclusion

From the contradictory statistics, figures and background it can be seen that no conclusion about the extent of electricity theft in South Africa can be made. However, the fact that publications were presented and figures calculated by authors indicates that electricity theft in South Africa is of a growing concern. Due to these efforts successes such as Ekurhuleni and Khayelitsha are recorded.

Chapter 3

Time Domain Pulse Reflectometry

3.1 Introduction

One problem with electricity theft is the problem of identifying an individual perpetrator. This chapter investigates the possibility of extracting information about the distribution network using Transmission Line Theory and Time Domain Pulse-Reflectometry (TDR) to identify perpetrators.

Reflections caused by loads and mismatched cables[1] are exploited and studied. By determining the time it takes for a signal to travel to and from a mismatched load, the distance to the load can be determined. The magnitude of the reflections can give an idea of the load situation at each reflection in order to detect possible faults and tampering. The problem with using TDR on a LV reticulation network is that uninterrupted power, meaning $230\ V/50\ Hz$ AC signal, must be delivered to the consumer at all times, while a sharp pulse is sent on the network and images are generated.

This chapter investigates the practical aspects of using TDR as a way of detecting electricity theft.

3.2 TDR background

TDR is a technique used to exploit reflections caused in transmission lines by mismatched loads or imperfections. A pulse propagating in a transmission line at a definite speed [39] is reflected back and forth by discontinuities. Pulses reflected back to the measuring point is used to build an image of the network and transmission line. The discontinuities are caused by mismatched cables, connections and loads due to their characteristic impedances. The propagation equations (3.1,3.2), reflection- (3.3) and transmission (3.4) coefficients shown,

$$V_0^- = V_0^+\Gamma \tag{3.1}$$

[1]It is assumed that cables and loads in TDR have roughly the same characteristics

$$V_r^+ = V_0^+ T \tag{3.2}$$
$$\Gamma = \frac{Z_r - Z_0}{Z_r + Z_0} \tag{3.3}$$
$$T = 1 + \Gamma \tag{3.4}$$

with,

$V^{+,-}$ the amplitude of the pulse propagating in the positive or negative direction,

x the distance from the connection of the two transmission lines,

Z the respective surge impedance of the transmission lines,

Γ the reflection coefficient and

T the transmission coefficient.

are illustrated in Fig. 3.1, where the two transmission lines are denoted by using subscripts $0, r$. These equations are derived from the telegraphist equation [46, 65, 39, 35]. Further explanations and derivations are given in Appendix B.

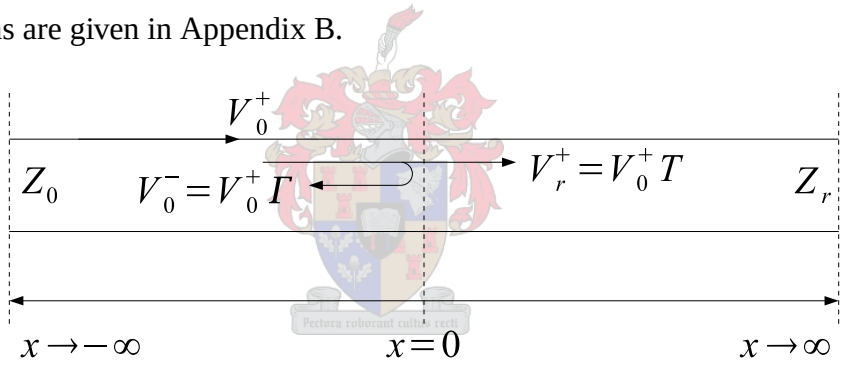


Figure 3.1: *Reflection and Transmission Coefficients of a Transmission Line Configuration*

3.3 Pulse Generator Topology

As mentioned earlier in this chapter the power supply may not be interrupted, and the power provided to the consumers should be within regulations [67, 68]. Various pulse generators were evaluated, see Appendix D, and the parallel-switch topology shown in Fig. 3.2 was found to be a practical solution.

The idea with this topology is to produce an inverted pulse on the $50Hz$ signal of the power signal as shown in Fig. 3.3 by dissipating energy, instead of injecting energy into the system. The advantage of this topology is that the device can be clipped onto the grid at existing split pole's terminals, without interrupting the supply.

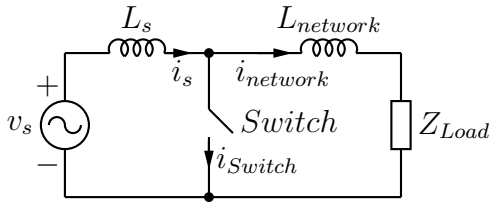


Figure 3.2: *Pulse Generator Topology*

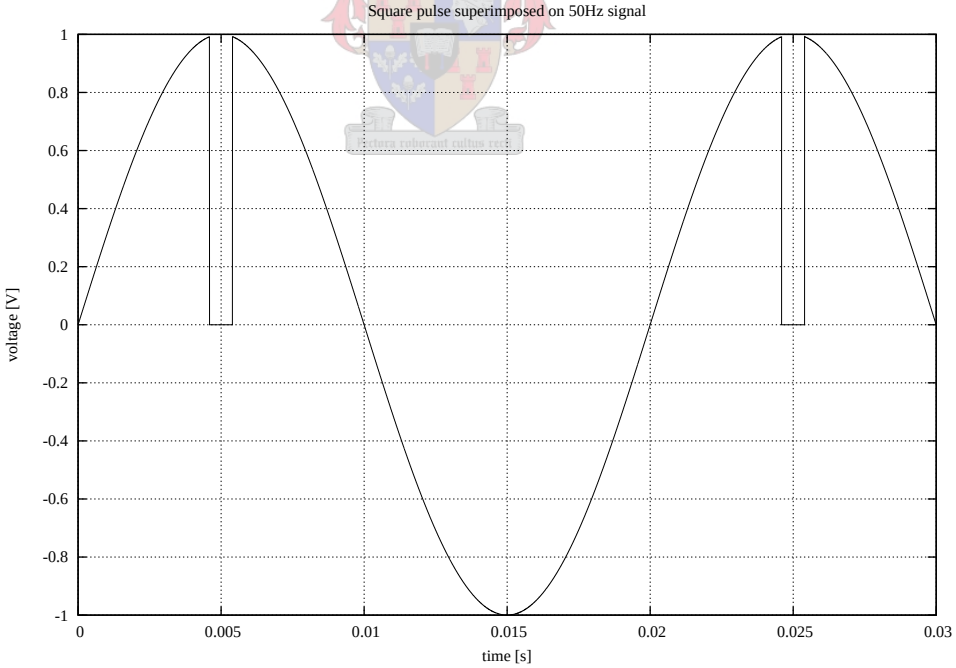


Figure 3.3: *Pulse Generation*

3.4 Design of Prototype Parallel-Switch

The pulse must be characterized by high dV/dt's and a small pulse width to increase resolution of the measurement, while simultaneously reducing harmonic content. A pulse width of between $40\ ns$ to $100\ ns$ was found to be practical. The $100\ ns$ pulse width is small enough compared to the $20\ ms$ period of the power signal and therefore the $50\ Hz$ component can be ignored.

The impact on power quality should be considered and preferably not be affected in any way. This implies that the pulse repetition rate should be kept as low as possible. The device implemented should not disturb the power flow of the current reticulation installation drastically during operation and installation.

3.4.1 System Specification

An Infineon CoolMOS 20N60S5 [41] was selected to perform the function of the main switch. The pulse should be initiated when a signal is received via an optic fiber interface in order to isolate the user and signalling equipment.

The CoolMOS MOSFET from Infineon [41] is considered with the following highlighted specifications:

$$V_{GS} = \pm 20\ V \tag{3.5}$$
$$V_{DSSmax} = 600\ V \tag{3.6}$$
$$I_{D_{max}} = 20A@25\ ^oC \tag{3.7}$$
$$= 13A@100\ ^oC \tag{3.8}$$
$$P_{tot} = 208\ W \tag{3.9}$$
$$R_{DS(on)} = 0.19\ \Omega \tag{3.10}$$
$$t_{d(on)} = 120\ ns \tag{3.11}$$
$$t_r = 25\ ns \tag{3.12}$$
$$t_{d(off)} = 195\ ns \tag{3.13}$$
$$t_f = 45\ ns \tag{3.14}$$

With the time limits on the MOSFETs the minimum pulse width can roughly be calculated as $100\ ns$. A maximum possible pulse width of $t_{pw} = 200\ ns$ is used in our calculations. Relative to the $200\ ns$ pulse the $50\ Hz$ can be considered as DC.

Fig. 3.2 shows the loads and currents flowing in the system. Using Kirchoff's current law $I_s = I_{network} + I_{switch}$. Due to the inductive nature of the transmission lines high $\frac{dI(t)}{dt}$ is not possible. This implies that when the switch is closed the additional current of I_{switch} will start flowing in L_s. When the switch is opened this additional current must be dissipated by the

snubber circuit. It is also clear that the main load current does not pass through the switch. Therefore, for the design of the switch, the load conditions in the circuit is not considered.

The following assumptions are summarized:

- $20\ ms >> 200\ ns$, therefore power source is considered to be DC.
- I_{switch} is absorbed by the snubber circuit.
- L_s is considered to include the transformer inductance and line inductance as shown in Fig. 3.2, where L_s is larger than the minimum inductance specified in (3.16).

3.4.2 Circuit Layout and Design

The total circuit layout is shown in Fig. 3.4. Components were selected from available stock in the laboratory and available samples from Texas Instruments. Due to the small amount of components only a few design considerations were taken. What is important, however, is to determine the specifications of the external circuit and how external components will influence the circuit behavior. The design is split into the following sections:

- The optical interface receives a signal from a computer or signal generator.
- The glitch generator is responsible for the pulse width.
- The driver circuit ensures that the CoolMOS has sufficient gate current to switch on and off.
- The switch circuit is used to generate the $0\ V$ pulse on the line.

Switch Circuit and Snubber Design

The switch and snubber circuit are considered as the main components of the system and will operate within the LV reticulation network as shown in Fig. 3.2.

Consider the network in operation as shown in Fig. 3.2. The current in L_s and in $L_{network}$ will be equal. Inductors cannot have discontinuous current flowing through them. When the switch is on for $40\ ns$ to $200\ ns$ the current in $L_{network}$ can be assumed to be continuous and will draw the required current from L_s for that instance in time. Therefore in the snubber design we need to compensate for the additional energy stored in L_s during the on state when the network is in full operation and when no load or $L_{network}$ is present.

Since L_s can change for each different setup, the minimum L_s should be specified. The device will probably be operated outside and can easily be exposed to maximum sunlight. Therefore, the maximum current at $100\ ^oC$ that may flow through the switch is $I_{max} = 13\ A$ from

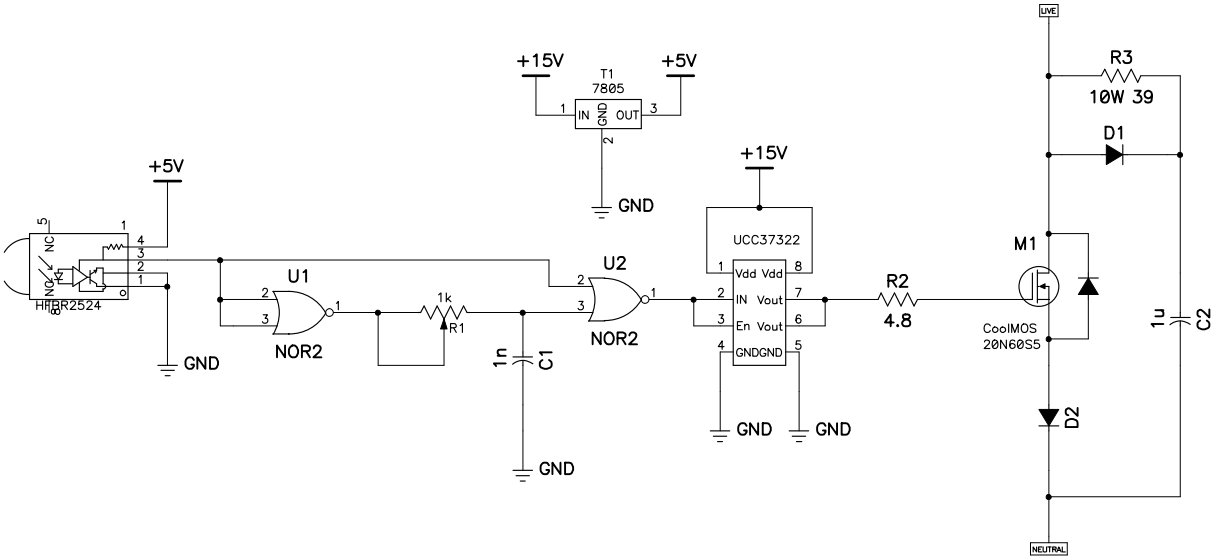


Figure 3.4: *Circuit Diagram of Complete Parallel Switch*

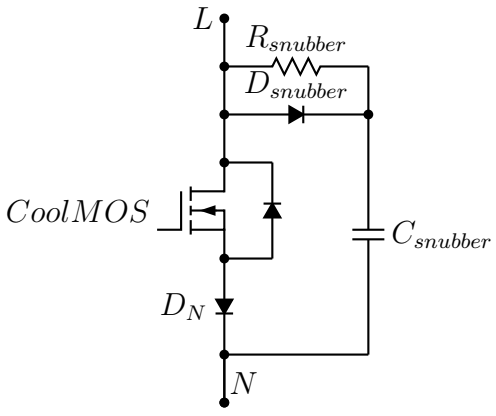


Figure 3.5: *Switch and Snubber Circuit*

(3.8), with a peak voltage of $V_{peak} = \frac{\sqrt{2}}{\sqrt{3}} 400\ V = 327\ V$, $t_{pw} = 200\ ns$ the maximum pulse width, is used.

$$L_s \approx \frac{V_{peak} t_{pw}}{I_{max}} \tag{3.15}$$
$$\approx 5.025\ \mu H \tag{3.16}$$

The device must be able to absorb the total energy stored in L_s to ensure that the proper functioning will occur under no load conditions. The energy stored in the inductor is

$$E_{s(min)} = \frac{1}{2} L_s I_{max}^2 = 424.578\ \mu J \tag{3.17}$$

when the MOSFET is conducting. The over voltage snubber capacitor sinks this energy when the MOSFET is turned off. The voltage over the capacitor is the same as the bus voltage, which can be taken as V_{peak}, just before the MOSFET is turned on again.

$$E_{snubber} = \frac{1}{2} C V^2 \tag{3.18}$$
$$C_{snubber(min)} = \frac{2E_{s(min)}}{V^2} = 7.941\ nF \tag{3.19}$$

A high frequency snubber capacitor with a value of $1\ \mu F$ was available in the laboratory. The energy storage available in this capacitor is

$$E_{snubber} = \frac{1}{2} C V^2 \tag{3.20}$$
$$= 53.465\ mJ \tag{3.21}$$

The problem however with the current configuration is that there is no real way to dissipate the energy stored in the capacitor. From the capacitor a resistor is connected to the live bus to drain the capacitor. The resistor should be chosen so that the capacitor is discharged when the next switching cycle starts. The following approximate identity holds [56] and $R_{snubber} = 39\ \Omega$ is chosen due to availability, therefore

$$t_{empty} > 2.3 C_{snubber} R_{snubber} \tag{3.22}$$
$$> 100\ \mu s \tag{3.23}$$

Considering that the stored energy in the inductor decreases as the inductor size increases, for the configuration, as indicated in (3.26),

$$E_L = \frac{1}{2} V I_{max} t_{pw} \tag{3.24}$$
$$= \frac{V^2 t_{pw}^2}{2L_s} \tag{3.25}$$
$$\Rightarrow \lim_{L \to \infty} \frac{V^2 t_{pw}^2}{2L_s} = 0 \tag{3.26}$$

Cable Inductance The minimum inductor size must not be less than $5.025\ \mu H$. To translate this into a physical cable distance from a source, neglecting source inductance, a $150\ mm^2$ ABC [2] cable is used[2] and

$$X_{L(min)} = 0.082\ \Omega/km\ @\ 50\ Hz \tag{3.27}$$
$$L_{min} = \frac{X_L}{2\pi f} \tag{3.28}$$
$$= 261.014\ nH/m \tag{3.29}$$
$$\ell = 19.252\ m \tag{3.30}$$

Transformer Inductance To calculate the $11\ kV$, $450\ V$ distribution transformer inductance the fault current is estimated at 10 times the transformer rating. With the line to phase voltage the impedance is calculated. Considering that the inductance is approximately ten times higher that the resistance, the inductance is calculated. For example:

$$S_{rated} = 800\ kVA \tag{3.31}$$
$$I_{\phi fault} = 10(I_{rated}) \tag{3.32}$$
$$= 11.594\ kA \tag{3.33}$$
$$Z = \frac{V_{rated_{LN}}}{I_{\phi fault}} \tag{3.34}$$
$$= 19.739i + 1.974\ m\Omega \tag{3.35}$$
$$L_{800\ kVA} = \frac{\Im(Z)}{2\pi f} \tag{3.36}$$
$$= 62.831\ \mu H \tag{3.37}$$
$$L_{100\ kVA} = 252.6\ \mu H \tag{3.38}$$
$$\tag{3.39}$$

The values are calculated to give a general idea of the impedances using the rule of thumb, and should be verified with the data of the actual transformer, prior to testing. The graph in Fig. 3.6 gives an idea how the rating of the transformer influences the inductance.

Driver Circuit

In order to switch with sharp rising and falling edges, which enhances the resolution, it is required to switch the MOSFET as hard as possible. The driver should be capable of providing all the required current for the MOSFET. A driver circuit, the UCC37322 (non-inverting) MOSFET with enable [90] from Texas Instruments was selected. The operating voltage range of

[2]The inductance according to the datasheets decrease as cable size increase. Since the minimum distance is calculated the largest available conductor is used.

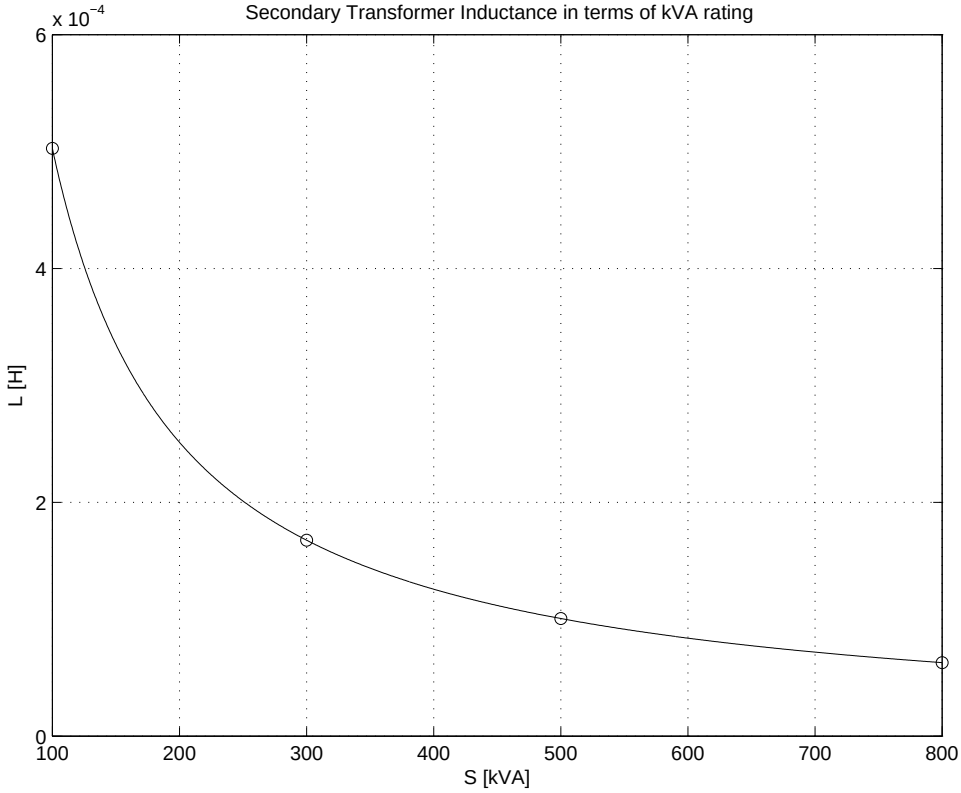


Figure 3.6: *Transformer inductance as a function of kVA rating*

about $15\ V$ can be applied, which is in the operating range for V_{GS}. The chip can sink $9\ A$ and have typical rise times of about $20\ ns$.

The gate resistor must be chosen such that the maximum current requirement for the driver is not exceeded and the time constant of the gate capacitor and gate resistor combination does not exceed the switching times of the MOSFET. The minimum resistance of R_G is calculated:

$$R_G > \frac{V_{on}}{I_{on}} \tag{3.40}$$

$$> \frac{15}{9} \tag{3.41}$$

$$> 1.6\ \Omega \tag{3.42}$$

The shortest switching time of the MOSFET is $t_s = t_{d(on)} + t_r = 145\ ns$ and the input capacitance $C_{iss} = 3\ nF$, according to [41]. The maximum resistance is then calculated to be:

$$R_G < \frac{t_s}{C_{iss}} \tag{3.43}$$

$$< 48\ \Omega \tag{3.44}$$

A value of $R_G = 4.8\ \Omega$ was chosen to drive the MOSFET.

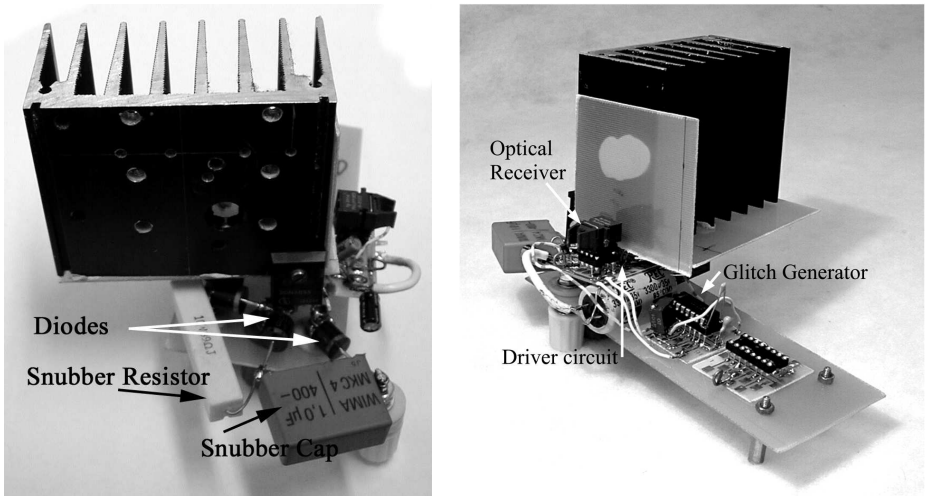


Figure 3.7: *Parallel Switch*

Glitch generator and circuit timing

To generate a pulse with small a enough pulse width, a Schmidt inverter logic circuit was used to construct the glitch generator. The idea is that the NAND logic is driven with two different signal paths, which can be achieved by running the one signal through another NAND gate. The circuit's time constant is increased using a RC circuit with a variable potentiometer, with a maximum of $R = 1\ k\Omega$ and a capacitor with $C = 1\ nF$. This allows the pulse width to be tuned between roughly $8\ ns \leq t_{pw} \leq 1\ \mu s$.

Optical Fibre input

The optical fiber input, HP R-2522, is used to start the pulse generator. An optical signal is received from either a signal generator or computer. This ensures that the computer and signal generator are isolated from the mains power. The optical fiber input feeds the glitch circuit.

3.4.3 Results

The final prototype is shown in Fig. 3.7. The device was tested by connecting it directly to the utility power. A signal was generated using a computer and the optical interface. The pulse width was set to around $80\ ns$. The result is shown in Fig. 3.8.

3.4.4 Practical Implementation Considerations

High Frequency Model

Due to the signals propagating in the transmission lines at high frequencies it is important to know how the pulse generator reacts under these conditions.

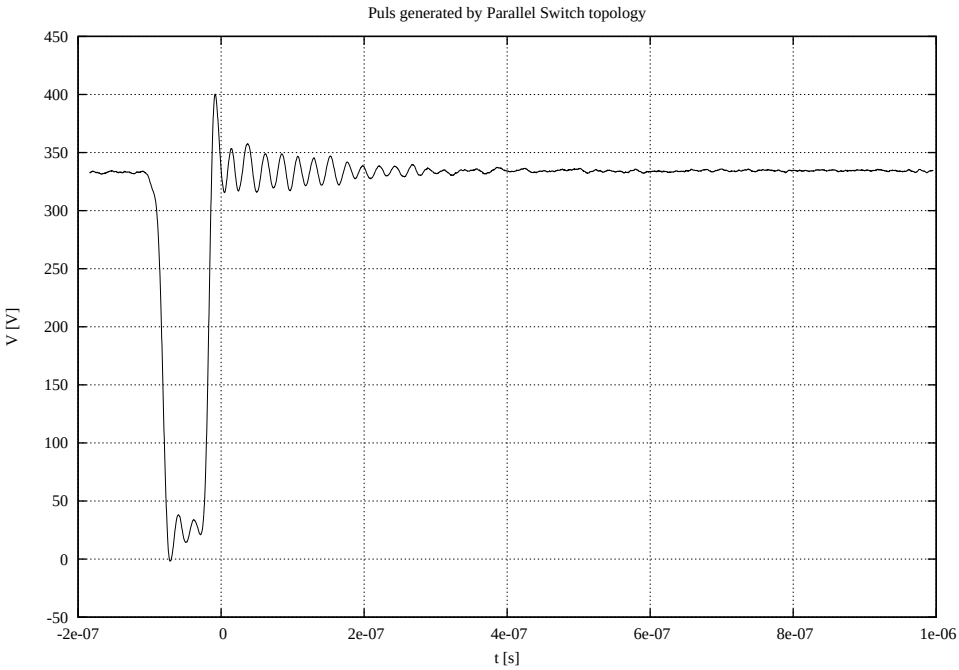


Figure 3.8: *Pulse Generated by Parallel-Switch*

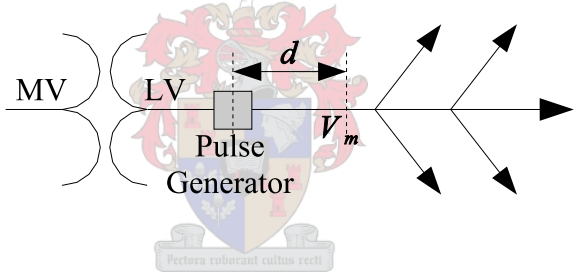


Figure 3.9: *Proposed Pulse Generator Position*

The model used for the parallel-switch is obtained by considering the snubber circuit and switching components. As shown in [58], for a diode it can be seen that the capacitance present acts as a short under HF conditions, which poses a problem. The MOSFET with C_{DS} present can also be modeled roughly as a short as indicated in [58], under high frequency conditions.

System Position within Reticulation network

Fig. 3.9 shows a generalised overview of a low voltage reticulation network. From the high voltage or medium voltage distribution network the transformer brings the voltage to an acceptable voltage, $V_{LL} \approx 400\ V$. From the transformer the power is distributed in a typical tree structure scheme[3]. The placement of the pulse generator and the measurement device must be considered. Due to the fact that the high frequency characteristics of the pulse generator react

[3]See Appendix E

as a short circuit and, therefore, the measurement device must be put some distance from the pulse generator as indicated by V_m and distance d. Placing the pulse generator at the end of the network won't have the desired effect, since the transmission line will have a hyperbolic effect on the pulse as it moves nearer to the transformer [46]. The distance from the transformer must also be considered. Using an inductor L_{min} it is calculated in (3.30) using (3.28) that the minimum length to the transformer should be $\ell = 16.5\ m$.

3.5 Experiments

Experiments were conducted to test the Parallel Switch Pulse Generator and verify the theory in previous sections. The tests are conducted at a low voltage, $V_{peak} = 327\ V$, at the end of the supply line. Therefore, a large inductance L_s, Fig. 3.2, is expected.

Two experiments are presented to illustrate the parallel switch functionality. The experiments are sketched using signal flow graphs [65]. Please refer to Section B.4 in the Appendices for a summary of the Signal Flow Graphs Theory.

3.5.1 Measurement Position

This experiment investigates the positioning of the measuring device relative to the pulse generator.

The pulse generator is connected to the power outlet (V_{50Hz}) at measuring point V_{switch}. The $15\ m$ flex cable[4] ($|A|\, e^{-\gamma\ell}|_{15m}$) extends the transmission line to the second measuring point V_{15m}. A $50\ m$ parallel wire transmission line ($|A|\, e^{-\gamma\ell}|_{50m}$) is added in the second experiment set.

The signal flow graph in Fig. 3.10 shows the expected propagation paths of the pulse through the setup. The open circuit at the end of the $50m$ parallel wire is indicated with the reflection coefficient, $\Gamma \approx 1$. The pulse generator's reflection coefficient is shown by $\Gamma \approx -1$. Although some small reflections occur at V_{15m} these are small enough to be discarded for this experiment's purpose.

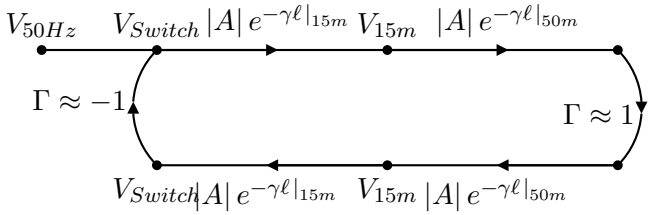


Figure 3.10: *Signal Flow graph for measurements at switch*

[4] 3-core $1.5\ mm^2$

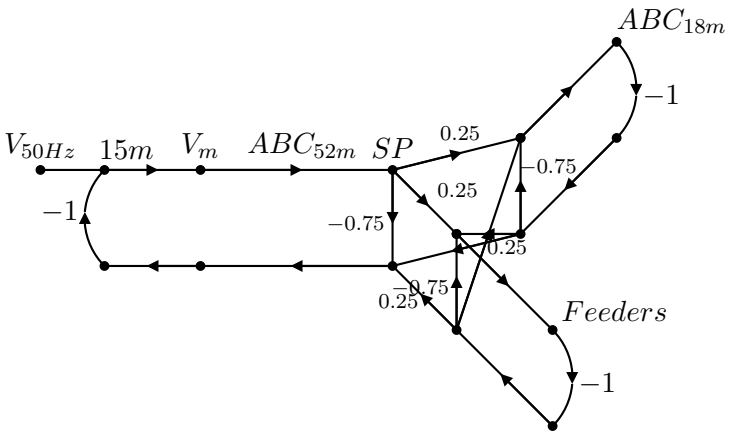


Figure 3.11: *Signal Flow graph for ABC measurements*

3.5.2 ABC and Split poles

For theft detection it is important to measure changes when new connections are made. The setup of this experiment is based on a simple LV network and therefore a slightly different configuration is used than in the previous experiment (Section 3.5.1). This experiment investigates the possibility of TDR to picking up changes in an LV network configuration.

The measurements are all taken $15\ m$ from the pulse generator. This is done to ensure that pulse widths do not overlap when the signal is propagating towards the PG and reflected back. The measurement distance from the pulse generator is calculated using (3.45)

$$\ell_{back\&forth} = \frac{v_c t_{pw}}{2} \approx 15\ m \tag{3.45}$$

with,

ℓ the distance from the pulse generator to the measuring point,

v_c the propagation speed of the pulse in the transmission line and

t_{pw} the pulse width of the initiated pulse.

The $50\ m$ parallel wire is replaced by a $70\ m$ Aerial Bundled Conductor (ABC) which is used for low-cost LV reticulation networks. At $52\ m$ a split pole configuration was added, with 4 feeders with open circuits at approximately the same length. The feeders can be disconnected from the ABC by a circuit breaker. This experiment is shown by the signal flow diagram in Fig. 3.11, which includes the added feeders at $52\ m$ at the split pole.

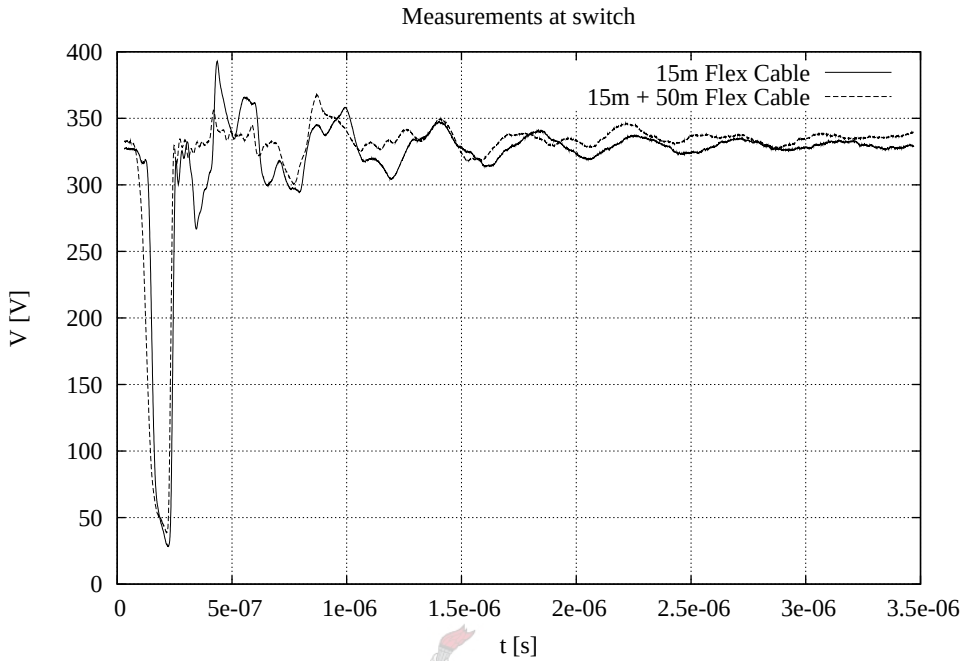


Figure 3.12: *Results from measurements at switch*

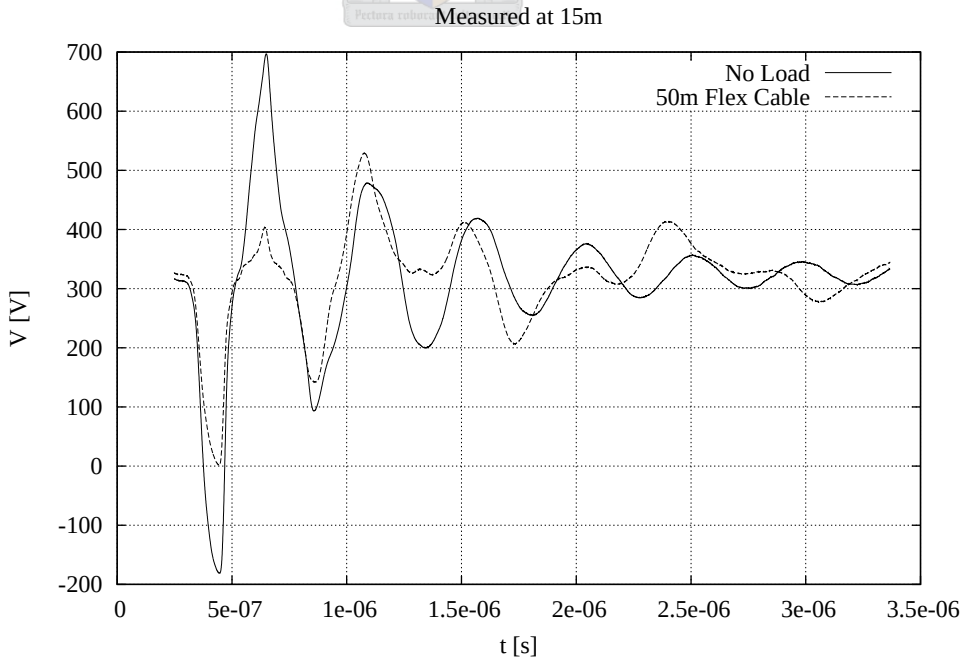


Figure 3.13: *Results from measurements* $15\ m$ *from switch*

3.6 Results

3.6.1 Measurement Position

It can be seen from the results in Fig. 3.12 that no significant signal can be detected when measuring at V_{switch} due to the capacitance of the pulse generator. By moving the switch and comparing the measurements at V_{15m} substantial differences in results can be seen in Fig. 3.13. The cable lengths and setup, in terms of the signal flow diagram, are indicated in the legend of Fig. 3.10.

3.6.2 Aerial Bundled Conductor and Split pole

The results of the aerial bundled conductor (ABC) and split pole experiment are shown in Fig. 3.14. For the measurements with only the ABC connected to the pulse generator the small reflected pulses can be attributed to bends and non-uniform characteristics of the ABC. The second measurement shows the image with the four feeders connected to the ABC configuration. A substantial difference can be seen when the two images are subtracted from each other in the third data set shown.

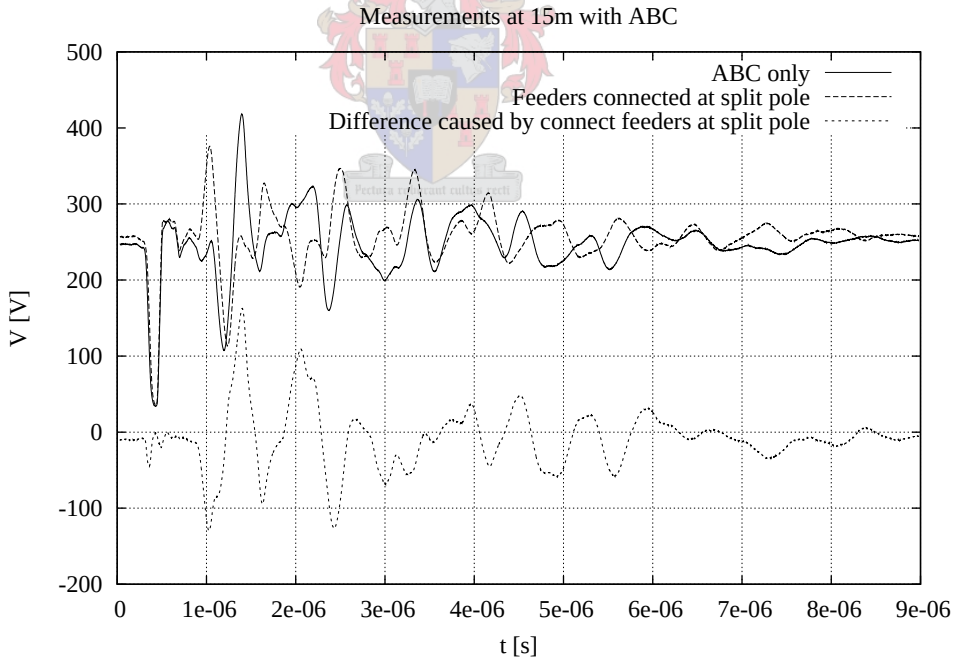


Figure 3.14: *Results from measurements with aerial bundled conductor*

3.7 Simulation

By confirming the models with a SPICE[5] package, larger and more complex networks can be simulated. The parameters used were calculated using the equations provided in Appendix B.5 and the results from an experiment with coaxial cable in Appendix D.2.3.

3.7.1 Cable Parameter Calculation

The data from the experiment performed in Section 3.6.1 was used to determine the parameters. In order to simplify the problem the assumption is made that both cables express similar characteristics and, therefore, they have the same per unit length parameters.

The surge impedance is taken as $Z_0 = 105.545\ \Omega$ from (D.13). From the results of the experiment, the propagation time for an additional $50\ m$ section is measured at $15\ m$ from the switch at V_{15m}, $\tau = \frac{420}{2}\ ns$. The attenuation A as calculated using $V_{t=0ns} = 323\ V$, $V_{t=420ns} = 187\ V$, where $t = 0\ ns$ corresponds with first peak on the results shown in Fig. 3.13, and where the relation $V_{t=420ns} = V_{t=0ns}\ A^2\Gamma$ is used, the attenuation is calculated as

$$A = \sqrt{\frac{V_{t=420ns}}{V_{t=0ns}\Gamma}} \tag{3.46}$$

$$= 0.7609 \tag{3.47}$$

Using the available physical values the following parameters are calculated using the equations in Appendix B.5:

$$R = 571\ m\Omega \tag{3.48}$$

$$L = 439\ nH \tag{3.49}$$

$$C = 40\ pF \tag{3.50}$$

$$G = 52\ \mu S \tag{3.51}$$

$$\tag{3.52}$$

3.7.2 Single Line Simulation

The transmission lines are setup in a simulation which is similar to the experiment in Section 3.5.1. The model used is shown in Fig. 3.15.

When the results in Fig. 3.16 are compared with the results of the actual $50\ m$ Flex Cable measurements in Fig. 3.13, it matches fairly well and confirms the model and measurements. Although the timing and the magnitude of each pulse are roughly in the same area the model does not introduce much distortion and, therefore, gives results that are much better than expected.

[5]Microsims' PSPICE Evaluation Version 9.1

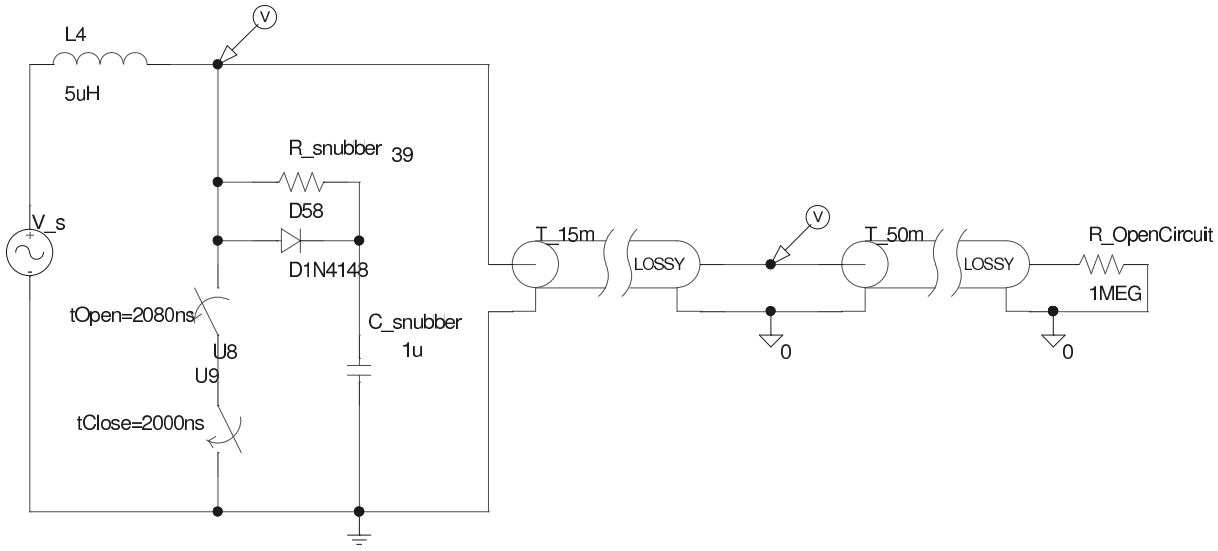


Figure 3.15: *Circuit used for simulation of parallel-switch connected to transmissionlines*

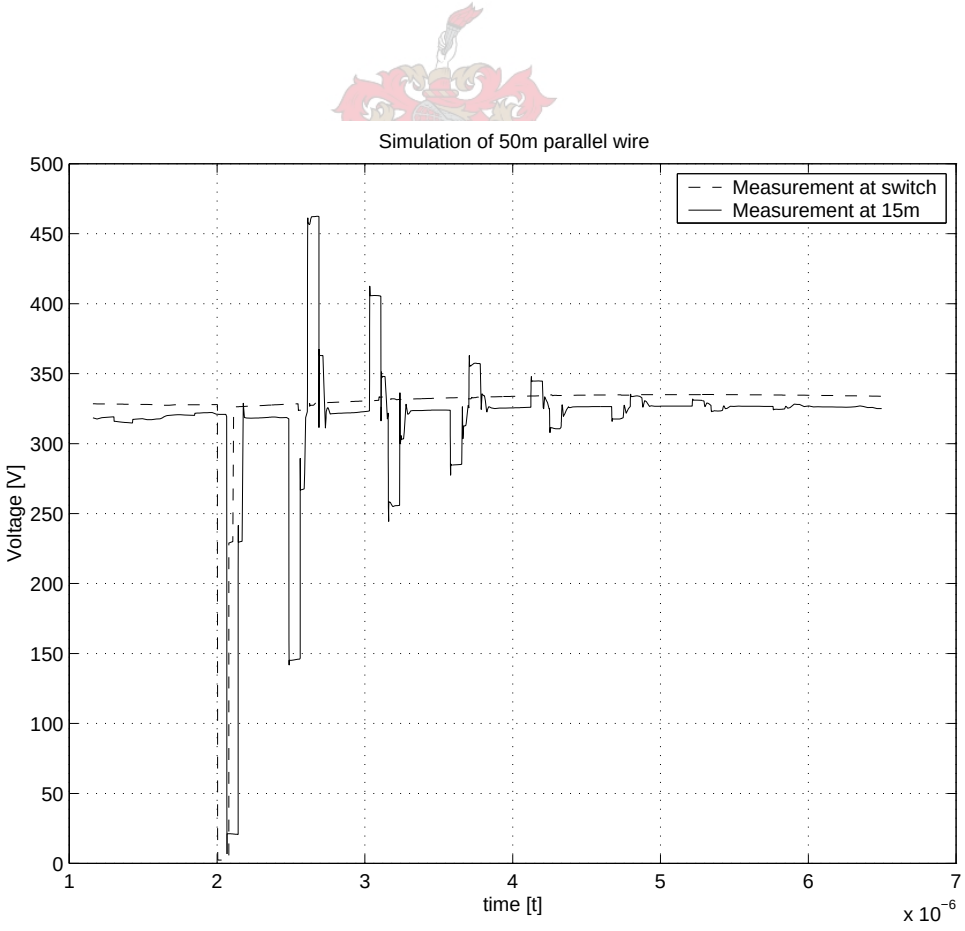


Figure 3.16: *Results from Parallel Switch Simulation*

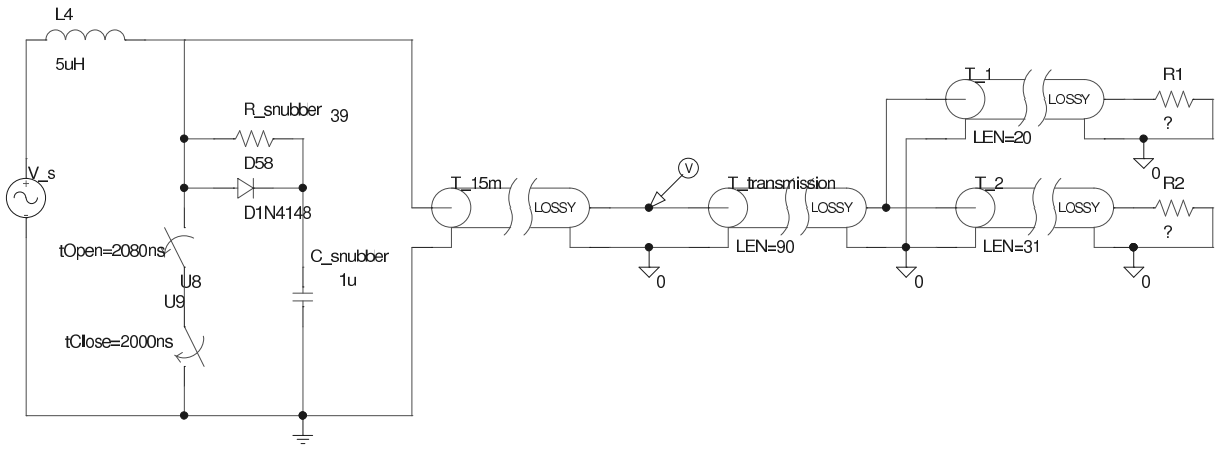


Figure 3.17: *Extended Small Network Circuit*

3.7.3 Split pole Simulation

A second simulation was done to see what effect more connections would have. The simulation performed is a split pole scenario with two connections. The two connections of different lengths, $T_1 = 45\ m$ and $T_2 = 31\ m$, were added to a $90\ m$ cable, as shown in Fig. 3.17. The lengths were chosen to be close multiples of the total length of the first cable. Resistors R_1 and R_2 were used to terminate the two transmission lines, T_1 and T_2 respectively. The measurements were taken after the $15\ m$ transmission line.

The resistor values for the four experiments are shown in Table 3.1.

Experiment	R_1	R_2
1	$1\ M\Omega$	$1\ M\Omega$
2	$1\ M\Omega$	$0\ \Omega$
3	$0\ \Omega$	$1\ M\Omega$
4	$0\ \Omega$	$0\ \Omega$

Table 3.1: *Resistor values for split pole simulation experiment*

The results of the simulation are shown in Fig. 3.18, with the plot only showing data from the first reflection from the split pole connection. The first reflection is found at $t = 2.823\ \mu s$ and travels at a speed of $v = 241\ Mms^{-1}$. The influence of the change in the impedances at R_1 and R_2 can now be seen when the reflection from the switch, $t = 2.948\ \mu s$, meets the reflection from R_1 at $t = 2.992\ \mu s$. The reflection of R_2 is seen at $t = 3.067\ \mu s$. The trailing end of the first reflection from the switch can be seen at $t = 3.0255\ \mu s$.

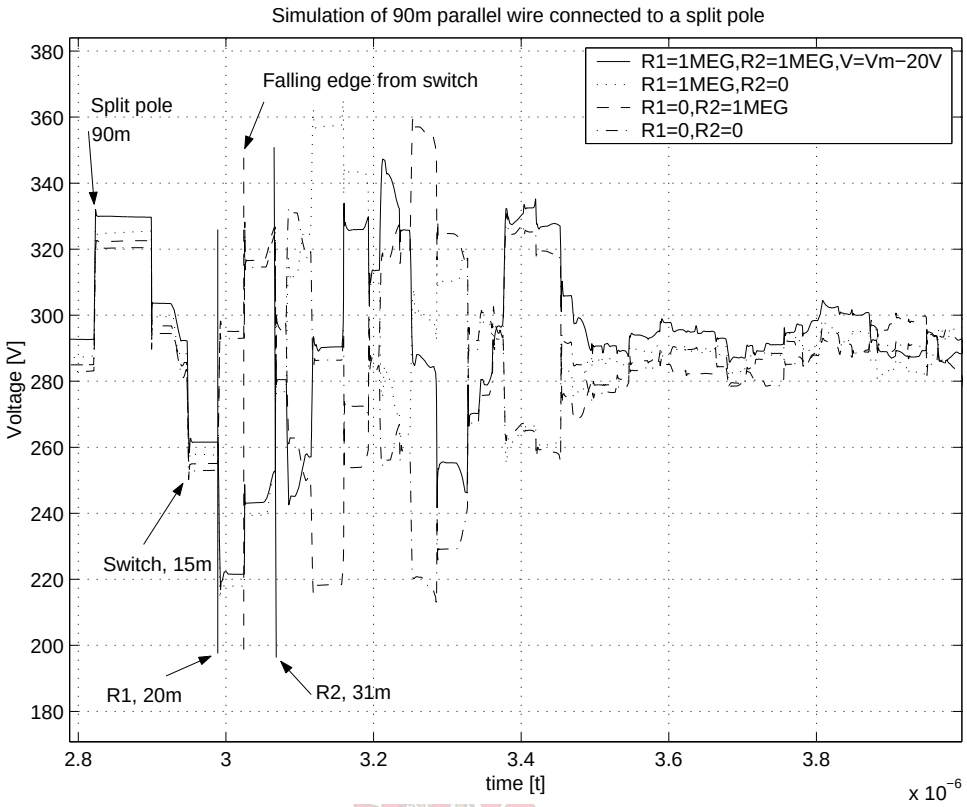


Figure 3.18: *Results of Simulation on small network*

3.8 Evaluation

From the measurements and simulations it can be seen that only after a few hundred meters the accuracy of TDR declines rapidly due to the superposition of multiple reflections. It is also extremely difficult to interpret these signals and map them to a two-dimensional network. Although, as Nielson (see Appendix H.3) proposed, the use of directional couplers may reduce the effect of the reflections against the switch , the problem of echoes however will remain at other split connection further down the line. In the simulation the effect of distortion is not even present. It can, therefore, be expected that field measurements will be even less interpretable as the results presented here. Due to these anticipated inaccuracies it may be required that many units be placed over the whole network section to increase accuracy. This has a direct impact on the total cost of this solution.

An expensive part of implementing this system will be the high voltage, high sample rate, high accuracy data acquisition component. Combined with the running costs of a data link, this could become an expensive solution, even per unit. Since DSP techniques are not yet developed to decode the signals, human interpretation is required. This adds to the running cost of the system. At this stage any automated process would require an accurate model of the implemented network.

With a densely populated network it can be seen from the measurements and simulation that the signal complexity will grow and will become distorted beyond recognition.

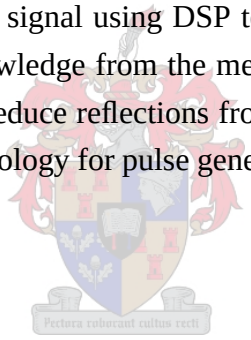
Although not suitable for electricity theft detection, this concept can be used for fault detection [21] on a live network, and possible implementation for cable theft detection on MV lines can be considered.

3.9 Conclusion

This chapter shows that TDR can be applied to an LV reticulation system without interrupting the supply. This enables real time on-line TDR imaging of the network. The placement of different system components is crucial. The capacitance of the PG and pulse width must be taken into account when assembling the system.

Although differences can be detected, the extraction of data from the one-dimensional signal for a two-dimensional problem is too complex to implement in a working system for accurate theft detection.

Modeling and interpreting the signal using DSP techniques can be considered for investigation purposes to gain extra knowledge from the measurements. The use of directive bridge couplings can be investigated to reduce reflections from the pulse generator. Possible applications, using the parallel switch topology for pulse generation, are on-line medium voltage cable theft and fault detection.



Chapter 4

Theory of Constraints applied to Electricity Theft

> If the 1980's were about quality and the 1990's were about re-engineering, then the 2000's will be about velocity. About how quickly the nature of business will change. . . The successful companies of the next decade will be the ones that use digital tools to re-invent the way they work. - Bill Gates, 1999

4.1 Introduction

To address the problem of electricity theft Goldratt's Theory of Constraints (TOC) [36] is used. The initial focus of this publication was at optimization of processes in production plants. However, since its publication in 1992 it has found many other applications and is still regarded as highly relevant in change management today [9].

4.2 The Goal

The main idea behind the TOC philosophy is to establish the goal of an operation, and to optimize the operation towards achieving its goal. Consider a general electricity distribution company. Such a company should be managed in a sustainable manner. This points to one main goal. The company should generate money and profits for the shareholders. In order to achieve this goal the company needs to supply customers with a product or service. In this case, cost effective high quality power. In order to deliver the product as defined, the company should apply a Continuous Improvement Strategy to improve on their past performances and provide affordable electricity to both industry and consumer.

For the purpose of the thesis the goal is extrapolated further and narrowed down to our constraint, namely the "Reduction of Non-Technical Losses". This can be seen as the goal for this thesis as shown in Fig. 4.1.

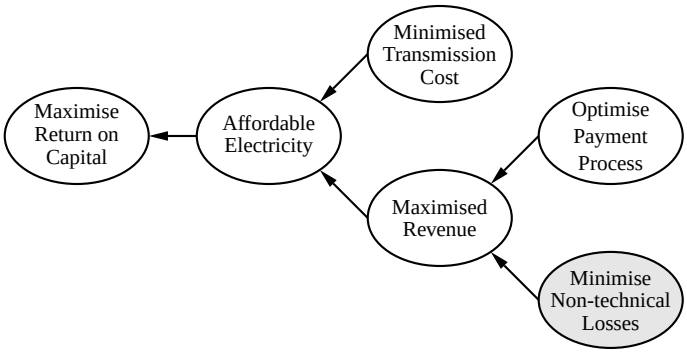


Figure 4.1: *Identifying constraints acting on the goal*

4.3 The TOC Process

According to [31] there are five steps to ongoing improvement. These steps are reduced to four steps for our purposes[1] of handling electricity theft. The process included in these steps is shown in Fig. 4.2.

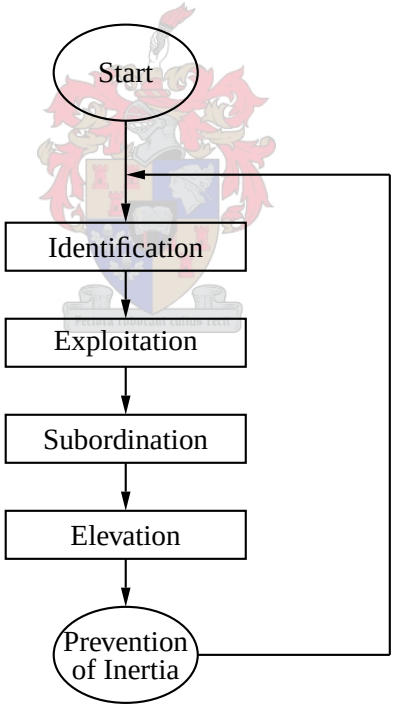


Figure 4.2: *TOC: a Process of Ongoing Improvement*

[1] Elevation is not relevant

4.3.1 Identification

The identification process must be seen in the context of the goal. Anything that brings the whole process closer to the goal is good, and anything that pushes it further away from the goal can be seen as a constraint. For electricity theft, the problem can be subdivided into locations based on the area each different transformer provides with power. Since electricity flows in one direction this makes the search for the weakest link a fairly straight-forward process. The weakest link can be seen as the area where the most electricity theft occurs. This area is where the focus for optimisation should be.

The decision of area size and how many transformers are included in this area will depend on the various situations.

4.3.2 Exploitation

With the facts known with regards to the specific area, for example the losses is known, various methods can be implemented. Below is a list of some known methods:

- If an individual is identified, prosecution is an option (see Appendix F.4).
- Implementing a full-scale remote measurement system has proved to be effective for Ekurhuleni Municipality (see Appendix I).
- Meter sweeps and meter audits can also be done.
- The use of another energy form, for example a $700\ V$ DC, has already been developed[57] for the purpose of eradicating electricity theft.

For each method a business plan can be drawn up in order to show the sustainability prior to initiating such a project. The evaporative cloud can be used here to determine the specific conflict to be addressed for solving this area's problem. Such a cloud is presented in Fig. 4.3, where the prerequisites are in conflict with each other.

An example of such a conflict could be the following:

In order for the electricity utility to provide cost effective electricity requires payment of electricity bills. The perpetrators do not have money and therefore cannot pay their accounts. This clearly results in a direct conflict.

4.3.3 Subordination

Depending on the nature of electricity theft a certain priority and urgency must be given to the relevant project [74]. Depending on the priorities of the organization, that is in terms of distributing the electricity, everything must subordinate in order to reduce electricity theft in the area.

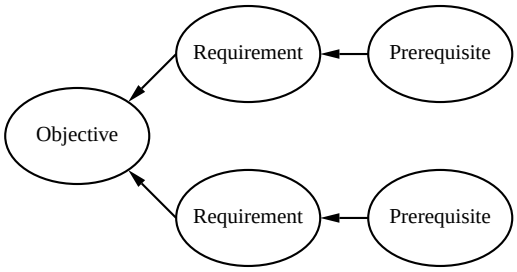


Figure 4.3: *Evaporative cloud for the general case*

4.3.4 Prevention of Inertia

When the situation in the specified area is under control, a new area can be identified as the constraint. The process will now repeat itself when the new constraint is identified.

4.4 Gap analysis and proposed process

As can be seen in the above section there are solutions to address electricity theft and these are available for implementation when an electricity theft hot spot is identified. Management should take care of *subordination* and *prevention of the inertia*. The critical factors for selecting a method is the return on investment made or effort spent and the local context of the electricity theft problem. What is lacking is an effective way or method of monitoring and detecting electricity theft.

With the help of TOC a process has been defined. This process is illustrated in Fig. 4.4.

The process proposed in the thesis consists of the following steps:

1. With all the available data on the various levels one can investigate the current losses within the network and determine if it is feasible to initiate a further investigation concerning the losses of the system in monetary terms. This would typically involve the electricity bill from the incoming feeders into the network and the total consumption of all the customers on the network. The statistical meters on the medium voltage network can also be used. If it makes sense, continue with the process.

2. Identify the measurement points and how energy flows relative to these points.

3. Determine possible “hot spots” within these measurement points and try to reduce or eliminate known technical losses as much as possible by choosing the measuring points carefully, for example:

   - Measure on the LV side of the transformer, to reduce the transformer losses.
   - Reduce the human factor in the information-gathering chain.

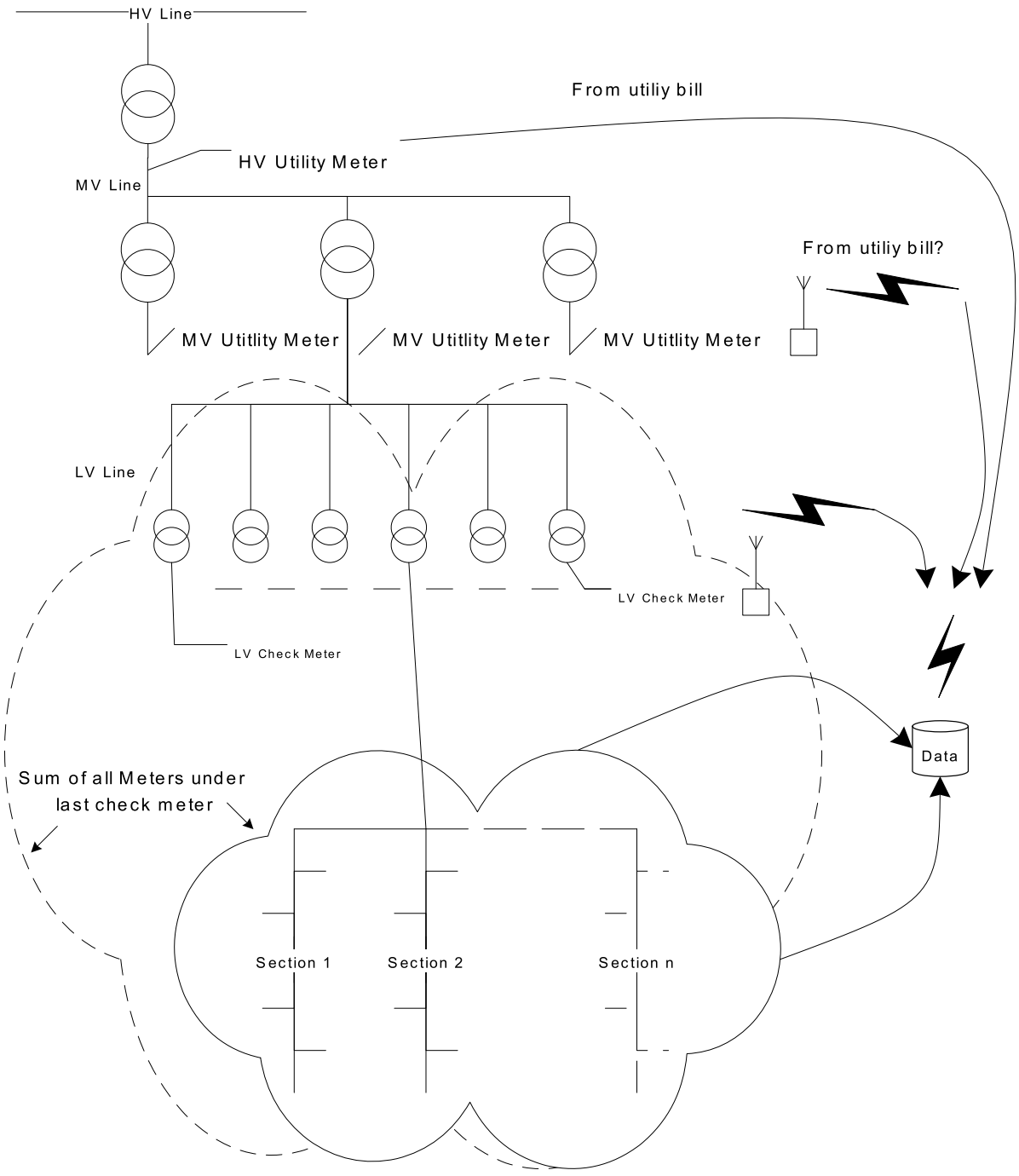


Figure 4.4: *Finding the constraint in the network*

4. With all the decisions that need to be made, keep the costs and benefits in mind through the whole process, and focus on the goal in terms of TOC.

5. If the area is successfully determined and electricity theft has dropped significantly, using *exploitation*, *subordination* and *prevention of inertia*, repeat the identification process again.

To aid this process the following needs to be investigated:

1. The available resources at hand, within the current network.

2. A platform for storing the appropriate data.

3. Investigation and possible development into an efficient system in the information chain. If these systems are required the specifications should be determined.

4.4.1 Resources

Available

In the general distribution network, power flows into the network and all power is consumed in some form. The distributor pays for incoming power and the consumers pays for the used energy. For the utility to charge the distributor, the energy flowing into the distribution network is measured and charged. In some networks, statistical meters may be installed on the MV lines using VT-CT converters or units, as seen in Grabouw in Fig. E.3, which can be used to identify the energy flow into one specific area.

The available resources can be summarized in the following points:

- Incoming meter readings on the HV lines.
- Consumed meter readings (e.g. all the paying consumers)
- Possible statistical measurements on the MV lines.

Required

With all the data available, from the consumers, statistical meters and the utility accounts, the net losses can be calculated. Since all the data is stored in various formats and in different databases a software package is required to represent this data in a useful format, taking into account the possible delays in measurements and stochastic sample times.

Lacking in this setup is a meter on the LV side of the distribution network. Since electricity theft is a dynamic occurrence, the meter should be extremely mobile and weatherproof.

With a platform in place where losses for an area can be calculated one last area is not addressed. The meter readings for the conventional meters are done using a hand system for

data collection. This means that the meter readers write the readings into a notebook and the clerks type the data into the municipality's debtor system. This leaves a lot of room for human error.

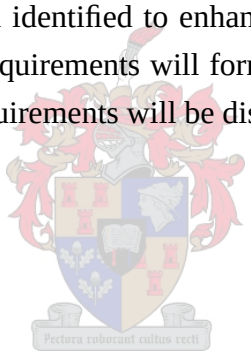
The following are identified to be lacking in the system:

- A data representation package or light-weight data mining application
- An LV mobile remote check meter
- A mobile data collection or meter reading solution

4.5 Conclusion

In this section a process for detecting and identifying and addressing electricity theft has been discussed. Implementing the TOC will elevate revenue protection to a new level. This process is in line with the South African Government Municipal Systems Act [74].

Three requirements have been identified to enhance the TOC thinking process, when applied to electricity theft. These requirements will form a platform for further research. In the following chapters these three requirements will be discussed and expanded.



Chapter 5

Data Mining Software Development

5.1 Introduction

With available technology vast amounts of data is captured daily. Data appears in a variety of formats and locations in distributed systems. These data sets are stored with inconsistent time stampperiods. This requires special methods to extract and represent the information from these data sets.

> Data Mining is the process of extracting previously unknown, valid and actionable information to make crucial business decisions. [10]

The art is then to effectively use the available data and turn the data into information. This is normally done by implementing simple short rules [51], instead of complicated models. This in short is the essence of this chapter: to extract and model the required data to determine the losses in the network to as close a perimeter as possible by accurately presenting the information retrieved.

5.2 Methodology

A data mining solution can be broken up into three different components[10]:

- Data warehousing and collection.
- Data mining or modeling
- Presentation

5.2.1 Measurement and Data Collection

As explained in Section 4.4 in the reticulation environment, incoming and paid consumption is known. The following data points are available for evaluation:

- Incoming feeders from the utility
- Usage of bulk and conventional meters
- Consumption of prepaid meter users

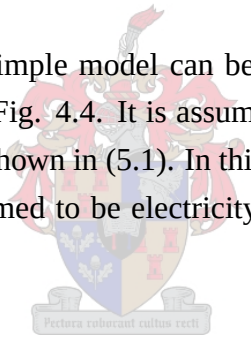
The data coming into the network from the utility is contained in the bills from the utility and is fed into the administration or financial system.

The consumed values for conventional metering and bulk metering is assumed to be stored in a debtor or administration system. These systems normally have some flat file export function or a direct database link via SQL (Structured Query Language). This means that these records can be accessed by external software.

The prepaid customers records are kept in a data base. From a site visit to Actaris in Cape Town, these systems are built on SQL servers and the data can be extracted using an SQL connection.

5.2.2 Modeling

With the data point identified a simple model can be built up using the power flow within a reticulation network as shown in Fig. 4.4. It is assumed that power is only consumed and not generated within this network as shown in (5.1). In this context E_{in} is the total power delivered to the network, $E_{consumed}$ is assumed to be electricity paid for, and E_{losses} includes technical and non-technical losses.



$$E_{in} = \sum E_{losses} + \sum E_{consumed} \tag{5.1}$$

In general, most distribution companies have statistical meters at substations or redistribution points on the medium voltage networks, covering some transformer area. Equation (5.1) can now be expanded.

$$E_{in} = \sum_{area=1}^{areas} E_{losses_{area}} + \sum_{area=1}^{areas} E_{consumed_{area}} \tag{5.2}$$

$$= \sum_{area=1}^{areas} (E_{losses_{area}} + E_{consumed_{area}}) \tag{5.3}$$

$$= \sum_{area=1}^{areas} E_{in_{area}} \tag{5.4}$$

where

$$E_{in_{area}} = E_{losses_{area}} + E_{consumed_{area}} \tag{5.5}$$

In the same manner $E_{in_{area}}$ can be broken down into smaller areas covered by a transformer or an area within a transformer area. This breaking up of the areas reveals the same pattern

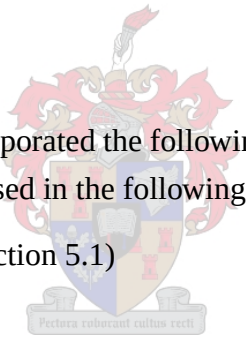
repeating itself recursively, for example, by comparing (5.1) with (5.5). This can easily be implemented into a tree structure as shown in Fig. 5.1 where the area is represented by a node with a relation or connection to a parent and child.

Each measurement point or area can be seen as a node within the network. The node is responsible for extracting information from the data warehouse. Each node is linked to a parent node and can have children nodes. The node is responsible for importing , pre-processing and formating the data for use by the rest of the software elements and nodes.

5.2.3 Presentation

To present the data, a graphical user interface is used. The nodes should be accessible and easy to use. From the modeling of the network a tree structure fits this representation criteria perfectly. Each node should represent the losses in the structure below it, in terms of its children or connected database. To determine trends the software should be capable of plotting the data contained in each node.

5.3 Design



The design of the application incorporated the following phases [10] in the data mining process into a layered design and is discussed in the following subsections:

1. Objective determination (Section 5.1)
2. Data preparation
3. Data mining
4. Analysis and presentation

The software is developed in the Java environment using the Java SDK [83] and Netbeans.org IDE [59] for the following reasons:

- Easy integration and re-use with already developed software classes
- Support and documentation on the internet
- Available classes on the internet

Objective determination

The objective of this application is to determine the losses in an area as effectively as possible, in order to detect abnormalities. These abnormalities will indicate either a highly inefficient system on the technical side or, as expected, an indication of electricity theft in the specific transformer area.

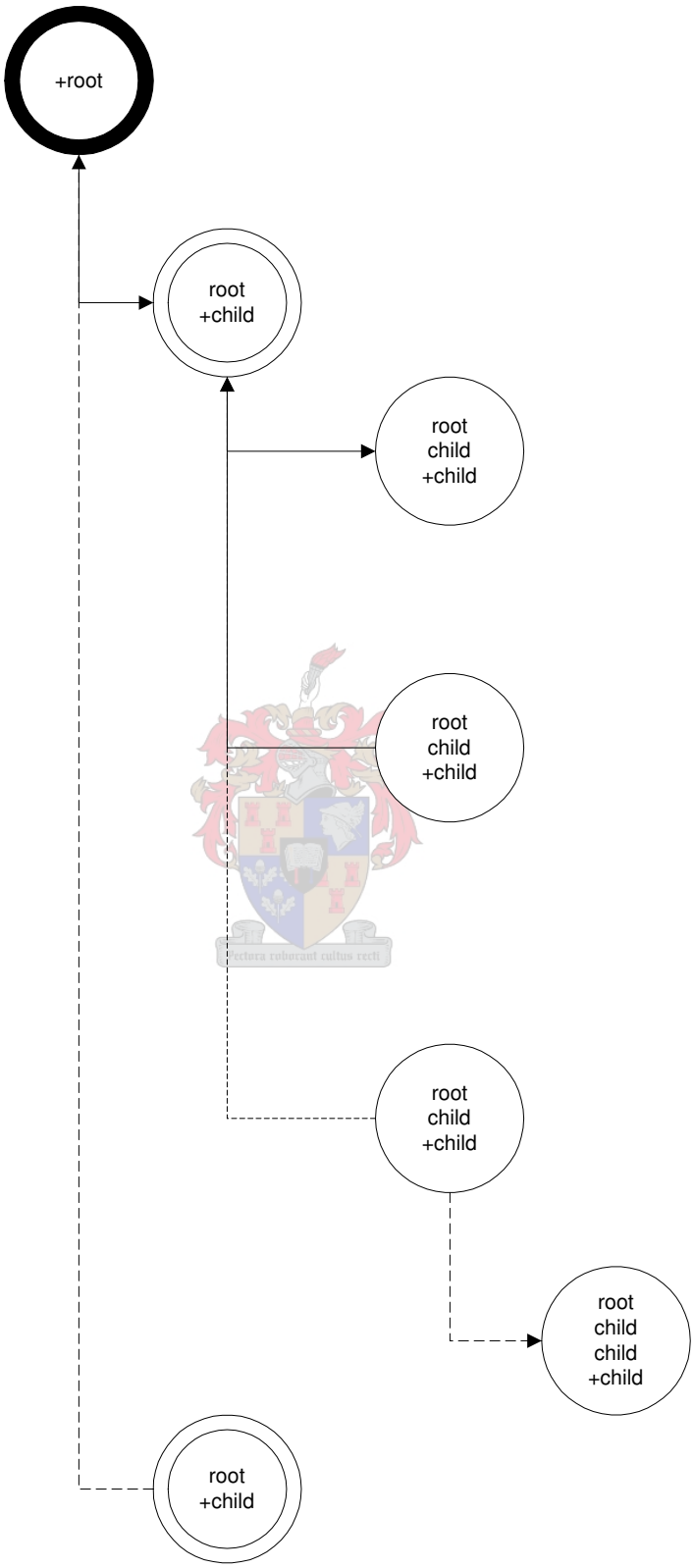


Figure 5.1: *Tree structure diagram where + indicates the depth in the structure*

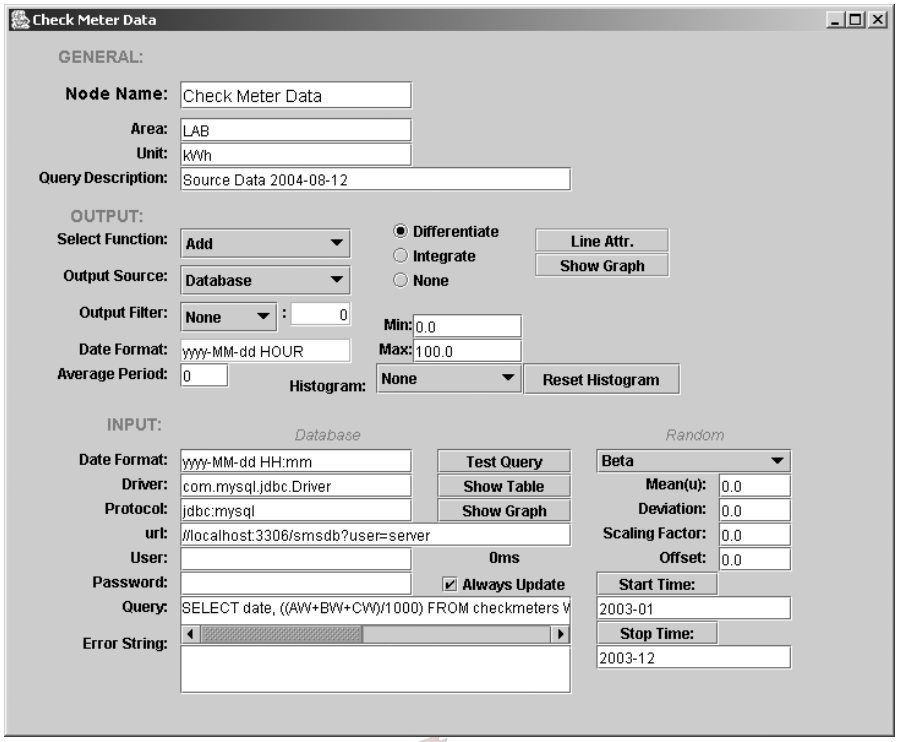


Figure 5.2: *Node Setup Screen Shot*

Data preparation

The data is available in various different forms. The data could be in a flat-file or preferably in a database with SQL capabilities. If the data is available in a database, the correct setup and drivers to access the database are required. Since there are so many available forms, it was decided to implement the software using a middleware client such as JDBC (Java Data Base Connectivity) [85]. To further broaden the scope of the possible data connections a JDBC-ODBC bridge is available to interconnect with ODBC (Open Data Base Connectivity) [94] driven databases.

The data retrieved from measurements in the field is sampled at extremely variable intervals. A grouping function should be implemented to scale the time axes to the relevant interval period, typically a monthly interval. Filtering and other mathematical functions such as integration and differentiation are also required, since the energy readers tend to accumulate their values from sample to sample.

The data preparation is configured in each node. A screen shot of the node setup is shown in Fig. 5.2.

For scenario planning, using the current data to sketch a situation under various circumstances can be extremely helpful, therefore two simulation functions are added to the data preparation part. A histogram function is added, to gather the data of several iterations and a random function generator is included, as shown in Fig. 5.2

An overview of how the data preparation is handled is shown in Fig. 5.3 and is implemented in `public dataCollection toDataCollection()` in Appendix Q.7.2.

Data mining

The data mining part is based on the area subdivision model developed in Section 5.2.2. Although very simple, the various nodes should have the capability to add and subtract the specific samples. As mentioned all the nodes are placed in a tree structure. This allows a routine to search through the tree structure recursively and calculate the value of each sample as specified by the node.

Presentation

Each node is identified in the tree with text. The text gives the name and the context of the node. The average of the total of the node is represented in the text as well as an indication of how efficient the node is with regards to its children. The data of each node can be plotted, therefore allowing a user to identify trends and to react to those trends. The plots give an idea of the current nodes data as well as the children. This screen shot can be seen in Fig. 5.4. For the user interface generation two mechanisms are used namely:

1. The tree structure
2. Data representation using graphs

The tree structure is represented by `javax.swing.JTree` [82] class provided in the standard Java classes.

The graph representation is created using the Scientific Graphical Toolkit provided by EPIC [23].

5.4 Implementation

5.4.1 Overview

The code for the application is held together and coordinated by the `DataMinor.myTree` class. An overview of all the interactions between the classes is given in Fig. 5.5.

5.4.2 Data Connectivity

To connect to the databases the `peg.sql.DBConnect` class is developed. This class uses the JDBC API. JDBC uses drivers provided by the database vendors. These drivers are available either from the vendor's website or from [85].

To setup the database the following parameters are required:

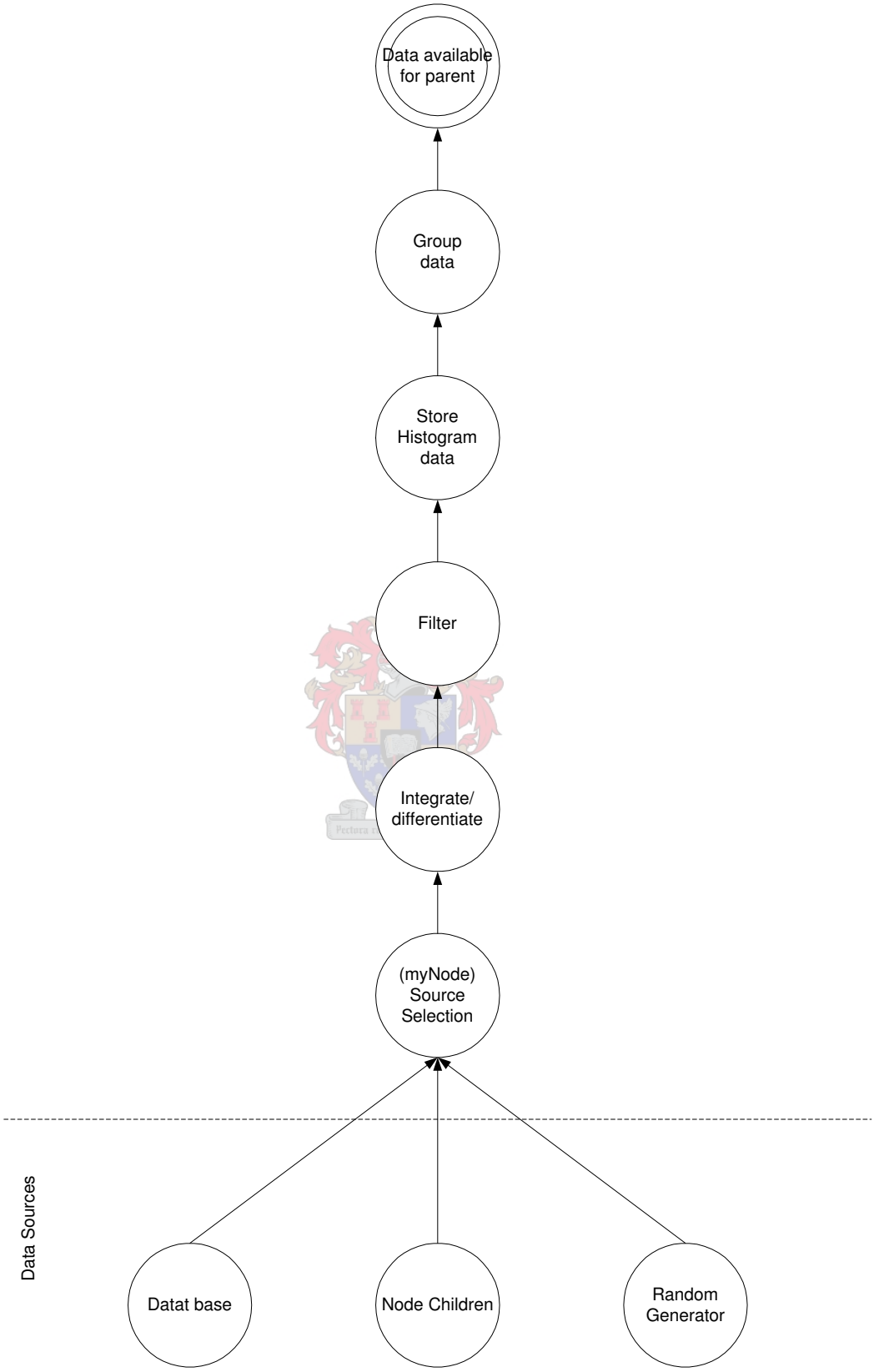


Figure 5.3: *Data Preparation Flow*

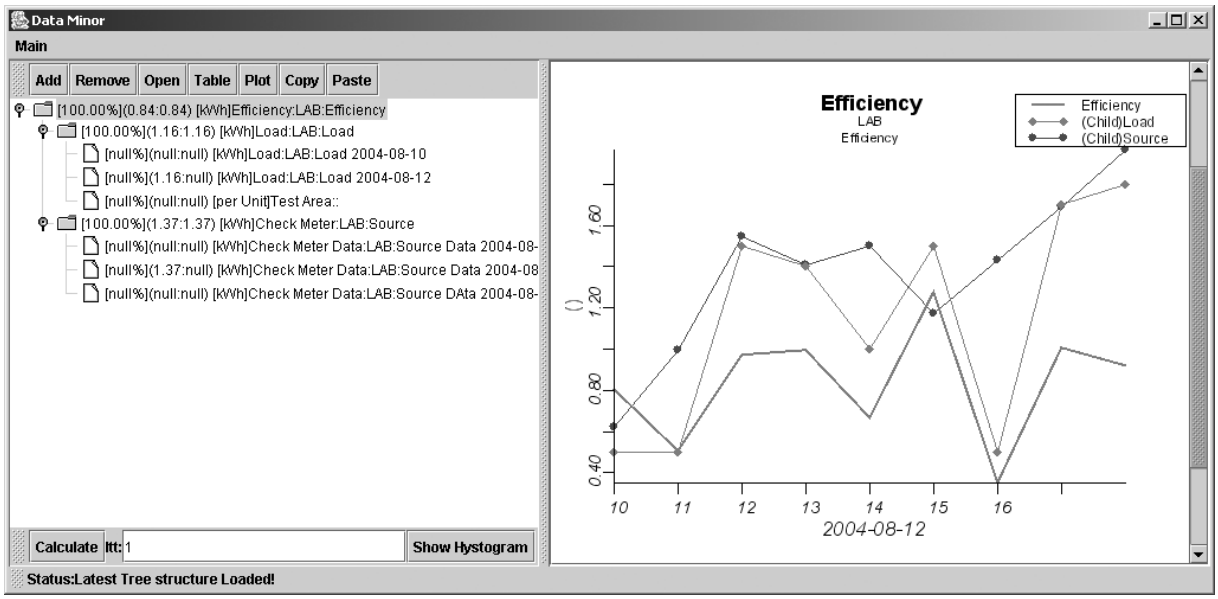


Figure 5.4: *Screen Shot of Application*

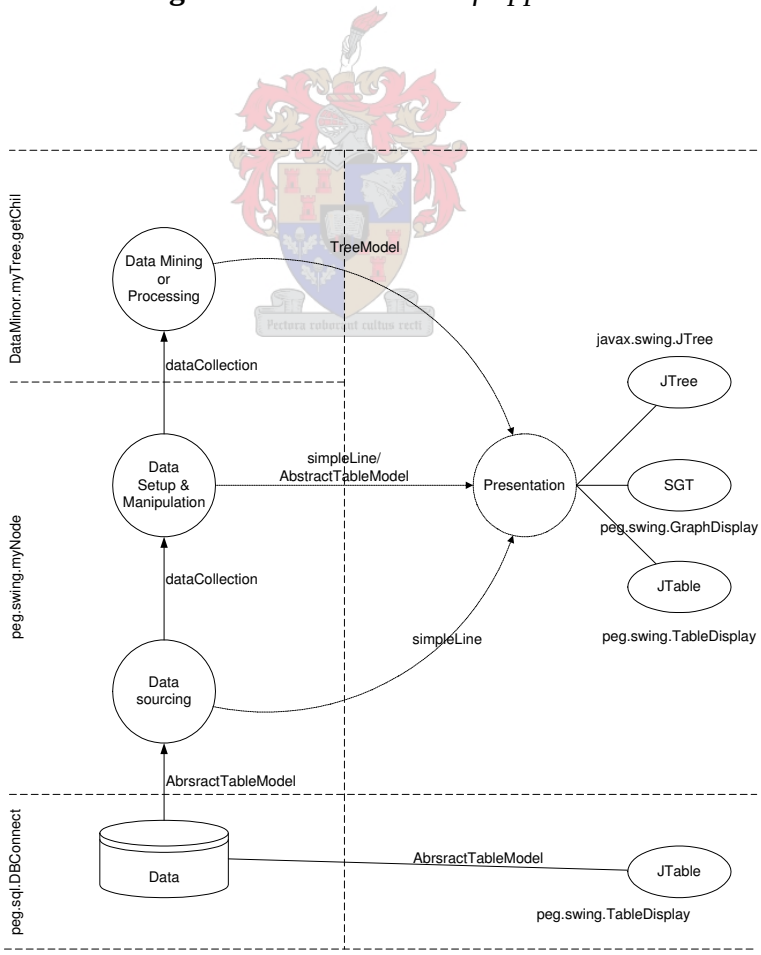


Figure 5.5: *Software Overview*

- the driver class, for example `com.mysql.jdbc.Driver`
- a protocol, for example `jdbc:mysql`
- a connection url to locate the database on the network or PC, such as `//localhost:3306/smsdb?user=server`
- a username, for example server[1]
- a password, such as somepassword[2]
- the query to be sent to the database in order to extract the required information.

This class interfaces to other classes using the `javax.swing.table.AbstractTableModel`. This model can be used to present the data in a table format and can easily be accessed by other classes.

5.4.3 Nodes

The node is represented by `peg.swing.myNode` class. The class is responsible for retrieving the data from the `peg.sql.DBConnect` class and preparing the data for processing in the rest of the software.

The `myNode` class relies on the `peg.utils.dataCollection` class to store the data and handle all the conversions and manipulations of the data. This class keeps the data in a `Vector()` of data-points. The `dataPoints` consist of the time and value of each point. The `dataPoints` can be manipulated in this environment as indicated by the functions in Fig. 5.6.

The data is retrieved from the node using the `toDataCollection()` method in the `myNode` class. The logic of this method is shown in the flowchart in Fig. 5.7.

The node is also responsible for the random processes. These random numbers are generated using classes provided by [71]. The node also keeps track of all the passed values in a `peg.utils.hystogram`(see Appendix Q.8.4) class. This can provide simulation functionality as seen in Section 8.2.

5.4.4 Data Calculation

Each node has the capability of storing the result of all the children's values as a `dataCollection` object. This is used by the recursive procedure to calculate the various values of each of the nodes, which are shown in Fig. 5.8.

---

[1] mySQL requires the user name to be in the connection url.

[2] mySQL requires the password to be in the connection url.

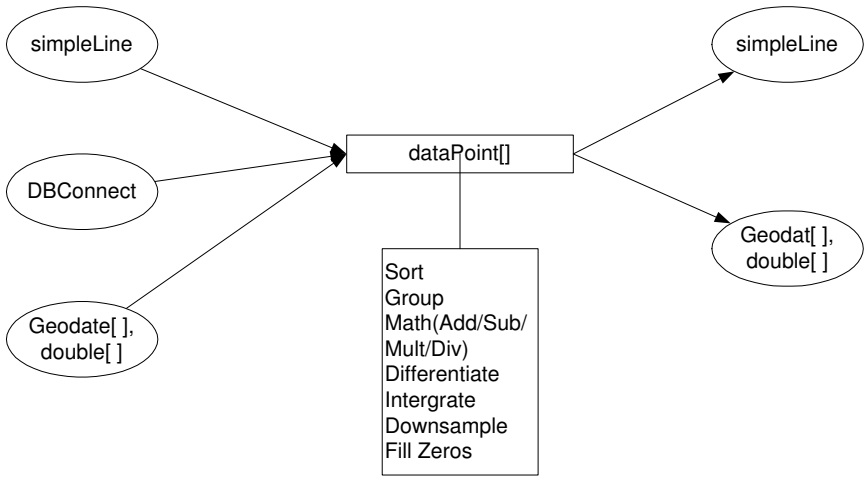


Figure 5.6: *dataCollection Class Functional Overview*

5.4.5 Presentation

The presentation part is implemented using the `javax.swing` classes available in the J2SE [81].

The `JTree` class keeps all the data together. From the `JTree` all the nodes are accessed and controlled, for example getting data and plotting the data and tables.

Two extra classes are developed to extend the `JPanel` class and can be imported into the forms designed in the Netbeans environment.

The `GraphDisplay` class relies completely on the Scientific Graphic toolkit, and provides a simple interface for the `myTree` class. The `TableDisplay` class is written with the same idea, but implements the JTable to display all data extracted from the database associated with a node.

Plotting the data for the user is initiated by pressing the PLOT button on the user interface. Depending on the type of node and how the node acquires its data the plot is performed according to Fig. 5.9.

5.5 Summary

This chapter develops software to assist managers and distribution personnel by providing a tool to determine the losses in transformer areas. By intelligently placing check meters and adapting the consumed data within the node, it is possible to zoom in and identify the cause of the high losses in the area.

By adding random processes in the nodes simulation capability is provided and combined with available data. From this models can be generated to aid in scenario planning. This can be used prior to developing an area to simulate possible network layouts. With an already existing network the expansion possibilities can be investigated.

The node analysis currently being done is based on comparing the averages of the various

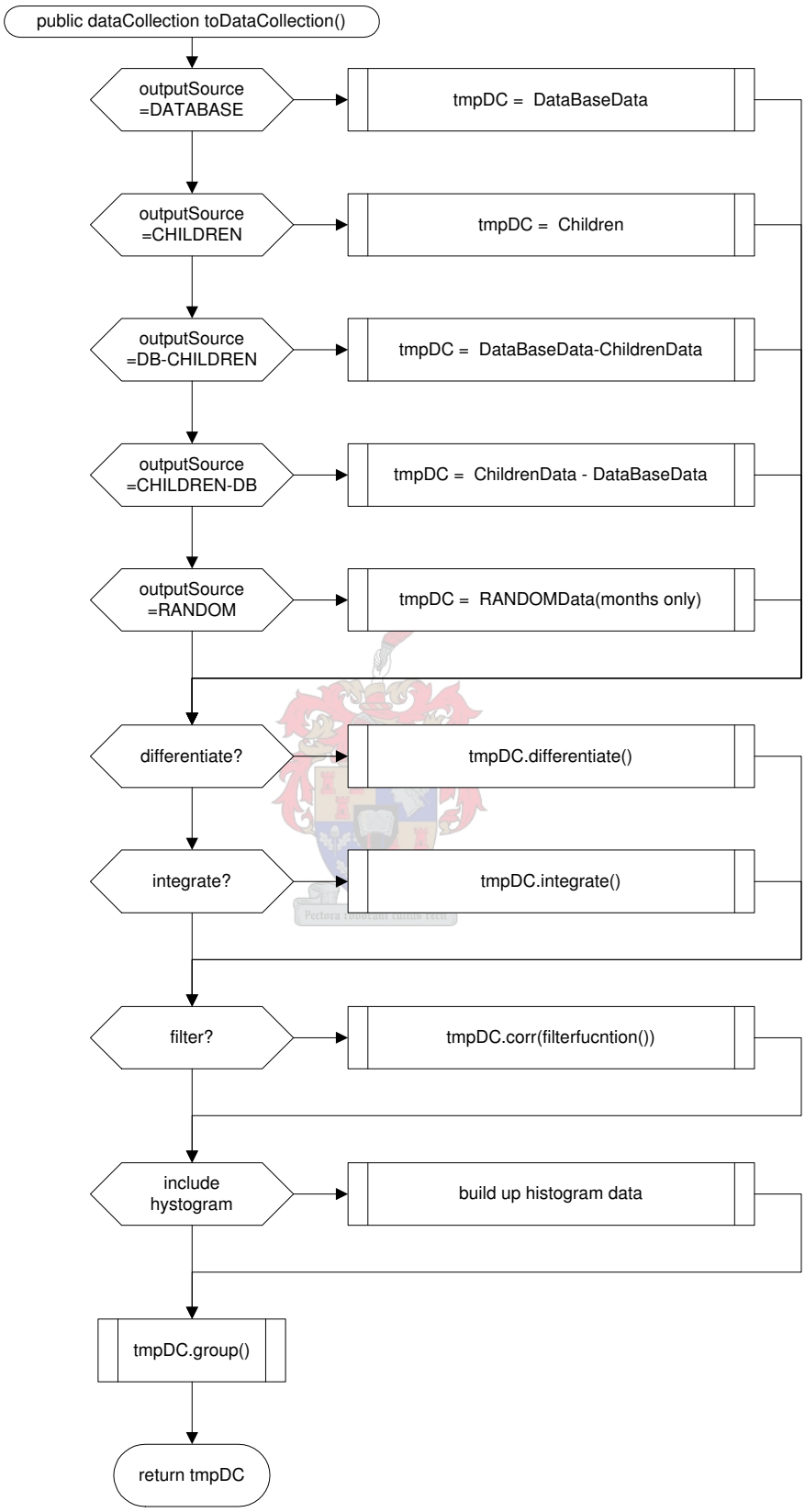


Figure 5.7: *Node Data Output Program Flow*

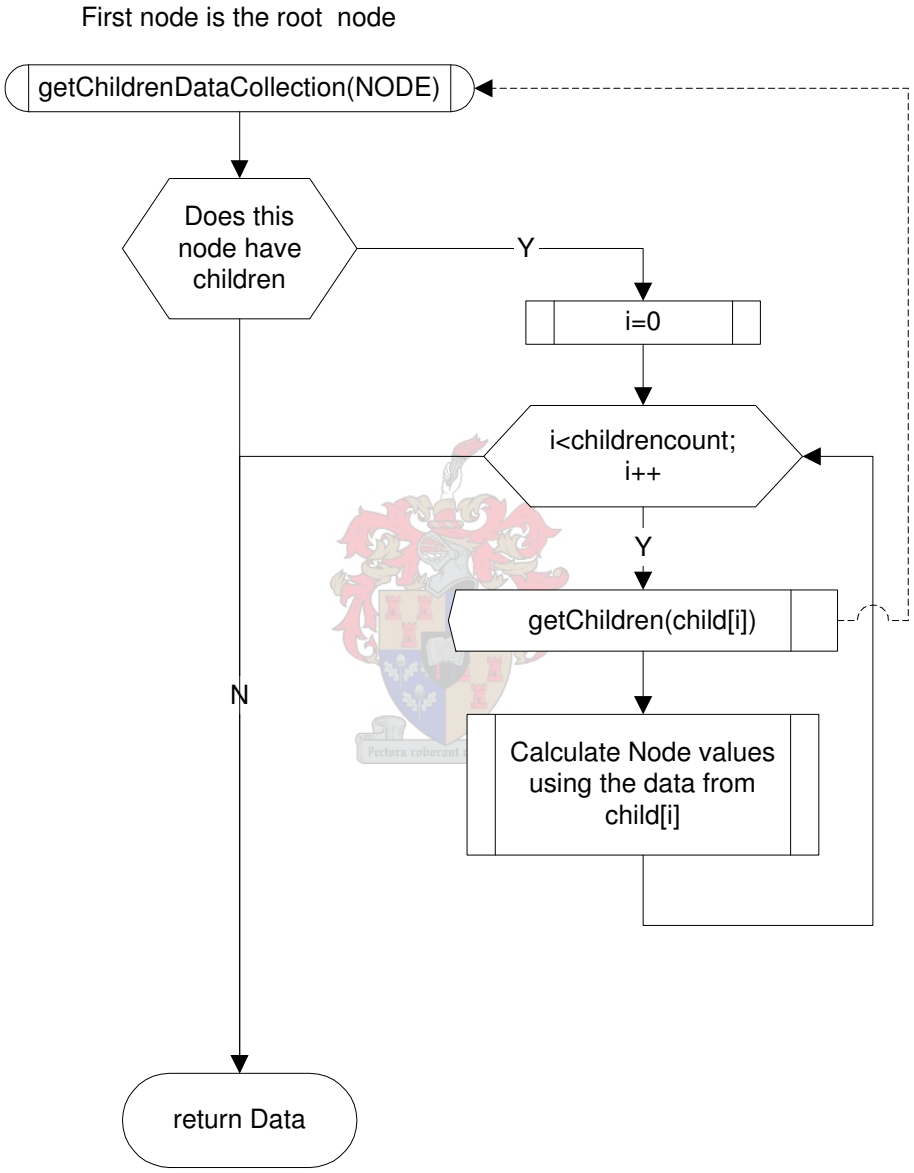


Figure 5.8: *Recursive Node Calculation Procedure*

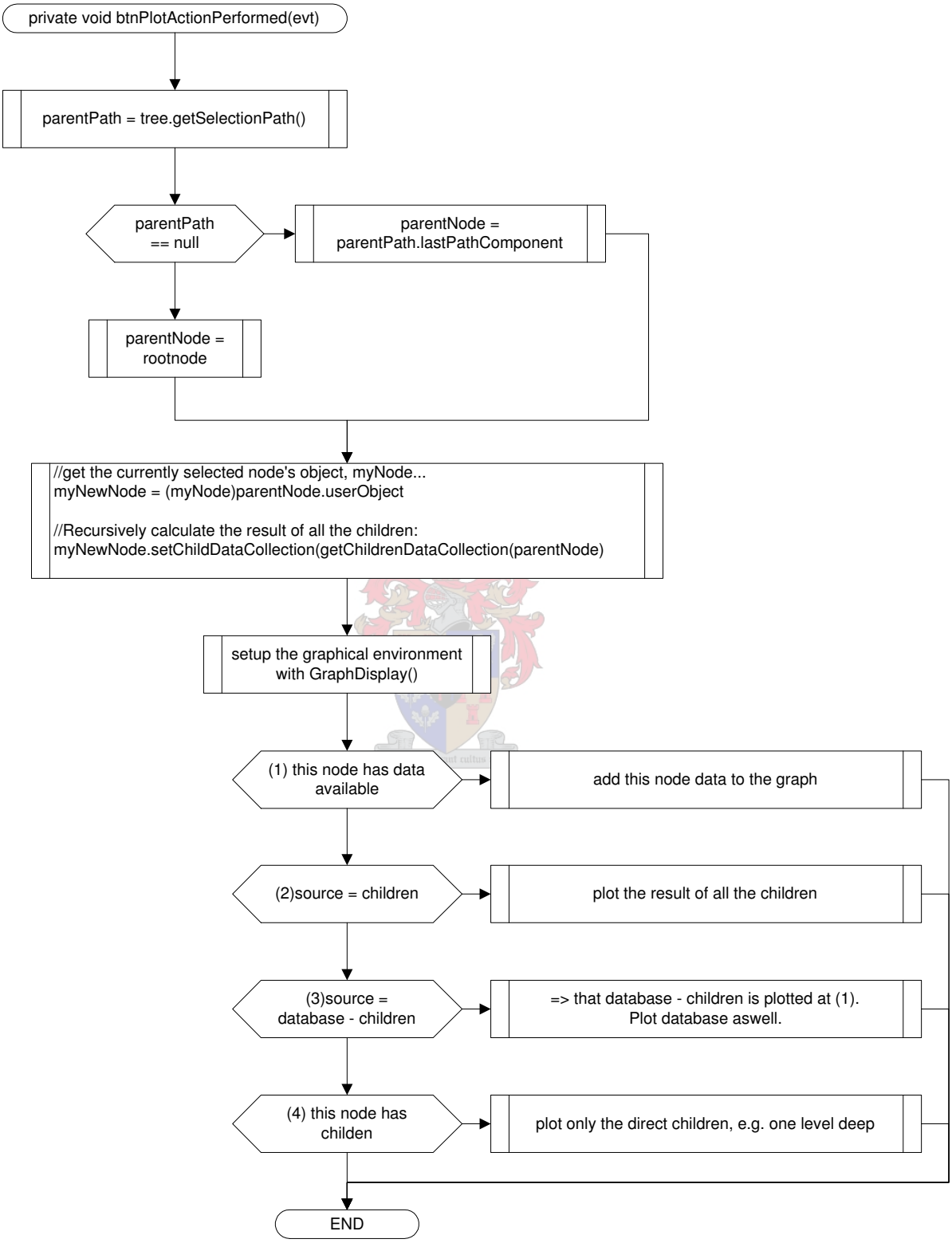


Figure 5.9: *Flow diagram of plotting functionality*

children and data of the node. Implementing statistical indicators and the bootstrap [8] paired samples analysis method in each node can further expand the capability of the software. Incorporating the work of Fourie [32] could also prove extremely useful.

This package is also in line with the performance criteria as set out in the Municipal Act 2000 for municipalities [74], whereby an interface is provided to track targets and monitor the systems performance. The software package can easily be used and integrated to aid other performance criteria applications such as water losses.

Chapter 6

Mobile Remote Check Meter Design

6.1 Introduction

In Section 4.4 it is seen that quantifying electricity theft is difficult and in the information chain, a gap exists here. Cloete [14] achieved excellent results using previously installed conventional meters to check the new pre-paid consumers and hence applied the check meter principle successfully.

Populating a total LV reticulation network with new check meters is a costly exercise. By isolating different areas within the reticulation network and installing check meters, losses can be determined based on the Theory of Constraints. In order to reduce the technical losses the check meter is placed at the LV side of the transformer, instead of the MV side, which eliminates the technical losses due to transformer inefficiencies as well as the need for CT's and VT's. Applying the ideas in Chapter 4 using a check meter, it is now possible to manage losses in a specific area. From the simulations done by [20] and in Section 8.2, it seems feasible to implement this strategy and the results generated justified looking deeper into the implementation of a remote check meter.

With new advancements in communications and measurement technologies, a decision was made to research the design of a low cost, highly configurable, weatherproof, low maintenance automatic remote check meter that is also easy to implement.

6.2 System specification, selection and overview

For the purpose of checking the energy flow into an area the system should consist of the following:

Checkmeter A three phase meter

Data Warehouse Some database server

A data connection A link of some kind to transfer the data from the check-meter to the data warehouse.

For the purpose of the study, this check-meter solution should comply with the following specifcations:

3ϕ **Y** Measure three phase power on a LV reticulation network, which implies $V_{LN} = 230V$, $\pm 10\ \%$, with an added $10\ \%$ for contingency the design should at least be capable of handling $V_{LN} \approx 280V$.

800kVA Should be able to measure a current of up to $I_\phi = 1160A$

Weatherproof The device should be able to operate in an all-weather environment.

Low Cost As with all research, the cost associated with the device should be as low as possible. This cost should include maintenance and running costs.

Accuracy For this specific function, namely of comparing its measurements against other measurements monitored at various arbitrary intervals, an accuracy within 1% to 2% should be sufficient. This accuracy is in line with a Class 2 meter specification.

Remote Operation The meter should be able to operate without any form of human interaction for the lifetime of the meter.

Flexibility The solution should form a platform where researchers can use and change devices in software as required.

Ease of use The system should be easy to install

With these specifications a comparison was made with other available products. Table 6.1 shows a comparison of available products against the criteria as set out in the previous paragraph [1].

Comparing the available products with the requirements that are necessary revealed that it would be worthwhile designing a new system and thereby take advantage of newly developed components and reducing the total system cost[2]. With the design of such a meter in combination with a data software package as described in Chapter 5 a flexible platform for future in-house research will be developed. To ensure a fully integrated platform and solution, an overview of the proposed system is showed in Fig. 6.1.

[1] Prices determined August 2004

[2] The costs of a potential meter at this stage does not include R&D or manufacturing costs.

Criterea	Actaris SL7000[4, 3]	Elite Pro[16]	EnerMax[79]
Voltage Range	57-230V	600V	380V
Current Range	5A or CT	CT dep	100A or CT
Weather Proof	No info	No info	Indoor appl.
Cost	Not available	R10793	R8798
Accuracy	Class 1	1%	Class 1
Remote Comms.	via RS232 RS 485 Infra Red DLMS-Cosem	RS232	RS232 or RS485
Modem included	No	No	Yes
Data logging	Energy and tariff registers	128k 25000 readings	160k 300days at 30 min
Flexibility	Proprietary products and software req.	Easy data imports e.g. excel SQL	Exports data to flat files
Installation	Fair	Very Easy	Fair
Data Solution	Eclipse Not available	ELog R1628	E-Man LU R 2000

Table 6.1: *Overview of available technologies*

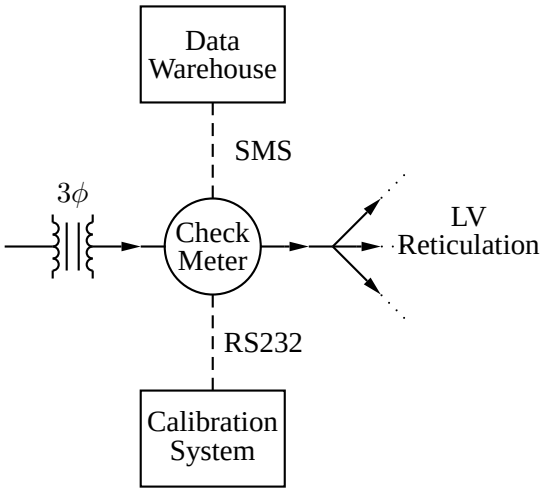


Figure 6.1: *System Overview of Check Meter*

6.3 Hardware Design

The check-meter consists of several subsystems: an energy measurement component, a controller unit, a power supply and a telemetry system. The design and construction is discussed in this section and the schematics, printed circuit board layouts and component list can be found in Appendix L. The total cost is roughly estimated at $R\ 4\ 700$ excluding the current transformers.

6.3.1 System Overview

The topology chosen is based on the ADE7758 energy measurement chip from Analog Devices and a micro controller, MSP430F149 from Texas Instruments as discussed in Section 6.3.2. Both are low power, low cost chips with almost no additional external active components required. The ADE7758 is capable of measuring active energy with true RMS up to 14kHz with a error of less than 0.1%[7]. To simplify the design a Traco power supply was chosen. The telemetry is done with a GSM[3] modem. The WISMO M2106B Integra modem is compatible with other vendor's modems, for example Falcom, which makes sense for production purposes. The system layout of the hardware is presented in Fig. 6.2.

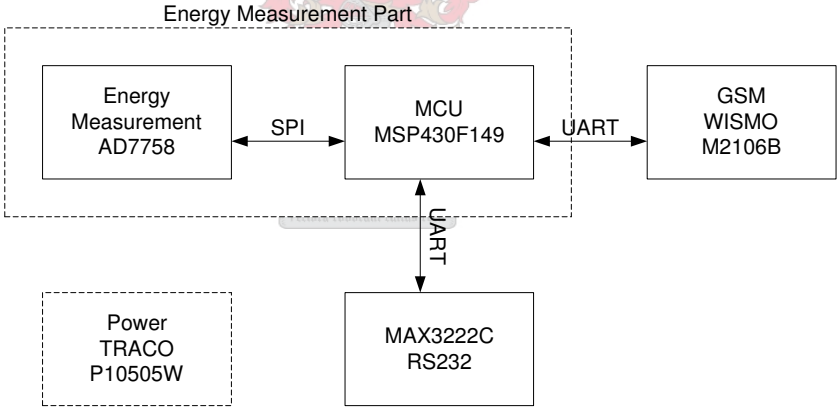


Figure 6.2: *Component Overview of Check-Meter*

6.3.2 Energy Measurement

DSP and ADE7758 evalutation

For the energy measurement two options were evaluated: a DSP based and pre-programmed energy measurement chip based solution.

Using a DSP, the energy measurement and controller unit can be implemented and combined using software. However, a DSP requires an amount of external components, for example

[3]Global System for Mobile Communications

level shifters, and requires commercial compilers. A DSP based design will require extensive software development, since the energy measurement system has to be built up from scratch in the software. The DSP would have been more flexible in terms of calibration methods, such as using a lookup table instead of two-point calibration to increase accuracy.

Analog devices released a new energy measurement chip, the ADE7758. The ADE7758 can be interfaced via a SPI bus to a Micro Controller Unit (MCU). This MCU should then handle the interfacing to different components. The ADE7758 does not require any additional active components due to the differential inputs which use a resistor divider network for voltage conditioning.

A quick overview of the costs in terms of the proposed components is shown in Table 6.2. The costs are based on a six channel measurement system. Prices available from the various companies [5, 91] on 17 August 2004.

Component	DSP 1	DSP 2	ADE7758
TMS320F240	$13.00	$13.00	
AD620(x6)	$18.84		
AD7715		$10.30	
ADE7758			$6.88
MSP430F1491			$5.60
Total	$31.84	$23.30	$12.48

Table 6.2: *Evaluation of different energy measurement strategies*

The following factors influenced the selection of the ADE7758 instead of a DSP:

- Accuracy
- Lower power consumption
- Lower cost (see Table 6.2)
- Calculates all the desired energy values required for the application
- Easy interfacing possible using SPI bus
- No level shifting required, hence no additional active components
- Simple and quick calibration process

ADE7758

Inside the ADE7758, the data acquisition is done by differential input analog to digital converters. These inputs require an input signal smaller than $500\ mV_{peak}$. Simple voltage divider

networks for both the current and voltage inputs are used with a capacitor added over the input to act as an anti-aliasing filter. For the current sensors an additional $100\ nF$ capacitor is added to increase the phase lead.

In order to protect the input stage of the ADE7758, two diodes are placed parallel, each in the opposite direction, to the inputs of the converters. This is to ensure that the inputs are not driven extensively past their desired input range of $\pm\ 500\ mV_{peak}$. Metal Oxide Varistors capable of $275\ V_{RMS}$ are placed parallel to the voltage measurement inputs, prior to the voltage dividing network. This is to ensure that spikes are damped and the device is operated within limits. The voltage measuring probes are fitted with $500\ mA$ fuses to protect the device.

Two LEDs were connected to the chip to help with calibration. The chip is driven by a $10\ MHz$ crystal. The SPI bus voltage levels are $5\ V$ TTL. A dual voltage buffer is used to interface with the $3.3\ V$ logic of the micro controller.

The circuit diagram of the energy measurement system is shown in Fig. 6.3.

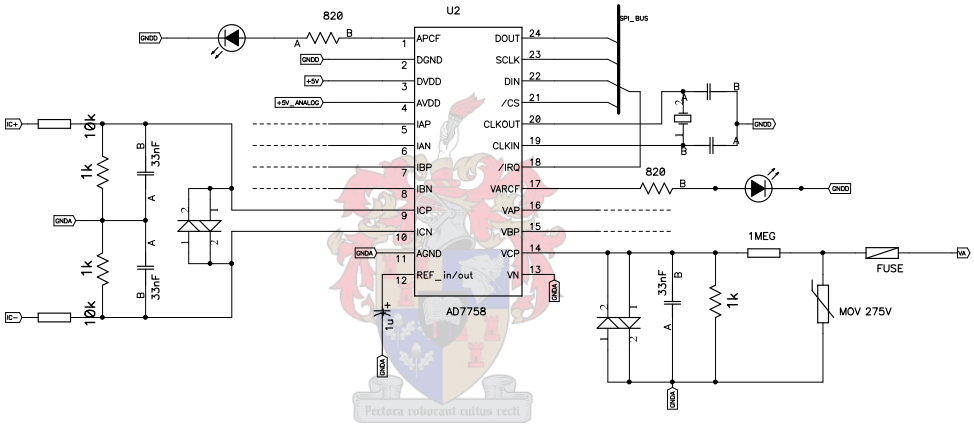


Figure 6.3: *Energy Measurement Configuration*

Signal Conditioning

Voltage The ADE7758 has software selectable input ranges for example $\pm 500\ mV_{peak}$, $\pm 250\ mV_{peak}$ and $\pm 125\ mV_{peak}$. The nominal measurement voltage should be around $V_{LN} = 230\ V_{RMS} = 325\ V_{peak}$. To keep things simple, it was decided to design the hardware for a nominal measurement voltage of between $\pm 250\ mV_{peak}$ and $\pm 500\ mV_{peak}$ input, which allows for measurement of both higher and lower voltages. A simple resistor divider network is used, as shown in Fig. L.3.

For $R_2 = 1\ k\Omega$ is chosen. R_1 is then calculated:

$$R_1 = R_2 \frac{V_{in} - V_{out}}{V_{out}} \tag{6.1}$$

$$= 1000 \frac{325 - 0.375}{0.375} \tag{6.2}$$

$$\approx 870\ k\Omega \tag{6.3}$$

In order to keep the values simple, a resistor value of $R_1 = 1\ M\Omega$ was chosen. This implies that with $V_{LN} = 327\ V_{peak}$, the voltage at the ADE7758 will measure $325\ mV_{peak}$, which is in the acceptable range of operation.

Since the voltage over R_2 is negligibly small, the selected resistor R_1 should be able to handle all voltage that is put over the terminals. The selected Vishay HTS 68 resistor can handle a voltage of $2\ kV$ over its terminals.

When measuring[4] the capacitance of the $1\ M\Omega$ resistor is $0.02\ pF$ and with $C_2 = \frac{R_2}{R_1}C_1$ results in $0.2\ pF$, which is negligibly small.

Current The current sensing range and capability are conventionally associated with the CT ratios. The selection of the sensing device is, therefore, important for the design of the check-meter. For this specific application a clip-on CT or probe should be sufficient since no DC offset should be present. The selected technology should be able to allow for a wide variety of current specifications. The device should preferably be able to operate in an all weather conditions. Cost should also be considered. The various options are considered and summarized in Table 6.3 for $I_{nominal} = 1000\ A$.

Criteria	Conventional CT	Typical LEM	Current Probes
Clamp On	Wide range	No	Yes
Maximum Current [A]	w	1200	10-1000
Safety	Fair	Good	Excellent
Voltage Output	No	Yes	Yes
External Power	No	Yes	No
Cost	R1800*	R1805**	1900*
Model		LT 505-S	

Table 6.3: *Evaluation of different current measurement solutions.* **June/August 2004,* ***September 2003*

For current sensing, LEM current probes were used with voltage output instead of conventional CT's. The main advantage is that the voltage output reduces the need for a burden resistor. For safe operation, when clamping the current sensor over the live conductor, this implementation also reduces the risk of electric shock when no burden resistor is connected to a conventional CT. For this application it is only required to measure the AC component, due to the connection at the transformer. The LEM, however, has a $3\ ^o$ phase shift. This is corrected by putting a $100\ nF$ capacitor parallel to the conditioning network.

[4]@ $75\ kHz$, tested 23/08/2004

The current inputs on the ADE7758 can also be selected for inputs ranging similarly to the voltage inputs. For the prototype the laboratory current probes were used mainly as a consequence of availability. The specific current probes, LEM M2, measure a nominal input current of $20\ A$ and a conversion ratio of $output = \frac{5\ V}{20\ A} = 250\ \frac{mV}{A}$.The other available probes similar to these provide an $output = 10\ \frac{mV}{A}$ and $1\ \frac{mV}{A}$. This equates to a maximum output of $V_{out} = 4.242\ V_{peak}$ when the $200\ A$ LEM probes are used. Considering that the ranges are software selectable, the a resistive network with $R_2 = 1\ k\Omega$ and $R_1 = 10\ k\Omega$ is used. The capacitance difference between these two resistors is relatively small. With a $V_{out} = 4.242\ V_{peak}$ the input voltage into the ADE7758 would be $V_{in} = 385\ mV_{peak}$.

Anti-aliasing Filter In order to prevent aliasing a capacitor is placed parallel to R_2. The ADE7758 samples at

$$f_{sample} = f_{CLKIN}/128 = 10.10^6/128 = 78.1\ kSPS$$

[7]. This means that the anti-aliasing filter should reject frequencies higher than $44\ kHz$. However, the evaluation board design [6] suggests a cut-off frequency of $4.8\ kHz$, which requires that the RC network should consist of a $33\ nF$ capacitor parallel to $R_2 = 1\ k\Omega$.

Over voltage protection

The check-meter should be able to operate safely at $V_{LN} \approx 280V$ (see Section 6.2). To ensure safe operation and protection against spikes and short over voltage bursts, two measures are taken to protect the circuitry:

1. The voltage channels are fitted with MOVs and the input leads are fitted with fuses.
2. In order to protect the Analog to Digital converters of the ADE7758, two diodes are placed parallel to each other, in opposite directions, to ensure the voltage may never exceed $\pm 0.7\ V_{peak}$.

The Metal Oxide Varistors (MOV) are placed across the input channels of the check-meter to ensure that energy in spikes is dissipated and the voltage is clamped. This should cause an increase in current flowing into the device. If the current exceeds $500\ mA$, the fuses placed in series with the voltage measurement channels should blow. From the data sheets, the fuses will blow when a constant voltage of $V_{rep} = 580.0V$ is placed over the lines due to the current flowing through the MOVs. This value implies that measurements will not be affected by the voltage clamping in the operating range, but will instead protect the device against over voltages and spikes.

The energy measurement and conditioning circuits should be able to handle the constant voltage at $V_{LN} = 580\ V_{RMS}$ but the power supply will not operate in this region, since it is specified at $V_{LN} = 264\ V_{RMS}$ ans can handle $1\ kV$ surges.

The fuses are placed into the probes used for the clipping onto the measurement points.

The selected MOVs have proved [5] to protect the circuit during testing.

6.3.3 Micro Controller

Since most of the calculations are done by the ADE7758 only a micro controller is required. Various controllers are on the market, but it was decided that the MSP430F149 will be more than sufficient for the purpose of interfacing the energy measurement chip with the communications components. The micro controller was chosen due to the following features:

- Low power
- 2 available UARTS
- SPI interface capability on one of the UARTs
- Ample capacity for future expansions
- Low cost
- In-system programming
- Free compilers available
- Good support on internet forums [72]
- Availability

Clock System

The clock frequency selected for the MCU is $f_{xtal} = 7.372\ MHz$. This crystal, XTAL, frequency is the highest available multiple of 9600, which increase the reliability of inter device communications.

XTAL1 is configured to run, instead of the internal clock of $32\ kHz$[87].

The following code is implemented to start the clock correctly:

```
unsigned int j;
BCSCTL1 |= XTS;                       // ACLK = LFXT1 = HF XTAL

do {
  IFG1 &= ~OFIFG;                     // Clear OSCFault flag
  for (j = 0xFF; j > 0; j--);         // Time for flag to set
} while ((IFG1 & OFIFG) == OFIFG);

BCSCTL2 |= SELM1+SELM0;               // MCLK = LFXT1 (safe)
```

[5] An over voltage was generated on 2004/08/03 which caused one of the MOVs to blow. The MOV was replaced and the system was working as normal again. Unfortunately at that stage of testing the fuses were not in place.

SPI Interface

The first UART is configured as an SPI bus to communicate with the ADE7758. The micro controller operates on $3.3\,V$ logic and hence, has to interface via a $3.3\,V$ to $5\,V$ bus transceiver. The transceivers, SN74LVC4245A from Texas Instruments, are able to allow signals to travel in both directions [88]. Two transceivers were used and hard wired to allow for up and down conversion of the various control and data lines.

MAX3222C

The second UART is shared between the GSM modem and normal RS232 communications. The RS232 is used for both debugging and calibration purposes. The RS232 must be operated via an electrically isolated link to prevent damage to devices due to different ground potentials.

For communications with a PC, a RS232 line driver is used. Since two different devices, the GSM modem and the PC, operate on the same UART-bus from the MCU, a line driver capable of going into high a impedance state is required, to prevent the possibility of the different devices driving each other. The RS232 line driver is only required when calibrating the device. For the rest of the time the device is switched off. While the line driver is switched on the GSM modem can be shut down and is put in a high impedance state.

6.3.4 Mobile Communications

The GSM modem is a Wismo Integra M2106B from Wavecom. This unit was selected due to availability and the compatibility between other vendors. It also offers good value for money for this specific application. The modem is only required to send SMSs[6] to the server. Therefore, a GSM instead of GPRS solution was selected. An easy AT command interface was used to communicate with the modem. In standby mode the unit consumes a low amount of power, and since only SMSs are sent, the transmission power used is minimal.

6.3.5 Power

To provide power to the system, a TRACO TML10505 module is used. This module is capable of supplying $2\,A$ at $5\,V$. Two capacitors are used to reduce the ripple on the $5\,V$ output. This is ample current to supply the various devices, including the GSM modem [96] at the full power of $1\,A$.

A 3.3V voltage regulator [78] for ST Microelectronics was used to supply the 3.3V circuit with power. This regulator is rated for a maximum of 1.5A.

---

[6]"Short for Short Message Service. Similar to paging. SMS is a service for sending short text messages to mobile phones" [94]

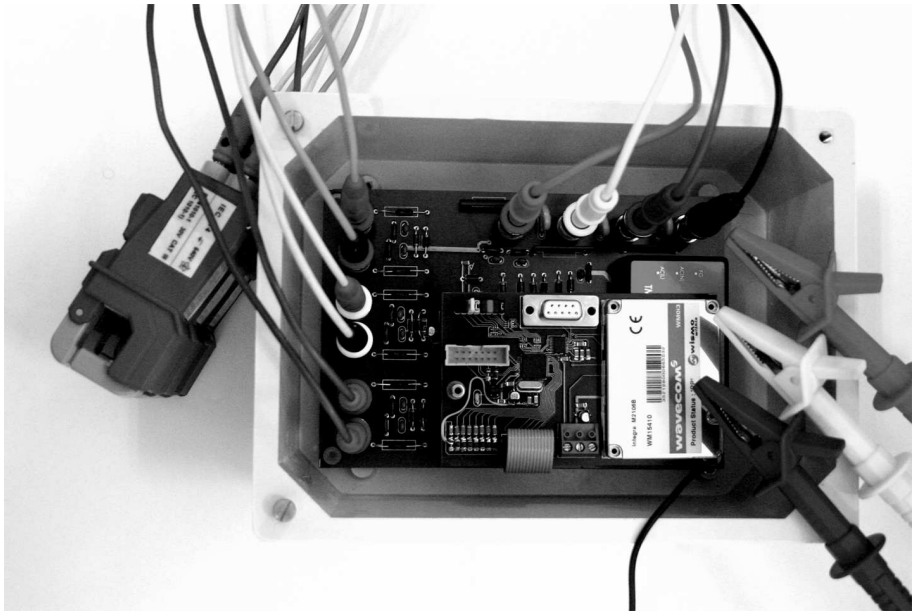


Figure 6.4: *Check Meter Prototype*

6.3.6 Construction

In order for the check-meter to function in a real-world environment, some thought must be given to the physical implementation of a possible final prototype, although not yet tested in the field.

The prototype is constructed on two stacked PCB's. The unit is placed in a weatherproof container to allow for all-weather installation. See Fig. 6.4. Because of its proposed use in rural areas, the container can be pole mounted as well.

Enclosure

The check meter is constructed using an ABB J2-S enclosure with dimensions of $200 \times 145 \times 80\ mm$. This console is specifically selected for its ability to be mounted on a pole, in an all-weather environment. An opening should be made at the bottom of the console and closed off with some cladding to ensure that water does not leak or spatter into the console in extreme weather. The exact cladding is not yet decided on.

Printed Circuit Boards

The check meter consists of two separate Printed Circuit Boards (PCB), which are assembled in two layers on top of each other. The constructed PCB's are shown in Fig. 6.5.

These should be mounted on a mounting plate supplied for this Voltex console.

Figure 6.5: *Layers PCB stack*

Probes

The probes are connected to the board via $4\ mm$ safety sockets, allowing for $32\ A$ and $8.2\ kV$ isolation without flash-over. The probes selected for the assembly are fused with $500\ mA$ ($50\ kA$, $1\ 000\ V$) and rated for $600\ V$ [66].

6.3.7 Connection and Installation

Before any connections are made, the box should be mounted onto the pole or safely fixed near the transformer or connection points. The leads coming out of the check-meter should hang down, to prevent water leakage into the meter. Depending on the intended use, it is suggested to strap the check-meter onto the pole using $10mm$ zip-ties or with brackets, to allow for easy removal.

To connect the device electrically to the network the following steps are suggested:

1. Connect the current probes to the check-meter.
2. Clamp each current probe over the conductor.
3. Connect the neutral voltage probe to the check-meter and connect it to the network. Take care to connect it to the neutral and not any of the other cables. This is important to ensure no damage to the device and to ensure that the correct measurement is made.
4. Connect each phase separately, firstly by connecting the probe to the check meter and then to the power grid. While connecting the probe to the transformer or the line, ensure that no physical contact is made by the user to the clamps of the probe to reduce the risk of electric shock.
5. After completing the electrical connections, ensure that the leads at the bottom of the console are wrapped with some form of cladding.
6. Ensure that the LEDs are flickering with in the correct sequence.
7. Close the lid of the check-meter and ensure its position by tightening the screws provided with the enclosure.

6.4 Check Meter Software Design

Each of the various components, except for the power supply as shown in Fig. 6.2, are coordinated by the MCU using software. The different component software is discussed in this section. Firstly, the design philosophy will be discussed. The state-event machine implementation is explained further in the section hereafter. The section following the state-event explanation will concentrate on some of the higher level components, such as calibration and GSM software.

6.4.1 Check Meter Software Design Philosophy

The check-meter is designed to run and operate in remote areas and should have minimal human contact for operation. The check-meter is an integral part of the data collection process and reliable data is required from this device. The software must, therefore, be extremely stable and reliable.

For this purpose the design was based on a state-event driven system. For this approach the following is required:

- No infinite loops may be present, except for one main loop in the software.
- The total state-event cycle time should be fast enough to allow for reliable information capture from the fastest device whose data can possibly be lost.
- The state-event machine should run in a stable environment.

The software is further broken up into different layers and parts. These are run as drivers which service the periphery and interface with other drivers. Each driver is programmed as a state-event machine. Events are either triggered by devices or calls to the drivers. The calls set specific flags which tell the state-event machine to start a task or process. The various levels, and their subsystems are outlined as follows:

GlobalHardware Ensures that the basic hardware is setup correctly for the MCU to ensure correct instruction execution.

- `main.c` The main application coordinating and implementing the system.
- `clock.h` Sets the correct clock for the system.

Data This layer provides interfacing with the internal periphery available on the MCU.

- `timer.h` Provides reliable timing functionality.
- `LEDs.h` Allows LEDs to be used for debugging purposes.
- `DIPswitch.h` Allows the user to provide input for the software by means of DIP switch selection.
- `spi.h` Provides a communication via an SPI interface with external components on serial port 1.
- `uart.h` Provides communication via the UART interface to a computer using serial port 2 on the MCU.
- `flash.h` Allows storage on the internal flash module of the MCU.

Protocols Communications to the external periphery

- `ad7758.h` Provides the interface to address the registers in the ADE7758 energy measurement chip.
- `rs232.h` Allows communications with the calibration software on the PC via the RS232 port.
- `atCommand.h` Interfaces with the GSM modem's AT command set.

Applications Provides functionality to the system

- `initMeter.h` Responsible for correct setup of all the global functions of the system.
- `energy.h` This header is responsible for measuring and storing the energy data.
- `onLine.h` Provides the calibration and real-time monitoring routines when connecting to a PC.
- `gsmMode.h` Allows the check-meter to communicate to a remote server using the available GSM modem.

When using this state-event approach the various processes can run concurrently. Each process is allocated the execution of one state. The execution sequence is done in the `main.c` code. Depending on the current mode of operation either the `onLine.h` or `gsmMode.h` drivers are executed. The following code snippet is implemented to do the process coordination:

```
for (;;) {                  // State Event Machine
    //Main Loop Heartbeat
    LEDs_toggleLED6();

    //MCU internal periphery
    LEDs_driver();
    DIPs_driver();
    SPI_driver();
    UART_driver();
    FLASH_driver();

    //External Hardware Communications
    AD7758_driver();

    //Application Drivers
    switch (mode) {   //mode is determined by the DIP switch/jumper setting
       case MODE_RS232:
          RS232_driver();
          OnLine_driver();
          break;
       case MODE_GSM:
          Energy_Driver();
          AT_driver();
          GSM_driver();
          break;
    }
 }
```

6.4.2 The State Event Machine

Depending on the function of the driver a state-event machine is implemented. Although `LEDs.h`, `dipSwitch.h` and `timer.h` performs some updating function the functionality is such that no state-event machine is required. For the drivers requiring a state-event functionality the following general guidelines were used to ensure stability in the system:

- Each driver has its own variable storing the state of the driver.
- Each driver should have a set of defined states which is unique to the driver
- The state may only change in the state-event driver.
- Each state should have at least one state-change code.

The state-event machine is programmed using the `switch` and `case` statements provided in C. This approach is preferred above the lookup table approach to increase execution efficiency and aid the understanding of the state-event programming for other programmers. This approach also has an advantage in that typing and logical errors are converted into syntax errors. Debugging the state-event machine using this approach proved to be fast and effective.

The structure of a state-event machine can be broken up into the following:

Pre-amble The code starts with an explanation of the driver, with which part and at what level it interfaces with other parts in the system. This includes all the declarations in terms of definitions and variable declarations for the machine to operate in. For example:

```
/*********************************************************************
*  ExampleDriver (ED)
*
*  Comments are entered here to explain the basic functionality to
*     the users and programmers.
*********************************************************************/

// Any includes may be specified here, which is specific to the driver, e.g.
#include <io.h>
#include <signal.h>

// State Event Declarations
//-------------------------------------------------------------------
//States
#define ED_stInit             0x000
#define ED_stIdle             0x001
#define ED_stFirstState       0x002
  |
  |
#define ED_stLastState        0x0FFFF


//Variables
int ED_state = ED_stInit;           //The current state within a the ED driver
```

```
//Flags
#define ED_InitFlag           BIT0
#define ED_NewCommand         BIT1
 |
 |
#define ED_LastFlag           BITF
int ED_flags = 0x00;               //Flags to allow the driver to keep track of user input

unsigned char ED_SomeVariable = 0x0;
   |
   |
char *ED_AnotherVariable = "";
```

User Calls The second part consists of code which can be called by other users or programs within the system. This code is responsible for setting the flags and variables so that the driver has all the required information to execute the request successfully. These calls are required to send the execution status, so that the user knows if it has to re-send the call to the driver.

```
// ED_SomeUserCall
//
// The user requests this function call to be executed.
// Returns:
//    0 if successful que placement
//    -1 if a character is already placed for a write itteration
//
//-------------------------------------------------------------------------------
int ED_SomeUserCall(unsigned char someVariable) {
   if ((ED_flags & (ED_NewCommand)) == 0x00) {
      //Do some initialization
      ED_flags |= ED_NewCommand; //Set the flag to let the driver know an action is requested
      return 0;
   } else {
      return -1;  //If action is already requested and not completed.
   }
}
```

Actions The third section allows for code which is called by the driver, which is normally called within the states. The code appearing in this section can theoretically be executed in the state, but depending on the implementation it might be more appropriate to implement it as a auxiliary function or action.

```
//  ED_init
//
// Intialize the ED driver.
//
//-------------------------------------------------------------------------------
void ED_init(void) {
   // Initialization code is entered here...
   ED_flags |= ED_InitFlag;  //And the flag is set, indicating a successful initialization
}
```

Driver The driver part is called by a higher level application or code. Normally this is called in `main.c`. It consists of a case statement responsible for selecting the correct state. The

driver is only allowed to execute one state at a time and should then exit. The format of the case statement implements this automatically, but should however be kept in mind when programming. To keep code maintainable the states should only change within the case statement. This is done by setting `ED_state=stNewState`, prior to exiting the case statement with `break`.

```
// ED_driver
//
// The state event machine implementation is done here.
// In the main loop this code is called by ED_driver();
//
//-----------------------------------------------------------------------------
void ED_driver(void) {

   switch (ED_state) {

      case ED_stInit:
         ED_init();
         ED_state = ED_stIdle;   //Change the state
         break;

      case ED_stIdle:
         if ((ED_flags & ED_NewCommand) == 0x0) {
            ED_state = ED_stFirstState;
         }
         break;

      case ED_stFirstState:
            //Does some thing, and wait if required.
            //If all is complete and the requirements is satisfied the state may be changed.
            // Remember that even if the requirements are not satisfied, the state must exit,
            // but the state does not need to change.
            ED_state = ED_stLastState;
         break;
      case ED_stLastState:
            //With the last state executed, the state event machine's state should
            //  return to idle
            ED_state = ED_stIdle;
         break;
   }
}
```

6.4.3 The Initialization Process

Prior to making any measurements, the system is required to run through an initialization process. All the different components, either in hardware or software, are initialized. The strategy behind the initialization and software design is to ensure that a stable environment exists for the state-event machines to operate in.

The process is started in `main.c`, and follows the following process:

1. The watchdog timer is disabled, to ensure that no unexpected resets occur.
2. The clock is setup to use the external crystal in XTAL1.

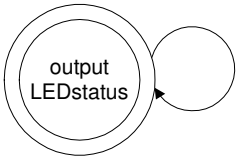


Figure 6.6: *State diagram of LED driver*

3. The `LEDs.h`, `dipSwitch.h` and `timer.h` code is initialized.

4. The mode is determined according to the DIP switch settings on DIP switch 7. With the dip switch closed the check meter will run in RS232 mode. With an open DIP switch the GSM mode is selected.

With all the basic functionality in place the state-event machine is started and each driver is then responsible for initializing all its own functions. On the application level however, a special driver is called, namely `initMeter.h`. This is called within `onLine.h` and `gsmMode.h`. The reason for this, is that this driver is only required for initialization and would after initialization take up unnecessary system time. The following code, shows how `initMeter.h` is called in `onLine.h`:

```
case OnLine_stInitStart:
   if (INIT_init() == 0) {
      OnLine_state = OnLine_stInitSystem;
   }
   break;
case OnLine_stInitSystem:
   if (INIT_isReady() != 0) {
      INIT_driver();
   } else {
      TIMER_startTimer(tmrWDTOnLine);
      OnLine_state = OnLine_stIdle;
   }
   break;
```

A similar implementation is done in `gsmMode.h`. The initialization state machine will be discussed in Section 6.4.4.

6.4.4 Software Sub-systems

This section will briefly discuss and show the implementations of each of the implemented software sub-systems of the check-meter.

LED driver

The LED driver consists of one state. This state ensures that the LED status, which is changed by the calls to the driver is always updated, as shown in Fig. 6.6.

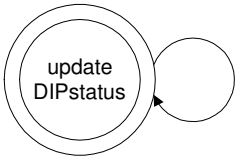


Figure 6.7: *State diagram of DIP switch driver*

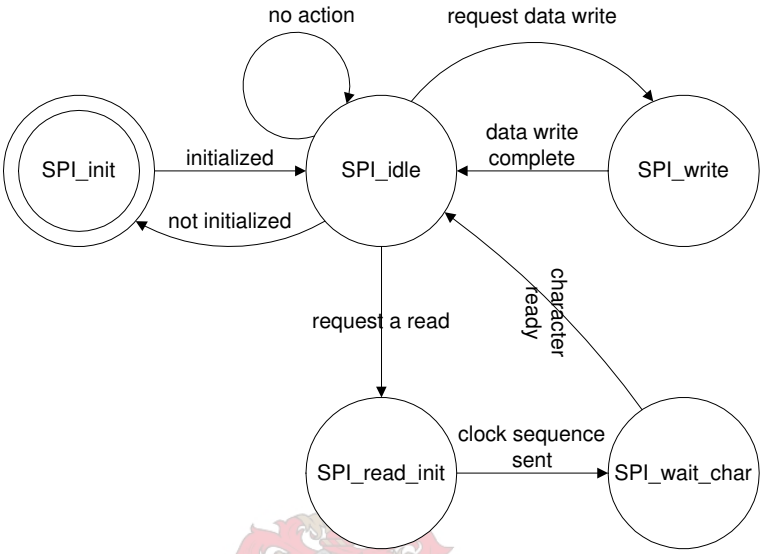


Figure 6.8: *State diagram of SPI driver*

DIP switch driver

The DIP switch driver updates the `DIPstate` register, as shown in Fig. 6.7. When other drivers test for the current DIP switch status they access the `DIPstate` variable. In future this could also be handy when implementing debouncing for keypads or buttons routines.

SPI driver

The SPI driver ensures that the data layer is correctly setup and communications can take place between any SPI device on the bus. The setup of all the registers is done in the `SP_init` state. The state diagram is shown in Fig. 6.8 and consists of an initialization part, a character write part and a character read part.

UART driver

The UART driver is used to communicate with the PC and with the GSM module. It also forms the data layer to these devices. This driver is also responsible for switching between these devices using the power off function of the MAXC3222C[89].

Incoming data is polled using the `UART_isCharReady()` function of the driver. The

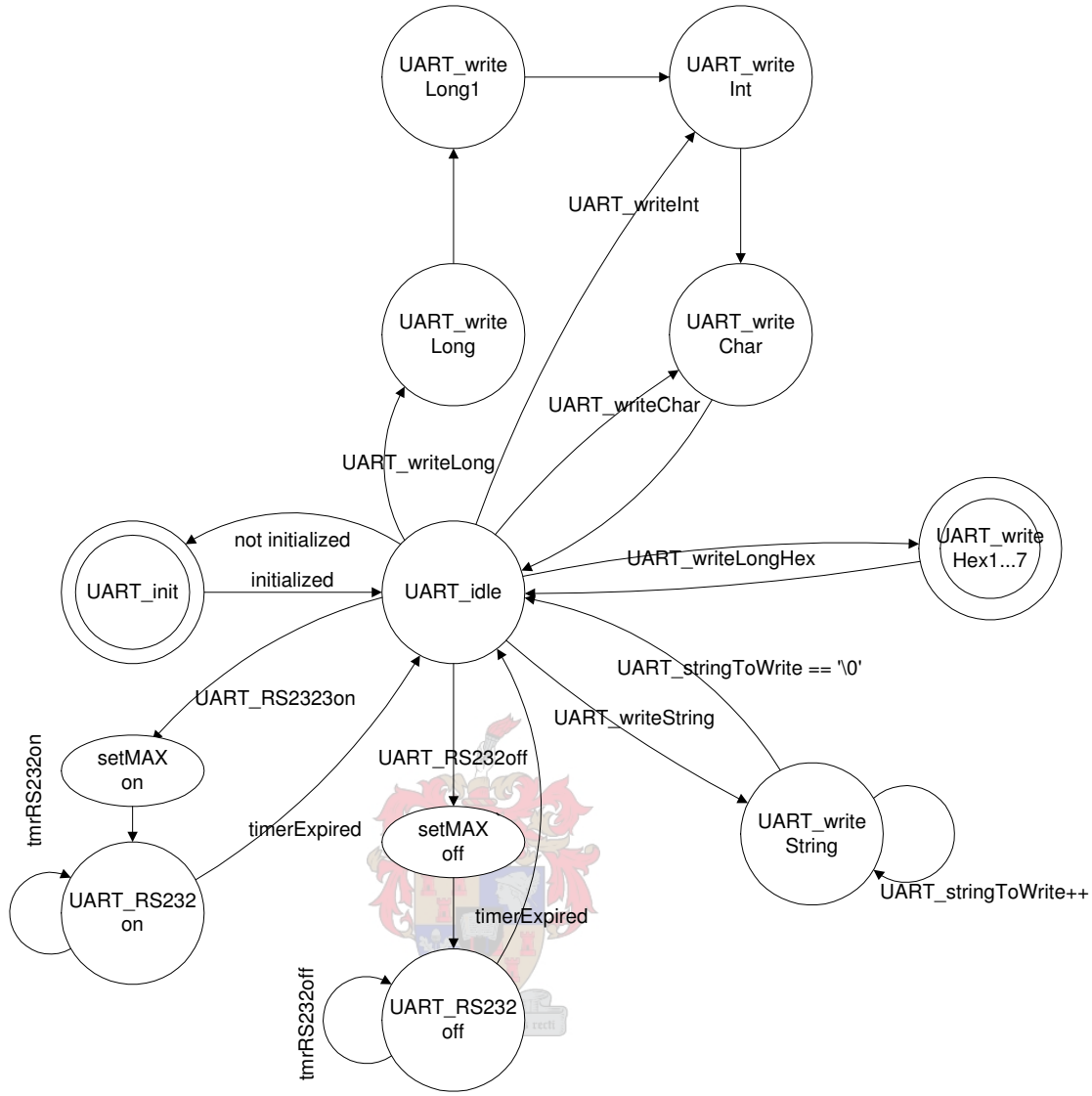


Figure 6.9: *State diagram of UART driver*

driver can send data in 8bit, 16bit and 32bit mode and also includes a process to send a 32bit character in hex format. Strings are sent using a single state which iterates through the string until the null character is found.

The state machine is shown in Fig. 6.9.

FLASH driver

The flash driver interfaces with the internal flash memory of the MCU. During the initialization the correct clock is setup. After each state the BUSY register is polled to see if the flash unit is free for a new command. A pointer is used to write to the memory. The read operation is a straightforward read from the correct address and is accessed by users of this driver with

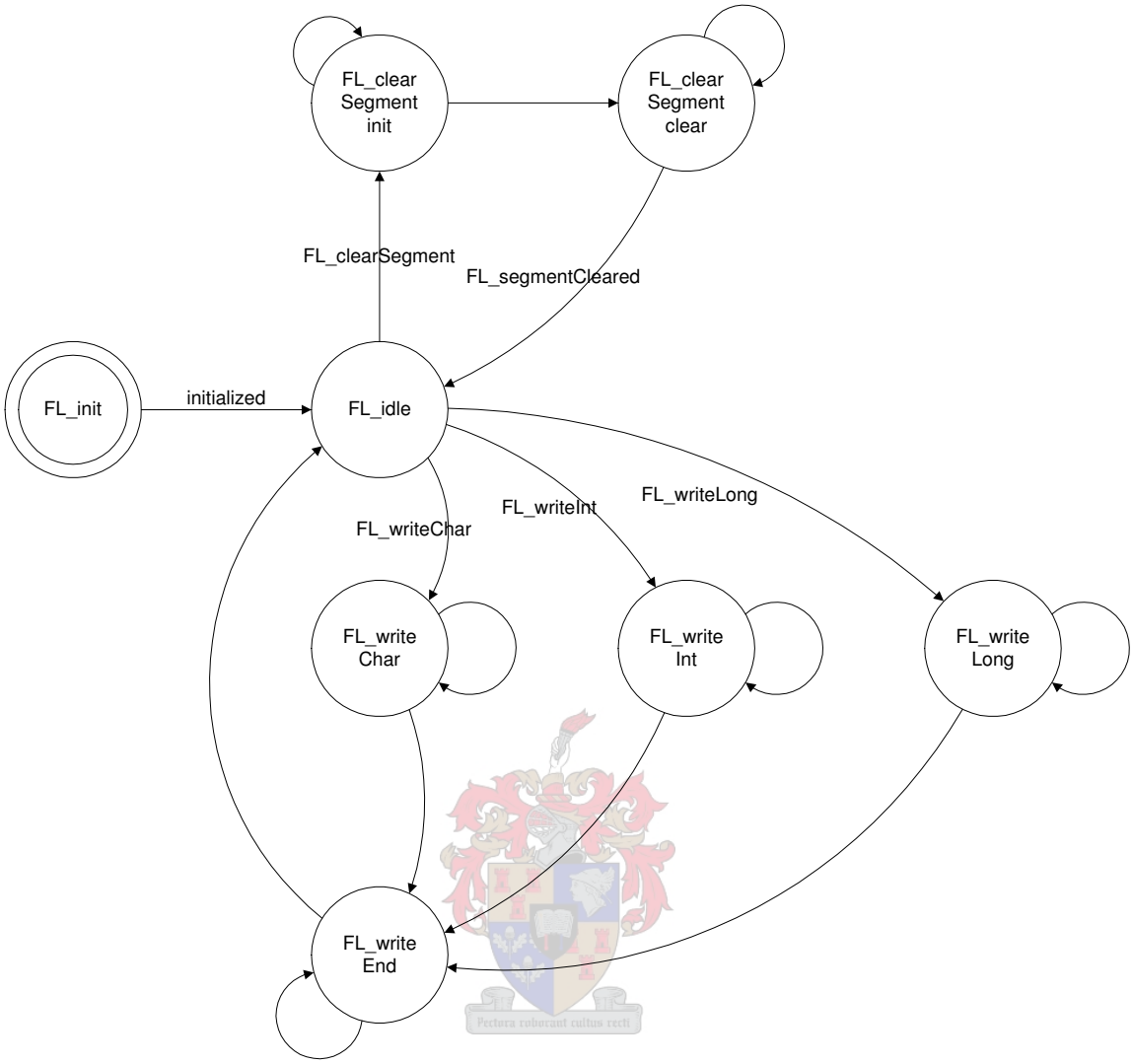


Figure 6.10: *State diagram for FLASH driver*

`FLASH_readChar()`, `FLASH_readInt()` and `FLASH_readLong()`. All the storage registers are defined in this driver.

The various states are shown in Fig. 6.10.

AD7758 driver

The AD7758 driver is responsible for handling all the communications via the SPI driver to the ADE7758 energy measurement chip. This driver contains all the register codes and handles all the various data types that might be sent and received to and from the device. The driver is also responsible for handling the CS signal. All the data sent or received is preceded by a register byte. Depending on the register byte the ADE7758 will expect 1 to 3 bytes of data. A read operation is indicated by setting the most significant bit with a 0 and for a write it should be set to 1.

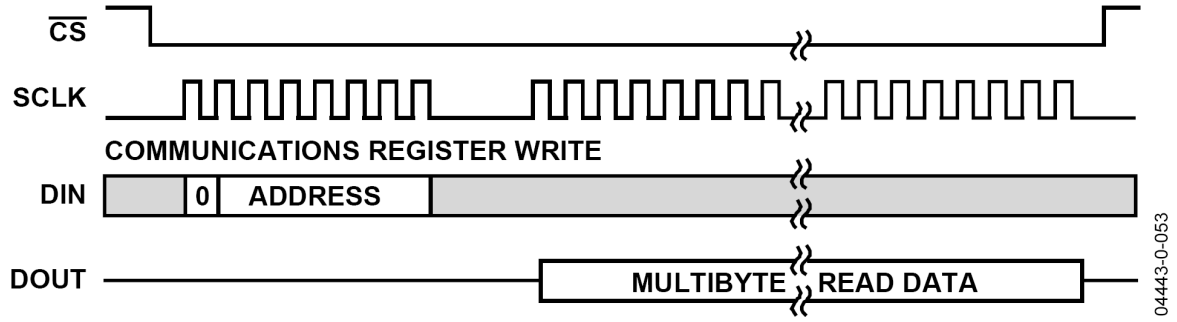


Figure 6.11: *Reading Data from the ADE7758 via SPI, taken from [7]*

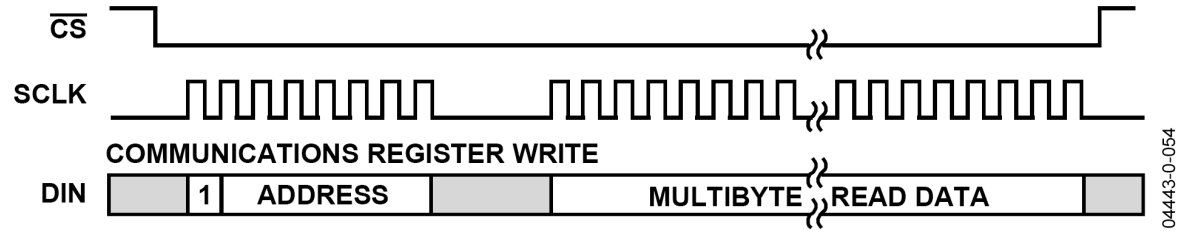


Figure 6.12: *Writing Data from the ADE7758 via SPI, taken from [7]*

Fig. 6.11 and Fig. 6.12 shows the timing requirements as set out in the ADE7758 data sheet [7].

The states are shown in Fig. 6.13.

For the read sequence the register requested by the AD7758 driver is written out to the SPI driver and then the state machine waits for the register to be clocked out. This is followed by the read states. The multiple read states consists of the following states:

1. Read a byte from the SPI driver.
2. Wait and continues to read the next byte until the read operation is complete.

The write operation is fairly similar except that the when a call is received by the AD7758 driver to write data to the AD7758, the register is included in the byte array and can be written out.

RS232 driver

The RS232 driver is responsible for encoding and decoding data to and from the PC. The driver is not responsible for error checking on the data link. Table 6.4 shows the implemented structure of the protocol as implemented in Appendix R.4.2.

The RS232 driver consists of a receive and send component. This is implemented using the following code:

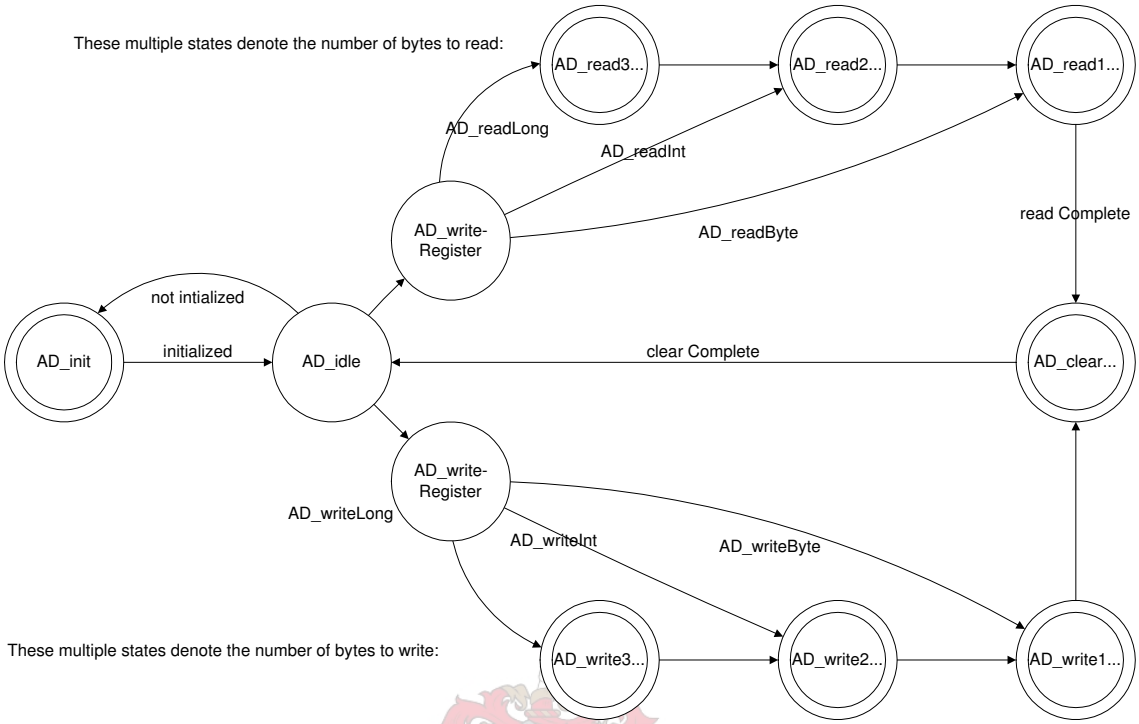


Figure 6.13: *State diagram of AD7758 driver*

Byte	Meaning
0	Start Character
1	Length
2	Device
3	Device Register
4	Data[1]
5	Data[2]
⋮	⋮
N	Data[Length-4]
N+1	Last Character

Table 6.4: *Protocol elements*

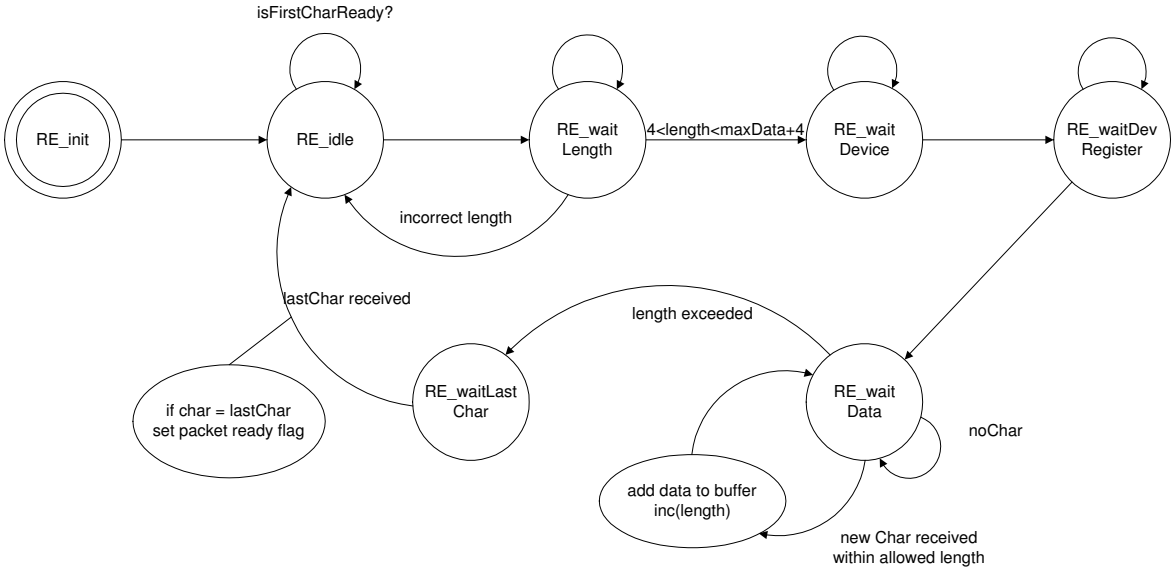


Figure 6.14: *State diagram of receiving RS232 driver*

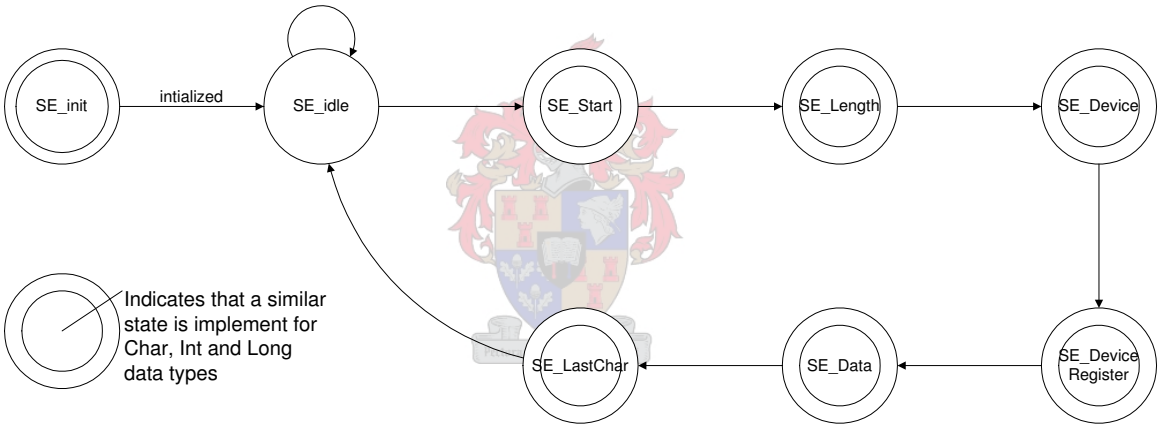


Figure 6.15: *State diagram of sending RS232 driver*

```
void RS232_driver(void) {
   if (UART_isRS232on() == 0) {
      RS232_receiveStateDriver();
      RS232_sendStateDriver();
   }
}
```

The receive state machine is showed in Fig. 6.14.

The sending state machine is showed in Fig. 6.15.

AT command set driver

The AT command set is supplied by Wavecom [93] and complies with ETSI specifications [26, 28, 27, 29, 30], ITU-T recommendations [42] and 3GPP technical specifications [1].

The AT command interface is a text interface that allows easy communications to the GSM

module. Each command is started with the text `\n\rAT` and submitted using the newline and carriage return characters, `\n\r`. The driver is responsible for checking the communications to and from the module and does some error checking to ensure that the commands are sent correctly.

This driver consists of a send and a receive part and can only operate when the RS232 driver chip is switched off. This is implemented using the following code:

```
void AT_driver(void) {
   if (UART_isRS232off() == 0) {
      AT_receiveStateDriver();
      AT_sendStateDriver();
   }
}
```

The send state machine is shown in Fig. 6.16. Simple error correction is handled here. The possibility to extend this exists, but for this application only checking for `OK` and `Error` is required. Using these two strings provides ample error detection. If an `Error` is detected, the AT command is repeated.

In the AT driver the sending of the Energy Report, via SMS, is also implemented. All the energy registers are sent using hex representation. Each register consists of 8 alpha-numeric characters. The following text gives an overview of how the text messages in the SMS are constructed, where W denotes the real power, VAR the reactive power and VA the active power measured per phase A, B or C:

```
WAAAAAAAWBBBBBBBWCCCCCCC:VARAAAAAVARBBBBBVARCCCCC:VAAAAAAAVABBBBBBVACCCCCC:
```

The receive state machine is used to capture the AT commands received from the GSM module. Only limited functionality is required and decoding of the various messages are done by other drivers or part of drivers. The state diagram is showed in Fig. 6.17.

Initialization driver

The initialization driver runs in a single sequence. The flow of this state machine is shown in Fig. 6.18. This driver is initiated upon startup of the application layer, in both `GSMmode.h` and `onLine.h`. This is done to ensure that both applications have all the resources setup is exactly the same way.

Energy measurement driver

The energy driver is responsible for giving other drivers the current energy readings. With a cycle time of $T_{cycle} = 100\ ms$ the registers are read from the AD7758 driver and stored. Other drivers can access these registers directly from the memory. Each multi state in Fig. 6.19 consists of a request to the ADE7758 for a specific register and an accumulator update for all three phases, namely A, B and C.

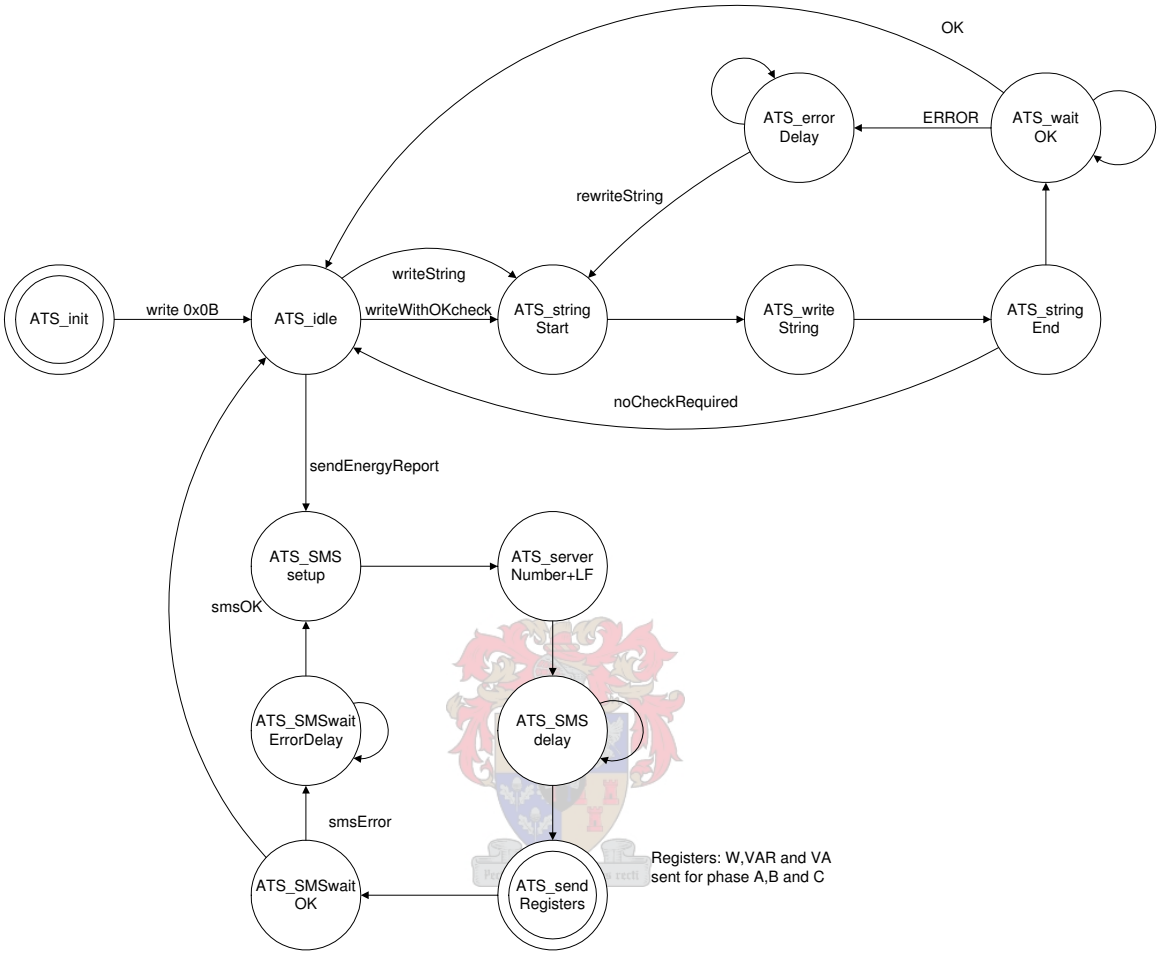


Figure 6.16: *State diagram for sending part of AT driver*

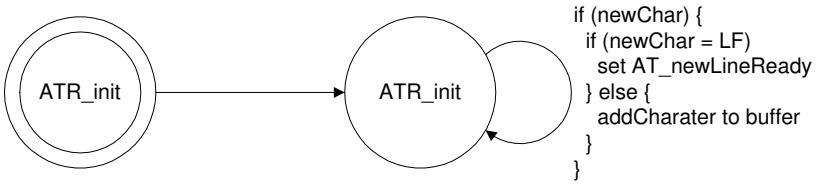


Figure 6.17: *State diagram for reception part of AT driver*

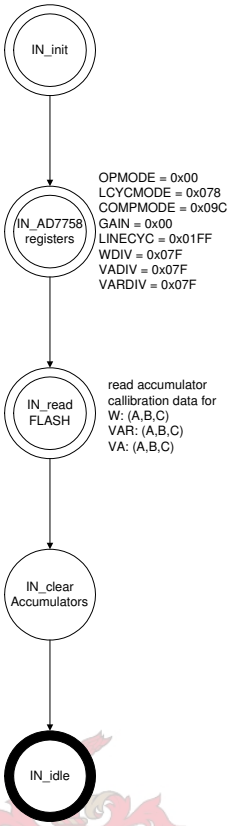


Figure 6.18: *State diagram of initialization process*

The strategy for measuring the energy is by reading the energy accumulators from the ADE7758. The ADE7758 is set up to reset the accumulator each time the registers are read. These register values are then added to the interim energy accumulators in the micro controller. If these are larger than the dividing factor, the values are added to the final accumulators. The whole process is illustrated using the following code for the real power calculation of phase A:

```
energy_W_a = energy_W_a + signedIntToLong(AD7758_getInt());
if ((energy_W_a >= MCU_WDIVA) | (energy_W_a <= -MCU_WDIVA)) {     //Scaling
   tmpLong = energy_W_a/MCU_WDIVA;
   energy_W_A = energy_W_A + tmpLong;           //Add new unit to final accumulator
   energy_W_a = energy_W_a - tmpLong*MCU_WDIVA; //Correction for rounding errors
}
```

Online mode

Since the `onLine.h` is responsible for handling the calibration routines, this driver is by far the largest driver of them all. Interaction takes place using the RS232 port. PC software is designed to interface with the check-meter and is discussed in Section 6.5. The various steps of the calibration process are discussed in Section 6.7.

A state diagram is presented in Fig. 6.20.

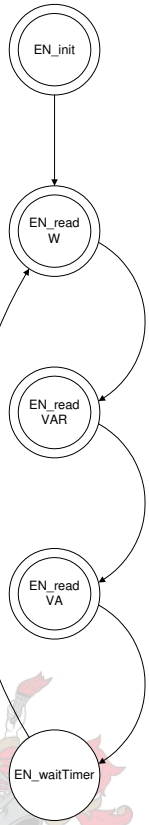


Figure 6.19: *State diagram of energy driver*

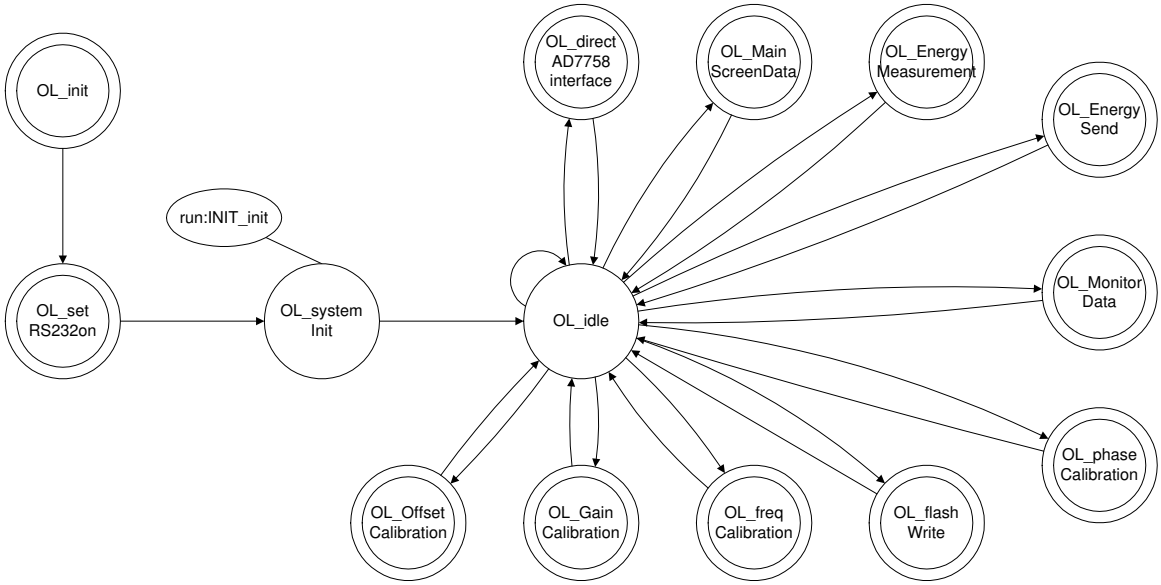


Figure 6.20: *State diagram of online driver*

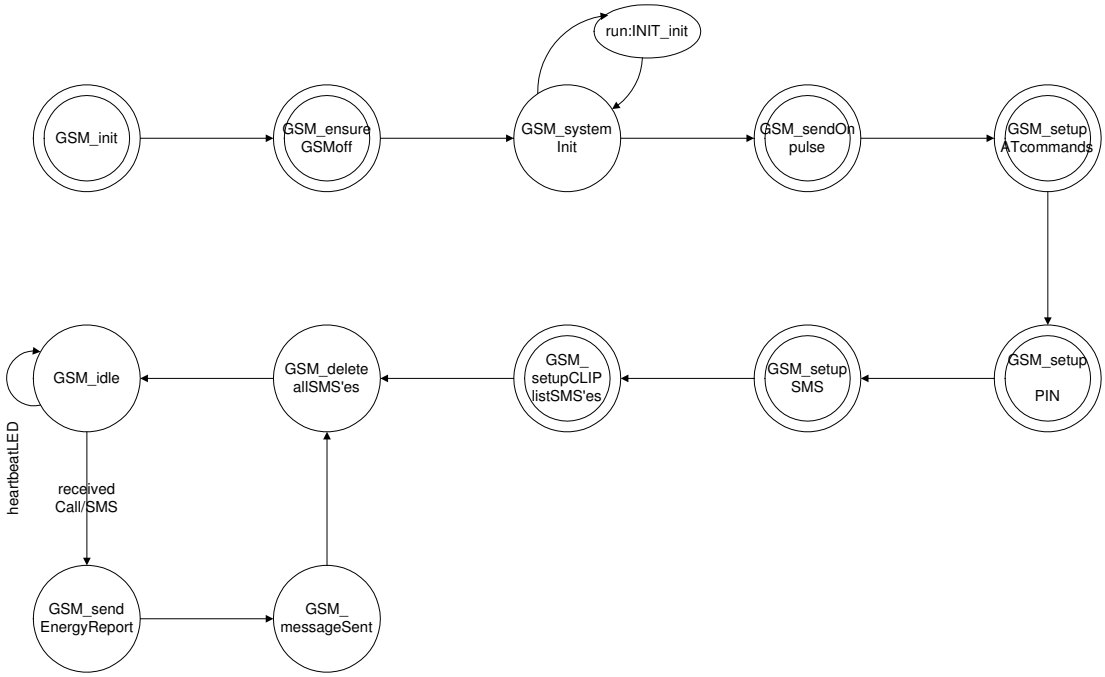


Figure 6.21: *State diagram of GSM mode driver*

GSM driver

The GSM driver uses the AT command driver extensively in combination with the UART driver. The UART driver is used to ensure that bus contentions do not occur, especially during startup.

An energy report is sent only when a mobile terminating call or message is received. In other words, when a message of some sort is received by the device an energy report is sent. It was decided to exclusively keep the logic at this level, to ensure that no “run-away” SMSs would be sent. The information frequency is set at the server. The client can only react to messages from the server, as can be seen in Fig. 6.21. The SMS server is responsible for capturing and initiating communications as discussed in Section 6.6.

6.5 Calibration PC Software

The calibration software was designed on the Java platform [83] in the Netbeans environment [59]. Since the data mining application in Chapter 5 was developed in Java, future integration and re-use of already written classes would be seamless.

The software is implemented using an event driven model based on user and communications input from the serial port. This section discuss the software while the calibration process will be discussed in Section 6.7 in more detail.

6.5.1 Communications

The software implements the serial communication using the `peg.io.serialInterface`(see Apendix Q.5.5) class using the Communications API [84] from Java to access the serial port. The `serialInterface` class is designed to aid and ease the use of the communications API.

The protocol used for communications is implemented in `peg.io.protocol` (see Appendix Q.5.4) and is designed to comply with the protocol used in Section 6.4.4.

6.5.2 Graphical User Interface

The user interface is done using Java's Swing API. The menu allows access to various tasks. The menu layout was established as follows:

File Allows the user to save the current setup of the energy measurement software.

Calibration Some general commands to aid the calibration process.

- **Calibration LEDs ON/OFF** Sets the LED's connected to the ADE7758 to be toggled on or off.
- **Write to Flash** Ensures the current calibration details are written to the Flash on the MCU.
- **Read Flash** Reads all the data on the flash and updates all the register in the MCU and ADE7758.

MCU This command requests that the micro controller sends a collection of important registers to the GUI.

Windows This menu provides access to additional windows for setup for specific tasks, as can be seen in Fig. 6.22. It is divided into a monitor and a calibration part. The following windows are available:

- **Energy Measurement** Allows the user to verify the calibration procedure.
- **Measurement** Allows the user to monitor voltage, current and the status of the energy registers on the ADE7758.
- **Other Measurements** This window shows the user information about temperature, frequency and the communications link status.
- **Registers** Various register are displayed here to provide feedback for the embedded software programmer.
- **Offset Calibration** A two-point gain and offset calibration is done on the RMS voltage and current.

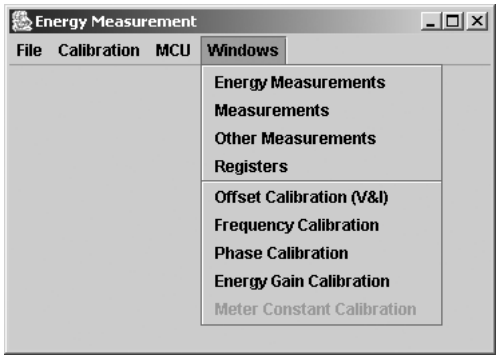


Figure 6.22: *Available windows in graphical user interface*

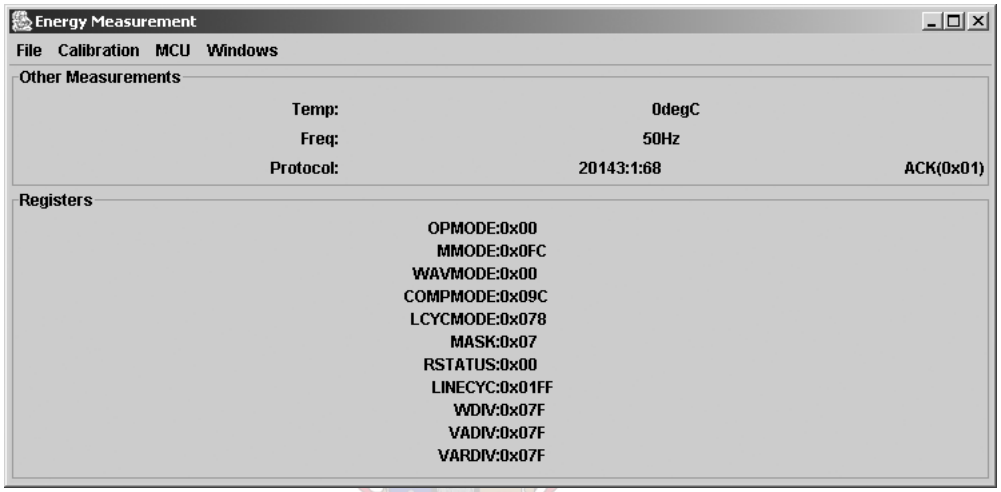


Figure 6.23: *Register and Other measurements*

Frequency Calibration The frequency is calibrated using this window.

Phase Calibration Slight phase errors introduced by the probes or current transformers is corrected for using the calibration window.

Energy Gain Calibration Depending on the current transformer or probe ratios the energy accumulators are calibrated.

Measurement Screens

The energy measurements screens are used to give feedback to the operator when calibrating to verify the calibration.

The *register* screen gives an overview of the important registers in the ADE7758, while the *other measurements screen* shows the current communication activities, line frequency and temperature. These screens are shown in Fig. 6.23. These screens are especially monitored when the unit is switched on for the first time to evaluate the correct initialization of the check-meter unit. All the registers can be dumped using the MCU — DUMP ALL REGISTERS command.

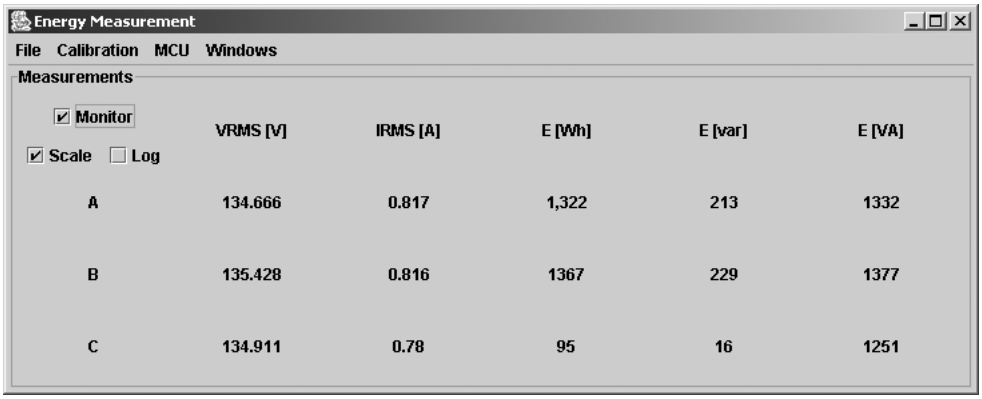


Figure 6.24: *Measurement screen*

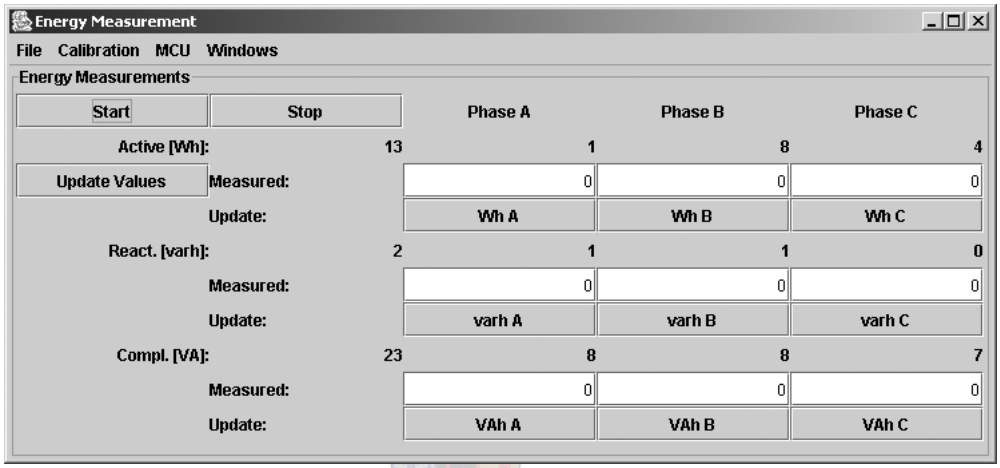


Figure 6.25: *Energy Measurement screen*

In order to verify the correct calibration values the *measurement* and *energy measurement* screens are used.

The measurement screen, as shown in Fig. 6.24, allows the user to activate the continuous update or monitor function to provide instantaneous feedback. The scale can be set to alternate between the actual register values of the ADE7758 and the stored calibrated values. With the log function enabled, the measured values are dumped to `c:\log.csv` and can be imported into a spreadsheet or database.

The energy measurement screen, see Fig. 6.25, presents the current micro controller accumulators, for all the phases and energy registers. The user can fine tune using the edit boxes on the screen. The calibration values are updated using the *Update Values* buttons. The accumulators and `energy.h` subsystem are started and stopped using the *Start* and *Stop* buttons provided on the screen.

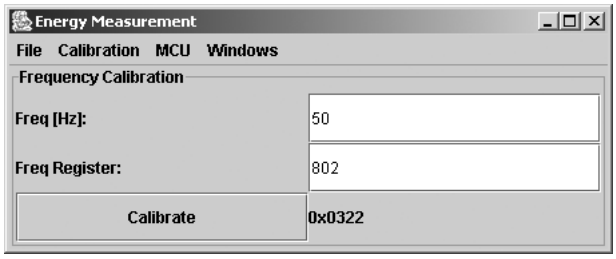


Figure 6.26: *Frequency Calibration screen*

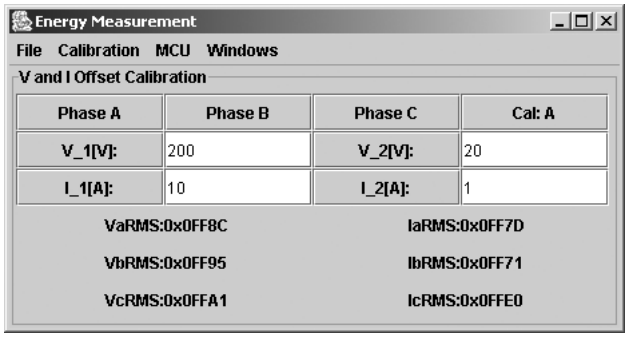


Figure 6.27: *Voltage and Current Offset Calibration screen*

Calibration screens

The first calibration screen is used for frequency calibration. This screen is fairly simple to use. The frequency value is entered in the appropriate text box and the *Calibrate* button is pressed to initiate the calibration process.

The Voltage and Current offset calibration screen is shown in Fig. 6.27. The top row buttons allow for selection of the current phase in the calibration process. The values are entered at two points in the text boxes. In order for the user to measure the value on the ADE7758, the button on the left of the text is clicked. This will start the measurement procedure for the specific value and currently selected phase. The system is calibrated for the specific phase using the top right button with the leading *Cal:* text.

To correct phase errors in the measurement path, the phase calibration screen is used, as shown in Fig. 6.28. For each phase, two power measurements at different power factors are measured. The previous measurements can be loaded using the *Update Form* button. The calibration data is calculated and sent to the check-meter by pressing the *Calibration* button.

To allow for various ranges of current transformers and probes the energy accumulators are calibrated using the Energy Accumulator Screen as shown in Fig. 6.29. For a first iteration the registers should be reset, using the *Reset* button. The current power flowing through the measuring point is entered and submitted using the *Cal Phase* buttons for the particular phase. Preferably this calibration should be done using a power factor of $pf \approx 0.7$. For fine tuning of these gain parameters the energy measurement screen can be used, as seen in Fig. 6.25.

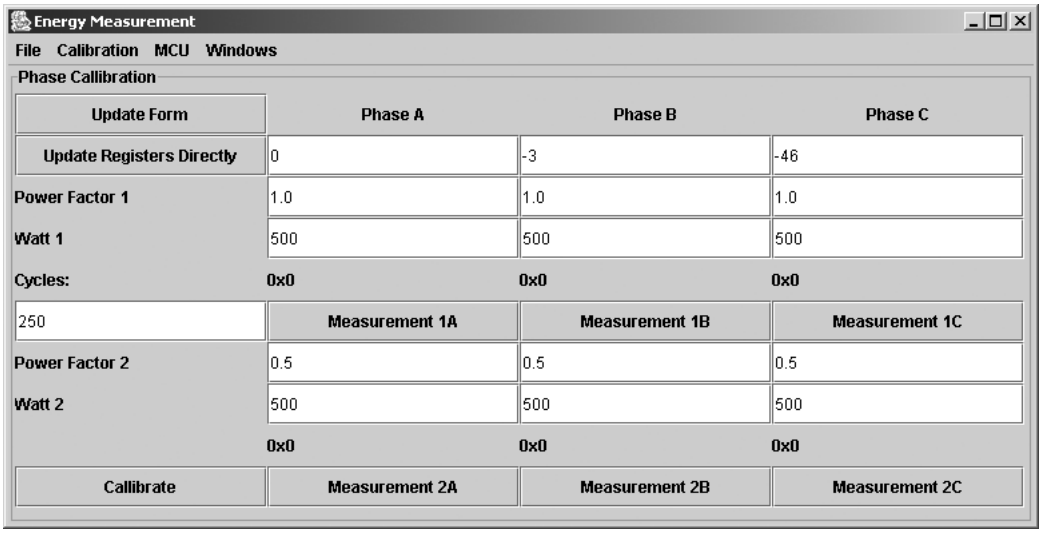


Figure 6.28: *Phase Calibration screen*

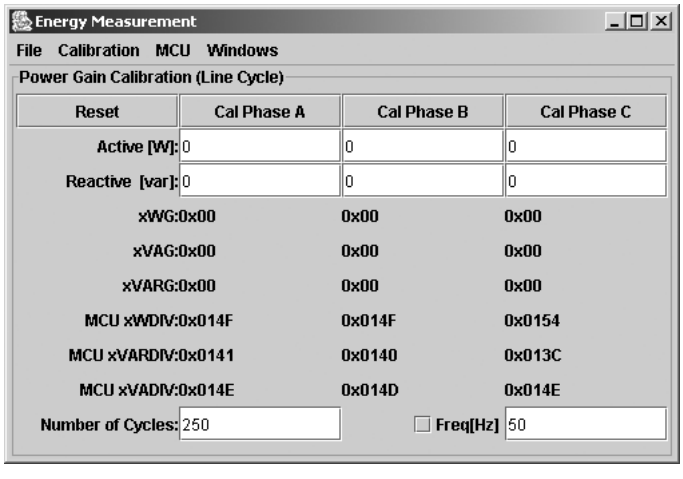


Figure 6.29: *Energy Accumulator calibration*

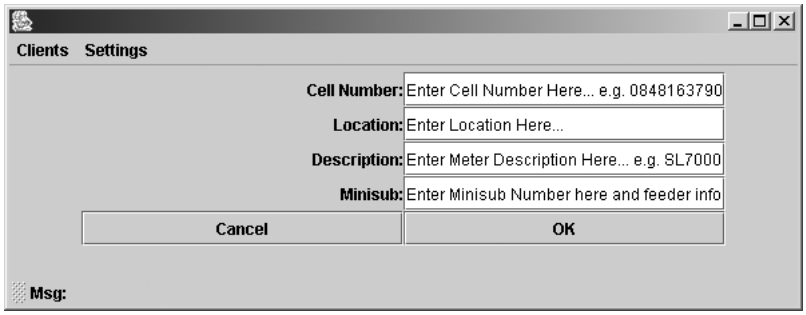


Figure 6.30: *Adding Clients screen*

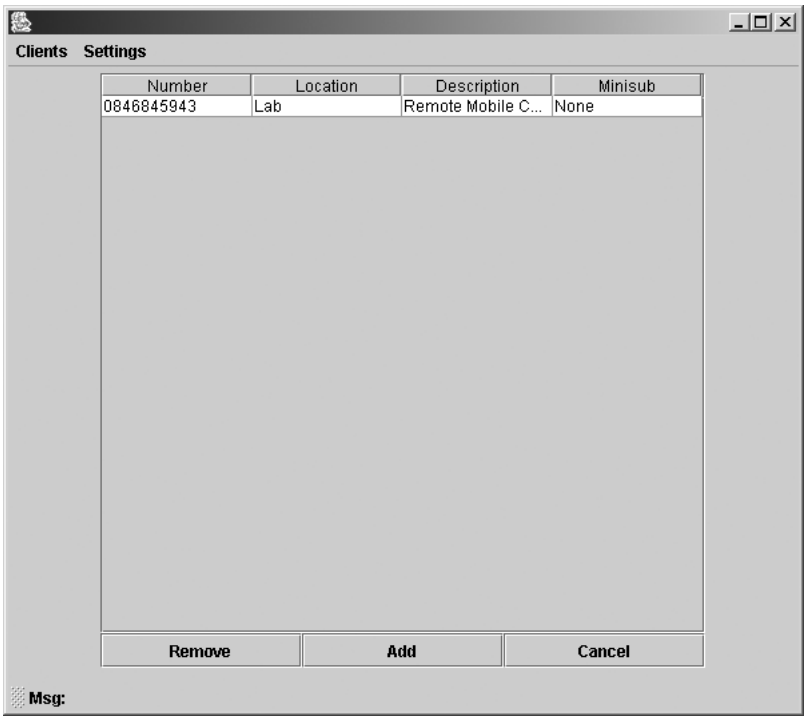


Figure 6.31: *View Clients screen*

6.6 PC Server Software

The PC server software allows the operator to send requests to the check-meter and to process reports received from the various check-meters and to store the data in a data warehouse.

A program was developed to allow users to easily set and modify server settings. Clients or check-meters can be added and viewed as shown in Fig. 6.30 and Fig. 6.31. A link to an SQL database can be set up using the Database settings screen in Fig. 6.32. The server settings allow a user to specify the numbers required for the GSM modem, the time the daily update should occur and the ability to force an update. The server settings are shown in Fig. 6.33.

The PC server is a simple eternal loop thread, as shown in Fig. 6.34. The thread interfaces to a GSM modem using the `peg.io.serialInterface`(see Apendix Q.5.5) class and an AT command set [93] class. Received SMSs are processed and stored in the database using the

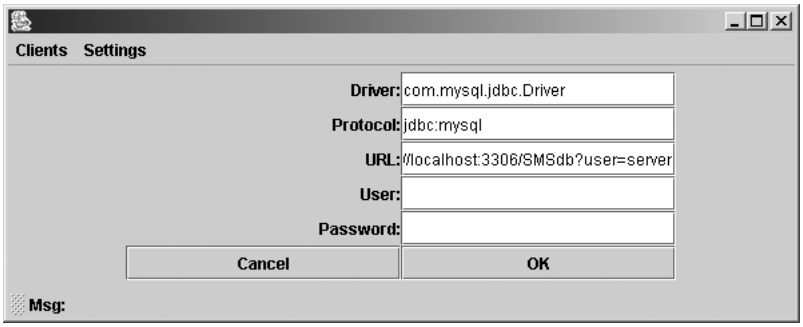


Figure 6.32: *Database Settings Screen*

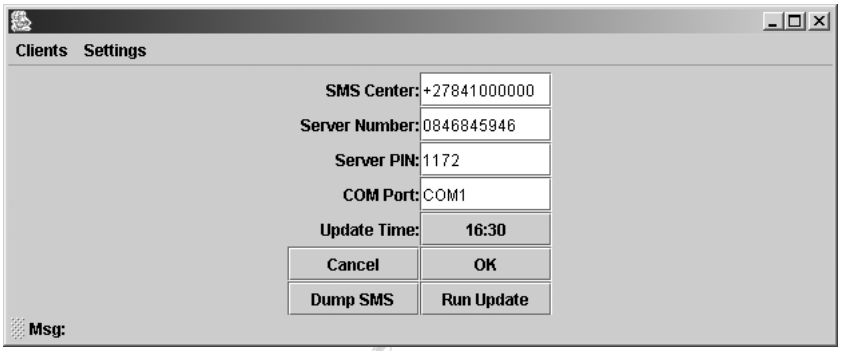
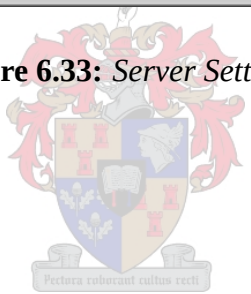


Figure 6.33: *Server Settings screen*



`peg.sql.DBConnect` class.

6.7 Calibration

Although it seems as if the ADE7758 requires an extensive calibration procedure, it can be broken down into four simple calibration steps. The calculations for the various parameters required by the ADE7758 and micro controller software, are done by the calibration software in the `energyMeasure.energyCalibration` class. When measurements from a user are required, it implies that an accurate measurement from an already calibrated device is used. It is important to note that the calibration accuracy is dependent on the accuracy of the device used to calibrate it against.

In order to understand the calibration process, the functional block diagram in Fig. 6.35 is presented to show how the ADE7758 calculates the various registers and how calibration registers fit into the equations.

The calibration process consists of the following steps:

1. The frequency is required to supply the other steps with correct timing data.
2. Offset calibration is used to increase the voltage and current accuracy.

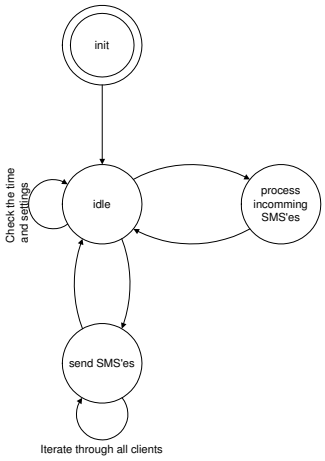


Figure 6.34: *State diagram of PC server program*

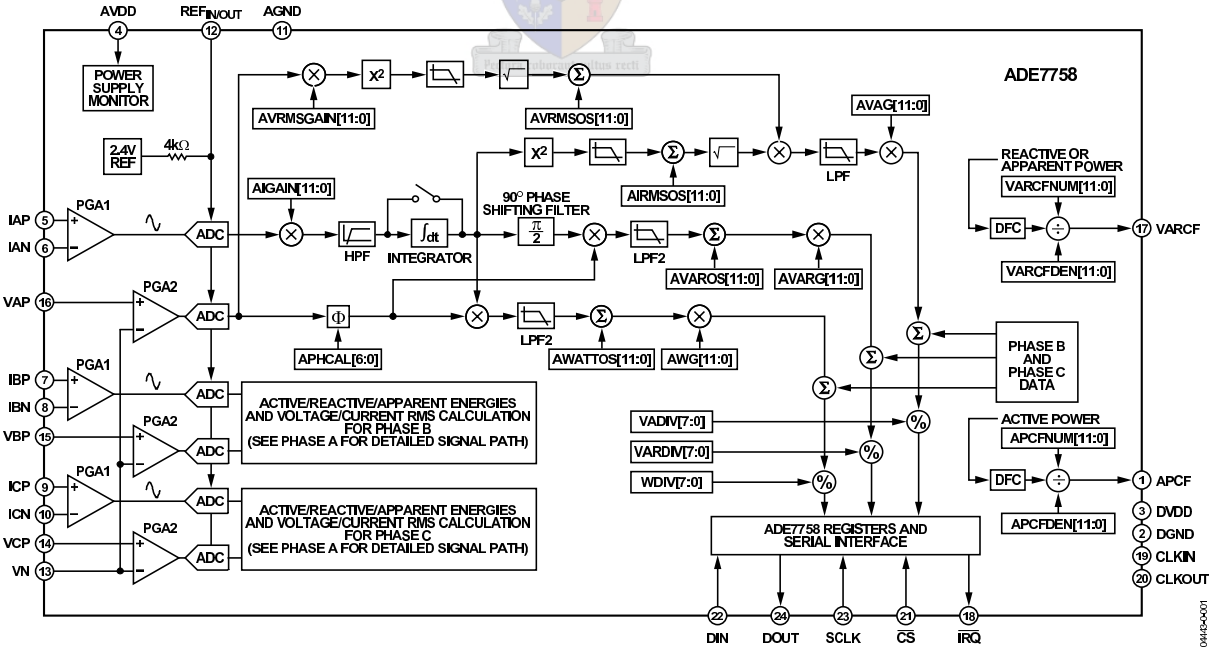


Figure 6.35: *ADE7758 Functional Block diagram [7]*

3. The difference in phase angle added to the current measurement path is calibrated to calculate the power factor and real power component accurately.

4. The energy accumulators are calibrated to include the additional gain of the current transformers and probes various ratios.

Each process can be broken down further into the following steps:

Screen selection The user is required to select the correct screen for calibration

Measurements Setup the equipment and measure the required values from the available equipment and input the values into the screens.

Acquire the ADE7758 registers The register values of the measurements on the ADE7758 must be retrieved and stored in the program.

Calculation and parameter storage Once all the measurements have been taken the parameters are calculated and stored on the ADE7758.

6.7.1 Frequency Calibration

The frequency calibration is done using a single point and calculating the LSB gain. This gain is only used for calibration purposes and is therefore, only stored in the application. The frequency calibration is done on Phase A. This, however, can be set in the software by changing the FREQSEL mnemonic of the MMODE register of the ADE7758.

When the frequency calibration process is initiated by the user, a request is sent to the check-meter to retrieve the current frequency value of the FREQ register. This is sent back to the calibration software and with the known frequency from the user input the gain is calculated. The process is outlined in Fig 6.36.

6.7.2 Voltage and Current Offset Calibration

This voltage and current offset calibration is not required for the active or reactive energy measurement, but is done firstly to give the user a good idea if the system is working and to determine the accuracy of the analog to digital converters of the ADE7758. Secondly, this calibration process is used by the apparent power calculation, which is not that important for this thesis's application. All measurements are done in RMS unless stated otherwise.

The LSB_{gain} calculation is only used in the calibration software, but the offset is sent to the ADE7758. The gain registers in the ADE7758 are not used to prevent any quantization errors, since fixed point number representation is used within the ADE7758, and the total gain is rather

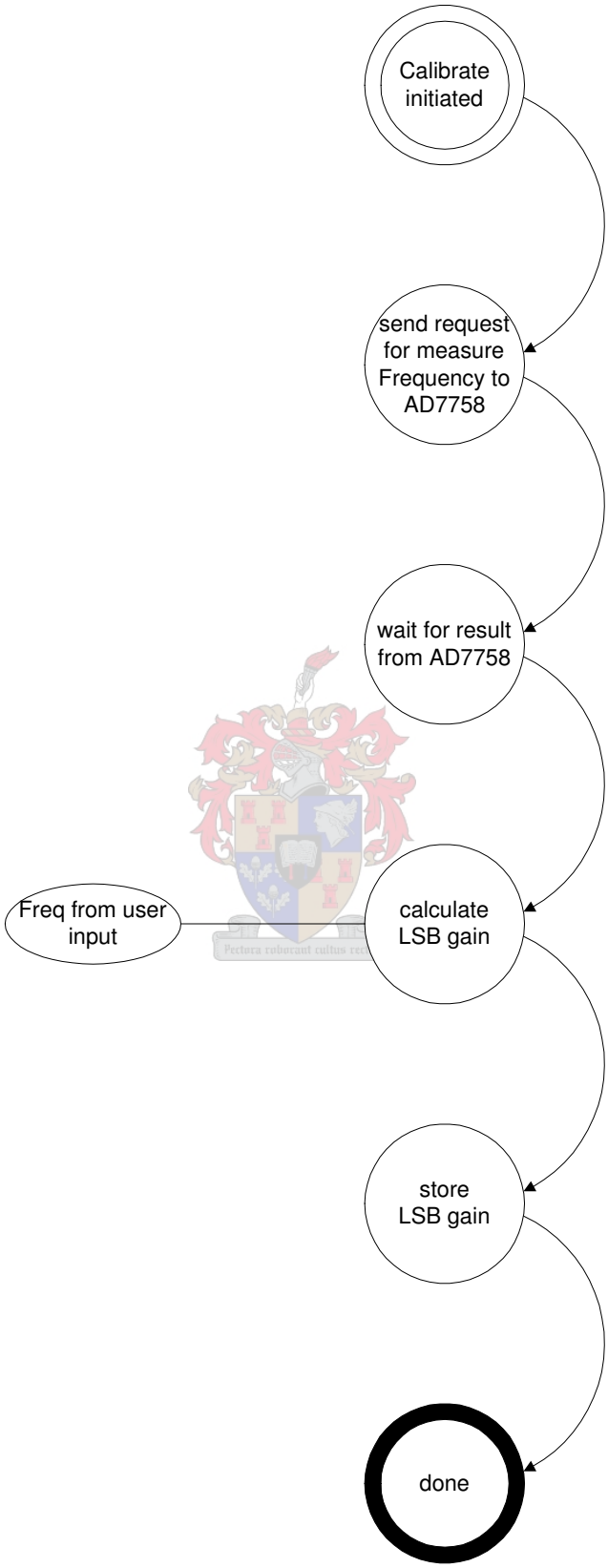


Figure 6.36: *Frequency Calibration process*

adjusted at the end by setting the gain for the energy accumulators. The voltages and currents are calculated in the ADE7758 using (6.4) and (6.5) according to [7].

$$V_{RMS} = V_{RMS_0} + 64 \times V_{RMSOS} \tag{6.4}$$

$$I_{RMS} = \sqrt{I^2_{RMS_0} - 128^2 \times I_{RMSOS}} \tag{6.5}$$

Using the calibrated V_{RMS} and I_{RMS} received from the ADE7758 the calibration software calculates the scaled values by multiplying the V_{RMS} and I_{RMS} with the LSB_{gain}.

To calibrate the voltage and current, two values are required in order to calculate the offset and the gain. After the two measurements for both current and voltage are taken by entering the values in the GUI, the calibration can be initiated (see Section 6.5.2 and Fig. 6.27).

When the measurement process is initiated by the user, the entered value is stored and the current measurement is requested from the ADE7758. The returned result is also stored. After all four values are gathered, the user can calibrate the selected phase using the CAL: button.

The voltage offset is then calculated in the application:

$$V_{offset} = \frac{V_1 V_{2_{ADC}} - V_2 V_{1_{ADC}})}{64(V_2 - V_1)}$$

The current offset in the diagram is found after the square root of the measurement is taken and therefore:

$$I_{offset} = \frac{I_1^2 I_{2_{ADC}}^2 - I_2^2 I_{1_{ADC}}^2}{128^2(I_2^2 - I_1^2)}$$

The LSB gain is calculated using the first value from the user input and is divided by the measured ADC value for current and voltage and stores it:

$$LSB_{gain} = \frac{M_1}{M_{1_{ADC}}}$$

where M indicates the measurement as either V or I.

The process is shown in Fig. 6.37.

6.7.3 Phase Calibration

CTs and current probes phase errors can be expected to vary from CT to CT [7] and probe to probe[48, 49, 47]. The phase error has a direct impact on the active energy calculations and therefore this is an important parameter to incorporate into the calculation path.

Hardware considerations

The ADE7758's phase correction can compensate for -2.72^o to 1.36^o at $50\ Hz$ using the xPHCAL registers. In some cases the lagging phase error could be as much as 10^o [47]. This lag should be corrected for realizing a phase lead by placing a capacitor parallel to current input stage of the ADE7758, as shown in Fig. 6.38.

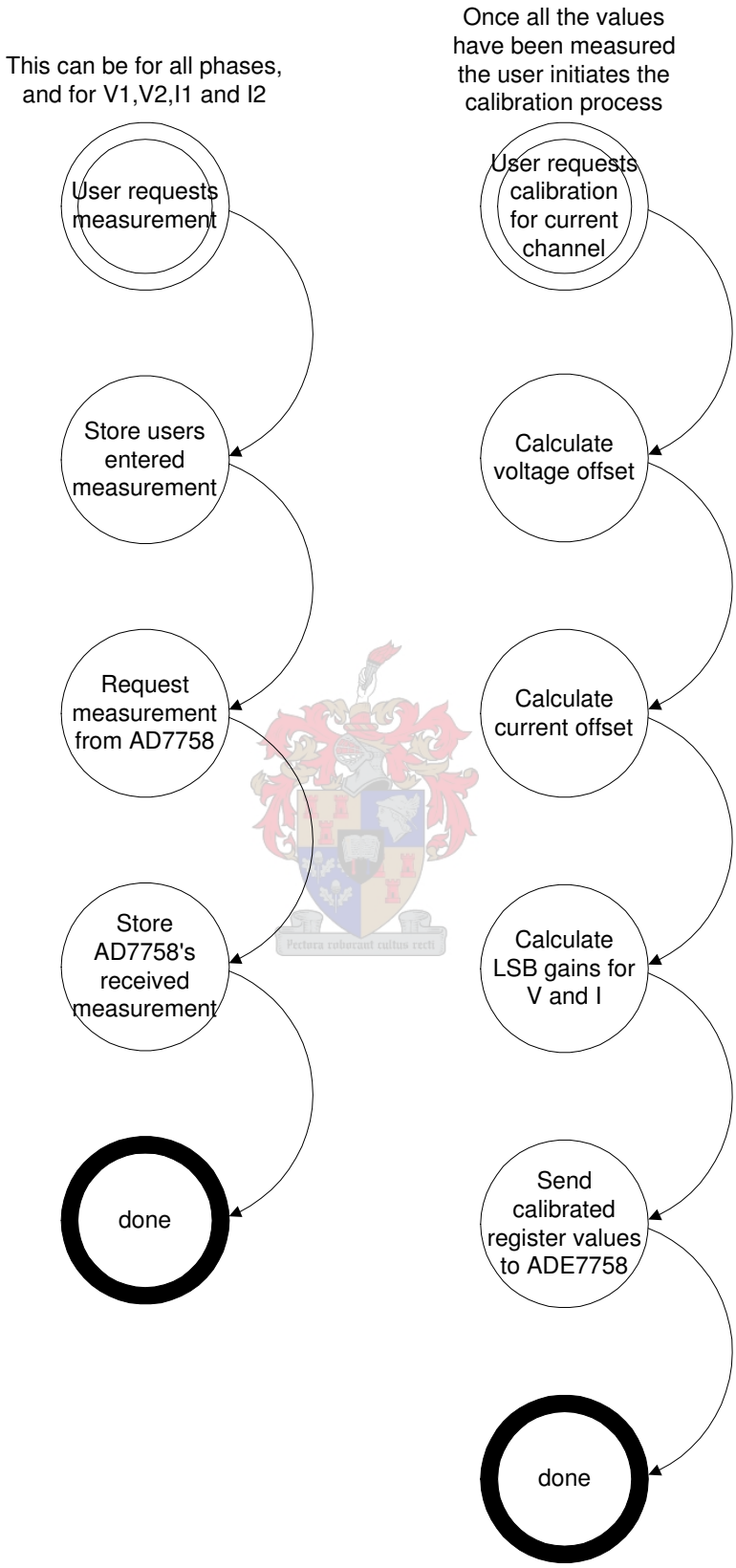


Figure 6.37: *Voltage and Current Offset Calibration process*

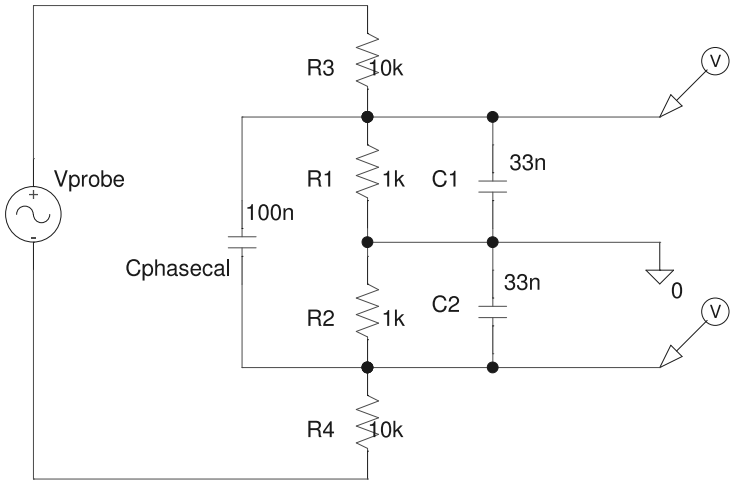


Figure 6.38: *Phase Calibration Hardware Implementation*

The required lead capacitor can roughly be calculated using the RC constant as follows:

$$Phase_{lead} = 3.6\,^{o} \tag{6.6}$$

$$t_{lead} = \frac{Phase_{lead}}{360^{o} f_{line}} \tag{6.7}$$

$$= 200\ \mu s \tag{6.8}$$

$$C = \frac{t_{lead}}{R} \tag{6.9}$$

$$= 100\ nF \tag{6.10}$$

where $Phase_{lead}$ is the necessary phase lead required, t_{lead} the lead time required, f_{line} the line frequency at $50\ Hz$ and C the required capacitance to realize the phase lead.

Software implementation

In the graphical user interface the phase calibration screen is used. This screen allows the user to enter two power measurements at two different power factors. Each of these measurements can be read from the ADE7758 using the appropriate button. Once all the measurements are taken the calibration calculations can be done and the registers of the ADE7758 can be updated.

The phase error is calculated in `calPhaseOffset` using the following equations. For the procedure the following values are required for each measurement:

pf_x the power factor of the measurement

W_x the measured power

$W_{x_{ADE7758}}$ the power measured by the ADE7758.

With this calculation the first measurement should have a power factor as close to $pf \approx 1$

as possible and the second measurement should be near $pf \approx 0.7$.

$$Error = \frac{\frac{W_1}{W_{1_{ADE7758}}} - \frac{W_2}{W_{2_{ADE7758}}}}{\frac{W_2}{W_{2_{ADE7758}}}} \tag{6.11}$$

$$\Delta pf = pf_2 - pf_1 \tag{6.12}$$

$$phaseError = \arcsin \frac{Error}{\sin \Delta pf} \tag{6.13}$$

If the phase error is negative

$$phCal = \frac{phaseError}{(2.4 \times 10^{-6} \times 360 \times 50)} \tag{6.14}$$

otherwise

$$phCal = \frac{phaseError}{(4.8 \times 10^{-6} \times 360 \times 50)} \tag{6.15}$$

The phase calibration register, $phCal$, is updated in the ADE7758. To further refine the values the register can be updated directly.

6.7.4 Gain Calibration

The final gain calculation stage is implemented in the Energy Measurement Driver. The code for calculating these registers is explained in Section 6.4.4. This step is a one point LSB_{gain} calibration. The meter is expected to send values in Watt Hours. Each LSB should represent one Whr, $VARhr$ and $VAhr$. The calibration values are entered into the calibration software with a $0.7 < pf < 0.76$ to have an active and reactive energy component. This measurement should be taken at nominal current and voltage as specified for the meter and current probes. Once all the measurements are taken the calibration constants can be calculated and the registers updated.

This measurement is done over a 250 line cycle period. The energy measured in the ADE7758 is then used with the externally measured power values.

The energy division registers are calculated as follows:

$$t_{totalCycle} = 250/(2 \times f_{line}) \tag{6.16}$$

$$LSB_{gain} = \frac{energy_{ADE7758} \times 3600}{t_{totalCycle} \times power} \tag{6.17}$$

where the power already takes the power factor into account.

The registers are now updated into the micro controller.

6.8 Calibration and Test Results

The calibration was done using a Voltec PM3300 Universal Power Analyser as reference. The system was calibrated using a three phase variac, with three paired $900\ W$ loads and a $25\mu F$ per phase capacitor bank to provide for a leading power factor.

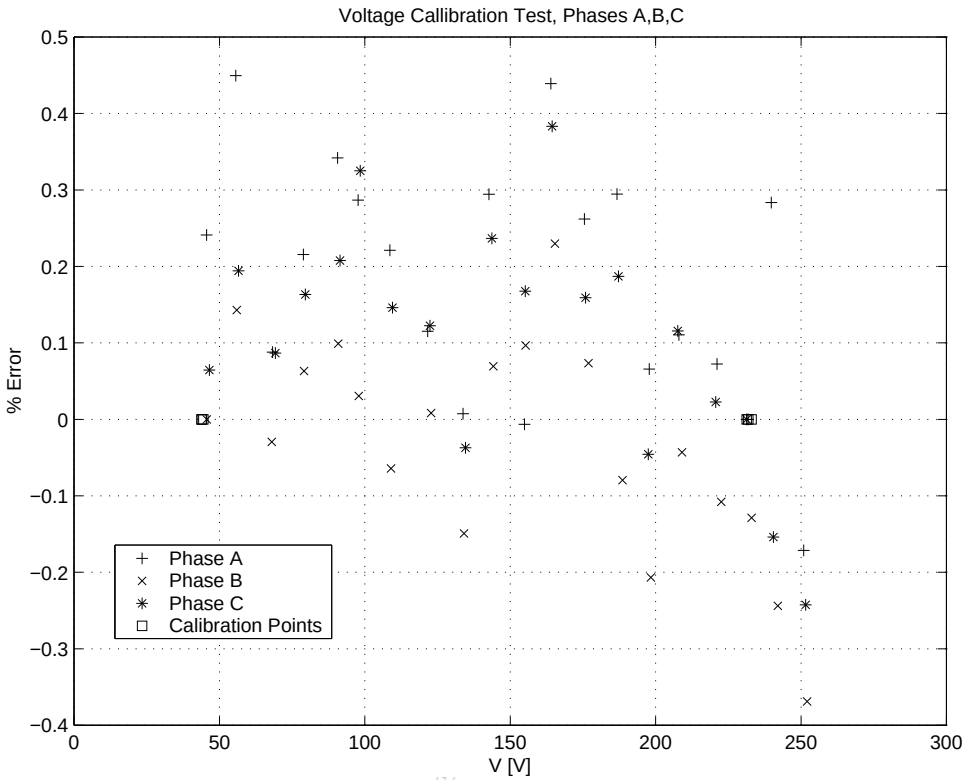


Figure 6.39: *Voltage Calibration results*

6.8.1 Voltage Calibration

The voltage calibration was done using the points in Table 6.5:

Measurement	Phase A	Phase B	Phase C
V_1	231.3 V	233.0 V	231.9 V
V_2	43.85 V	43.94 V	44.49 V
I_1	4.107 A	4.144 A	4.000 A
I_2	399.0 mA	404.0 mV	428.9 mA

Table 6.5: *Voltage and current offset calibration*

The results from the voltage and current calibration experiments are shown in Fig. 6.39 and Fig. 6.40.

Around the nominal point an error of $0.25\ \%$ is seen for the voltage and $0.3\ \%$ for the current calibration. This matches the typical expected error as provided in the data sheets [7].

6.8.2 Phase and Gain Calibration

The phase and gain calibration is done using the resistor and capacitor bank to absorb the required energy.

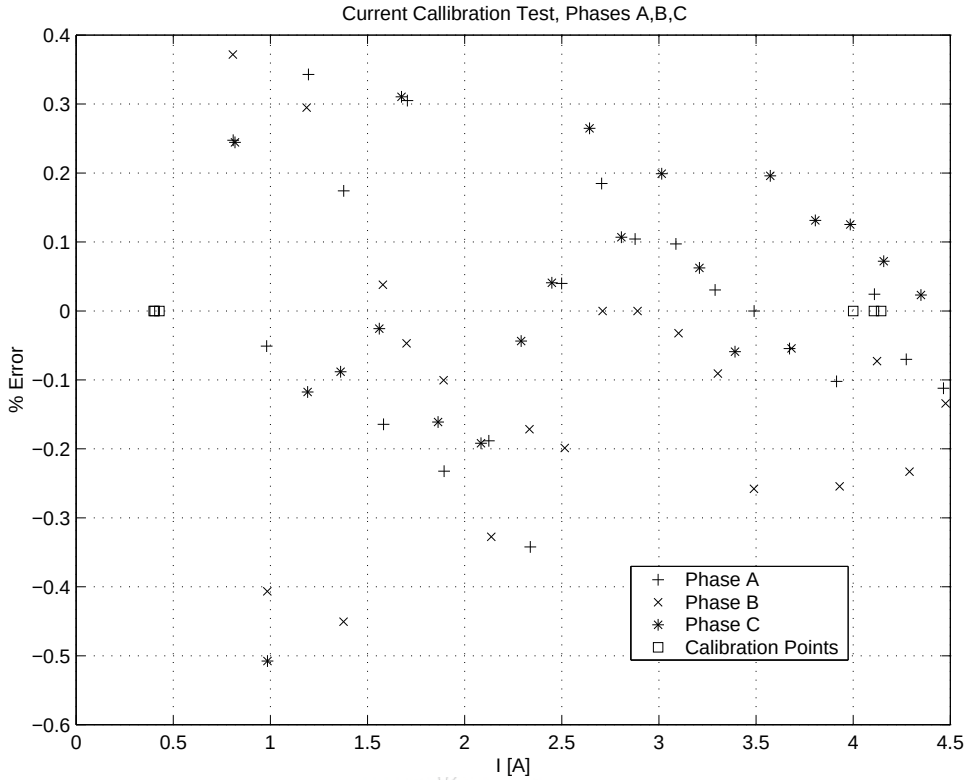


Figure 6.40: *Current Calibration results*

Prior to the gain calibration the phase calibration is done by setting the cycles for the calibration process to $1\ 000$. Two measurements for each phase are made and are shown in Table 6.6

Energy Register	Phase A	Phase B	Phase C
PF_1	0.982	0.986	0.984
W_1	350.2	380.6	373.4
$W_{1_{ADE7758}}$	0x0149	0x0161	0x015C
PF_2	0.715	0.728	0.734
W_2	351.8	375.0	376.4
$W_{2_{ADE7758}}$	0x0144	0x0161	0x0156

Table 6.6: *Values used for phase calibration*

The gain calibration procedure is done at $V_{nom} = 223\ V_{RMS}$ and $I_{nom} = 2.15\ A_{RMS}$ and fine tuned by allowing a $16\ min$ energy accumulation.

The results when testing the phase and gain calibrated check-meter yielded the results in Table 6.7.

Due to quantization and test synchronization errors caused by the radio link the percentage

Power Factor	Voltage [V]	Current [A]	Active E. [Whr]	Reactive E. [VARhr]	Apparent E. [VAhr]
0.7	223	2.15	1.30 %	0.48	-0.08 %
0.984	223	1.66	2.27 %	-8.47	0.09 %
0.984	225	1.66	1.49 %	-9.19	0.04 %
0.987	225	3.56	1.39 %	-11.99	-0.74 %
0.987	225	3.56	1.39 %	-12.58	-0.54 %
0.942	225	3.73	0.39 %	6.67	-0.29 %
0.003	225	1.84	0 %	-2.27	1.46 %
0.939	189	3.16	0.79 %	6.67	-0.63 %
0.939	189	3.16	1.47 %	-12.65	-0.91 %

Table 6.7: *Percentage Error per accumulator*

error decreases as time increases as can be seen in Fig. 6.41. Fig. 6.42 supplements Fig. 6.41 by showing only the active power component and the error associated with this accumulator. These results show that there is room for improvement with these measurements. For the purpose of this thesis, however, the results attained are good enough. Incorporating the energy accumulators offset register can be investigated to improve the measurements.

6.9 Conclusions and Recommendations

The check-meter was developed and tested according to the requirements.

The calibration tests gave good results for the voltage and current measurements, but the energy accumulator's results are only fair, and in this area there is room for improvement. An additional active and reactive energy offset can be considered to further increase measurement accuracy.

The construction, assembly and implementation requires some refinement, and should be tested for weather-proofing prior to any field tests.

The check meter is designed in such a way that it can easily be expanded or manipulated for other applications. Possible applications could include peak and sag or dip detection, which can be used for flicker and quality of service measurements. A general data logging capability is also a possibility. An auto memory dump should also be considered when power is removed from the meter. However timing and energy management should then quickly be applied to ensure the integrity of the measurements. The possibility of sending an off status SMS can also be investigated.

Combined with the developed data mining application this device can prove helpful for field analysis and energy measurements in remote areas.

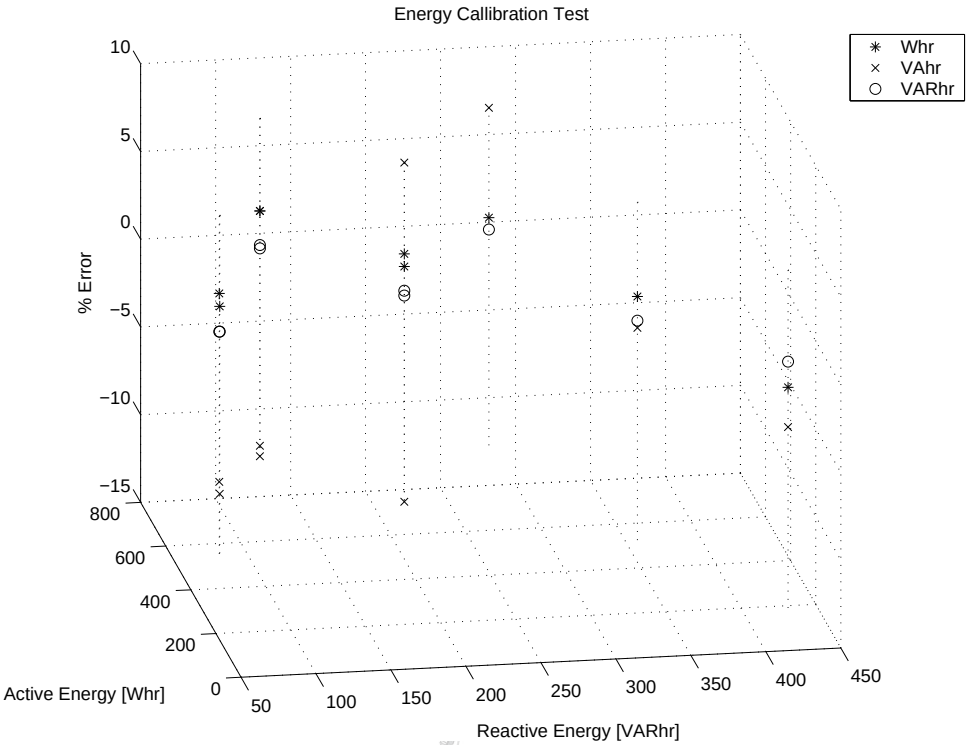


Figure 6.41: *Gain Calibration with Phase correction results*

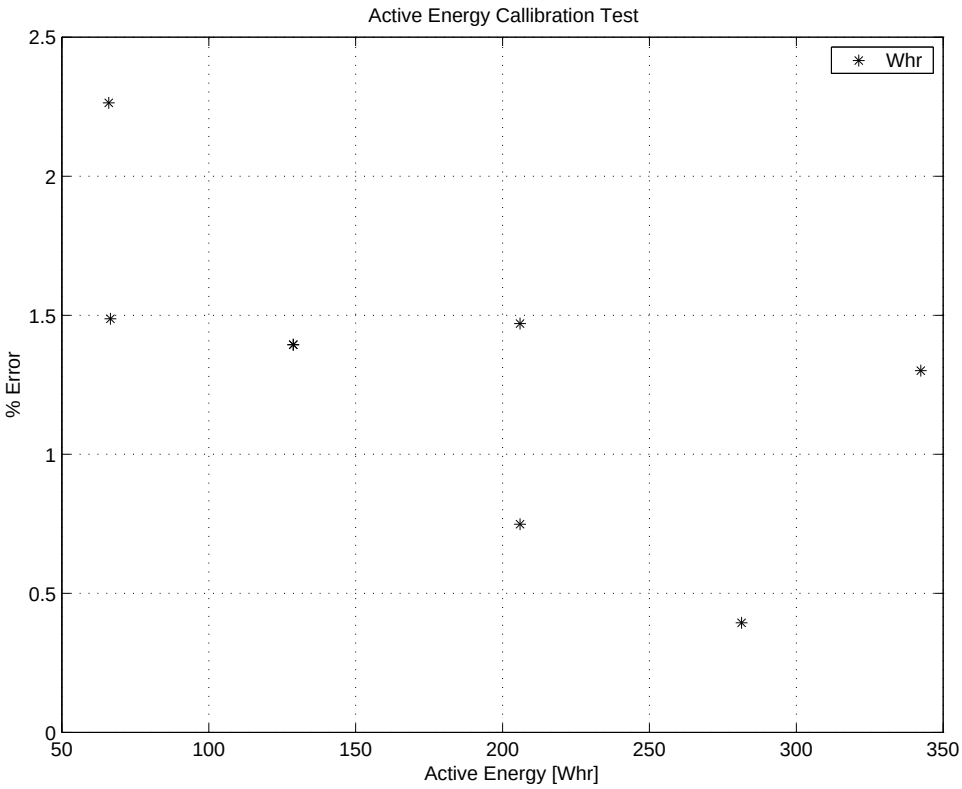


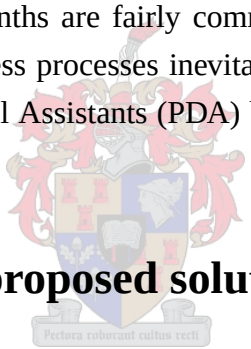
Figure 6.42: *Active Energy Calibration results*

Chapter 7

Data Collection Software Development

7.1 Introduction

Detection of electricity theft based on previously developed data mining solution depends on the quality of the data. Conventional meters are read by meter readers roughly every month, but periods longer than three months are fairly common. Longer measurement periods and a number of possible erroneousness processes inevitably have an effect on data quality. This chapter presents a Personal Digital Assistants (PDA) based solution, whereby data in the field is captured accurately.



7.2 Motivation and proposed solution

To bill consumers for their electricity usage means that their electricity meters must be read frequently and accurately. This is typically done by meter reading personnel who walk along the route and write the required values into a notebook. The readings are then entered into the municipality's debtor system by clerks in the back office.

The data collection involves two human processes. Each of these processes are prone to error. By using a hand held terminal this can be reduced to one process. The amount of errors can further be reduced by incorporating simple checks and intelligence into the hand held terminal before validating the data on the system. Reducing errors not only saves money in terms of incorrect bills and the client service associated with incorrect bills, but time is also saved by reducing the number of meter reading iterations per bill.

The idea to use hand held terminals is well known [54, 22, 95]. Models are currently available on the market, but these are expensive and bulky[95]. The recent advance in PDA technology and the affordable pricing of these units make it economically viable to implement a PDA based meter reading solution.

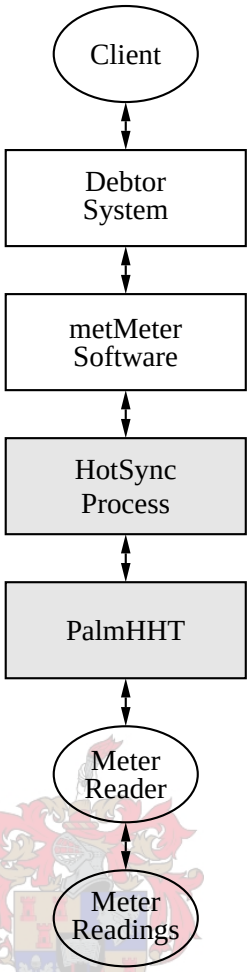


Figure 7.1: *Process Flow and System Overview*

7.3 Functionality of System Components and Platform

An overview of the proposed system is given in Fig. 7.1 with the system development for the thesis shaded in gray. The human processes are indicated in the ellipse areas.

The system comprises of the following components:

PDA The palm is responsible for collecting the data in the field.

PDA Software The software used for capturing and storing the data on the PDA.

Conduit The software necessary for uploading and downloading the meter readings out of and into a specified database of the metMeter Software.

7.3.1 Personal Digital Assistants

The PDA, introduced in 1993 [37] by Apple, is a small device able to provide mobile computing capabilities. Various products are available and this market is changing at a rapid pace with new

models being released daily. Because of this development it was decided to use a Palm OS based product, since Palm has been successful in this area since 1996 [61] and has provided an economical product. The development is tested on the Zire 21 and Zire 71 (Fig. 7.2) series.

Figure 7.2: *The Palm Zire 71 PDA*

7.3.2 Palm Software

The software development platform chosen for the project is Java[1] [86]. Java software can be ported to other platforms without major changes, since these devices are developing rapidly. Code can be written easily in a reusable manner. To run the software on the palm the Java code is compiled into a JAD file by the Java compiler. This file is then converted with the converter tool [86] into the Palm PRC file. The “palmHHT.prc” and “MIDP.prc” files are uploaded to the palm. The application is started by clicking on the “palmHHT” icon.

The meter reading software operating on the palm has a simple interface to even allow users with basic skills to enter data into the system.

The simple screen is displayed to allow the user to enter data as shown on Fig. 7.3. The following fields are displayed to aid the users in the entering the correct data:

Owner The owner of the premises or meter

Address The physical address of the meter

Meter ID The meter number, for example if more than one meter is located at the same address.

Meter Type Several types of meters can be read with this system, for example W-Water, E-Electricity and K-kVA or maximum demand meters[2].

---

[1] J2ME MIDP for Palm OS

[2] The meter type is indicated in the square brackets on the screen

Figure 7.3: *Palm software screenshot for user input*

Account Number The owner's account number.

Meter Number The serial number of the meter.

Messages Warning messages or special instructions are displayed in these fields, for example, "Fetch the keys at the neighbor's".

Depending on the specific setup[3] the following input fields are available:

Reading The reading displayed on the meter must be entered here. If the value is out by $\pm 10\%$ of the average consumption, the user will be requested to re-enter the reading.

Messsage This field allows the meter reader to indicate wether there is a reason for an incorrect or complete lack of a reading. For example if the meter is broken or locked.

New Message If a message different to the messages stored on the palm is required, the meter reader can create a new message and store it on the device.

Users can skip forward and backward between records with the BACK and NEXT button. The data is validated by pressing the OK button.

7.3.3 HotSync Conduit

Synchronization of the data on the palm is done with Palm's HotSync process. This software is supplied by Palm with the palm device for the Palm Desktop Environment. Each different program or application is required to have a conduit. The conduits knows the data structure of

[3]The setup is contained in the ".prc" file and generated for each user

the application on the palm, namely where and how the data should be stored or received on the PC. Fig. 7.4 shows the HotSync process screen.

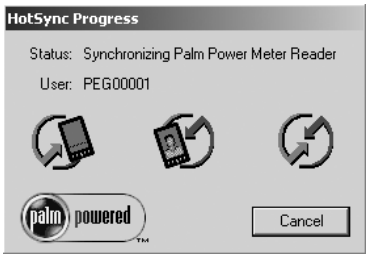


Figure 7.4: *A screen shot of the HotSync process*

The conduit was developed using Palm's JSync Java Conduit Development Kit [63].

When synchronizing, the conduit uses the specific palm unit's username to determine which route should be downloaded and uploaded. The completed route and messages are then downloaded and stored in flat files, ready for post processing by the metMeter[4] software. Next the new route, prepared by the metMeter software is uploaded and the messages for the specific route are updated.

7.4 Software Design and Implementation

The software of the system is broken up into two sections namely the Palm Software and the HotSync Conduit, which are discussed in the following subsections. Fig. 7.5 is shown to help the reader understand how the palm software fits into the rest of the system.

7.4.1 Palm Software

Program Flow

To program the hand held device the Mobile Independent Device Profile (MIDP) is used. This profile provides a common platform for programming mobile devices in Java to allows cross-platform portability. In the Java or MIDP environment an object is created which extends the `midlet` object. This object is first constructed in the MIDP environment and stays resident until the application is removed from the Palm[5]. On the Palm the program data and state are preserved when the power is switched off and program execution continues after the power is switched on later.

[4]This is the back-office software which interfaces with the debtor system and is used with permission from MetGovIS

[5]Source code can be found in Appendix Q.3

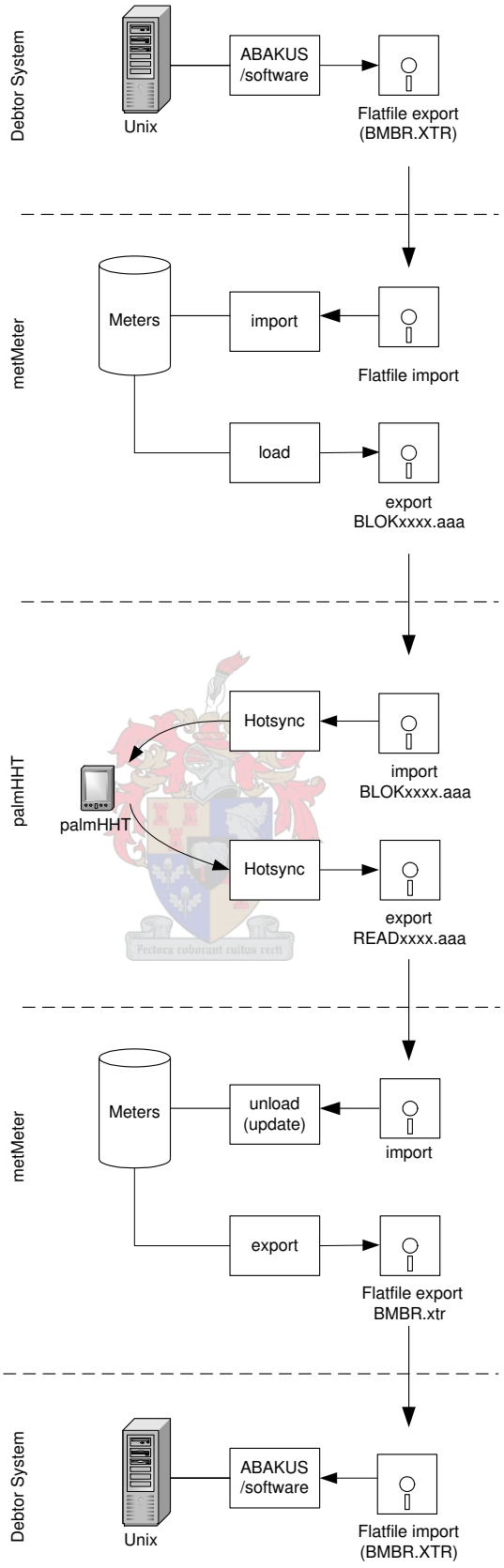


Figure 7.5: *The hand-held system within the total system*

As mentioned above, the program or object is initialized with a constructor `palmHHT()`. The constructor is responsible for ensuring that valid records are in place for the route, before the program can continue as shown in Fig. 7.6. If the route is not correctly uploaded a synchronization process should be initiated by the user. The constructor also sets up the first screen to be displayed (See Fig. 7.3).

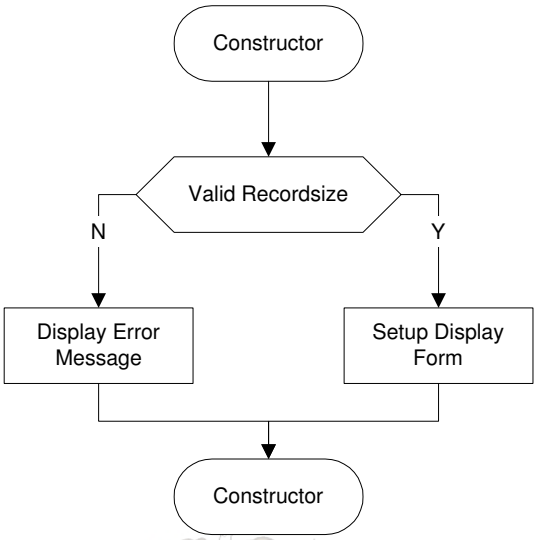


Figure 7.6: *Flow Diagram of Palm Software Constructor*

Each time the program is called or started by MIDP the `startApp()` method is called. The `startApp()` method activates the first screen and invokes the command listener.

The objects on the form or screen can be updated by the user. Once one of the buttons is ticked with the stylus, the `commandAction(...)` method is called and the events serviced. Fig. 7.7 shows the flow diagram of the implemented code.

Data structures

All data in the data records are stored as strings. This makes importing and exporting to and from the data structure simple. The fact that Strings are used does however require additional processing on the palm.

The data structures are stored in a resizable array on the palm. The data structure is accessed with the MIDP `RecordStore` object. The `RecordManager` object is created to encapsulates the `javax.microedition.rms.recordStore` for easy access to the stored data. This method uses the enumeration[6] method of the `RecordStore` to access the data. This is the only object allowed to directly access the records store.

In the palmHHT software the messages are stored in the `MessageInfo` object. The `HeaderInfo` and `MeterInfo` object are used to store all the route specific data. The

[6] A way to put the records in a sequence

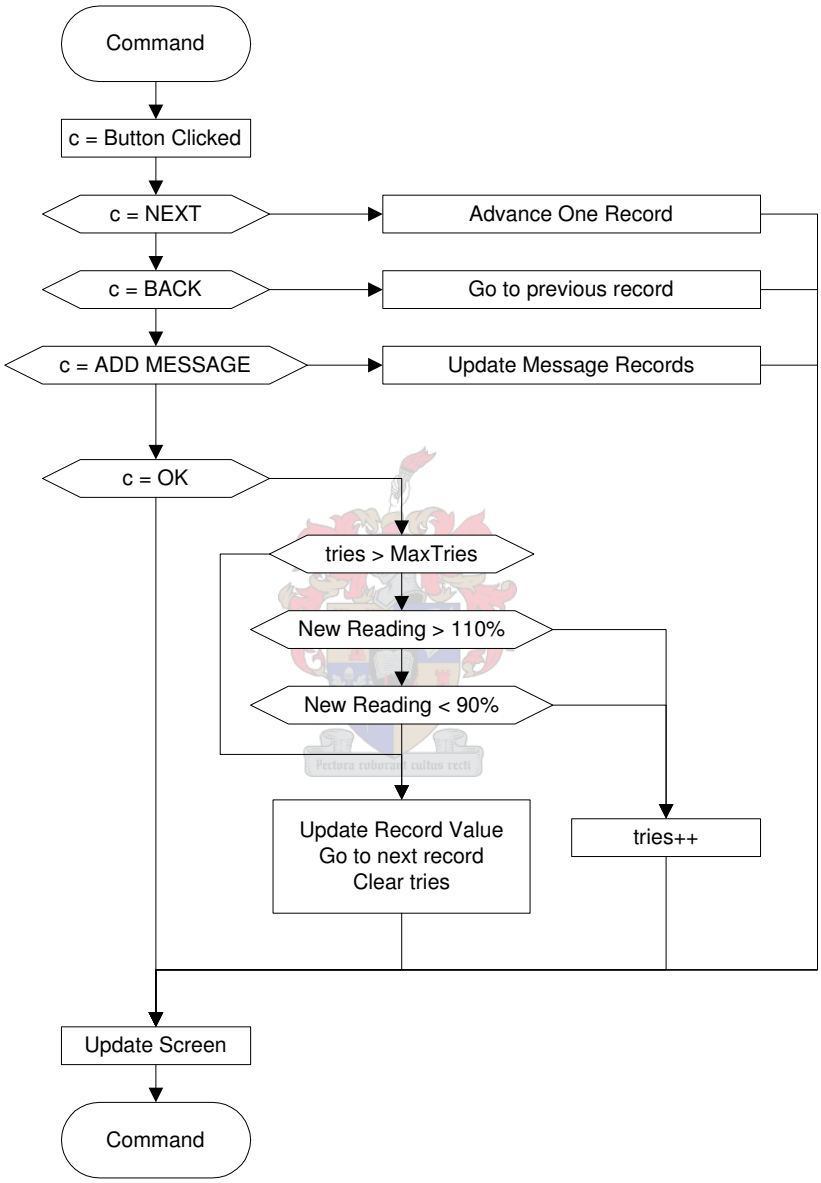


Figure 7.7: *Flow Diagram of Command Action Event*

`MessageRecords` and `RouteRecords` inherits methods to access the `RecordStore` from the `RecordManager` class. Fig. 7.8 illustrates the object components and inheritance of the data objects.

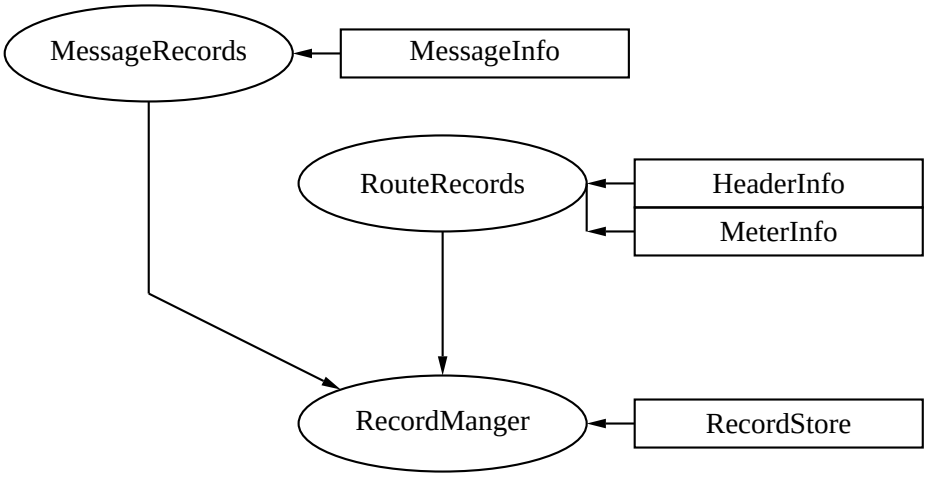


Figure 7.8: *PalmHHT software Data Structure*

The complete source code for the Palm Software can be found in Appendix Q.3.

7.4.2 HotSync Conduit

Program Flow

The HotSync code implements the Conduit abstract class which initiates the synchronization process by calling the `open(...)` method in `HHTCond`. An overview of the program flow is given in Fig. 7.9 and can be referenced in Appendix Q.4.

The setup is managed by the `green.HHTSync.HHTSyncGUI` object and creates a XML file containing the setup on the specific machine.

Prior to loading any data, the correct path, where the data files are stored and retrieved, is loaded from the setup file. First the palm User is retrieved from the palm to ensure the correct configuration file is read for the palm[7]. This file contains the header information which can now be uploaded onto the palm.

The data of the route completed prior to this synchronization process is now downloaded into a flat file. The route number is used to identify this block of data and the flat file is saved as “READxxxx.dat”, where xxxx is the route number.

With the old data stored the new data can be uploaded. This route number is used from the header information for the new block of data. The data from the “BLOKxxxx.DAT”, where xxxx is the new route number, is uploaded onto the palm.

[7]This is an “username.PCT” file

The last process downloads all the messages from the old route in a file “RTMSxxxx.DAT” and uploads this route’s messages from “MESSxxxx.DAT” where xxxx is the route number of both the old and new route.

Supporting methods are also included in the HHTCond and can be seen in the source code in App. Q.4.

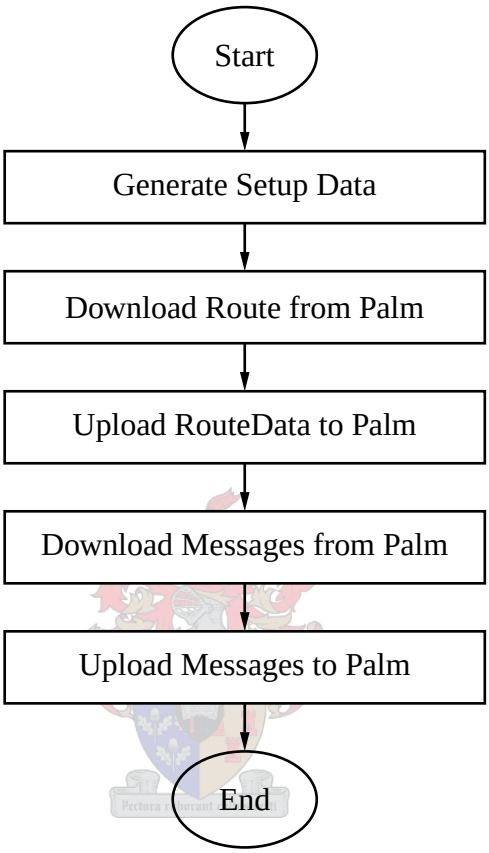


Figure 7.9: *HotSync software proces flow*

Data Structures

The data structure on the palm is accessed in the conduit by using the `StringRecord` which inherits the `palm.conduit.AbstractRecord`. Records on the palm are accessed using data IO streams. Each record in the `StringRecord` class is encoded or decoded by using the `setString()` and `getString()` after the record’s index is specified by `writeRec()` and `readRecByIndex()` of the `SyncManager()` class.

Two different record stores or sets are used, namely the `MessageRecords` and the `RouteRecords`. The `MessageInfo` records are stored sequentially. The `RouteRecords` contains a `HeaderInfo` part. Each `MeterInfo` is added sequentially after the `HeaderInfo`.

The data required for import is contained in a flat file generated by the metMeter software from the Debtor System. The format of these files have a fixed width and are accessed

by the `flatFile` class. The import of data by the `flatFile` class is done by defining a `dataStructure` array with the width of each field for example,

```
private int[] dataStructure = {12,3,14,4,1,18,2,8,8,8,3,4,8,2,8,1,6,1,3,3,3,8};.
```

This structure is then referenced to retrieve the number of fields and length of each field per line. Each line is then iterated and added to the route on the Palm to form the Route.

7.5 Installation

In order to get all the different components together an installation procedure is required. It was found that a solution from `http://www.handx.net` is most suitable for the application. The installation procedure is programmed in a XML file, which can be found in Appendix P.1.1. It was found that the JSync software is designed with Java 2 Standard Edition v 1.3. To be compliant with code created for the conduit the J2SE v 1.4 is required. In the installation process the Java 1.3. virtual machine is replaced with Java 1.4.

7.6 Testing and Implementation

Functionality of the system is tested according to the functional description and specification. The software is also tested according to the Palm Powered Solution Test [62]. The results of the tests can be found in Appendix O.1.

The system is currently undergoing tests at Theewaterskloof Municipality[8]. The software is provided in a single file compiled by pInstaller [38]. A training session was presented in Caledon to empower the Meter Reading Personnel. See Appendix N. The installation instructions can be found in Appendix M. The documented source code for this chapter is included in Appendix Q.3, Q.4, Q.5.2, Q.7.3 and Q.8.3.

7.7 Conclusion

By using the PalmHHT software with the combination of the metMeter post- and preprocessing software, the data collection of meter readings is optimized by reducing the number of human processes and consequently increasing accuracy. By increasing the accuracy and data quality, billing can be done more accurately. Also, using cleaner data should yield better results when investigating electricity theft. Both consumer and utility will gain from this investment.

[8] October 2004

Chapter 8

Simulations and Measurement results

8.1 Introduction

The testing and simulation of the data mining software and the testing of the total integrated system is discussed in this chapter.

The first part of this chapter focuses on the idea of a mobile remote check-meter in the field and uses data from Mooiwater in Franschoek to simulate the feasibility of such a system.

The second part integrates the complete system and shows how measurements can be made.

8.2 Simulation Functionality

8.2.1 Introduction

In order to determine the feasibility of implementing a mobile remote check-meter and to illustrate the capability of the developed package in Chapter 5, a simulation function was added, since no check-meter was available.

The simulation capability allows a user to simulate conditions on the network by introducing a random process in combination with real data. This required a histogram function to be included in each node as well as a random function generator [71] (see Section 5.4.3).

The calculations are iterated through and the values are calculated using the `toDataCollection` method. After each iteration these values are placed in the histogram. The iterations are started by pressing the CALC button and setting the number of iterations required for the simulation. The SHOW HISTOGRAM button is used to plot the histogram.

With this functionality [20] the capability of the software was illustrated and it simulated the technical losses in the network, according to Fourie's [33] method, in order to determine possible outputs of a check-meter for electricity theft detection, prior to the construction and implementation of such a meter.

8.2.2 Methodology

Variables

With the consumption data ($E_{consumed}$) available from a test site it is possible to simulate data for the required variables, for example the check-meter (E_{in}), technical losses (E_{tl}) and non-technical losses (E_{nontl}). The technical losses are calculated as described below:

- **Check meter data** With no implemented check-meter, the first step is to simulate a check-meter (E_{in}), by using available field data from consumers and by simulating the losses in the network. The data for the modeled check-meters consist of technical losses (E_{losses}) and metered energy usage ($E_{consumed}$) as in (5.1), where E_{losses} consist of the technical (E_{tl}) and non-technical losses (E_{nontl}).

  For the calculation of the check-meter reading it is assumed that all these meters are fully functional and no meter is omitted from the database. Therefore, $E_{consumed}$ is known and E_{nontl} is assumed to be zero in (8.1).

  $$E_{in} = E_{tl} + E_{consumed} + E_{nontl} \tag{8.1}$$

- **Technical losses** E_{tl} for a low voltage reticulation network, in general, is the algebraic sum of the loss contributions of all the consumers' respective phase resistances and the vectorial sum in the neutral resistance [33]. The main factor contributing to losses in the LV reticulation network is due to feeder losses (I^2R) [33]. Depending on the load and usage patterns of consumers, the losses will vary. E_{tl} is therefore approached with a Beta probability density function as shown by [33]. Although other methods, such as as Heunis [40] suggested, are available, the method implemented in using this [33] was found practical. The mean or expected value (μ) and the standard deviation (σ) is calculated for the losses in the area using circuit breaker sizes as well as the number of consumer and cable ratings [2]. These are combined in order to calculate the total technical losses for the area.

- **Non-technical losses** Electricity theft is modeled by removing specific or random readings from the initial data set and thereby altering $E_{consumed}$ and E_{nontl} in a controlled fashion.

Thresholds

To determine whether action should be taken the feasibility of such action should be assessed. This is done by including the initial capital put into a check-meter and by calculating the losses in monetary terms for a year period. The threshold, in terms of how many losses are measured by the check-meter, are calculated to aid the decision process in terms of action planning required for determining a suitable method for eradicating the illegal connections in the area.

Simulation and verification

A simulation is done with a number of expected illegal users to verify the expected electricity theft in the area. This is done to verify that the simulation and data package will assist an electricity distributor to determine the extent of the electricity theft in the area. For the simulation a time period, t, in months, is chosen and a number of n iterations are done. This is then evaluated for each of the different configurations, by comparing the results from the experiment against the case where no electricity theft is present in the simulation.

The different nodes of data sources are linked together with the tree structure as described in Fig. 5.1. Each node can be altered as seen in Fig. 5.2, to extract actual data from a data source or generate random data formed by a PDF. The mathematical method of connecting to the tree can also be specified, for example add, multiply, subtract or divide in the node.

The following is setup (XML as in Appendix P.2.2) and used in the Data Minor package.

Check-Meter The check-meter is modeled using all the samples stored in the database. The technical losses are added using a Gausian distributed random variable as found in Section 8.2.3.

Consumers The consumer database is changed by randomly selecting a number of consumers and excluding them from the accumulated energy consumption as measured over the total period under investigation.

Loss calculation The losses can now be determined by subtracting the measured value from the check-meter from the changed consumer consumption of the area.

8.2.3 Modeling

Site data

Stellenbosch Municipality granted access to their consumer data for the purpose of this study. The site under investigation is Mooiwater at Franschoek. See Appendix K.1 for more information. This data was used to extract consumer data and validate the simulations. Data for October 2003 to December 2003 was used to create the histogram in Fig. 8.1. The data for the parameter calculation is grouped together per consumer, for example each consumer's consumption is added together for the sample to give an indication of the consumption diversity.

From the sample a user profile was calculated in Matlab®.

```
============================
data3monthGroupedByUser.txt
============================
From load current: Calculations
----------------------------
```

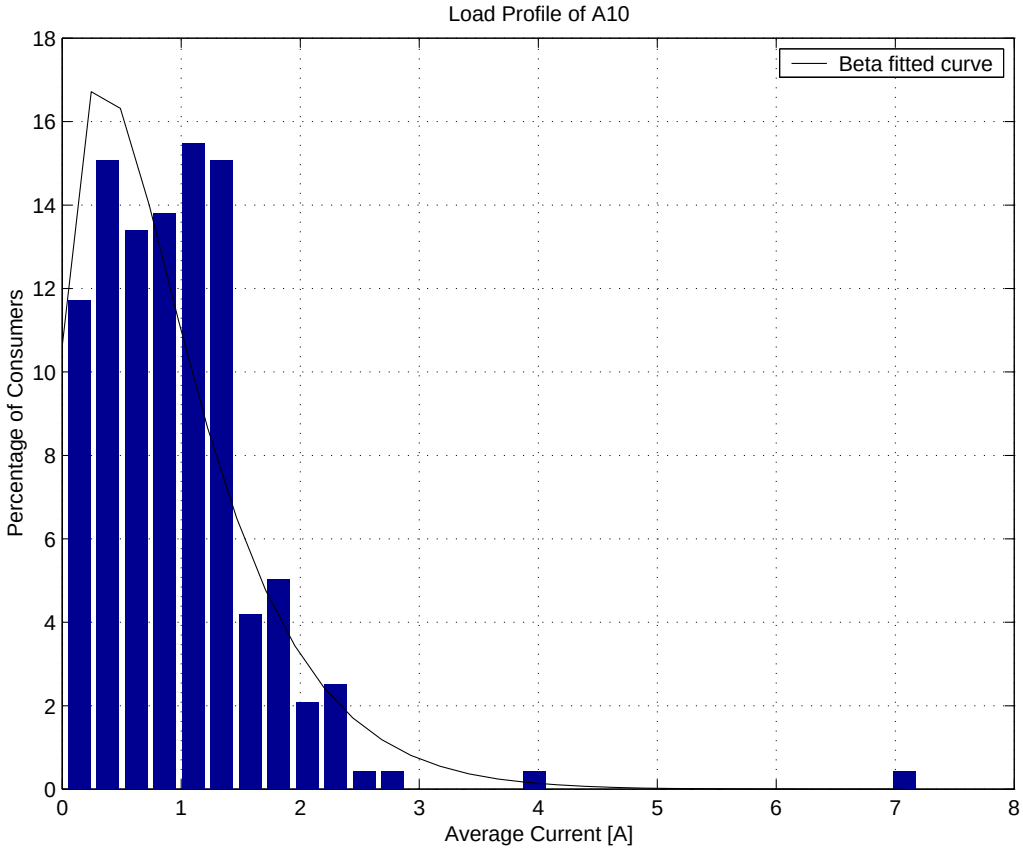


Figure 8.1: *Histogram of usage per consumer for Mini sub A10*

```
Connections   = 239[]
I RMS         = 1.215[A]
I Average     = 0.97938[A]
I VARiance    = 0.52544[A^2]
Average Power = 53836.5741[W per month]
Total Load    = 38762.3333[kWhr per month]
/Consumer     = 162.1855[kWhr]
alpha         = 1.6871
beta          = 32.7658

From beta: Calculations
----------------------------
I RMS      = 1.1114[A]
I Average  = 0.97938[A]
I VARiance = 0.52544[A^2]
```

where, from [33], the following is used:

$$\alpha = \frac{\mu\left(c\mu - \mu^2 - \sigma^2\right)}{c\sigma^2} \tag{8.2}$$

$$\beta = \frac{(c-\mu)\left(c\mu - \mu^2 - \sigma^2\right)}{c\sigma^2} \tag{8.3}$$

and

$$\mu = I_{Average} \tag{8.4}$$

$$\sigma^2 = I_{VARiance} \tag{8.5}$$

$$c = I_{max} \tag{8.6}$$

$I_{max} = 20\ A$ is used to scale the data and is normally an indication of the circuit breaker size.

Loss parameter calculations

Fourie [33] developed a spreadsheet solution (see Appendix K.3) to aid the estimation of the losses in the network base on Beta probability density functions used for E_{tl} simulation. From the spreadsheet the Beta attributes is taken and placed in the nodes in the software package of Chapter 5, to get a feel of the probability spread of the losses. Since a large amount of random functions are generated the histogram shown in Fig. 8.2, as expected, tends to have a very narrow and small standard deviation and approaches a Gausian probability density function with $\mu = 0.31\ \%$ and $\sigma = 0.03\ \%$. This simulation is done with $3\ 000$ iterations and the saved XML file is shown Appendix P.2.1.

These results were matched with simulations done in Retic Master 2004 and are available in Appendix K.

Threshold calculations and feasibility

The area under investigation includes 239 consumers. From the loss calculations, for this area and consumer base, the mean and standard deviation expressed as an percentage are $\mu \approx 0.31\ \%$ and $\sigma \approx 0.03\ \%$ of the total consumption.

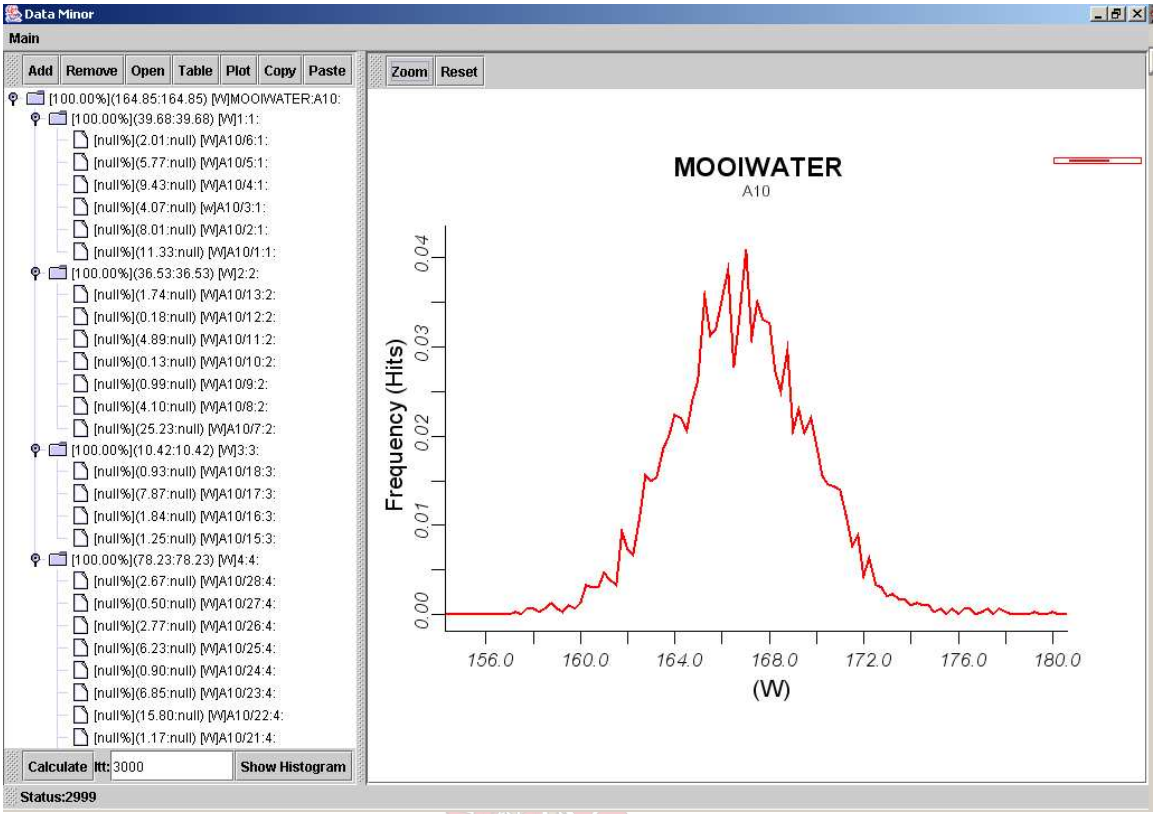


Figure 8.2: *Screen shot showing results of total technical losses simulation*

The cost of a mobile remote check-meter is estimated at $R\ 10\ 000$, since this device is mobile and can be moved to different areas. It is expected that this device can be moved four times a year. This suggests a cost per area of $R\ 2\ 500$. The selling price of electricity in this area is, on average, $R\ 0.33/kWhr$.

The cost per sweep is estimated at $R\ 27$ per consumer[1], which implies a total cost of $R\ 1\ 053$ per sweep. This includes the correction of any perpetrators. The total efficiency in terms of recurrence rate of illegal connections is assumed to be $10\ \%$.

Another option is to implement a DC reticulation network as proposed by Molepo [57]. This results in a cost of $R\ 700$ per consumer, but will require an extra rectifier circuit. Some other costs are estimated as shown in Table 8.1.

Network Simulation Results

The data from the Mooiwater mini substation MS A10 was used in the experiments.

For the simulation,the number of iterations $n = 500$ was selected. These were stored in a histogram for analysis. The output was also printed to the standard output buffer which allows easy data export to Matlab® or other software packages, if required.

[1]Cost of sweep received from Stellenbosch Municipality as an indicator

Threshold calculation			
Site data:			
Total Consumers	239		17
Total monthly consumption	38762	kWh	162.19
Loss percentage	0.31%		2757.15
Yearly Checkmeter measurement	466590	kWh	
Yearly User Consumption	465148	kWh	
Cost per kWh	R 0.33		
Check meter:			
Total Meter Cost	R 10,000.00		
Labour per shift	R 400.00		
Rotations per year	4		
Implementation cost	R 4,100.00		
Proposed solutions			
		Sweep solution	**DC Reticulation**
Assumed Efficiency		90.00%	100.00%
Overhead Costs		R 0.00	R 5,000.00
Cost per concumer		R 27.00	R 700.00
Total Cost		R 6,453.00	R 172,300.00
Including check meter		R 10,553.00	R 176,400.00
In terms of energy[kWh]		31979	534545
Energy including effieciency[kWh]		35177	534545
Percentage Threshold in terms of			
Payback period*		**Sweep solution**	**DC Reticulation**
1		7.54%	114.56%
2		3.77%	57.28%
3		2.51%	38.19%
4		1.88%	28.64%
5		1.51%	22.91%
*Payback period assumed with no devaluation of currency			

Table 8.1: *Spreadsheet for Threshold calculation*

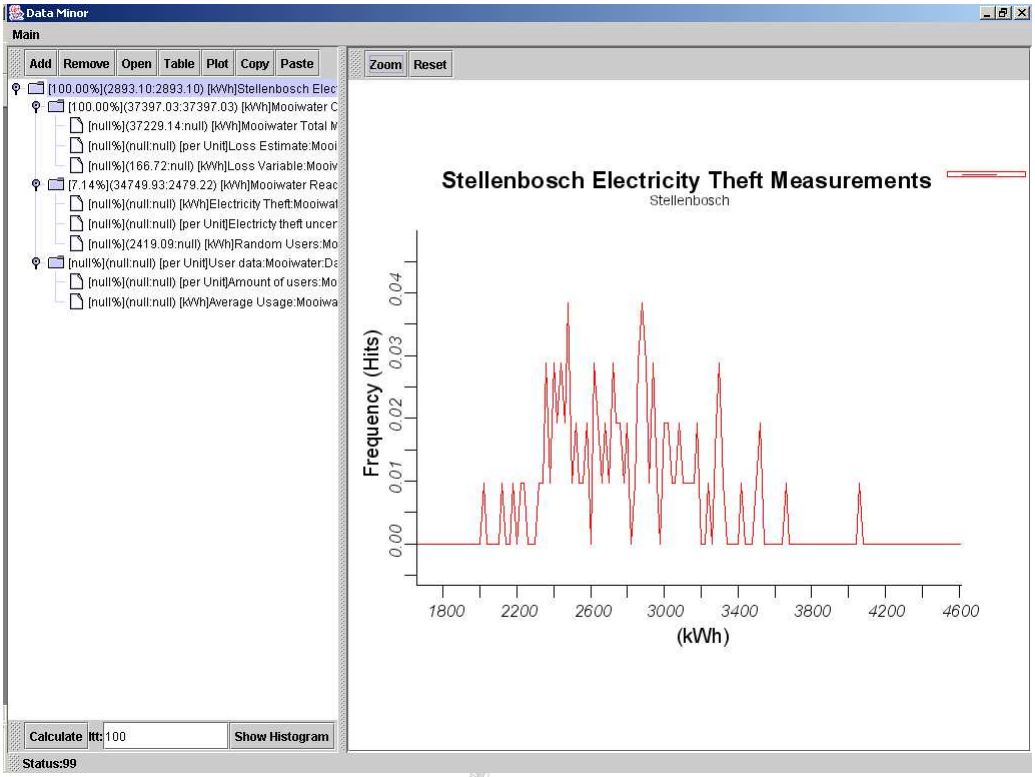


Figure 8.3: *Results from simulation with 17 illegal connections*

From the feasibility and threshold calculation it is shown that a minimum of $7.54\ \%$ losses should be measured by the check-meter. On average each user contributes 0.4% to the total energy consumption. This implies that roughly 19 consumers should actively benefit from illegal connections.

The results of the simulation with only 17 users, as in Appendix P.2.2, is seen in Fig. 8.3 and match the expected results.

8.3 Practical Laboratory Experiments

All the systems and subsystems are required to be tested and evaluated.

8.3.1 Setup

A laboratory setup was done to test the system as shown in Fig. 8.4.

The total interconnection of the system is shown in Fig. 8.5.

From the supply of the laboratory bench the Voltec Power Analyser is placed to confirm the measurements made by the check-meter and to indicate if power is switched on.

The check meter, as discussed in Chapter 6, is placed after $30\ A$ circuit breakers. These are used to isolate the meter from the power grid. A distribution board was made (see Fig. 8.6) to

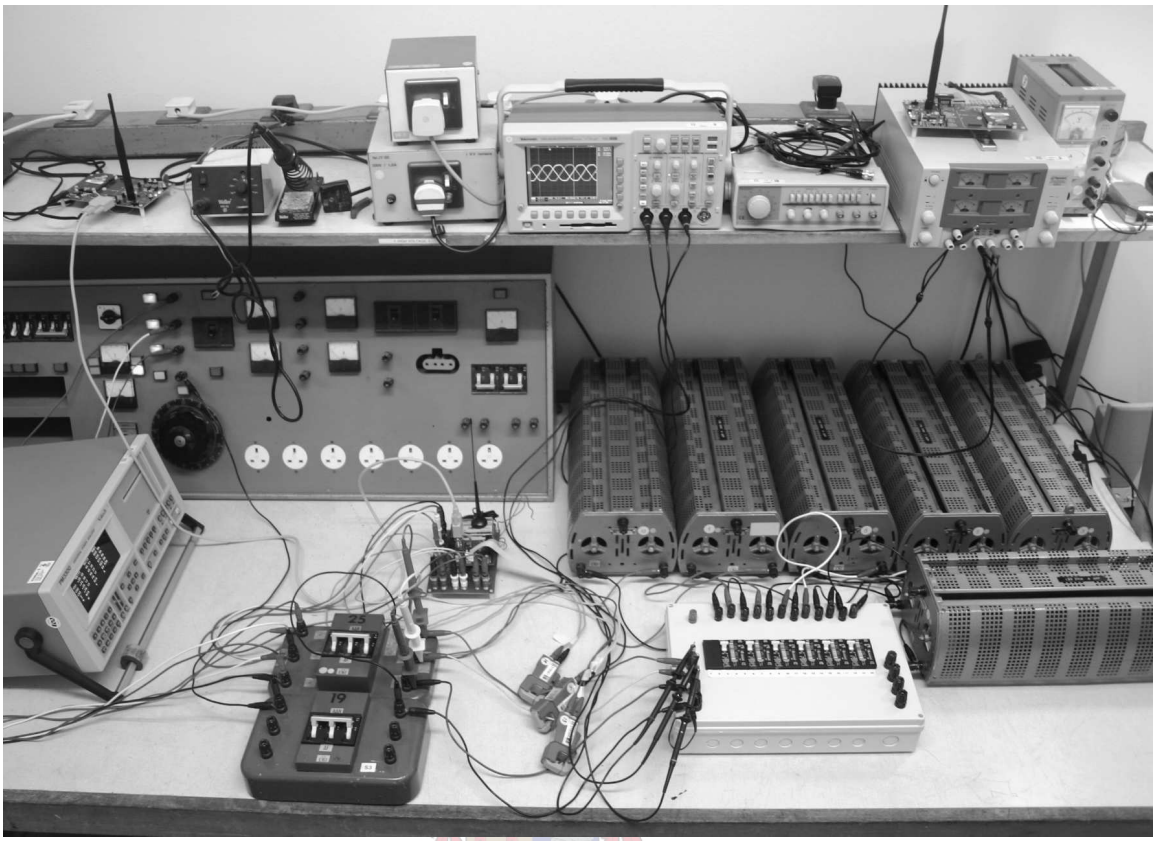


Figure 8.4: *Setup in laboratory*

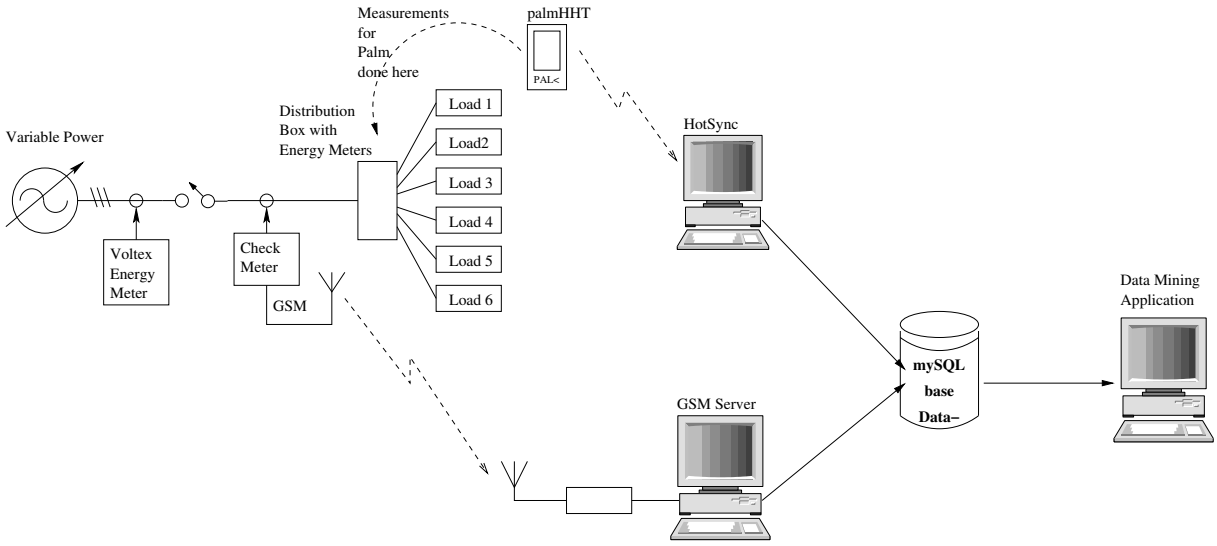


Figure 8.5: *Overview of test setup*

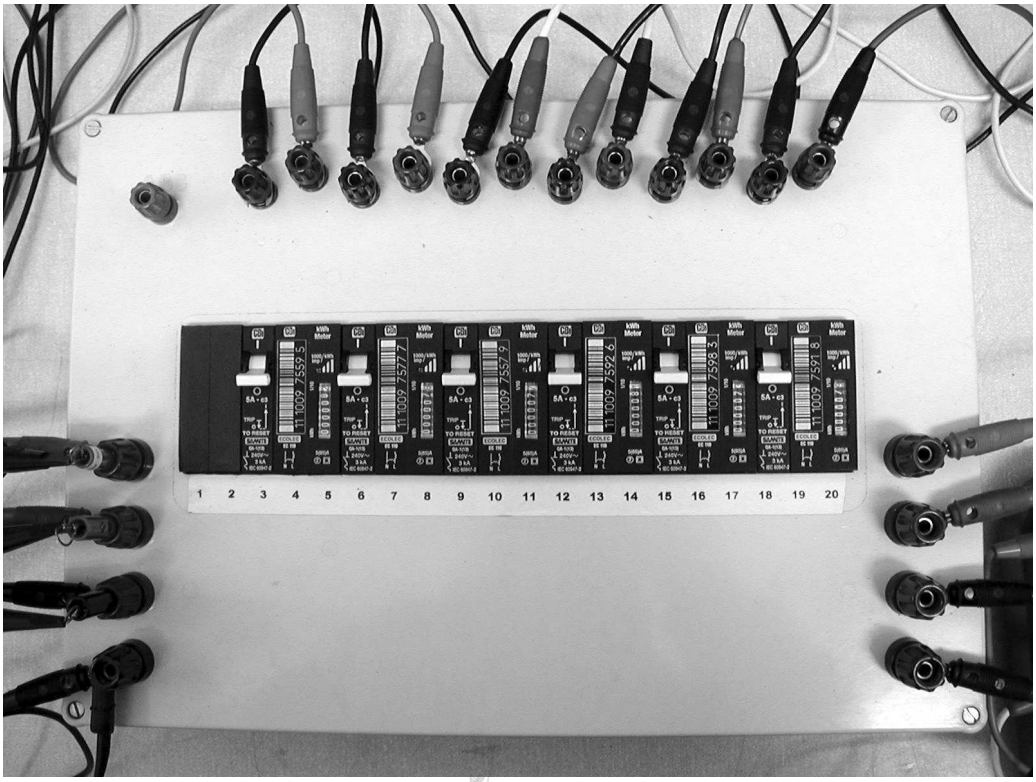


Figure 8.6: *Distribution board with energy measurement units*

allow individual load measurements to be taken. Six loads, capable of $I_{max} = 4\ A$, $R = 136\Omega$, are connected to the distribution board.

With the check-meter running the GSM server system is started. At preset daily intervals the server requests an update from the check-meter. Since the tests are conducted over an 8-hour period and requests are done on an hourly basis, the trigger times were advanced by an hour each time a request was made. In some instances during the tests the requests were made randomly. The data was captured on the database.

The load's individual measurements were made by using the palm's data collection software as described in Chapter 7. The data from the hand held is dumped into the database.

The results are presented using the data mining software developed in Chapter 5 and this completes the total information path.

8.3.2 Results

The results of the test are as expected. The total load measured by the individual meters and the check-meter match fairly closely. Some minor discrepancies were detected, but these can be attributed to the fact that the accuracy of the individual measurement units in the distribution board can only measure accurately up to $0.1\ kWhr$.

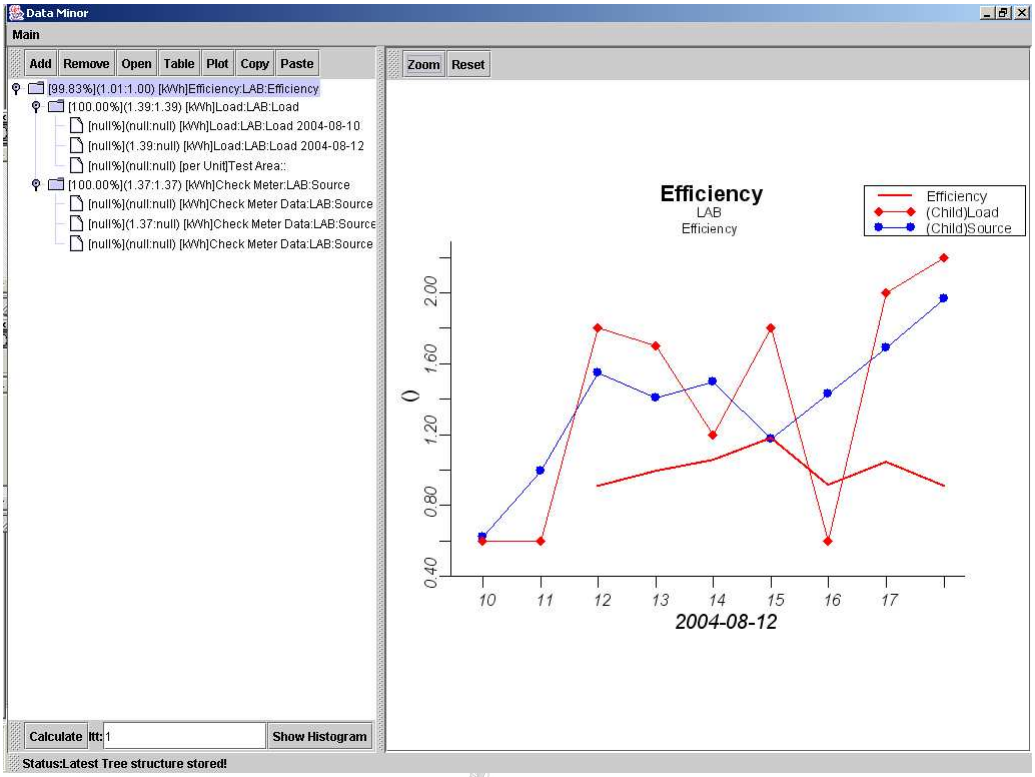


Figure 8.7: *All loads measured*

All metered loads connected

The first measurement set, as shown in Fig. 8.7, is done with the total load included. With a 3 sample moving average it can be seen that the average is around unity, as shown in brackets in text at the top node of the tree, indicating no electricity theft or major losses.

One load omitted from data set

The second set shown is done with one user disconnected from the database. This results in a $84\ \%$ efficiency and can be seen in Fig. 8.8.

8.4 Conclusions

The feasibility study done indicates that if appropriate eradication or action is taken, electricity theft can be addressed. The simulations helped with the modeling of the network and gave insight into the various aspects involved.

The test in the laboratory showed that the platform presented can be integrated successfully. The information path is demonstrated from measurement, data transmission and storage through to presentation and processing.

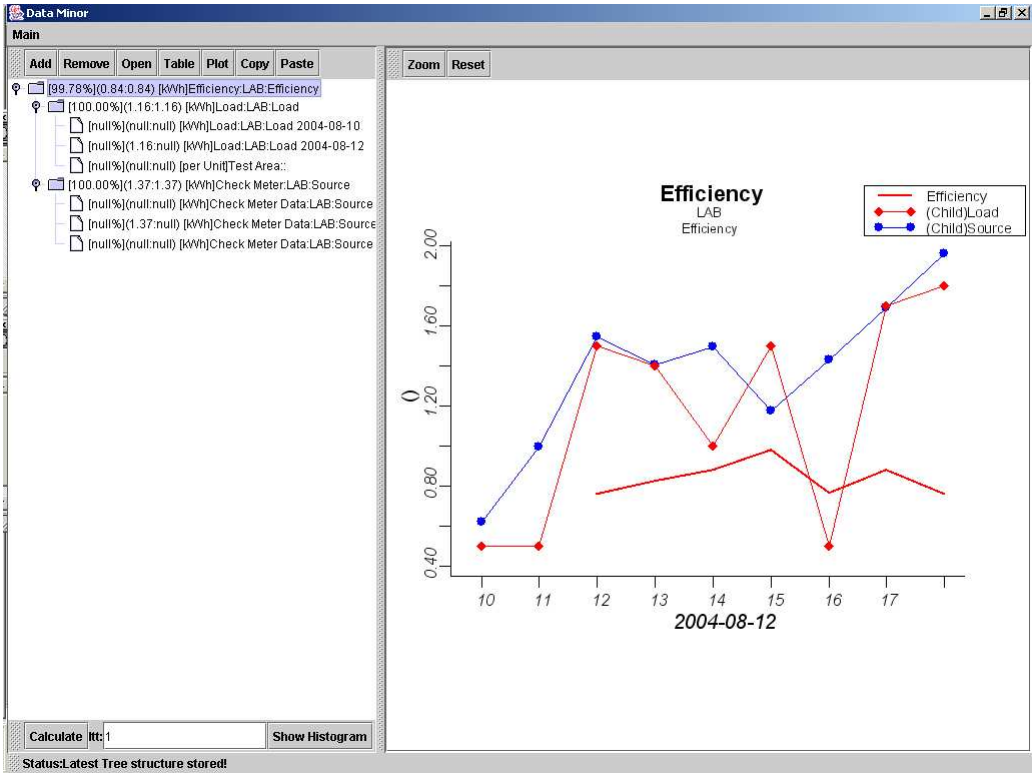


Figure 8.8: *One load omitted from data set*

Chapter 9

Conclusions

9.1 Overview

In South Africa, the electricity distribution industry is exposed to more unmonitored users each year due to the drive of the rural electrification projects and the installation of the pre-paid electricity meter since the early 1990s. With an extensive amount of infrastructure already in place the emphasis is on finding an economical as well as viable solution to reduce or eliminate electricity theft.

For this project, the following possible solutions were investigated:

- By generating a pulse on the network, Time Domain Pulse Reflectometry technology was developed.
- Based on the Theory of Constraints, information technology systems were developed to aid the management of electricity theft.

For these two methods, systems were developed for functioning in a low voltage reticulation environment. The devices developed were tested in a laboratory environment in order to evaluate their functionality.

9.2 Evaluations and Feasibility

The criteria for evaluation is fairly straightforward: how does the method or solution impact the bottom-line of the distribution network?

According to this criterion the following can be concluded from the research presented in this thesis:

- The reflections from the Time Domain Pulse Reflectometry are too complex to decode using known signal processing tools. This is due to the amount of reflections, distortions and signal attenuation. Therefore, a high capital layout would be required to acquire an

acceptable degree of accuracy. With the current technology level, a high level of human interaction is expected. Combined with high per unit costs and high density, the running costs would be extremely high. This solution does not seem viable in terms of electricity theft detection.

- Electricity theft must be managed using a performance driven system. There is no magic wand for this problem. Applying the Theory of Constraints in combination with the developed Data Mining platform, could have a payback period of less than a year for areas where electricity theft is expected to be more than 8%. This data driven approach adds value to already implemented investments by adding a minimal amount of additional infrastructure to provide an extremely powerful tool for electricity theft detection.

9.3 Recommendations and Future Research

9.3.1 Time Domain Pulse Reflectometry

Although Time Domain Pulse Reflectometry is not suitable for electricity theft detection the inverted pulse topology developed can be used for on-line real-time fault finding on cables. For further work this technology could be investigated for applications on medium voltage lines to provide a solution for cable theft detection.

9.3.2 Theory of Constraints and Data Driven Solution

The system developed in the thesis was tested in a controlled laboratory environment. These developments and this technology should be tested in the industry as soon as possible to determine its practical impact.

The Theory of Constraints proves to be an extremely helpful tool to address the problem of electricity theft at management level. During the research it was impossible to monitor this management strategy effectively and it should prove to be an interesting study to apply and monitor this strategy at a few local municipalities.

The developed Data Mining platform could be expanded to including learning algorithms on the macro, micro and individual scale by further developing the node capabilities. Features such as exception reporting and statistical analysis on the data of nodes can be included, such as the methods developed by Fourie [32], in a next release. Developing additional data collection software for input via PCs to assist performance management in other areas where no mobile electronic Data Collection applications, such as the PDA solution in Chapter 7, exist.

Due to the modular design of the check-meter, the systems feature set can be expanded to include the following:

- The design of single phase metering system.

- Implementation of a real time data logging system.
- Additional memory placed on the module can extend the real time data logging capabilities.
- The use of the $2.4\,GHz$ ISM band for inter-area data communications can be investigated.
- With the implementation of inter-area communications, the system can be used to perform pre-paid or conventional metering functions.
- Along with the metering functions the appropriate back office applications should be developed to support the system in the field.

9.4 Conclusions

In the field of electricity theft no extensive academic research has been done for the South African environment. A study was undertaken to establish the know-how for addressing the problem of electricity theft. During this process, gaps in the information chain were identified.

For the purpose of the thesis, tools and systems were developed from scratch to aid in the detection of electricity theft and to provide a platform for research into trends and other phenomena concerning electricity usage. This provides the ideal tool for electricity theft detection in a performance driven management culture. The evaluation of these new tools showed that a solution exists for determining the extend of electricity theft in an area by using a minimal amount of additional infrastructure and therefore, adding value to current investments.

Bibliography

[1] 3GPP, *ETSI GSM 07.05: Digital cellular telecommunications system (Phase 2); Use of DTE-DCE interface for Short Message Service (SMS) and Cell Broadcast Service (CBS)*. 3GPP, V6.9.2 edition, April 2001.

[2] ABERDARE, *ABC - Low voltage aerial bundle conductor*. Aberdare Power Cables, 2003. www.aberdare.co.za.

[3] ACTARIS, *ECLIPSE Prepayment Electricity Resource Management Software*. Actaris SAS, ZI de Chasseneuil Avenue des Temps Modenes 86361 Chasseneuil du Poitou cedex France.

[4] ACTARIS, *SL7000 Smart Comercial and Industrial Electricity Meter*. Actaris SAS, ZI de Chasseneuil Avenue des Temps Modenes 86361 Chasseneuil du Poitou cedex France.

[5] ANALOG DEVICE, "Analog Devices." `www.analog.com`. 2004.

[6] ANALOG DEVICES, *Evaluation Board Documentation ADE7758 Energy Metering IC*. Analog Devices, One Thenology Way, P.O.Box 9106, Norwood, MA 02062-9106, U.S.A., Rev.prb edition, August 2003.

[7] ANALOG DEVICES, *Poly phase multifunction Energy Metering IC with per Phase information, ADE7758*. Analog Devices, One Thenology Way, P.O.Box 9106, Norwood, MA 02062-9106, U.S.A., Rev.0 edition, February 2004.

[8] BRADLEY, T., *An introduction to the Bootstrap*. Chapman and Hall, Inc. 29 West 35th Street New York, NY 10001-2299: Chapman and Hall, 1993.

[9] BURTON-HOULE, T., "The Theory of Constraint and its thinking processes." `http://www.goldratt.com`. 2001.

[10] CABENA, P. *et al.*, *Discovering data mining from concept to implementation*. Prentice Hall, Inc., 1997. p. 12, p.43.

[11] CAMPBELL, K., "The good, the bad and the ugly." `http://www.sarpa.co.za/documents/document_list_all.asp`. 1999.

[12] CARTE BLANCHE, “Dangerous Sparks - 17 March 2002.” http://www.mnet.co.za/CarteBlanche/default.asp. 2002.

[13] CHAPMAN, T. and TERBLANCH, W., “History and Current Status of the Eskom dealings with non-payment in Soweto.” tech. rep., ESKOM, 2003.

[14] CLOETE, D. E., “Check meter principle in respect of tampering.” http://www.sarpa.co.za/documents/document_list4.asp?id=132. 2002.

[15] DAVIDSON, I. E., “Evaluation and effective management of non-technical losses in power networks.” *The Transactions of the SA Institute of Electrical Engineers*, September 2003.

[16] DENT INSTRUMENTS, *ELITEpro Recording Poly Phase Power Meter*. Dent Instruments, 65 NW Franklin Ave. Bend, OR 97701-2906 USA.

[17] DEPARTMENT OF MINERALS AND ENERGY, “BACKGROUND ON FREE BASIC ELECTRICITY (FBE).” http://www.dme.gov.za. 2004.

[18] DE VILLIERS AND MOORE CONSULTING ENGINEERS, “Mooiwater Housing Project - LV Reticulation Service and Connections.” CAD Drawing R3530/LV and SC1, Inframax DMS Western Cape, P.O.Box 472, Durbanville, 7551, November 2001.

[19] DICK, A. J. and MACEY, R., “Revenue protection in a competitive supply environment.” *Metering Tariffs for Energy Supply*, May 1999, No. 462, p. 229.

[20] DOORDUIN, W. A. *et al.*, “Feasibility Study of Electricity Theft detection using Mobile Remote Check Meters.” *AFRICON 2004*, January 2004.

[21] DOORDUIN, W. A. *et al.*, “On-line Time Domain Pulse Reflectometry in a LV reticulation network.” *SAUPEC 2004*, January 2004.

[22] DOUCETTE, S. and HAYWARD, R., “Handhelds Enhance Operation Excelence Program.” http://www.mt-online/articles/0504keyspan.cfm?pf=1. 2003.

[23] EPIC and DONALD, W. D., “Scientific Graphical Toolkit.” http://www.epic.noaa.gov/java/sgt/. October 2003.

[24] ESKOM, “Non payment of services.” http://www.eskom.co.za. January 2003.

[25] ESKOM, “Current Prepaid Related Projects.” http://www.eskom.co.za/electrification/current_projects.htm. 2004.

[26] ETSI, *ETSI GSM 03.38: Digital cellular telecommunications system (Phase 2): Alphabets and language-specific information*. ETSI.

[27] ETSI, *ETSI GSM 03.40: Digital cellular telecommunications system (Phase 2); Technical implementation of the Short Message Service (SMS) Point-to-point (PP)*. ETSI.

[28] ETSI, *ETSI GSM 04.80: Digital cellular telecommunications system (Phase 2): Mobile radio interface layer 3, Supplementary service specification, Formats and coding*. ETSI.

[29] ETSI, *ETSI GSM 07.05: Digital cellular telecommunications system (Phase 2); Use of DTE-DCE interface for Short Message Service (SMS) and Cell Broadcast Service (CBS)*. ETSI.

[30] ETSI, *ETSI GSM 07.07: Digital cellular telecommunications system (Phase 2); AT command set for GSM Mobile Equipment*. ETSI.

[31] FOCUSED PERFORMANCE, "The TOC thinking process." `http://www.focusesperformance.com`. Unkown.

[32] FOURIE, J. W. and CALMEYER, J. E., "A Statistical Method to Minimize Electrical Energy Losses in a Local Electricity Distribution Network." in *Africon*, vol. 02, pp. 667–673, IEEE, 2004.

[33] FOURIE, R. J., "Evaluation of losses in LV feeders using the Beta Load." Master's thesis, Stellenbosch Unversity, 1998.

[34] GIOVINAZZO, P., "The Fatal Currenty." `http://www.physics.ohio-state.edu/~p616/safety/fatal_current.html`. February 1987.

[35] GLOVER, D. J. and SARMA, M., *Power System Analysis and design*. Second edition. PWS Publishing Company, 1994.

[36] GOLDRATT, E. and COX, J., *The Goal: A Process of Ongoing Improvement*. Avraham Y. Goldratt Institute, 1992.

[37] HANDANGO.COM, "History of the Personal Digital Assistant." `http://www.handango.com/PDAHistory.jsp?siteId=1`. 2004.

[38] HANDX, "handX pInstaller." `http://www.handx.net/devtool/pInstaller/`. 2004.

[39] HAUSE, H. A. and MELCHER, J. R., *Electromagnetic Fields and Energy*. Pretice-Hall International, Inc., 1898.

[40] HEUNIS, S. W. and HERMAN, R., "Estimation of the annual resistive loss on LV residential feeders." *PMAPS, Naples*, September 2002.

[41] INFINEON, *CoolMOS Power Transistor SPW20N60S5*. Infineon Technologies AG, November 2002. www.infineon.com.

[42] ITU-T, *ITU-T Recommendation V.25 ter: Serial asynchrinous automatic dialing and control*. ITU-T.

[43] JAMIESON, D., "Conversion to pre-payment." tech. rep., Thembisa Ekurhuleni, June 2003.

[44] JAMIESON, D., "Saga of protective structures." tech. rep., Thembisa Ekurhuleni, June 2003.

[45] KRISHNA RAO, M. V. and MILLER, S. H., "Revenue improvement from intelligent metering systems.." *Metering and tariffs for Energy Supply*, May 1999, No. 462, p. 218. Conference Publication No. 462 IEE 1999.

[46] KU, Y. H., *Transient Circuit Analysis*. D. van Nostrand Company, inc., 1961.

[47] LEM, *Current Probe Model PR1200 MACV*. LEM.

[48] LEM, *Current Probe Model PR201 ACV*. LEM.

[49] LEM, *Current Probe Model PR221 ACV*. LEM.

[50] LOUW, D., "Legal Matters and RPAAU Proceedings." tech. rep., SARPA, 2002.

[51] MENZIES, T. and HU, Y., "Data Mining for Very Busy People." *IEEE Computer Society Journal - Computer*, November 2003, Vol. 36, No. 11, pp. 22–28.

[52] MICHAEL T. GOODRICH, R. T., *Data Structures and Algorithms in Java*. 2 edition. John Wiley & Sons, Inc., 2001.

[53] MICHIE, D. M., "Free basic electricity." tech. rep., Association of Municipal Electricity Undertakings, June 2001.

[54] MISETE, E., "New HandHelds Improve Worker Productivity." `http://www.integratedsolutionsmag.com/`. May 2003. Articles/2003_05/030503.htm.

[55] MOHAMED, Y., "Nie deur gure weer gekeer." *Die Burger*, August 2004, p. 6. 10 August 2004.

[56] MOHAN, N. *et al.*, *Power Electronics*. second edition. John Wiley & Sons, Inc., 1989.

[57] MOLEPA, S. A., "A multilevel inverter for DC Reticulation." Master's thesis, Stellenbosch Unversity, April 2003.

[58] NEAMEN, D. A., *Electronic Circuit Analysis and Design*. McGraw-Hill Companies, Inc., 1996.

[59] NETBEANS.ORG, "Netbeans downloads." http://www.netbeans.info/downloads/. 2004.

[60] OPENSHAW, C., "Report remote monitorig pre paid meters." ESKOM minutes, 2003.

[61] PALM, "Company Background." http://www.palmone.com/us/company/pr/background.html. 2004.

[62] PALM, "Compatibility Testing." http://www.palmos.com/dev/core/compatiblity/test.html. 2004.

[63] PALM, "Download Conduit Developer Kit (CDK)." http://www.palmos.com/dev/dl/dl_sdks/dl_cdk/. 2004.

[64] PALM, "palmOne Support: HotSync Technology Support Index." http://www.palmone.com/us/support/hotsync.html#network. 2004.

[65] POZAR, D., *Microwave Engineering*. Second edition. John Wiley and sons, inc., 1998.

[66] RS, "3 Phase Test Leads, 600V CATIII, RS #212-988." http://www.rssouthafrica.com/. 2004.

[67] SABS, "NRS048:1 - Electricity Supply - Quality of Supply." tech. rep., National Electricity Regulator, 1996.

[68] SABS, "NRS048:2 - Electricity Supply - Quality of Supply." tech. rep., National Electricity Regulator, 1996.

[69] SCHWARZ, A., "www.mikrocontroller.net - MSPGCC (english version)." http://www.mikrocontroller.net/mspgcc.en.htm. December 2003.

[70] SHINGAI, S., "The role of metering in revenue protection." *Metering and tariffs for Energy Supply*, May 1999, No. 462, p. 223. Conference publication IEE 1999.

[71] SIEGRIST, K., "Virtual Laboratories in Probability and Statistics." http://www.math.uah.edu/stat/objects/index.xml. 2004.

[72] SOURCE FORGE, "Mspgcc-users – GCC for MSP430 - http://mspgcc.sf.net." https://lists.sourceforge.net/lists/listinfo/mspgcc-users. 2004.

[73] SOUTH AFRICAN GOVERNMENT, "Electricity Act, Law 41 of 1987 article 27 (2,3)." 1987.

[74] SOUTH AFRICAN GOVERNMENT, “Municipal Systems Act 2000.” http://www.gov.za. December 2000.

[75] SOUTH AFRICAN GOVERNMENT, “OCCUPATIONAL HEALTH AND SAFETY ACT.” http://www.info.gov.za/gazette/acts/1993/a85-93.htm. 2003.

[76] STEHMANN, T., “Development and Optimsation of a Solid-State Pulsed Power Supply for a $C0_2$ TEA laser.” Master’s thesis, Stellenbosch University, April 2003.

[77] STELLENBOSCH MUNICIPALITY, “Stellenbosch Muncipal.” http://www.stellenbosch.org.za. 2004.

[78] ST MICROELECTRONICS, *1.5A Low Drop Fixed and adjustable positive voltage regulators*. ST Microelectronics, May 2000.

[79] STRIKE TECNOLOGIES, “ENERMAX.” http://www.strike.co.za/1024x768/content/Products/Enermax.htm. 2004.

[80] STROBER, K., “Revenue Protection - Proactively Changing Attitudes.” tech. rep., Cape Town, 2003.

[81] SUN, “javax.swing (Java 2 Platform SE v.1.4.2).” http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/package-summary.html. 2003.

[82] SUN, “JTree (Java 2 Platform SE v.1.4.2).” http://java.sun.com/j2se/1.4.2/docs/api/javax/swing/JTree.html. 2003.

[83] SUN, “Java 2 Platform, Standard Edition v 1.4 Overview.” http://java.sun.com/j2se/1.4/. June 2004.

[84] SUN, “Java Communications API.” http://java.sun.com/products/javacomm/. 2004.

[85] SUN, “JDBC Technology.” http://java.sun.com/products/jdbc/index.jsp. 2004.

[86] SUN, “MIDP for Palm OS.” http://java.sun.com/products/midp4palm/. 2004.

[87] TEXAS INSTRUMENTS, *Configuring the MSP430x11x(1)’s Basic Clock Module*. Texas Instruments Inc., Slaa082 edition, November 1999. www.ti.com.

[88] TEXAS INSTRUMENTS, *Octal bus transceiver and 3.3-V to 5-V shifter with 3-state outputs*. Texas Instruments Inc., Sca5375g edition, March 1999. www.ti.com.

[89] TEXAS INSTRUMENTS, *MAX3222 3-V to 5.5-V Multichannel RS-232 Line Driver/Receiver with 15-kV ESD protection*. Texas Instruments Inc., Slls408g edition, January 2000. www.ti.com.

[90] TEXAS INSTRUMENTS, *Single 9-A High Speed Low-Side MOSFET driver with enable*. Texas Instruments Inc., September 2002. www.ti.com.

[91] TEXAS INSTRUMENTS, "Texas Instruments." www.ti.com. 2004. www.ti.com.

[92] VAN DER MERWE, J., "Experiments on electromagnetic fields and waves." Master's thesis, Stellenbosch Unversity, September 1995.

[93] WAVECOM, *AT Commands Interface Guide*. Wavecom, 003 edition, December 2003.

[94] WEBOPEDIA.COM, "A Word Definition From the Webopedia Computer Dictionary." http://www.webopedia.com. 2004.

[95] WILLIAMS, D. F., "Meter readers make the switch to PDA." http://www.istart.co.nz/index. December 2003. /HM20/PC0/PV22447/EX24706/CS24777.

[96] ZENOU, B., *WMOi3 user's guide*. Wavecom, 1.4 edition, January 2001.

Appendix A

Additional Background Information

A.1 Eskom's Notice of Unlawfull Tampering

ESKOM

Aksie teen Dr,Mnr,Me .....
Action against Dr,Mr,Ms

in terme van Eskom se Voorsieningsvoorwaardes vir klein toevoere
in terms of Eskom's Conditions of Supply for small supplies

Met verwysing na bogenoemde dokument word u in kennis gestel dat u wederregtelik met Eskom se apparaat of toerusting gepeuter het en dat u aanspreeklik gehou word vir die koste verbonde aan die herstel van die toevoer en enige verliese deur Eskom gely. U het u blootgestel aan vervolging ingevolge die Elektrisiteitswet, nommer 41 van 1987, soos gewysig en die toevoer na u installasie is gestaak en die meter verwyder.

With reference to the abovementioned document you are notified that you have unlawfully tampered with Eskom's apparatus or equipment and that you are held responsible for the cost of restoring the supply and of any losses suffered by Eskom. You have exposed yourself to prosecution in terms of the Electricity Act, number 41 of 1987, as amended and the supply to your installation has been discontinued and the meter removed.

Installasie adres: _____ Installation address

_____

_____

Erfnommer _____ Erf number

Meternommer _____ Meter number

Eenhede (kWh) op meter _____ Units (kWh) on meter

Beskrywing van skade aan installasie / Description of damage to installation :

_____

	Naam / Name	Handtekening/Signature	Datum / Date
Eskom verteenwoordigers Eskom representatives	.....	.....	.....
	.....	.....	.....
Klant / Customer	.....	.....	.....

Nota aan klant:
Aansoek om heraansluiting kan by Eskom se Klantedienskantoor te Bellville ingedien word. Hierdie vorm sowel as 'n geldige identiteitsdokument moet deur applikant getoon word voor heraansluitingsfooi bereken kan word.

Heraansluitingsfooi is in kontant betaalbaar by indiening van aansoekvorm.

Note to customer:
Application for reconnection can be made at Eskom's Customer Service Centre in Bellville. This form, as well as a valid identity document must be produced by the applicant before a reconnection fee can be calculated.

The reconnection fee is payable in cash on submission of the application form.

VIR KANTOOR GEBRUIK / FOR OFFICE USE

Vorige remiediërende aksies
Previous remedial actions

Huidige aksie op PPS/CMS geplaas
Current action entered on PPS/CMS — Datum / Date ..... — Deur wie ingepons / Punched by .....

Heraansoek ingedien
Re-application submitted — Datum / Date ..... — Deur wie ontvang / Received by .....

PPS Geskiedenis Kode
PPS History Code ....

Bedrag betaalbaar
Amount payable R.....

WKV212

A.2 Electricity Theft Methods

This section is taken from [45] and placed to provide the reader with a reference as to what has been found in the field:

Pilferage through tapping of lines

- Direct tapping of lines without meter
- Tapping supply from behind the meter board by joining one end of a wire to incoming phase wire and the other ens to the main switch and bypassing the meter.
- Pilfering energy from cutouts where they are connected before the meter.
- Breaking the neutral connection and incorporating a switch in its circuit with efficient earth connection.

Pilferage through damage to meter mechanism without tampering terminal cover seals and meter seals

- Drilling small hole in the meter cover and inserting a fine needle to stop the rotation of the meter disc and covering it up with paint to avoid being noticed.
- Creating gaps in meter glass and inserting a tongue cleaner to obstruct the rotation of the meter disc.
- Creating fine gaps in between meter top and bottom covers and inserting thin bits of tongue cleaner or polymeric leaves or celluloid films to obstruct the rotation of the meter disc.
- Subjecting the meter to violent shock to render it totally inoperative without forming a dent on the surface.
- Breaking the meter glass and fixing another glass in its place with adhesive compound to have accessibility to recording arrangement.
- Keeping the meter in an inclined position to slow down the meter speeder keeping it horizontal to bring the meter speed to a stop.
- Burning the meter either by creating a loose connection at the terminal or by setting fire to it by pouring petrol or other inflammable material or creating a short circuit.

Pilferage through tampering terminal cover seals

- Reversing the connections between line (incoming) and load (outgoing) in the terminal block. If one phase is reversed in a 3 phase service, the meter runs in a forward direction whenever three phase load is switched on but records only $\frac{1}{3}^{rd}$ of the consumption.
- By keeping open the potential links in the meter terminal block for such meters which are not provided with internal potential links, so that the meter will not work due to non-availability of supply to potential coil.
- By providing a shunt wire to the current coil terminal in the meter terminal thus creating parallel path and partial recording of energy.
- Removing the phase and neutral wires from the meter terminal block and joining them externally so as to stop running of the meter.
- By reversing incoming phase and neutral wires in the terminal block and providing a switch with earth connection.

Pilferage through tampering meter cover seals

- Stopping the meter disc.
- Cutting potential wire, resulting in non-rotation of the meter disc.
- By providing a shunt wire to the current coil terminals inside the meter, thus creating parallel path and partial recording of energy.
- Reducing maximum demand in case of Trivector meters.
- Adjusting the consumption recorded at will.
- Providing parallel paths to current coil elements in 'R' phase and 'B' phase of Trivector meters by connecting up extra wires with switches fixed in parallel paths to switch off or on the parallel paths in order to reduce the flow of normal current in the mtering ciruit.

Pilferage through tampering seals of metering equipment(CTs & PTs)

- Providing parallel paths through CT secondary terminals.
- Increasing the CT ratio to a higher range by changing the CT secondary connections thus resulting in recording of lower consumption.
- Meddling of PT secondary terminal wires including opening/reversing the connections.
- Blowing or removing PT primary fuses from inside the metering cubicle.
- Short circuit current transformers on secondary side.

- Change the current transformer ratio on the primary side.

Pilferage through wrong readings and desk top readings

- Meters are not read properly and ofter desk readings are entered.
- Often consumers are not available at their premises to enable a meter reader to record meter reading.
- Some consumers bypass the meter but when the meter reader comes to check the reading they revert it back to the original connection.

Appendix B

Transmission Line Theory

Some of the theory behind time domain pulse reflectometry will be explained here. This is included as a summary of Transmission Line Theory and is based on the derivations of Ku and Hause & Melcher.

B.1 Telegraphist Equation

Consider the transmission line in Fig. B.1 and the derivations in [46], which provide the necessary equations for the transmission line model. Let

R = resistance per unit length of the line,

L = inductance per unit length of the line,

G = conductance per unit length of the line,

C = capacitance per unit length of the line and

x = the distance measured from the sending end of the transmission line.

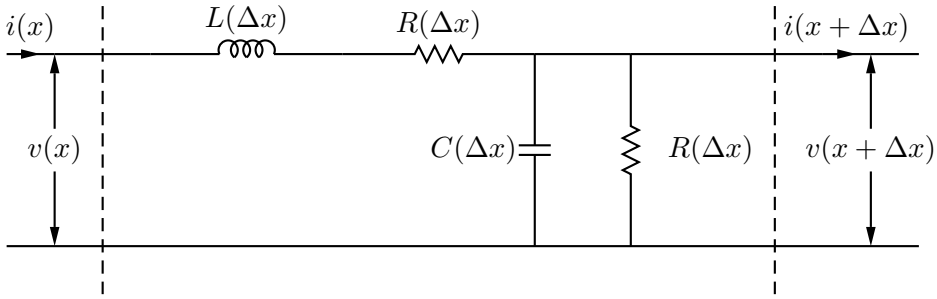


Figure B.1: *Section of Transmission Line*

The voltage drop and decrease in current in length Δx is given by (B.1) and (B.2)

$$-\frac{\partial v}{\partial x}\Delta x = (Ri + L\frac{\partial i}{\partial t})\Delta x \tag{B.1}$$

$$-\frac{\partial i}{\partial x}\Delta x = (Gi + C\frac{\partial v}{\partial t})\Delta x \tag{B.2}$$

This voltage and current functions are in the time domain where

$$v = v(t, x) \tag{B.3}$$

$$i = i(t, x) \tag{B.4}$$

The part of interes is in the transient response of the transmission line and by using the Laplace transformation with respect to t and dividing both (B.1) and (B.2) by Δx, (B.5) and (B.6) is obtained.

$$-\frac{\partial V(s, x)}{\partial x} = (R + Ls)I(s, x) \tag{B.5}$$

$$-\frac{\partial I(s, x)}{\partial x} = (G + Cs)V(s, x) \tag{B.6}$$

By differentiating (B.5) with respect to x and substituting the result in (B.6) yields (B.7):

$$\frac{\partial^2 V(s, x)}{\partial x^2} = \gamma^2 V(s, x) \tag{B.7}$$

Similarly, by differentiating (B.6) with respect to x and substituting the result in (B.5), (B.8) is obtained

$$\frac{\partial^2 I(s, x)}{\partial x^2} = \gamma^2 I(s, x) \tag{B.8}$$

where

$$\gamma = \sqrt{(R + Ls)(G + Cs)} = \sqrt{ZY} \tag{B.9}$$

and

$$Z = (R + Ls) \tag{B.10}$$

$$Y = (G + Cs) \tag{B.11}$$

General solutions for (B.7) and (B.8) are

$$V(s, x) = V^+ e^{-\gamma x} + V^- e^{\gamma x} \tag{B.12}$$

$$I(s, x) = \frac{1}{Z_0}(V^+ e^{-\gamma x} - V^- e^{\gamma x}) \tag{B.13}$$

where

$$Z_0 = \sqrt{\frac{(R + Ls)}{(G + Cs)}} = \sqrt{\frac{Z}{Y}} \tag{B.14}$$

and $V^+(s, x)$ and $V^-(s, x)$ are used to denote the signal traveling in the $+$ and $-$ direction of the transmission line[39, ch 14.3, (14.3.2), p.641], depending on our chosen reference point.

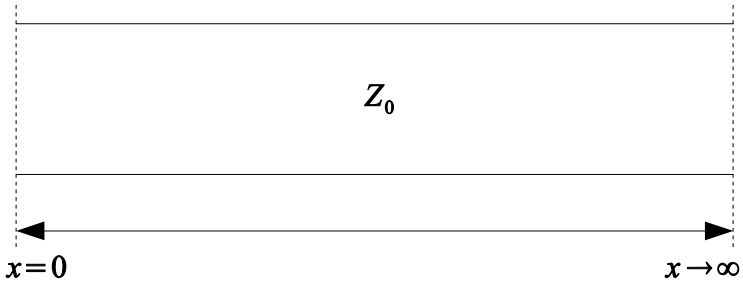


Figure B.2: *One Way Infinite Line*

B.2 Propagation in Transmission Lines

B.2.1 Infinite Lossless Line

Consider the infinite transmission line, Fig. B.2, without the lossy components and therefore $R = 0$ and $G = 0$ with the end of the transmission line at $x \to \infty$. Using boundary conditions and (B.12) it follows from [46, ch 10-4, p.291] that

$$\lim_{x\to\infty} V(s,x) = 0 \Rightarrow V^- = 0 \tag{B.15}$$

$$V(s,0) = V^+; \tag{B.16}$$

and results in

$$V(s,x) = V(s,0)e^{-\gamma x} \tag{B.17}$$

$$= V(s,0)e^{-\sqrt{LC}sx} \tag{B.18}$$

which, with the use of the inverse Laplace transform results in

$$v(t,x) = v_s\left(t-\sqrt{LC}x\right) \tag{B.19}$$

$$= v_s\left(t-\frac{x}{c}\right) \tag{B.20}$$

where the voltage at the sending end $v_s(t)$ is the inverse transform of $V(s,0)$ and $c = \frac{1}{\sqrt{LC}}$. When considering the infinite lossless line in Fig. B.3 in both directions, the reference point is moved to the middle of the transmission line and setup the initial conditions from here. The following initial conditions [39, ch 14.4, p.647] are substituted

$$V(s,0) = V(s) \tag{B.21}$$

$$I(s,0) = I(s) \tag{B.22}$$

in (B.12) and (B.13) in order to solve for V^+ and V^- and obtain

$$V^+ = \frac{1}{2}\left(V(s)+Z_0I(s)\right) \tag{B.23}$$

$$V^- = \frac{1}{2}\left(V(s)-Z_0I(s)\right) \tag{B.24}$$

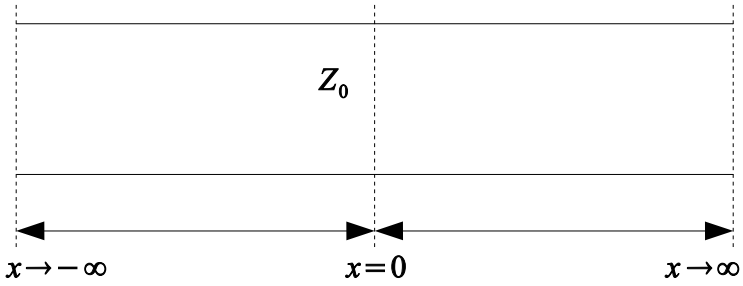


Figure B.3: *Both Way Infinite Line*

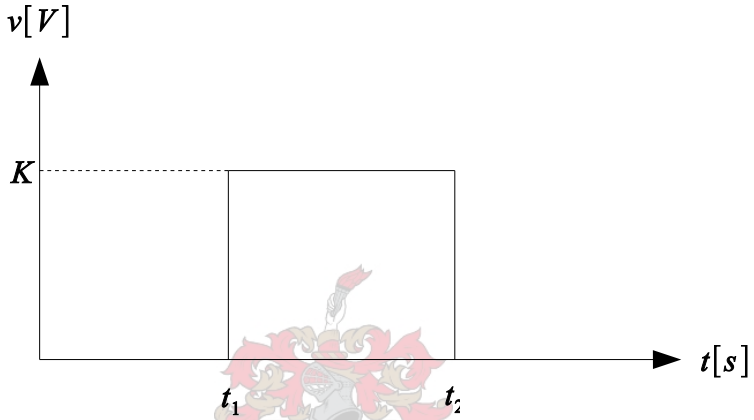


Figure B.4: *Rectangular Pulse*

B.2.2 Infinite Line with Distortion

For an infinite line, Fig. B.2, (B.18) is used and rewritten from (B.9) to obtain

$$\gamma = \frac{1}{c}\sqrt{\left[(s+\rho)^2 - \sigma^2\right]} \tag{B.25}$$

where

$$\rho = \frac{R}{2L} + \frac{G}{2C} \tag{B.26}$$

$$\sigma = \frac{R}{2L} - \frac{G}{2C} \tag{B.27}$$

Substituting (B.25) in(B.18) gives

$$V(s,x) = V(s)e^{-\frac{1}{c}\sqrt{\left[(s+\rho)^2-\sigma^2\right]}x} \tag{B.28}$$

For a rectangular pulse in Fig. B.4

$$V(s) = \frac{K}{s}\left(e^{-t_1 s} - e^{-t_2 s}\right) \tag{B.29}$$

(B.29) is used to obtain the inverse transform of (B.28) to realize (B.30)

$$\begin{aligned} v(t,x) = & Ke^{-\frac{\rho x}{c}}\left[u\left(t-t_1-\frac{x}{c}\right)-u\left(t-t_2-\frac{x}{c}\right)\right] \\ & +K\frac{\sigma x}{c}\left\{\left[\int_{\frac{x}{c}}^{t-t_1} e^{-\rho\tau}\frac{I_1(\rho z)}{z}d\tau\right]u\left(t-t_1-\frac{x}{c}\right)\right. \\ & \left.-\left[\int_{\frac{x}{c}}^{t-t_2} e^{-\rho\tau}\frac{I_1(\rho z)}{z}d\tau\right]u\left(t-t_2-\frac{x}{c}\right)\right\} \end{aligned} \tag{B.30}$$

where $z=\sqrt{t^2-\frac{x^2}{c^2}}$ and

$$I_1(\sigma z)=\frac{\sigma z}{2}\left[1+\frac{(\sigma z)^2}{2(4)}+\frac{(\sigma z)^4}{2(4)(4)(6)}+\cdots\right] \tag{B.31}$$

is the modified Bessel function of the first kind.

B.2.3 Distortionless Line

Heavyside first discovered the distortion less line, which is characterised by

$$\alpha=\frac{R}{L}=\frac{G}{C} \tag{B.32}$$

From the line with distortion (B.25) is taken and by using (B.32) the following relationship is shown:

$$\rho = \alpha=\frac{R}{L}=\frac{G}{C} \tag{B.33}$$

$$\beta = \frac{1}{c}=\sqrt{LC} \tag{B.34}$$

$$\alpha\beta = \frac{R}{Z_0} \tag{B.35}$$

For an infinite line propagation equation is expressed as

$$v(t,x)=e^{-\alpha\beta x}v_s(t-\beta x) \tag{B.36}$$

From (B.36) it can be seen that the $e^{-\alpha\beta x}$ term shows the attenuation factor, A.

B.3 Nominal and Equivalent Networks

For short transmission lines it is found convenient to replace the distributed constants with lumped constants in nominal T and Π networks as shown in Fig. B.5 and Fig. B.6. With

$$Z = (R+Ls)d \tag{B.37}$$

$$Y = (G+Cs)d \tag{B.38}$$

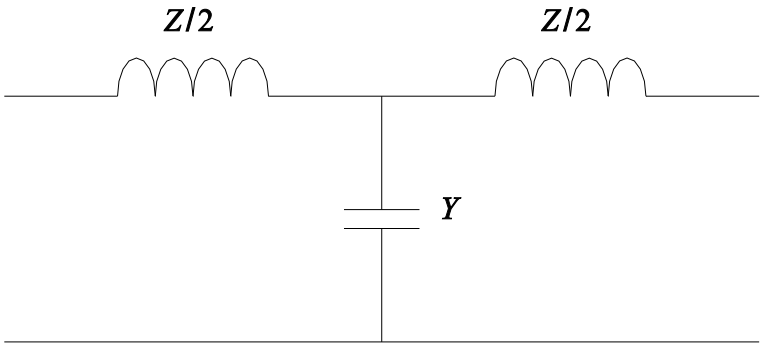


Figure B.5: *Nominal T Network*

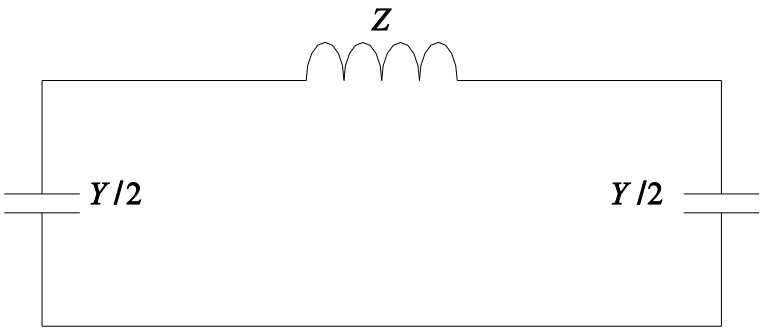


Figure B.6: *Nominal Π Network*

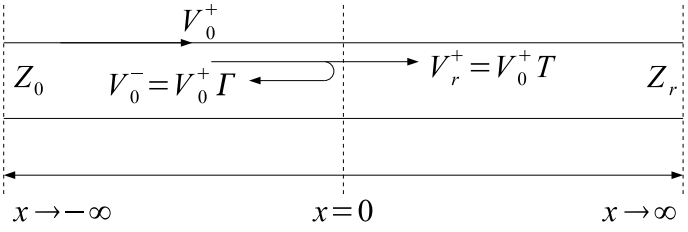


Figure B.7: *Reflection and Transmission Coefficient*

the following can be shown[46, ch.10-3, p.288]

$$I_r = I_s\left(1+\frac{ZY}{2}\right) - V_s Y \tag{B.39}$$

$$V_r = V_s\left(1+\frac{ZY}{2}\right) - I_s\left(2+\frac{ZY}{2}\right)\frac{Z}{2} \tag{B.40}$$

where s and r denotes the sending and receiving side of the transmission line. This allows the modeling of the transmission line with conventional circuit theory.

B.4 Reflection Coefficients

B.4.1 One line and Load Topology

The reflection coefficient is the main focus of this study. Reflections in the transmission line occur due to mismatching of the network elements due to impedance differences. The idea of TDR is to exploit this phenomena and use the reflections received from network elements to determine if additional illegal lines or loads are added to the reticulation network. The additional lines and loads are handled together, where Z_r can be replaced with Z_L. The extra transmission line, Fig. B.7, is considered with boundary conditions[39, ch 14-4, p.647]

$$V(s,0)_0 = Z_r I(s,0)_0 \tag{B.41}$$

$$V_0^+ + V_0^- = \frac{Z_r}{Z_0}\left(V_0^+ - V_0^-\right) \tag{B.42}$$

from which

$$V_0^- = \Gamma V_0^+ \tag{B.43}$$

can derived with

$$\Gamma = \frac{Z_r - Z_0}{Z_r - Z_0} \tag{B.44}$$

Pozar [65, ch 2.3, p.71] shows that the incident signal is transmitted onto the second line with a voltage amplitude given by a transmission coefficient multiplied by the incident signal amplitude. For $x < 0$

$$V(s,x)_0 = V_0^+\left(e^{-\gamma x} + \Gamma e^{\gamma x}\right) \tag{B.45}$$

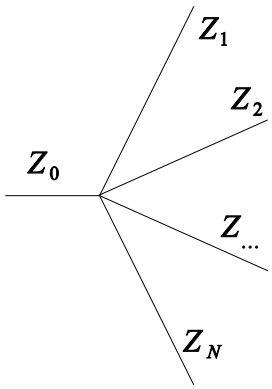


Figure B.8: *Split Pole Configuration*

V_0^+ is the voltage of the incident voltage signal on the feed line. It is assumed that a infinite long line in the positive direction and therefore do not expect any reflections and thus an absence of a new V_r^+ and can then be written for $x > 0$ as

$$V(s,x)_r = V_0^+ T e^{-\gamma x} \tag{B.46}$$

therefore at $x = 0$, $V(s,0)_r$ and $V(s,0)_0$ is equated to obtianed

$$T = 1 + \Gamma \tag{B.47}$$

B.4.2 Split Pole Configuration

Consider the split pole configuration as given in Fig. B.8.
The following conditions hold [35, ch.12.3, p.491]

$$V_0^+ + V_0^- = V_1^- = V_2^- = \cdots = V_N^- \tag{B.48}$$

$$I_0^+ + I_0^- = I_1^- + I_2^- + \cdots + I_N^- \tag{B.49}$$

which, by using $I_0^+ = \frac{V_0^+}{Z_0}$, $I_0^- = \frac{-V_0^+}{Z_0}$, $I_1^- = \frac{V_1^-}{Z_1}$, $\cdots$, $I_N^- = \frac{V_N^-}{Z_N}$, in (B.49), (B.50) is obtained

$$\Gamma_N = (Z_0||Z_1||Z_2||\cdots||Z_N)\left(\frac{1}{Z_0} - \sum_{k=1}^{N}\frac{1}{Z_n}\right) \tag{B.50}$$

and by using (B.48)

$$T_N = 1 + \Gamma_N \tag{B.51}$$

is obtained.
From Fig. B.9 it can be seen that for the thesis the reflections from a split pole with more than 4 connections can result in a large reflected signal, which will result in a small signal propagating further. For a configuration with four, 4-connection split poles, the reflected signal

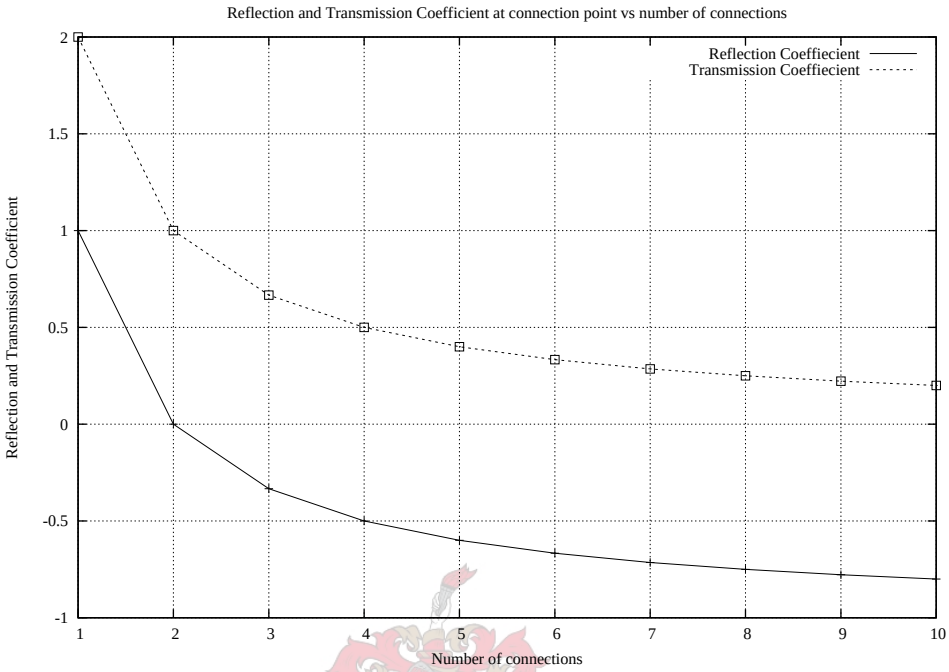


Figure B.9: *Results from Split Pole simulation*

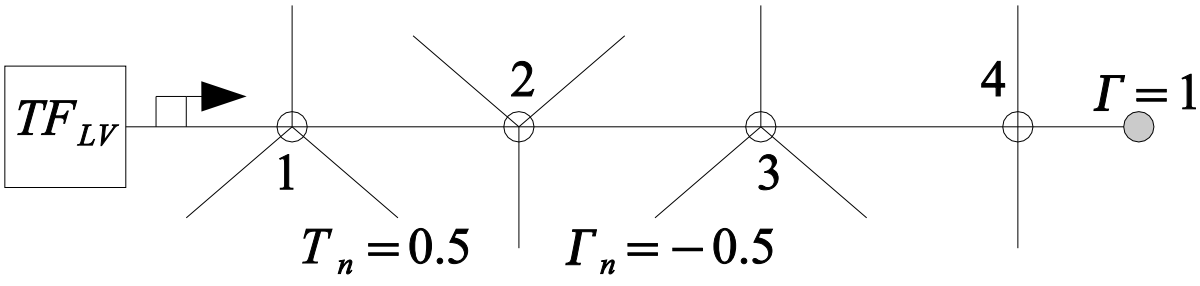


Figure B.10: *Example of Four Split Poles*

from an open line connected to the last split pole, as shown in Fig. B.10, will have a attenuation factor[1] of $1:2^8$ when the signal is measured at the low voltage transformer. For a standard 220V AC peak pulse it would mean a return voltage of roughly 1V, which is fairly small.

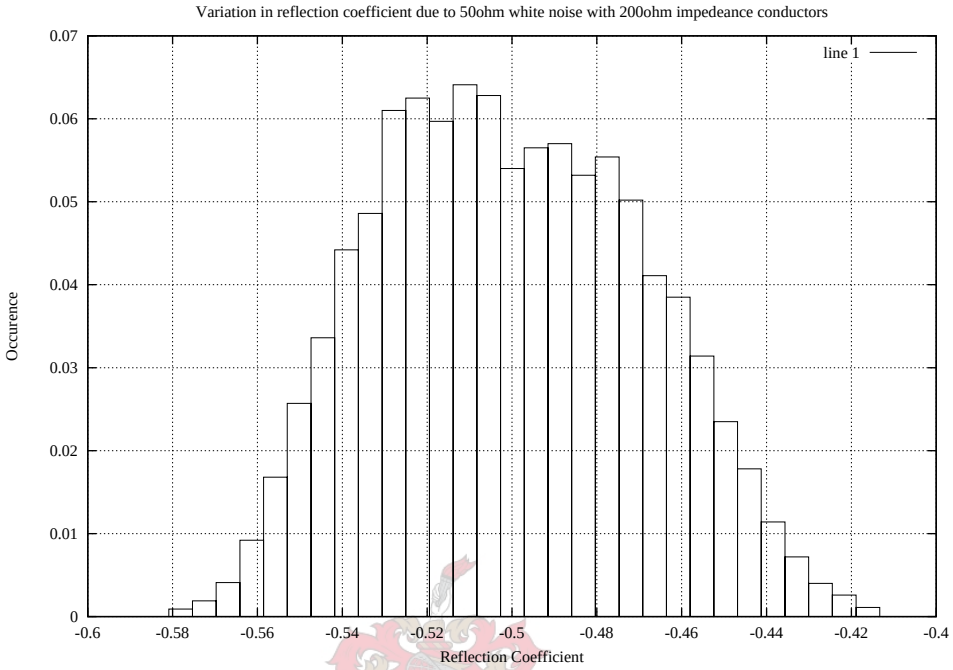


Figure B.11: *Reflection Coefficient Variation*

Power lines are not designed to transfer signals at high frequencies, therefore it can be expected that the impedance tolerances are low. Considering a split pole configuration with four connections with 12.5% tolerance we get the distribution as shown in Fig. B.11 when simulated with 10000 samples.

B.4.3 Scattering Matrix

It is of interest to represent the reflection coefficients. As indicated in (B.44) these can be calculated for one transmission line. It is required to retrieve and model many reflections. The use of the scattering matrix[65, ch 4.3, p.196] can prove helpfull. The scattering matrix is defined as

$$\begin{bmatrix} V_1^- \\ V_2^- \\ \vdots \\ V_N^- \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} & \cdots & S_{1N} \\ S_{21} & & & \\ \vdots & & & \vdots \\ S_{N1} & & \cdots & S_{NN} \end{bmatrix} \begin{bmatrix} V_1^+ \\ V_2^+ \\ \vdots \\ V_N^+ \end{bmatrix} \tag{B.52}$$

[1] Excluding normal signal attenuation due to line losses.

for a N-port network and can also be written as

$$[V^-] = [S][V^+] \tag{B.53}$$

$[S]$ can be calculated with

$$S_{ij} = \left.\frac{V_i^-}{V_j^+}\right|_{V_k^+, k \neq j} \tag{B.54}$$

For the configuration in Fig. B.8 it is found that with values $Z_0 = Z_1 = Z_2 = \cdots = Z_N$.

$$[S] = \begin{bmatrix} \Gamma & T & \cdots & T \\ T & \Gamma & & \\ \vdots & & & \vdots \\ T & & \cdots & \Gamma \end{bmatrix} \tag{B.55}$$

Pozar[65, ch.4.3, p.203] shows that the reference plain of the scattering matrix can be shifted by using a reference matrix and results in

$$[S'] = \begin{bmatrix} e^{-\gamma x_1} & & & 0 \\ & e^{-\gamma x_2} & & \\ & & & \vdots \\ 0 & & \cdots & e^{-\gamma x_N} \end{bmatrix} [S] \begin{bmatrix} e^{\gamma x_1} & & & 0 \\ & e^{\gamma x_2} & & \\ & & & \vdots \\ 0 & & \cdots & e^{\gamma x_N} \end{bmatrix} \tag{B.56}$$

where

$$[V'^-] = [S'][V'^+] \tag{B.57}$$

and x is the distance measured from where the reflection occurs.

B.4.4 Signal Flow Graphs

Pozar[65, ch.4.5, p.213] presents us with a method of graphically representing the scattering matrix. For completeness these rules and methods are summarized below:

Mapping A typical two port network is shown in Fig. B.12 and the equivalent signal flow graph in Fig. B.13.

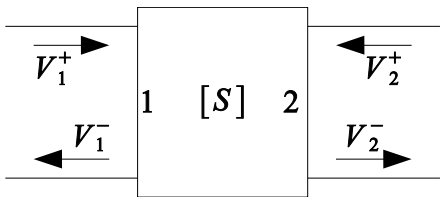


Figure B.12: *Two Port with Scattering Matrix Representation*

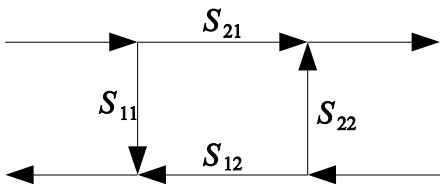


Figure B.13: *Two Port Signal Flow Representation*

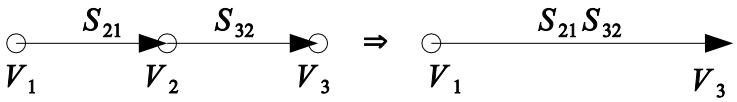


Figure B.14: *Series Rule*

Series Rule Two branches whose common node has only one incoming and one outgoing signal, can be combined to form a single branch as in Fig. B.14.

Parallel Rule Two branches from one common node to another may be combined as shown in Fig. B.15.

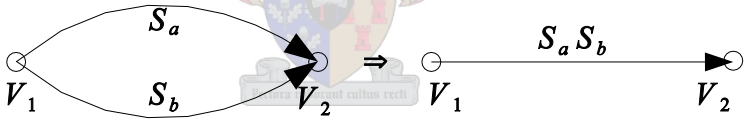


Figure B.15: *Parallel Rule*

Self-loop Rule When a branch begins and ends at the same node it can be eliminated as shown in Fig. B.16.

Splitting Rule A node may be split into two separate nodes as long as the resulting flow graph contains, once and only once, each combination of separate (not self loops) input and output branches that connect to the original node as shown in Fig. B.17.

B.5 Parameter Calculation

For simulation purposes it is required to calculate the cable parameters from the measured signals. The model in spice requires five parameters, e.g. L, C, G, R and distance of transmission line, ℓ. From (B.36) it can be seen that with the transmission line parameters can

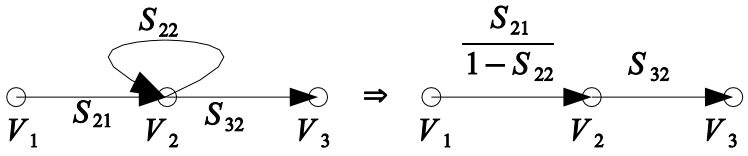


Figure B.16: *Self-loop Rule*

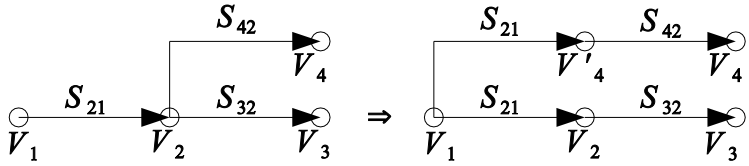


Figure B.17: *Splitting Rule*

be calculated by using the, attenuation A, line length ℓ, propagation time τ over the distance and surge impedance Z_0 and the distance of the transmission line. Since only four parameters are available a distortion less line is assumed and using the relations given in Section B.2.3 can be used to derived R, L, G and C:

$$A = \frac{|V_{in}|}{|V_{out}|} \tag{B.58}$$

$$A = e^{-\alpha\beta\ell} \tag{B.59}$$

$$\Rightarrow \alpha\beta = -\frac{\ln A}{\ell} \tag{B.60}$$

$$\beta = \frac{1}{v} = \frac{\tau}{\ell} \tag{B.61}$$

$$R = \alpha\beta Z_0 \tag{B.62}$$

$$L = Z_0\beta \tag{B.63}$$

$$C = \frac{\beta^2}{L} \tag{B.64}$$

$$G = \frac{RC}{L} \tag{B.65}$$

These parameters can now be put into the spice model and used for simulations.

B.6 Conclusion

Equations for the propagation, distortion and attenuation of signals in a transmission line with distributed parameters were derived and shown. It can be seen that the solutions, regarding the infinite line with distortion, are not easy to solve analytically. The lossless and distortion less solutions can be used to model the transmission lines.

Techniques giving us a better insight into transmission line reflections is presented in the form of signal flow graphs and the scattering matrix. These pose helpful when analizing the results of experiments.

Appendix C

Parallel Two Wire Parameters

L	$\frac{\mu}{\pi}\cosh^{-1}\left(\frac{D}{2a}\right)$
R	$\frac{R_s}{\pi a}$
C	$\frac{\pi\epsilon^{'}}{\cosh^{-1}\left(\frac{D}{2a}\right)}$
G	$\frac{\pi\omega\epsilon^{''}}{\cosh^{-1}\left(\frac{D}{2a}\right)}$

Table C.1: *Transmission Line Parameters for Parallel Two Wire configuration[65, Table 2.1, p.62]*

where

μ is the permeability,

R_s is the surface resistance

$\epsilon^{'}$ is the real part and $\epsilon^{''}$ is the imaginary part of the permeativity constant, where $\epsilon = \epsilon^{'} - i\epsilon^{''}$ and can be interpreted as $\epsilon = \epsilon^{'} - i\tan\delta$ [65, ch 2.2 ,p.61],

$\epsilon^{''}$ is $\tan\delta$,

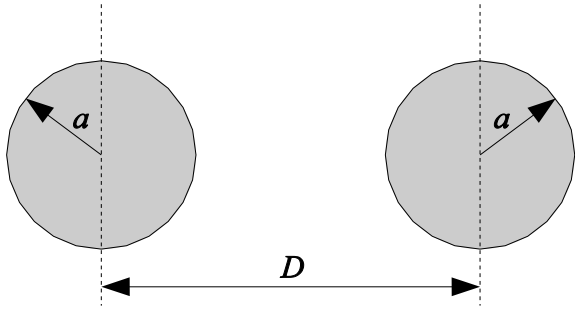


Figure C.1: *Parallel Two Wire Configuration*

ω is the applied frequency

L is the inductance per unit length,

R is the resistance per unit length,

C is the capacitance per unit length and

G is the conductivity per unit length.

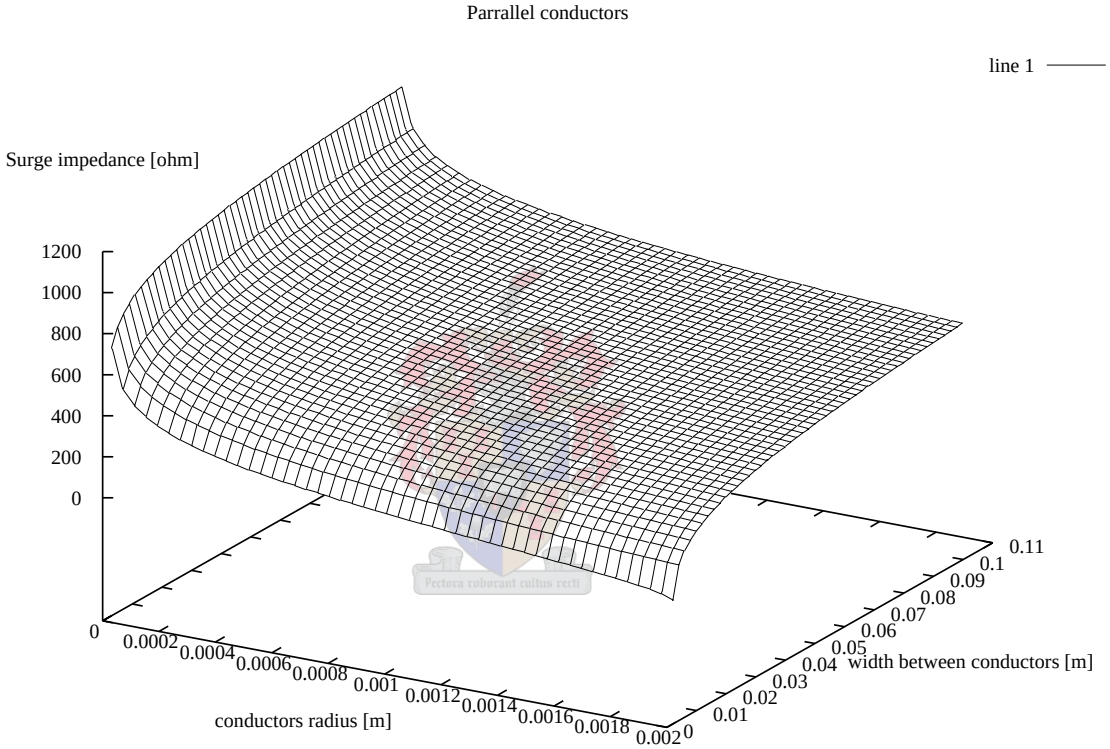


Figure C.2: *Parallel Two Wire Impedance*

Fig.C.2 shows the surge impedance of the parallel two wire configuration in terms of width between the wires and the radius of the conductors.

Appendix D

Pulse Generators

In order to implement Time Domain Reflectometry a suitable pulse generator is needed. Different Pulse generator technologies are discussed here.

D.1 Signal and Pulse Generators

Two power electronic device topologies, namely the half-bridge and parallel-switch are shown in Figs.D.2 and D.3. In our research some initial experiments were conducted using a few variations of these as well as a standard signal generator and glitch generator.

D.1.1 Signal Generator

A standard signal generator was implemented to start with the initial experiments as shown in Fig.D.1. With the signal generator it is easy to quickly get started with some experiments and we know that the impedance Z_0 is roughly 50Ω.

D.1.2 Glitch Generator

[92] uses a glitch generator in experiments to generate a pulse with a $-3dB$ pulse width of $40ns$. This device is extremely useful when a short pulse is required with a low repetition rate.

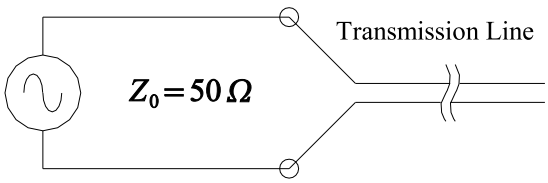


Figure D.1: *Standard Laboratory Signal Generator*

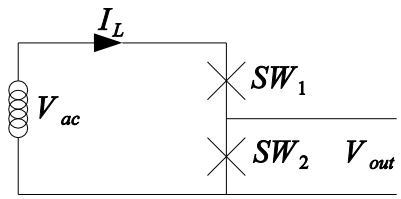


Figure D.2: *Half-Bridge Topology*

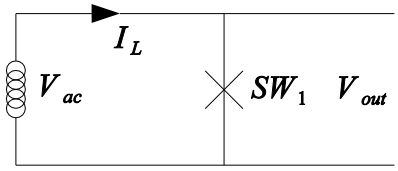


Figure D.3: *Parallel Switch*

The main part of this circuit is used in the parallel-switch design and discussed briefly in Section 3.4.2, page 18, and shown in Fig.3.5.

D.1.3 Half-Bridge

For higher voltage applications a half bridge switch (Fig.D.2) is provided in the laboratory. To get a feel for the power electronics it was decided to do a few experiments with this topology. The drawback of this configuration is that all the power delivered to the load must run through both SW_1 and SW_2 which, depending on the load on the network, can be around 25-200kVA[1] per transformer.

D.1.4 Parallel-Switch

The main advantage of the parallel-switch is that only during pulse generation time power is dissipated in the device. Therefore the power rating on the device is calculated depending on its position in the system.

D.1.5 Other Topologies

MPC The Magnetic Pulse Compression[76] unit can provide quick high voltage and energy pulses. Due to the mixed signal and possible harsh industrial environment the construction of the non-linear inductors was found too complex and therefore not used in the experiments.

[1]Typical values gathered from ESKOM. See Appendix E

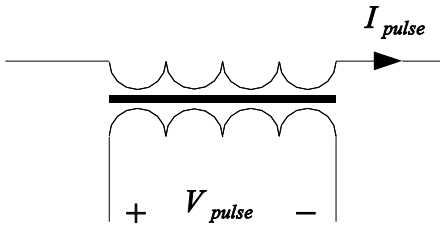


Figure D.4: *Current Transformer Topology*

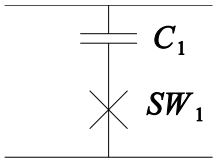


Figure D.5: *Capacitor coupled pulse generator*

Induction coupled pulse generator The use of an current transformer to induce current into the circuit transformer was investigated by sending a pulse through windings around small circular ferrite as shown in Fig.D.4. The results from these experiments were not significant enough and terminated.

Capacitor Coupled Pulse Generator The capacitor coupled topology as shown in Fig.D.5 was also considered and did not yield good enough results.

D.2 Experiments and Results

The experiments done in this seciton is for verification of the theory and for evaluation of the functionality of the various pulse generators. This section only covers the the following test phases namely, small signal experiments with a signal- and a glitch generator[92, ch.24].

D.2.1 Small Signal Signal-Generator Experiments

These experiment are variations of the one described by [92, ch.24]. The signal generator is used to generate the desired pulses. Reflection coefficients are easily controlled due to the fact that the signal generator and cables are matched at 50Ω. The typical setup is illustrated in Fig.D.6. We assume that the test setup does not have any reflection coefficients and therefore we will only consider the cable and devices under test in the explanation.

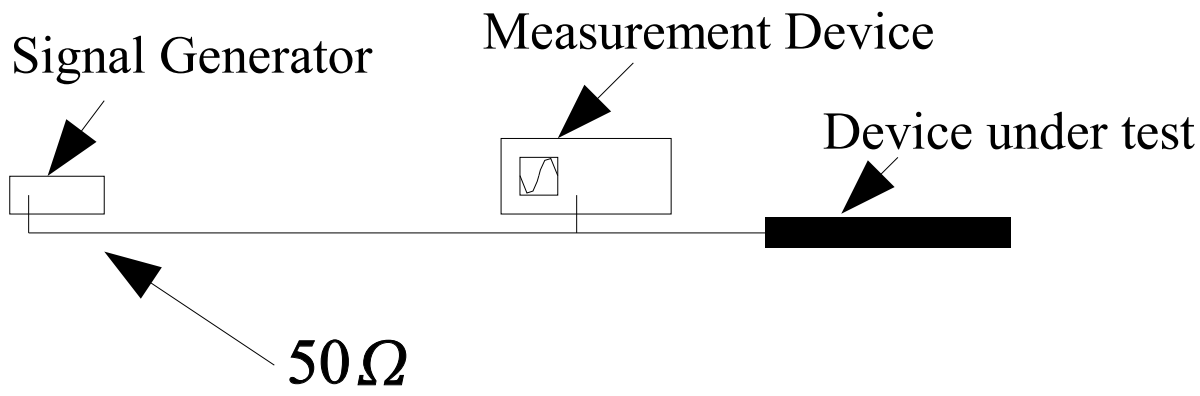


Figure D.6: *Default Test Configuration*

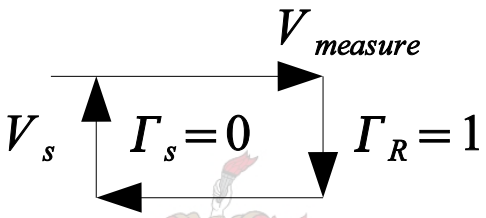


Figure D.7: *Signal Flow Graph of Open Circuit at Measurement Device*

Coaxial Line Reflection Coefficient Experiment

The idea behind the test is to get a idea of how reflection coefficients effect the circuit and give a feel for signal propagation in transmission lines.

Open Circuit Test From (B.44), with $Z_0 = 50\Omega$ and letting $Z_r \to \infty$ we get $\Gamma = 1$. Fig.D.7 shows the signal flow graph of the experiment. It can be seen from the results in Fig.D.10 that the reflected signal V^- adds to the forward going signal V^+.

Matched Circuit With a terminator of 50Ω added on the coaxial cable at the measuring point no reflections occur and the cable is matched. The signal flow graph is shown in Fig. D.8. Due to the fact that no reflection occurs at the terminated cable end, we only measure V^+ as seen in Fig. D.10.

24.6m Cable with Open End Consider a 24.6m 50Ω coaxial cable with an open circuit and repeat the experiment. A signal flow graph for the setup is shown in Fig. D.9. We can see from the results in Fig. D.10 that the pulse take some time to travel. Unfortunately the pulse width is

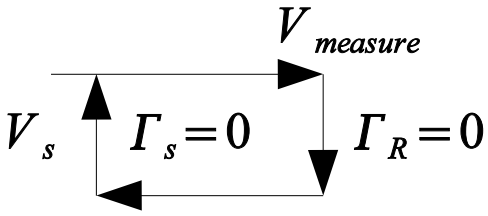


Figure D.8: *Signal Flow Graph of Matched Cable*

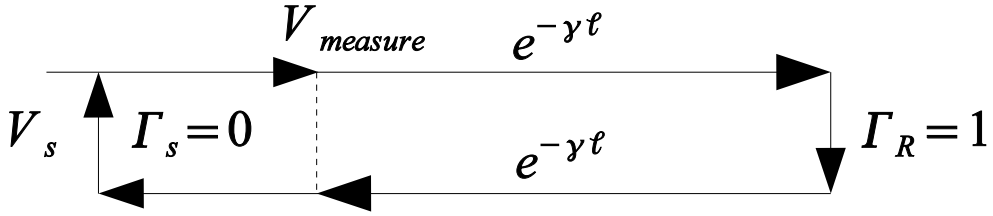


Figure D.9: *Signal Flow Graph for 24.6m Cable with Open Circuit*

too wide. The result is that the last portion from V^+ and V^- overlaps and therefore the small peak. From the rising edge of the first edge to the second edge is the total time for the pulse to return. From these it can be calculated that the speed in the coaxial cable with

$$t = 250\ ns \tag{D.1}$$

$$\ell = 2 \times 24.6\ m \tag{D.2}$$

$$v_{coax} = 196.8\ Mms^{-1} \tag{D.3}$$

This value is confirmed by [92, ch.24] with $v \approx 200\ Mms^{-1}$.

Results The results of the above experiments are shown in Fig. D.10. From the results we can confirm that the $V(0) = V^+ + V^- = V^+ + \Gamma V^-$.

Pulse Delay Experiment

The setup of this experiment is similar to the default experiment setup, except that the extra channel of the oscilloscope is used to measure the pulse at the end of the cable. The cables are matched and therefore only V^+ is measured. The signal flow graph for the two experiments are exactly the same as, shown in Fig.D.11, except that the delay and length of the different types of cables.

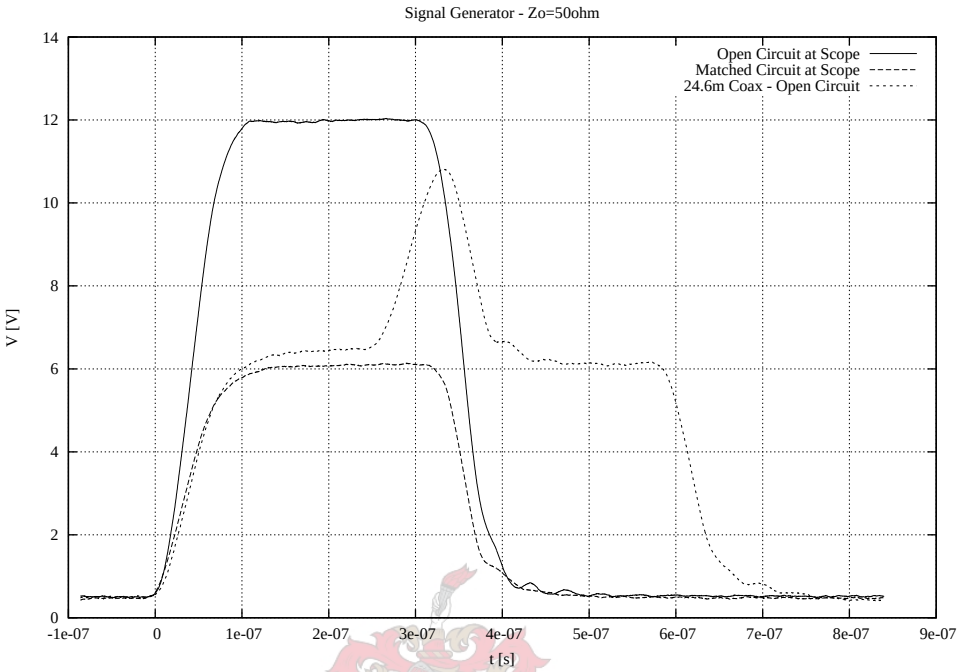


Figure D.10: *Results from Initial Experiment*

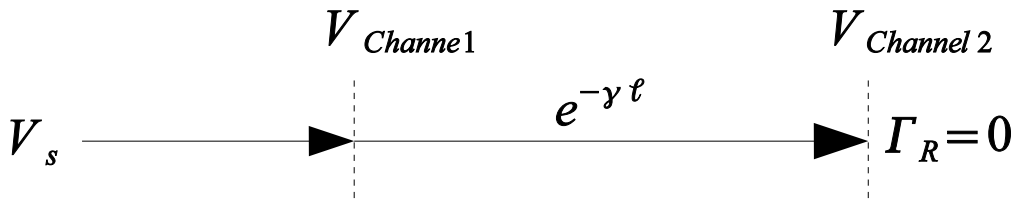


Figure D.11: *Signal Flow Graph for Delay due to Cable Length*

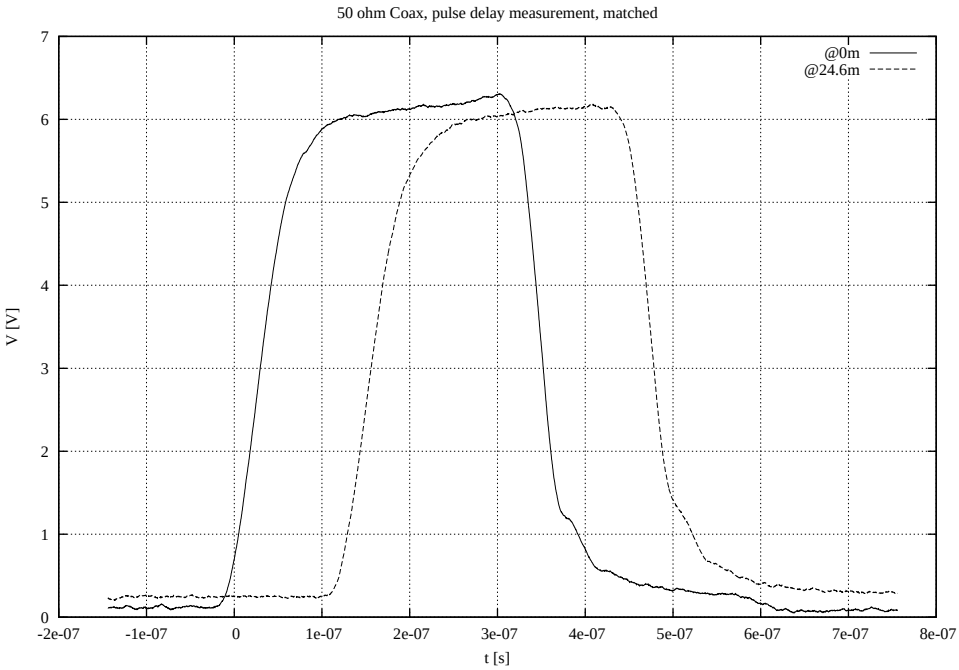


Figure D.12: *Delay Measurement in* 50Ω *Coaxial Cable*

Coaxial Cable A 24.6m coaxial cable is used for measurement. The results is shown in Fig.D.12 confirms the speed in a coax, where $t = 123.944ns$ and $\ell = 24.6m$ as $v_{coax} = 198.477Mms^{-1} \approx 200Mms^{-1}$.

Parallel Wire The $2.5mm^2$ parallel wire configuration is used. The wire is spaced roughly 5mm-15mm from each other. The isolation on the conductors were roughly 1mm. To match the cable a variable resistor was used. The results are shown in Fig.D.13. From the results the delay $t = 246.154ns$ over a distance of $\ell = 50m$ a speed of $v_{pw} = 203.125Mms^{-1} \approx 200Mms^{-1}$ is calculated.

Remarks The results shows clearly that pulses move along transmission lines as predicted by theory. The parallel wire and coaxial cable roughly has the same propagation speed.

Cable Junctions

For this experiment, the 24.6m coaxial cable is connected to the initial configuration shown in Fig.D.6. The cable is terminated with N-number coaxial cables, all having the same surge impedance. The results of the experiment is shown in Fig.D.14. The practical measure reflection coefficients are shown in Fig.D.15 against the calculated values as determined in (B.50). A summary of the results are given in the Table D.1.

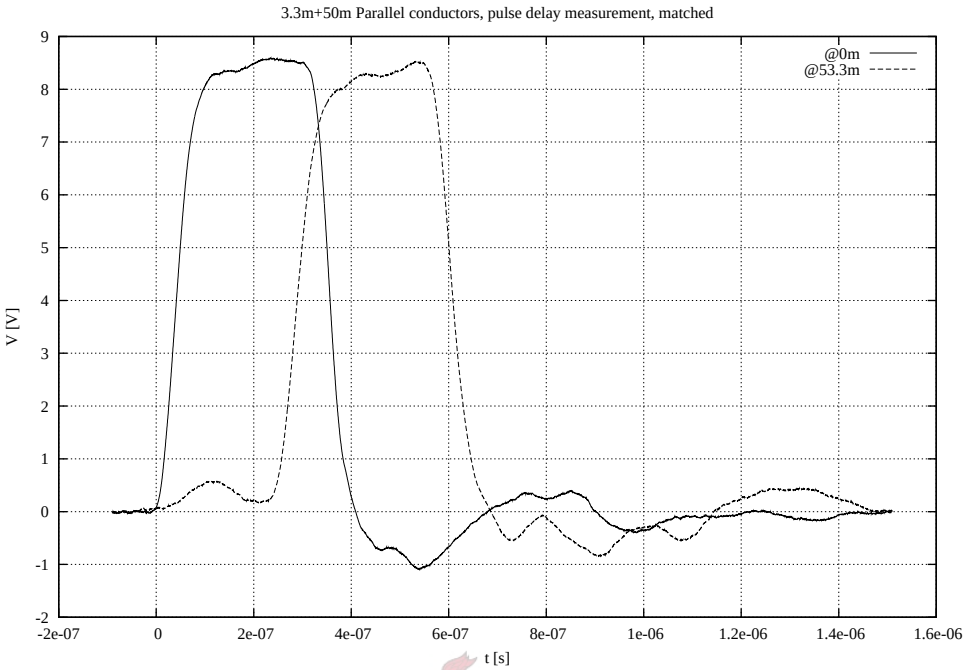


Figure D.13: *Delay Measurement in* $2.5mm^2$ *Parallel Wire*

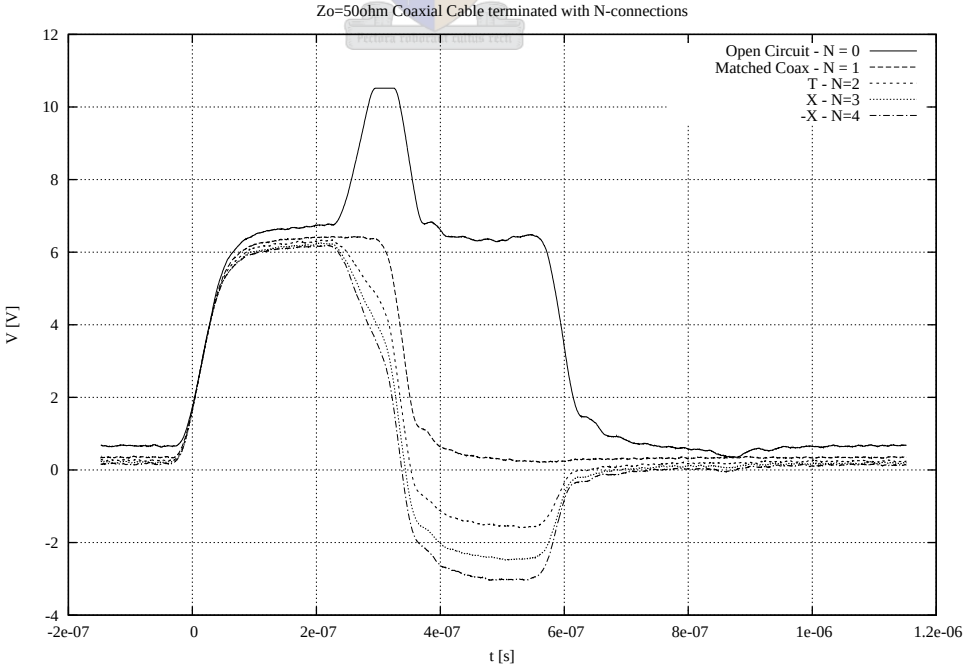


Figure D.14: *Measure Signals from N-conductor Split pole Configuration*

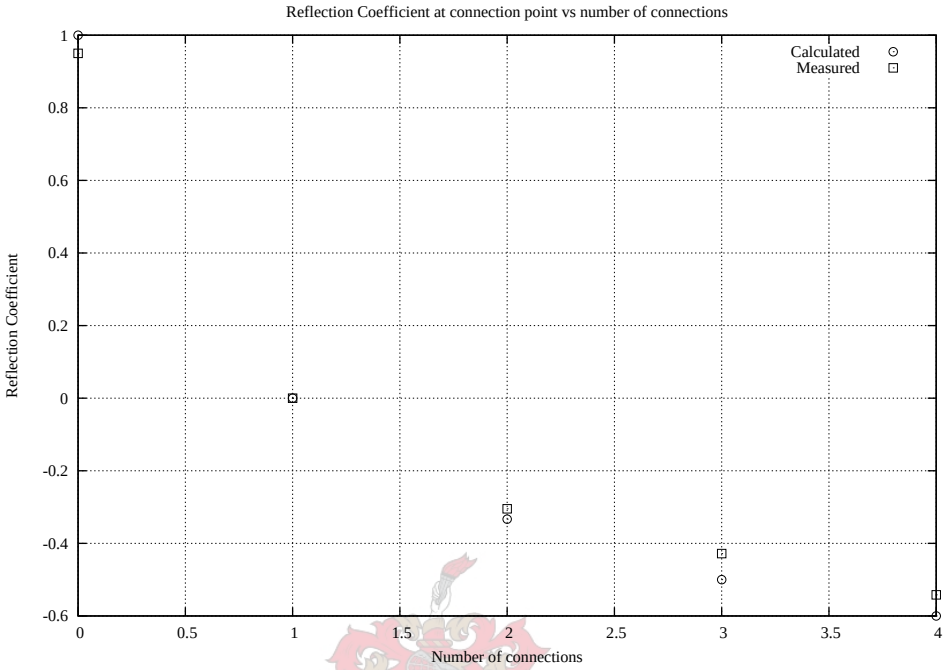


Figure D.15: *Theoretical vs Measured N-conductor Split Pole configuration*

N	$\Gamma_{measured}$	$\Gamma_{calculated}$
0	0.987	1
1	0	0
2	-0.308	-0.333
3	-0.428	-0.5
4	-0.542	-0.6

Table D.1: *Reflection Coefficient for N-conductor Split Pole configuration*

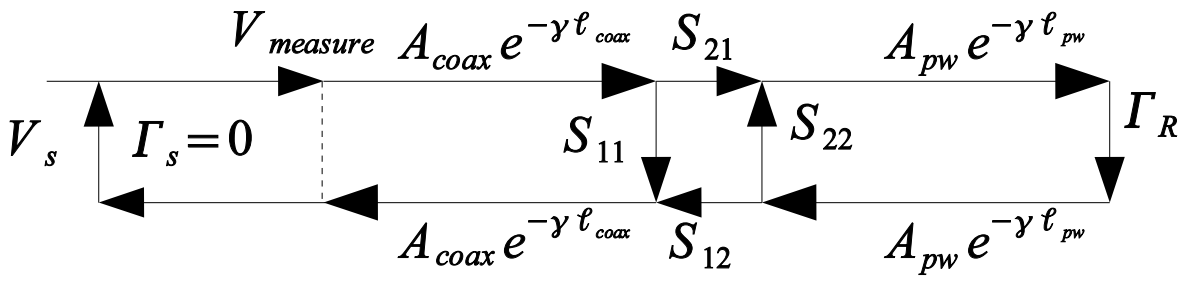


Figure D.16: *Signal Flow Graph of standard coaxial cable and parallel wire transmission lines*

D.2.2 Small Signal Glitch-Generator Experiments

To create a finer resolution and longer delay between pulses a glitch generator as proposed by [92, ch.24] was built and used for the mixed cable experiment.

D.2.3 Mixed Cable Experiment

For this experiment a 50Ω coaxial cable and the parallel wire transmission line is used and combined with the setup as shown in Fig. D.6. The parallel wire for this case was closely spaced next to each other.
First the open circuit reflection coefficient of the coaxial cable is measured. The next step is to extend the line with the 50m parallel wire. Two reflections is measured. For the second experiment with the open connection parallel wire fitted to the coaxial cable, four reflections are measured.
The signal flow graph for both experiments is shown in Fig.D.16. From the results in Fig.D.17 the different parameters for the scattering matrix where the junction occur in Fig. D.16 is calculated.

Coaxial Cable with Open Connection The 50Ω coaxial cable is considered. This is a well known standard and reference for the rest of the system. The coaxial cable is terminated with an open connection, which gives a $\Gamma = 1$. From here the attenuation A_{coax} is calculated:

$$V_{in} \approx 2.35V \tag{D.4}$$

$$V_{return} \approx 1.7V \tag{D.5}$$

where

$$V_{return} = V_{in} A^{+}_{coax} \Gamma A^{-}_{coax} \tag{D.6}$$

$$\Rightarrow A_{coax} = \sqrt{\frac{V_{return}}{V_{in}\Gamma}} \tag{D.7}$$

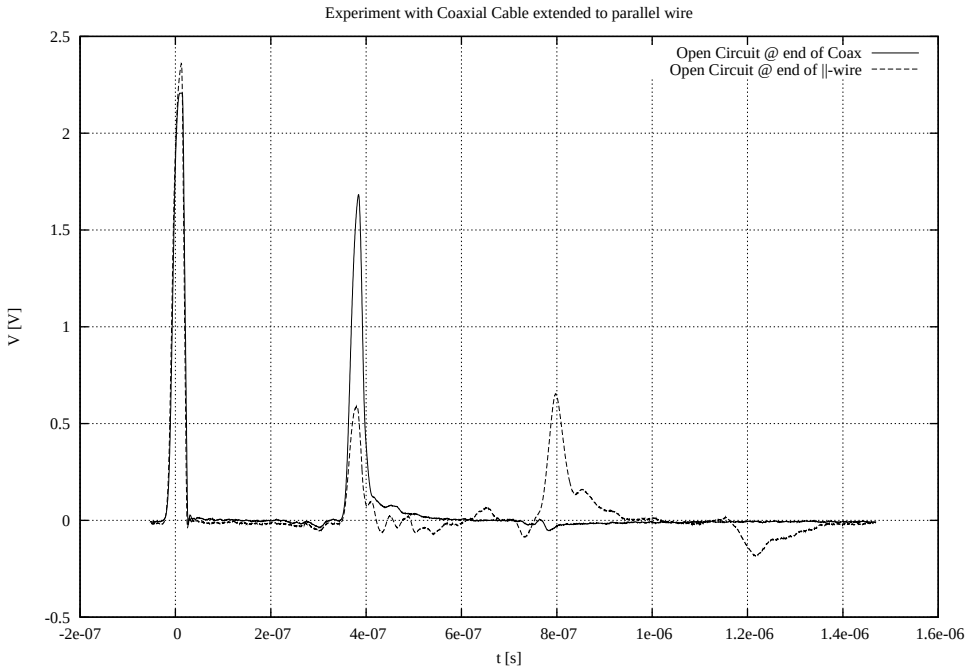


Figure D.17: *Reflections from Coaxial cable parallel wire experiment*

$$\approx \quad 0.850 \tag{D.8}$$

Parallel Cable with Open Connection Now consider the case where the parallel wire is connected. The first reflection with

$$V^{+} = V_{return_{coax_{open}}} \quad = \quad 1.7V \tag{D.9}$$

$$V^{-} = V_{return_{coax_{parallelwire}}} \quad = \quad 0.6 \tag{D.10}$$

is measured and hence the reflection coefficient is calculated as

$$\Gamma = S_{11} = 0.352941 \tag{D.11}$$

The transmission coefficient in the forward direction is calculated

$$T = S_{21} = 1 + S_{11} = 1.352941 \tag{D.12}$$

From (D.11) and (B.44) the surge impedance for the parallel wire is calculated

$$Z_0 = 104.545\Omega \tag{D.13}$$

Now S_{22} and S_{12} is calculated

$$S_{22} \quad = \quad -0.352932 \tag{D.14}$$

$$S_{12} \quad = \quad 0.647068 \tag{D.15}$$

The attenuation in the parallel wire is calculated. Due to the open circuit at the end it can be calculated with

$$\Gamma = 1 \tag{D.16}$$

$$\tag{D.17}$$

that

$$V^{+} = 2.35V \tag{D.18}$$

$$V^{-} = 0.65V \tag{D.19}$$

The total signal flow path for the first reflection from the open circuit is expressed as

$$V_{measured} = V_{sent} A_{coax} S_{21} A_{pw} \Gamma A_{pw} S_{12} A_{coax} \tag{D.20}$$

$$\Rightarrow A_{pw} = 0.661286 \tag{D.21}$$

With these known the next reflection coefficient is calculated

$$V_{measured} = V_{sent} A_{coax} S_{21} A_{pw} \Gamma A_{pw} S_{22} A_{pw} \Gamma A_{pw} S_{12} A_{coax} \tag{D.22}$$

$$\approx -0.1V \tag{D.23}$$

which is close enough to $-0.18V$ for a first iteration, although roughly 44% accurate, which can be attributed to possible distorion in the cable. When considering the time delay between the last two delays we see that these are roughly the same, $400ns \approx 420ns$. Which in turn give a propagation speed of $250Mms^{-1}$.

Discussion A difference in the propagation speed and the surge impedance for the parallel wire was seen when comparing the results of previous experiments. This can be attributed to the fact that for this experiment the cables were placed closer and precisely next to each other.

D.2.4 Conclusion

The experiments yielded good results. From the measured data the transmission line theory was verified. The fact that reflection coefficients at the split poles are large, as well as the relative high attenuation in the parallel wires is of great concern. The distortion levels in the parallel wire are larger than expected.

Appendix E

ESKOM Grabouw Field Visit - Nicky Schneider

A field trip was organized at ESKOM distribution - Grabouw to investigate the practical aspect of the project. Good insight was gained in how a Low Voltage Reticulation network is designed. The main network components were seen. Typical layouts and tampering cases were showed.

Weekday	Day	From-To	Location
Friday	04/06/2003	9:00-12:00	ESKOM - Grabouw
Project:	LV Electricity Theft		
Title:	LV Reticulation Field visit		

Participants			Notifications		
Company	Name	Location	Company	Name	Location
US-PEG	Riaan Doorduin	Stellenbosch	US-PEG	Prof. HdT Mouton	Stellenbosch
ESKOM	Nicky Schneider	Grabouw			

E.1 Contact Details

Tel.: (021) 859 4163

E-mail.: nicky.schneider@eskom.co.za

E.2 Network Components

The main power of an area is fed from the 3ϕ 11kV Δ network. A VT-CT combination device measures the power flow into this area. From this device 11kv:400V transformers are installed to supply the client side Υnetwork. Typical transformers seen ranged from 25kVA to 200kVA. This is illustrated in Fig.E.1.

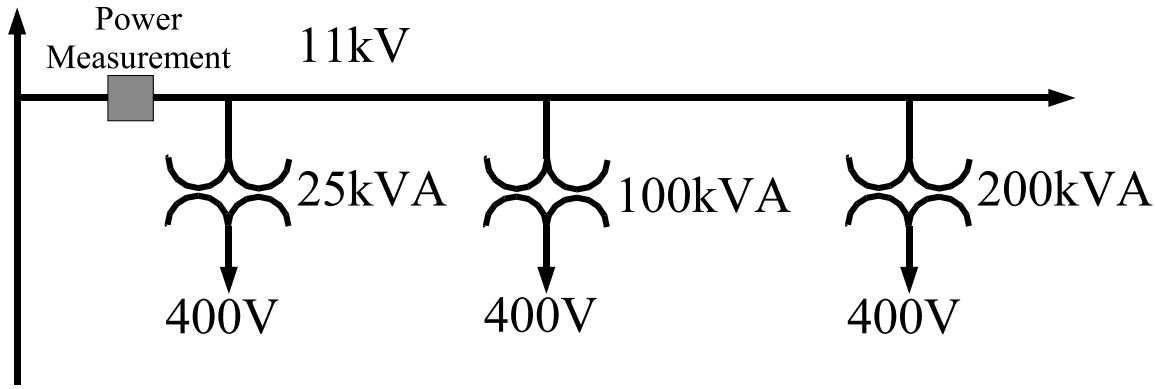


Figure E.1: *Medium Voltage distribution*

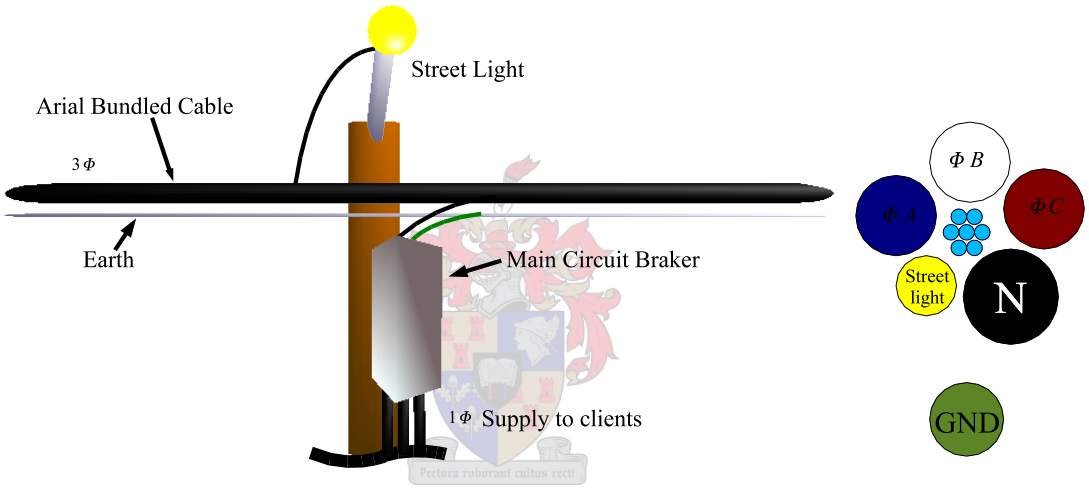


Figure E.2: *Cable configuration on overhead distribution poles and front view of ABC*

From the transformer the network is fed with ABC's[1]. These consist of a steel supporting cable(light blue), a neutral return path(black), street light feeder(yellow) and the 3ϕ's, blue, white and red. On the poles of the overhead network a earth cable(green) is also included. A typical pole and cable configuration is shown in Fig.E.2 with the MCB[2] isolating the single phase running to the clients houses. Up to nine houses can be connected to one of these MCB's.

[1]Aerial Bundled Conductors

[2]Main Circuit Breaker

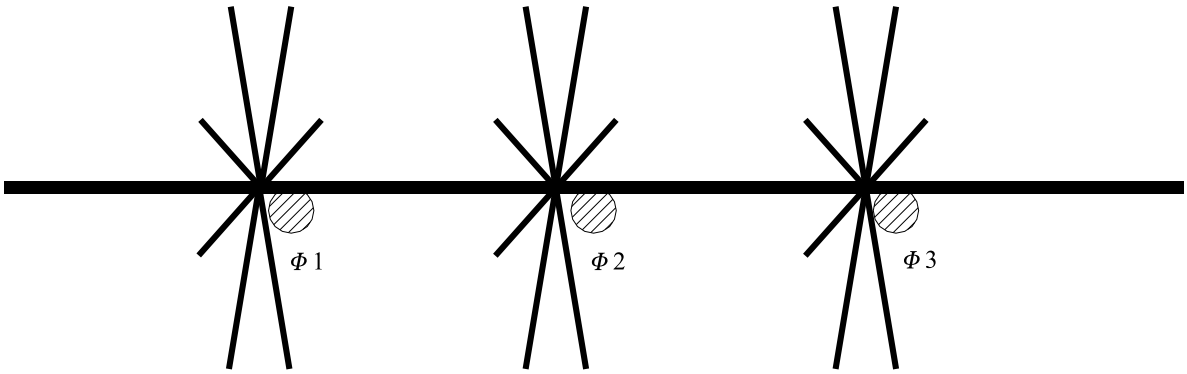


Figure E.3: *3 phase load distribution*

E.3 Network Layout

The LV network is laid out to optimize both cable usage and balance the load per phase. In Fig.E.3 it is shown how each third distribution pole use a separate phase, which optimize load-phase balancing.
The LV transmission lines are laid out similarly to a tree or root like structure. A typical layout is shown in Fig.E.5 and Fig.E.4. It can be seen that the ABC provide the cable to the nearest pole with a MCB. From the MCB a feeder cable provides power to the client.

E.4 Electricity Theft and Tampering

In Grabouw, Electricity theft and tampering is not common practice, although it was seen a few times. Tampering with the meters is reduced by changing to a new meter system and providing electricity to more houses, which reduce the need for private extensions of the network. However, some isolated cases were seen. Typically tampering occurs inside the huts. Flex and thin cables run either over and along the roads or through trees as shown in Fig.E.6. Typical flex, thin cable and $5mm^2$ wiring is used. This means that a higher than normal reflection coefficient could be expected due to a high probability of mismatching with an TDR solution.

E.5 General and Conclusions

The exposure to the a practical network gave incredible insight in both technical and social understanding of this particular environment. Network components and layout was discussed as well as typical tampering methods. The large distances between transformers and substations or ESKOM buildings will pose a problem setting up a possible measurement and

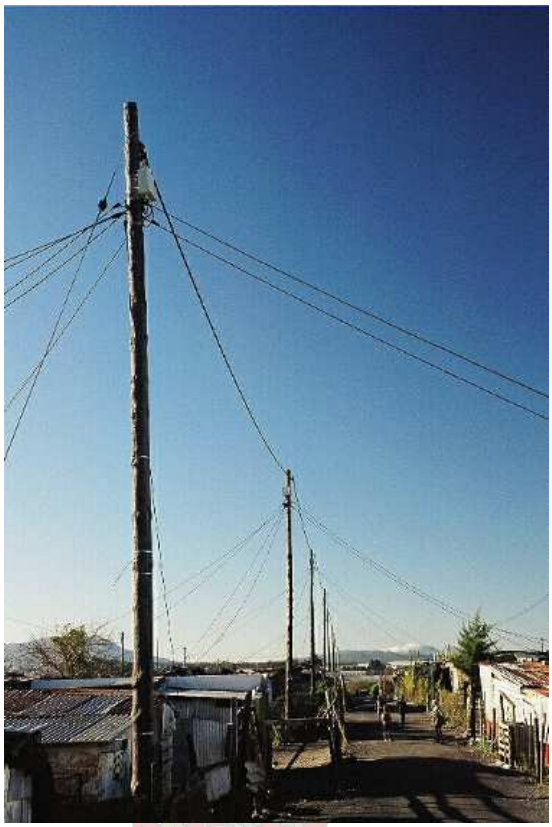


Figure E.4: *Reticulation Example*

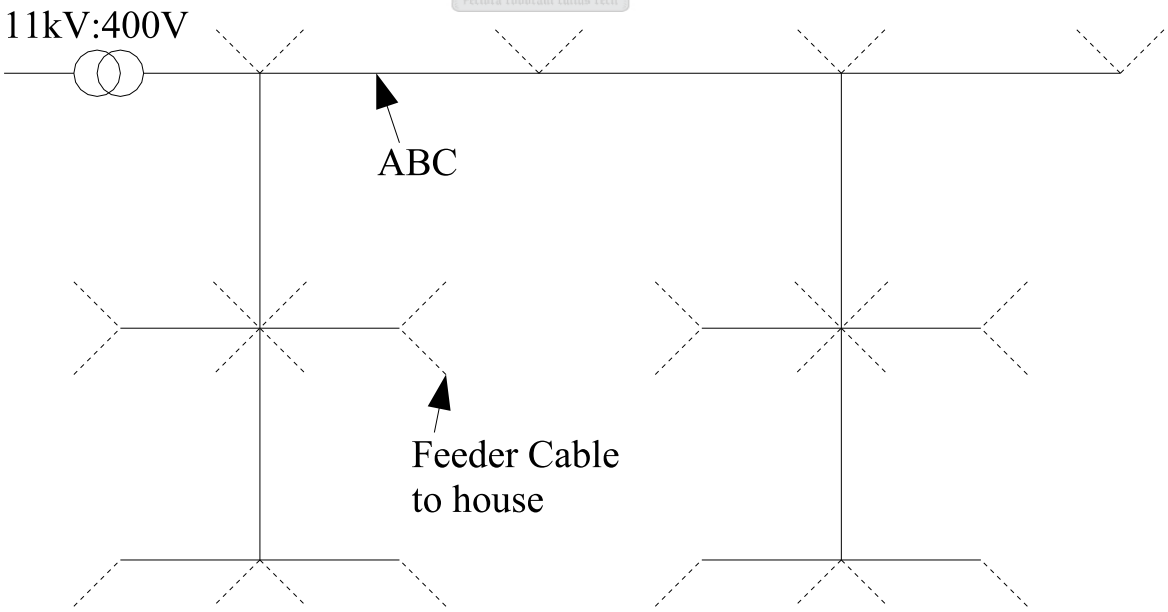


Figure E.5: *Low Voltage Reticulation Example*

Figure E.6: *Examples of Electricity Theft*

detection scheme. The length and size of the reticulation network may have a larger attenuation effect on the pulses generated with TDR than expected and must be investigated.

I would like to thank Nicky Schneider and Robert at ESKOM, Grabouw whom have made this valuable visit possible.

Appendix F

ESKOM Brackenfell - Andre Truter

Andre Truter is Security Manager at ESKOM Brackenfell. He works closely with SAP and all Electricity Theft cases in the Western Cape is handled by his department.

Weekday	Day	From-To	Location
Friday	30/07/2003	10:00-11:00	ESKOM - Brackenfell

Project: LV Electricity Theft

Title: Impact on Security

Participants			Notifications		
Company	Name	Location	Company	Name	Location
US-PEG	Riaan Doorduin	Stellenbosch	US-PEG	Prof. HdT Mouton	Stellenbosch
ESKOM	Andre Truter	Brackenfell	US-PEG	Dr. JH Beukes	Stellenbosch

F.1 Contact Details

Tel.: (021) 980 3533

E-mail.: andre.truter@eskom.co.za

F.2 Project Discussion

The overview of the current research was discussed up until 2003/07/30 and a presentation was given.

F.3 How does Eskom experience electricity theft and tampering?

F.3.1 Cable Theft

There are many ways to steal electricity and in broader terms this also includes the theft of cables.

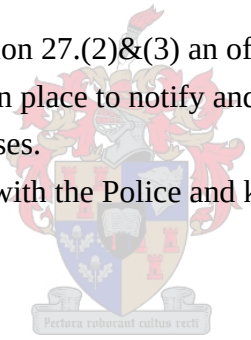
The most common practice is the theft of cables, especially the high voltage cables, which leads to many deaths. Effort is spent on educating the communities about electricity by means of the direct approach e.g. showing pictures of people shocked to death. The results of this initiative is still debatable.
Another approach is to remove the incentive of stealing cable, by applying strict regulation to the depots buying copper and cable. This problem is not limited to ESKOM and therefor a forum was established to initiate actions and ideas around the topic.

F.3.2 Electricity Theft

Cases of electricity theft and tampering on the low voltage network are known. Meters are either tampered with or bypassed. This happens on the split poles or ABC, on the airdacs and inside the huts, wherever the most appropriate place may be for the offender.

F.4 There is the law and then there is justice

According to Act 41 of 1987 Section 27.(2)&(3) an offender can be prosecuted for theft. ESKOM also has documentation in place to notify and offender of such activities and can disconnect the power to the premises.
Andre and his team work closely with the Police and keep trends of perpetrators.



F.5 Suggestions

Andre main concerns and suggestions are:

1. People must be sensitized
2. Mechanisms and systems must be automated.
3. Manpower is becoming a problem
4. Confronting perpetrator are extremely dangerous
5. Upfront vending as means of sales should be considered. - Units are prepaid for by the vendor to reduce ESKOM’s responsibility in terms of revenue. This will however imply that the vendor can sell the electricity at a higher price.
6. A cashless vending principle should be considered.

F.6 Contacts

The following people can be contacted for further information:

Jan Scholz	(021) 980-3122
	Power line communication and availability
	of data points on HV lines
Rens Bindeman	(021) 959-5352
	(082) 850-5318
	rens@pn.co.za
	SARPA - chairperson

F.7 Conclusion

Regarding a solution relevant to my study, LV Electricity theft prevention, the focus should be on automation, thus reducing the human factor as much as possible. If the study's solution can include cashless vending it would be an added benefit.

Appendix G

ESKOM Bellville - Monde Moletsane

Monde Moletswane is the Regional Revenue Protection Manager of the Western Cape.

Weekday	Day	From-To	Location
Friday	04/07/2003	13:00-14:00	ESKOM - Bellville
Project:	LV Electricity Theft		
Title:	Revenue Protection		

Participants			Notifications		
Company	Name	Location	Company	Name	Location
US-PEG	Riaan Doorduin	Stellenbosch	US-PEG	Prof. HdT Mouton	Stellenbosch
ESKOM	Monde Moletswane	Bellville	US-PEG	Dr. JH Beukes	Stellenbosch

G.1 Contact Details

Tel +27 21 915 2799
Fax +27 21 915 2928
E-mail monde.moletsane@eskom.co.za

G.2 Project Discussion

A brief idea of my current work was presented and discussed.

G.3 Current Situation

Cable theft is the main concern and is the main contributer to losses in the Western Cape region.
Regarding Electricity Theft and tampering, getting access to property for inspections and follow ups is a problem.

The customers experience ESKOM personnel as police and therefor cooperation is not good. Physical threats are common. There is expectation of permanency which results in demands in terms of electricity. Some consumers view it as a basic right.
Another problem is the illegal distribution of electricity. This involves paying customers selling from their electricity to neighbors via extension leads. This pose a safety threat. Education is used as a means of opposing the problem.
Losses due to electricity theft in the Western Cape roughly amounts to 10% (R6 million per annum) of the total energy distributed.

G.4 Availability and Geographical Correlation

Most of the problem does not occur in the rural areas, but in settled urban areas, where the community have more criminal tendencies. Post and prepaid metering are both targeted. Regarding this issue, Monde, found that people in rural areas are generally willing to pay. And that the losses, in terms of Electricity theft, here are really small compared to urban areas.
As mentioned in my presentation I pose the question that providing electricity to illegal users, would then reduce the problem. It was agreed upon that the opposite is also true, that providing electricity to more users makes the network more vulnerable to tampering and electricity theft.

G.5 Processes

Meter audits are done on a door to door basis. This is the safest way of knowing what is happening in the field. To refine and narrow the audits exception reports are drawn from the available data. These reports are only as accurate as data is maintained in the system, and it does occur that meters with a zero reading was removed by ESKOM personel and not updated in the system.
Energy balances also indicate areas of high revenue losses where losses are calculated on the sold units vs. used units.

G.6 Prepaid Metering

Revenue protection had its origin in the 90s and grew hand in hand with the evolution of prepaid metering. Due to the fact that these meters are out of sight and that the meter is not visited once a month, it is more susceptible to tampering.
Prepaid metering also introduced ESKOM to a new type of customers.

G.7 Postpaid Metering

Monde explained that the current focus is on postpaid meters, since these meters are also tampered with. Although postpaid meters are less in number than their prepaid counterparts, postpaid meters account for more revenue.

G.8 Basic Free Electricity

Government indicated that, as with water, people have a right to energy. Monde argues that there are other resources which they can turn to and if this grant will help the poor is still highly debatable. In the same light it was found that the perception exists that electricity is an expensive form of energy and therefor people use electricity only for lighting purposes. I guess that education plays an important role here.
From the 1st of July people can apply for 50kWh per month at the local authority. Government budgeted R300 million for this year and R1.7 billion for next year. The shortfall here is that no profile exist for deciding on who should get this subsidy.

G.9 Conclusion

Concerns such as accessibility to premises and physical threats must be addressed. Human error must be minimized in terms of data collection. Postpaid meters must also be included in the analysis of electricity theft prevention and a possible solution must be applicable in both pre- and postpaid environments.

Appendix H

ESKOM TSI - Dr. Nielsen

Dr. Shawn Nielson is affiliated with both ESKOM TSI and the University of Pretoria. His current research involves detecting water in HV insulators.

Weekday	Day	From-To	Location
Thursday	14/08/2003	10:45-12:00	ESKOM-TSI - Cleveland

Project: LV Electricity Theft

Title: TDR in a LV Reticulation Network

Participants			Notifications		
Company	Name	Location	Company	Name	Location
US-PEG	Riaan Doorduin	Stellenbosch	US-PEG	Prof. HdT Mouton	Stellenbosch
ESKOM	Dr. Shawn Nielsen	Cleveland	US-PEG	Dr. JH Beukes	Stellenbosch

H.1 Contact Details

Tel +27 12 629 3569

Fax +27 12 629 5454

Cell +27 83 617 4820

E-mail shawn.nielsen@eskom.co.za

H.2 Project Introduction

A brief overview of the project was given, with the specific focus on TDR. Dr. Nielson was impressed by the parallel switch solution i.t.o. generating an in line pulse and the sharp dV/dt's measured using this method.

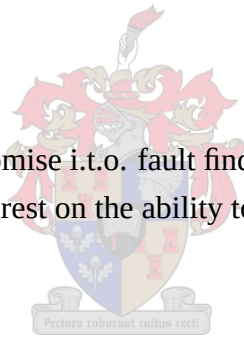
H.3 Tips & Tricks

Dr. Nielson proposed a directional bridge circuit to only measure pulses sent from the pulse genrator. This solution can propably be used to illiminate reflections from the pulse generator

itself.
By changing the pulse width information could be extracted using FDR (Frequency Domain Reflectiometry). The longer the line the higher the pulse frequency content must be. Standing waves would then be used to extract information from the network.
The main problem with TDR in a LV network would be to decode or interpret the signals received. Therefore the network must be modeled extremely accurate and methods should be investigated on how to estimated and determine connections. Since we are measuring in one dimension, GIS and other data can be used to deterimine the propability of certain phenomina measured by the system.
Temperature would not have a great impact on measurements since it would be spread evenly over a distance.
Dr. Neilsen was extremely positive about the current progress in the project. The possibility of other spins offs from the results of the study could be interesting. Detection of small faults could be made possible. However the relaxation of capacitive loads could pose a problem and must be investigated.

H.4 Conclusion

The study thus far shows great promise i.t.o. fault finding on cables. The success for detecting tampering in a LV network would rest on the ability to interpret the recieved signals.



Appendix I

Ekurhuleni Metropolitan Municipality - Dave Jamieson

Dave initiated the installation of a AMR (Automatic Meter Reading) system in the Thembisa area. Although expensive, the system have proved itself since 1996.

Weekday	Day	From-To	Location
Friday	15/08/2003	11:00-13:30	Thembisa
Project:	LV Electricity Theft		
Title:	AMR in a LV Reticulation Network		

Participants			Notifications		
Company	Name	Location	Company	Name	Location
US-PEG	Riaan Doorduin	Stellenbosch	US-PEG	Prof. HdT Mouton	Stellenbosch
EMM	Dave Jamieson	Thembisa	US-PEG	Dr. JH Beukes	Stellenbosch

I.1 Contact Details

Tel	+27 11 921 2477 / 926 2742
Fax	+27 11 975 4614 / 926 2804
Cell	+27 82 490 2627
E-mail	davej@ekurhuleni.com
Web	http://www.ekurhuleni.com

I.2 Project Introduction

Dave introduced me to the project at Ekurhuleni. He compiled a complete folder with presentations and articles.

I.3 Extract from Papers and Background

I.3.1 From ”Saga of protective structures”

Reference: [44]

Background

In January 1995 the Tembisa network with its 31500 consumers became the responsibility of Kempton Electricity with a bulk bad debt of R250 million. This debt was increasing with interest at prime and an alarming R3 million per month due to a non-payment campaign in the community. Networks were also overloaded due to a *self-help culture* in the form of illegal connections. An investment of over R100 million i.t.o. network renovation, electrification, meter replacement and the deployment of SCADA and telemetry was made to address this problem.

Why this system

Prepaid metering was introduced in the 1980s and was soon proved prone to electricity theft and fraud. The ingenuity of the domestic consumer was either totally overlooked or underestimated. Technical losses increased due to this and in 1996 more than 50% of all pre-payment meters in South Africa were bridged.

> Initially this loss represented a relatively small percentage of total energy sold, but with South Africa venturing into the ambitious electrification drive as it did in 1990 with less than 40% of all households being electrified at the time, it was realized that this sector would be growing cosiderably with resultant increasing losses to the industry. In 1995 the outstanding debt in this sector, country wide, amounted to a sum in excess of R2.0 billion.

Uniqueness of the system

System requirements were defined:

- Two way communications
- Meter and display separation (split meter)
- Telemetering, remote billing, cash collection and automated energy auditing
- Devices must have tamper detecting
- Long term cost effectiveness

System lay out

Up to 20 meters can be connected to concentrator in a mini sub. The substation communicates via a radio network to the control center. (Will be elaborated later in these minutes.)

Project implementation

Community involvement was considered to be of high priority. The first meters were installed in February 1997.

Protective structures

Tampering to the municipalities meters were common practice. This problem was addressed by experimenting with different protective structures. 10mm thick hydraulically operated steel structures were put over the metering kiosks and miniature substations at a cost of R24 million. With all these measures in place (including AMR) payment levels rose from ±R1million to R8.6million per month.

Latest developments

> The total cost for this investment in new technology(High-tech metering & protective structures) soon paid for itself not only from additional revenue received and reduce cost to suppliers (ESKOM) but also the savings in effective manpower utilization for maintenance and new projects as apposed to continually removing illegal connections and repairing vandalized equipment.

Regognition award

The Kempton Park/Tembisa Electricity Department together with Intelligent Metering Systems won the Residential Category of the prestigious 2002 ETA Award for promoting energy efficiency.

I.3.2 From "'Conversion to Pre-paid"'

Refference: [43]

Background

The removal of charges[1] was a strong motivation for customers to use to prepayment instead of conventional metering. Awareness and understanding of how the system worked was achieved

[1] R25 for the delivery of notice and R150 for discontinuing and restoring supply

by educating the community through a comprehensive market strategy.

The System

The system works in both post and pre-payed mode. A card is issued to each stand and is used to eliminate finger problem at the vending stations[2]. The metering system can communicate bi-directionally via various communication mediums to the control center. When a sale is made the new credit is loaded onto the stand's meter and can be used. To update the credit on the meter roughly takes 15 minutes. At the beginning of the month 50 units are loaded onto each consumers meter in order to assist those who cannot afford essential services.

Meter Advantages

- Both pre and post-payed functionality - without meter readers
- Remote management
- Tamper and bypass alarm
- Card eliminates finger problems at display terminal
- Credit is loaded automatic
- Monthly, weekly and daily records are stored per consumer
- SQL compliant database functionality implies easy integration
- Account details can be changed instantly
- Power can be switched on in case of an emergency
- Faulty equipment can easily be identified
- System priorities can be set in terms of maintenance tasks, e.g. Reconnection of customer, alarms and non functional meters
- Up to date information on customers display
- Maintenance reports can be printed daily
- Tampering alarms is monitored in the control center
- Display is operated via mains borne modem, therefore no additional communications cables are required
- Split meter principle i.t.o. meter and display (Accessible to Technicians)

[2]Ideal for the illiterate

Meter Disadvantages

- Expensive capital layout
- Not STS compliant
- Display does not function when electricity is cut off, due to mains modem
- Sole supplier of equipment

Conclusion

AMR has been found to be successful in terms of revenue protection in the Ekurhuleni Metropolitan Council.

I.3.3 From presentation ”Conversion to Prepayment”

Additional extracts from [43]

Levies vs. Payments

June 2002	30%
December 2002	51%
June 2003	59%

I.4 Technical Observations

The technical personal were introduced, as well as the equipment used in the system. A visit to the control room as well as printouts from the database was shown.
A technician showed how the system is implemented.
The vending system is very simple to operate and clerks are familiar with the process. People in general seem happy with the system.
Operation of the protective structures were shown. These are operated by hydraulic and electrical means. Mechanism to detect tampering are installed and in some cases these structures are connected to the metering system via optical fibre.
Hans said that in one case, a list of roughly 2500 were disconnected within 2 hours.
A substation was also shown.
The card used for customer identification in the system was shown. The magnetic strip only contains the account number. On the card, both the account number and stand number is shown. The stand number can be decoded as follows:

S72/	001/	0990	/0000
area	extension	stand	portion

I.5 Contacts

Further technical information can be requested at qurynl@ekurhuleni.com.

I.6 Conclusion

Ekuhurleni is currently reaping the benefit of a well thought out system. The load on human resources is lower than with the previous system. The combination of protective structures and AMR reduces tampering. The sole supplier situation is not healthy.

Appendix J

Stellenbosch Municipality - Kevin Bey-Liveld

Kevin is involved with the Municipalities' investigations into Automatic Meter Reading (AMR).

Weekday	Day	From-To	Location
Friday	22/08/2003	11:00-11:45	Stellenbosch
Project:	LV Electricity Theft		
Title:	Check Metering		

Participants			Notifications		
Company	Name	Location	Company	Name	Location
US-PEG	Riaan Doorduin	Stellenbosch	US-PEG	Prof. HdT Mouton	Stellenbosch
S.org	Kevin Bey-Liveld	Anglo Building	US-PEG	Dr. JH Beukes	Stellenbosch
S.org	Gerhard W.	Anglo Building			

J.1 Contact Details

Tel +27 21 808 8406

E-mail kevinb@stellenbosch.org

J.2 Project Overview

The project was briefly explained and how the involvement of Stellenbosch Municipality can help the project.

J.3 Discussion

Stellenbosch Municipality does not experience large quantities of electricity theft. However they are planning to start investigating AMR.
Their main concern are in the bulk electricity users. The total income per year from 400 users is R80 million per year. For the 12530 household meters the income per year is roughly R12 million. For the period from July to August R1.8 million per month was consumed by households.
The extend of possible theft in Stellenbosch is not determined yet. Proposals of using InfoPOD's (R150 per meter) to collect data was found an expensive solution.
It was suggested, that the focus of the LV electricity theft study must be changed to bulk electricity users, since the usage here justifies the extra costs incurred for equipment.
Actaris have currently been involved in the deployment of 300 AMR meters in the Blouberg area.

J.4 Joint Venture

The possibility of working together with the municipality was discussed. Stellenbosch Municipality have ten meters and modems to use as check meters. Involvement from the Power Electronics group could be beneficial if an arrangement between the two parties can be reached. The results from the venture can be used for research and a business plan, to motivate further AMR installations, can be drawn up.

J.5 Contacts

Gerhard V.	+27 21 808 8406	gerhardv@stellenbosch.org
Imran Mohammid	+27 83 306 2413	Actaris AMR

J.6 Action Items

Riaan Doorduin will discuss the proposal with Prof. Mouton. Depending on the outcome, the final proposal from the Power Electronics Group will be sent to Kevin. Kevin Bey-Liveld will discuss the proposal with Stellenbosch Municipality.

J.7 Conclusion

The possiblity of joining strengths could pose tremendous progress in terms of revenue protection for Stellenbosch Municipality and give the study a good practical insight into AMR

as well as real-world figures.

”What you can not measure, you do not know.” In general municipalities are not aware of how much electricity is stolen in various areas. The losses are then assumed to be technical losses in the network and are recovered by price increases. To develop a low cost remote check meter could speed up detection and identification of electricity theft.

Appendix K

Data Mining Loss Calculations

K.1 Site Layout

The site layouts is provided courtesy of Stellenbosch Muncipality. [77]. An aerial photograph from the GIS system is shown in Fig. K.1 and the design of the circuit is shown in Fig. K.2 [18].

Figure K.1: *Mooiwater Arial View*

Figure K.2: *Engineering Drawing of Site*

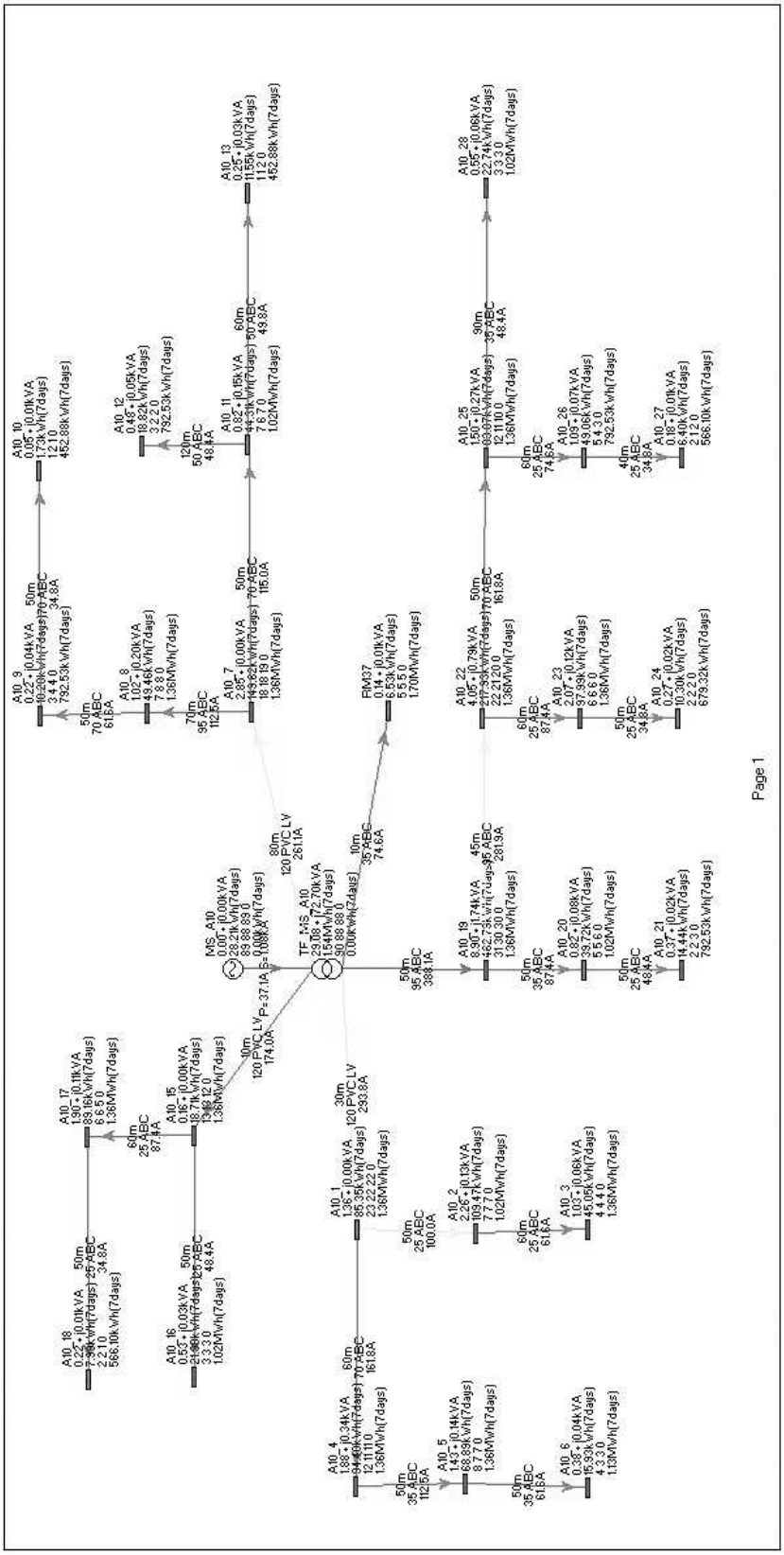


Figure K.3: *40A Standard user Retic-Master layout*

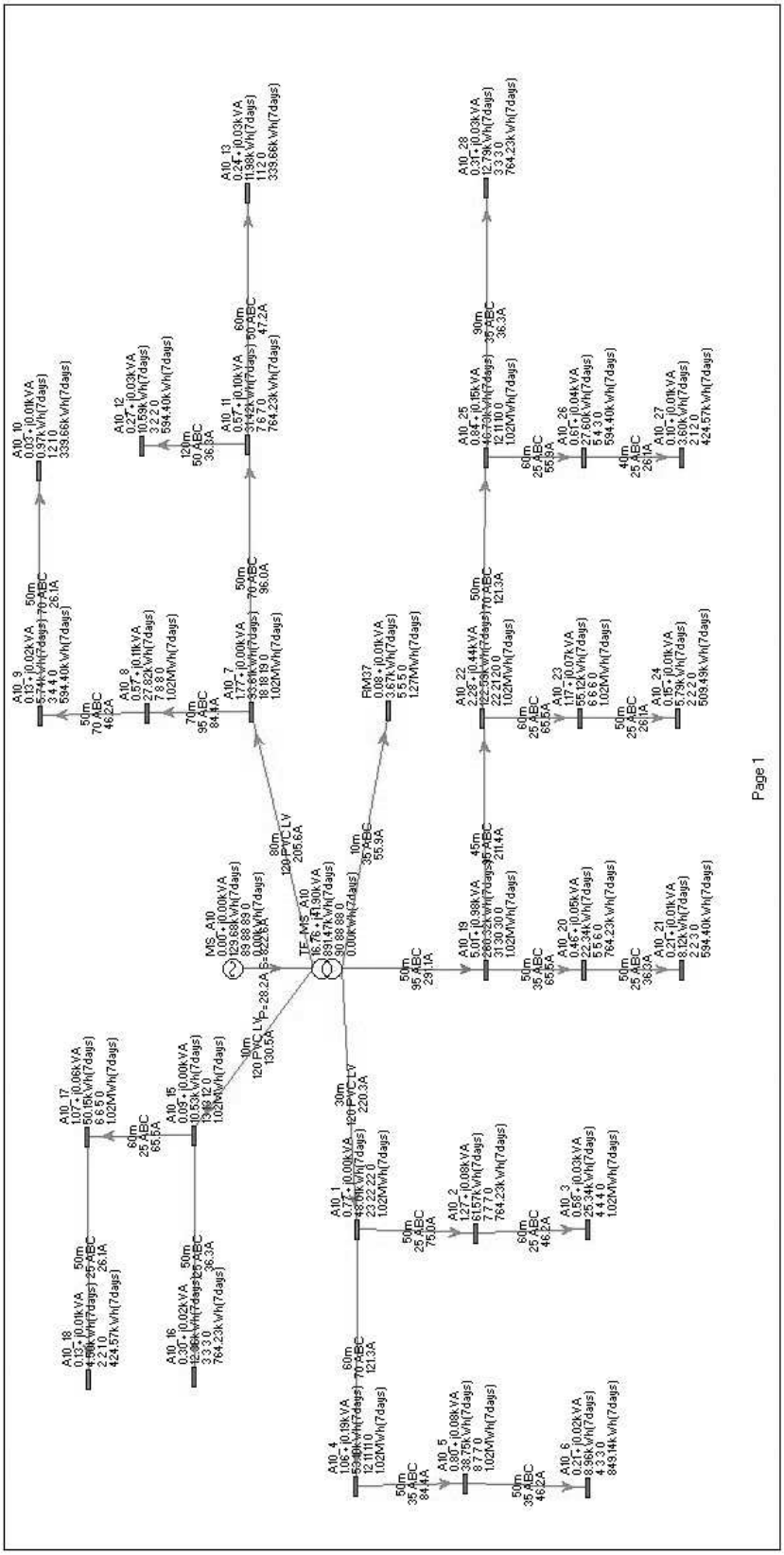


Figure K.4: *Retic-Master layout from data*

K.2 Reticmaster Simulations

K.2.1 Layouts

Two simulations were done using Retic Master. These graphical layout of the simulations are shown in Fig. K.1 and Fig. K.1.

K.2.2 Extracted Results

The following spreadsheets contains formatted results of the Retic Master simulations.

Standard consumer model

SORTED

From Node	Node Name	Total Load Loss Energy (kWh/7days)	Total Domestic Energy (kWh/7days)	% Total Losses (kWh)	Total Load (S,PF) (kVA)	Total Load Loss (S)	% Total Loss(VA)
-	MS_A10	3216.14	0	Err:503	813.96	0	0.00%
A10_1	A10_2	154.52	2377.6	6.50%	69.17	3.31	4.78%
A10_1	A10_4	179.23	3849.45	4.66%	107.75	3.72	3.45%
A10_11	A10_12	18.82	792.53	2.37%	26.97	0.48	1.79%
A10_11	A10_13	11.55	452.88	2.55%	27.85	0.25	0.90%
A10_15	A10_16	21.98	1018.97	2.16%	34.3	0.53	1.55%
A10_15	A10_17	97.15	1924.73	5.05%	57.77	2.12	3.68%
A10_17	A10_18	7.99	566.1	1.41%	20.6	0.22	1.07%
A10_19	A10_20	54.16	1811.51	2.99%	53.81	1.2	2.22%
A10_19	A10_22	487.48	7132.81	6.83%	182.31	9.81	5.38%
A10_2	A10_3	45.05	1358.63	3.32%	41.46	1.03	2.49%
A10_20	A10_21	14.44	792.53	1.82%	26.45	0.37	1.40%
A10_22	A10_23	108.29	2037.95	5.31%	56.83	2.35	4.14%
A10_22	A10_25	161.27	3736.23	4.32%	99.12	3.34	3.37%
A10_23	A10_24	10.3	679.32	1.52%	22.36	0.27	1.21%
A10_25	A10_26	55.46	1358.63	4.08%	39.64	1.27	3.21%
A10_25	A10_28	22.74	1018.97	2.23%	31.5	0.55	1.76%
A10_26	A10_27	6.4	566.1	1.13%	19.2	0.18	0.94%
A10_4	A10_5	84.82	2490.82	3.41%	71.3	1.81	2.54%
A10_5	A10_6	15.93	1132.19	1.41%	35.69	0.38	1.07%
A10_7	A10_11	74.68	2264.38	3.30%	77.51	1.57	2.02%
A10_7	A10_8	61.38	2604.04	2.36%	75.55	1.32	1.75%
A10_8	A10_9	11.93	1245.41	0.96%	39.6	0.28	0.72%
A10_9	A10_10	1.73	452.88	0.38%	17.38	0.05	0.29%
MS_A10	TF_MS_A10	3189.64	30116.31	10.59%	772.54	101.53	13.14%
TF_MS_A10	**A10_1**	419.09	7585.69	5.52%	205.11	8.37	4.08%
TF_MS_A10	**A10_15**	137.85	4302.33	3.20%	122.65	2.81	2.29%
TF_MS_A10	**A10_19**	801.86	10302.95	7.78%	263.88	20.06	7.60%
TF_MS_A10	**A10_7**	285.69	6227.06	4.59%	179.65	5.71	3.18%
Totals for TF_MS_A10		1644.49	28418.03	5.79%	771.29	36.96	4.79%

Consumer model based on site data

Sheet1

**********	**********	**********	**********	**********	**********	**********	**********
From Node	Node Name	Total Load Loss Energy (kWh7da	Total Domestic Energy (kWh7da	% Loss (kWh 7)	Total Loss (kVA)	Total Load (S,PF) (kVA)	
**********	**********	**********	**********			**********	
-	MS_A10	50.57kWh(7days)	0.00kWh(7days)			0	93.7
MS_A10	TF_MS_A10	50.22	4336.61	1.16%	1.37	92.09	
TF_MS_A10	A10_1	4.39	1092.3	0.40%	0.08	21.17	
TF_MS_A10	A10_7	12.61	896.67	1.41%	0.23	32.59	
TF_MS_A10	A10_15	1.47	619.52	0.24%	0.03	12.68	
TF_MS_A10	A10_19	10.38	1483.58	0.70%	0.19	27.99	
Totals		28.85	4092.07	**0.71%**	0.53	94.43	**0.57%**
Percentage Losses							
A10_1	A10_2	1.64kWh(7days)	342.36kWh(7days)			7.49	
A10_2	A10_3	0.48kWh(7days)	195.64kWh(7days)			4.64	
A10_1	A10_4	1.88kWh(7days)	554.30kWh(7days)			11.45	
A10_4	A10_5	0.90kWh(7days)	358.67kWh(7days)			7.79	
A10_5	A10_6	0.17kWh(7days)	163.03kWh(7days)			3.99	
A10_7	A10_8	0.65kWh(7days)	374.97kWh(7days)			8.1	
A10_8	A10_9	0.13kWh(7days)	179.33kWh(7days)			4.32	
A10_9	A10_10	0.02kWh(7days)	65.21kWh(7days)			1.91	
A10_7	A10_11	7.36kWh(7days)	326.06kWh(7days)			22.14	
A10_11	A10_12	0.20kWh(7days)	114.12kWh(7days)			2.98	
A10_11	A10_13	4.01kWh(7days)	65.21kWh(7days)			16.87	
A10_15	A10_16	0.24kWh(7days)	146.73kWh(7days)			3.68	
A10_15	A10_17	1.03kWh(7days)	277.15kWh(7days)			6.25	
A10_17	A10_18	0.09kWh(7days)	81.52kWh(7days)			2.28	
A10_19	A10_20	0.58kWh(7days)	260.85kWh(7days)			5.92	
A10_20	A10_21	0.16kWh(7days)	114.12kWh(7days)			2.98	
A10_19	A10_22	5.09kWh(7days)	1027.09kWh(7days)			19.9	
A10_22	A10_23	1.15kWh(7days)	293.45kWh(7days)			6.52	
A10_23	A10_24	0.11kWh(7days)	97.82kWh(7days)			2.63	
A10_22	A10_25	1.71kWh(7days)	538.00kWh(7days)			11.09	
A10_25	A10_26	0.59kWh(7days)	195.64kWh(7days)			4.62	
A10_26	A10_27	0.07kWh(7days)	81.52kWh(7days)			2.26	
A10_25	A10_28	0.25kWh(7days)	146.73kWh(7days)			3.65	
TF_MS_A10	RM37	0.07kWh(7days)	244.55kWh(7days)			5.64	

Page 1

K.3 Spreadsheets according to Fourie's Method

K.3.1 Standard 40A Consumers

Consumer constants			Load Current Parameter for Loss Calc	
Cb	40	1.7	Alpha=	1.700
Confidence	100%	4.5	Beta=	4.500
Vsupply[V]	230		E(Yi)=	2.74E-01
Period[hrs] (year)	168	1.7	E(Yi2)=	1.03E-01
Load Current		4.5	E(Yi3)=	4.64E-02
Mean (RMS) [A]	13.2765		E(Yi4)=	2.37E-02
Mean (AVE) [A]	10.97		E(Yi,Yk)=	7.52E-02
Variance (AVE)	44.2251600	50.09%	E(Yi2,Yk)=	2.82E-02
Consumption			E(Yi3,Yk)=	1.27E-02
Total ma	87		E(Yi,Yj,Yk)=	2.06E-02
Total mb	85		E(Yi2,Yj,Yk)=	7.73E-03
Total mc	85		E(Yi,Yj,Yk),Yl)	5.65E-03
Total m	257		E(Yi2,Tk2)=	1.06E-02
Max Cons.[W]	2364400	W		
Alpha	1.700			
Beta	4.500			
E(Yi)	0.27			
E(Yi2)	0.1			
E(Yi,Yk)	0.08	**From Total PDF**		
E(Ptot) (mu)	648301		From alpha,beta	From data
E(Ptot2)	420895112879	**Max Loss(Scaling)**		199561.46
E(Ptot)2	420293858561	**E[PtotLoss]**	16086.9381	16086.94
VAR(Ptot)=(STDdev)^2	601254318	**E[PtotLoss^2]**		277237749.63
STD DEV	24520	**E[PtotLoss]^2**		26985865.98
Alpha	507	**VAR**	142005.78	142005.78
Beta	1342	**Std DEV**		376.84
Consumption[kW]	771.29	**Alpha**		1675.40
Consumption[kWh]	129.58	**Beta**		19108.27
	0.01	**Total Loss**	17975.724	
%Loss (90%)	**4.91%**	**DEV_loss**	994.81%	Loss
E[%loss]	**2.48%**	**DEV_total**	48.85796%	

Section:	1	1	1	1	1	1	2	2	2	2	2	2	2
Subsection	A10/6	A10/5	A10/4	A10/3	A10/2	A10/1	A10/13	A10/12	A10/11	A10/10	A10/9	A10/8	A10/7
Type:	35	35	70	25	25	120	50	50	70	70	70	95	120
Res/Km@t2	0.58	0.58	0.3	0.81	0.81	0.19	0.41	0.41	0.3	0.3	0.3	0.23	0.19
Section length[m]	70	50	60	60	50	30	60	20	50	50	50	70	80
Rphase[ohm]	0.0405	0.0289	0.0178	0.0484	0.0404	0.0056	0.0246	0.0082	0.0149	0.0149	0.0149	0.0158	0.0148
Cumulative ma	4	8	12	4	7	23	4	3	10	1	3	7	21
Cumulative mb	3	7	11	4	7	22	4	2	9	2	4	8	21
Cumulative mc	3	7	11	4	7	22	4	2	9	1	4	8	21
Max Loss[Wh]	*2265.76*	*7537.12*	*11034.14*	*3718.66*	*9490.32*	*13302.24*	*1889.28*	*236.16*	*6248.88*	*166.32*	*997.92*	*4505.54*	*31328.64*
E(Yp)a=	1.10E+00	2.19E+00	3.29E+00	1.10E+00	1.92E+00	6.31E+00	1.10E+00	8.23E-01	2.74E+00	2.74E-01	8.23E-01	1.92E+00	5.76E+00
E(Yp)b=	8.23E-01	1.92E+00	3.02E+00	1.10E+00	1.92E+00	6.03E+00	1.10E+00	5.48E-01	2.47E+00	5.48E-01	1.10E+00	2.19E+00	5.76E+00
E(Yp)c=	8.23E-01	1.92E+00	3.02E+00	1.10E+00	1.92E+00	6.03E+00	1.10E+00	5.48E-01	2.47E+00	2.74E-01	1.10E+00	2.19E+00	5.76E+00
E(Yp2)a=	1.31E+00	5.03E+00	1.12E+01	1.31E+00	3.88E+00	4.04E+01	1.31E+00	7.60E-01	7.79E+00	1.03E-01	7.60E-01	3.88E+00	3.37E+01
E(Yp2)b=	7.60E-01	3.88E+00	9.40E+00	1.31E+00	3.88E+00	3.70E+01	1.31E+00	3.56E-01	6.34E+00	3.56E-01	1.31E+00	5.03E+00	3.37E+01
E(Yp2)c=	7.60E-01	3.88E+00	9.40E+00	1.31E+00	3.88E+00	3.70E+01	1.31E+00	3.56E-01	6.34E+00	1.03E-01	1.31E+00	5.03E+00	3.37E+01
E(Yp3)a=	1.70E+00	1.20E+01	3.89E+01	1.70E+00	8.21E+00	2.63E+02	1.70E+00	7.70E-01	2.29E+01	4.64E-02	7.70E-01	8.21E+00	2.01E+02
E(Yp3)b=	7.70E-01	8.21E+00	3.02E+01	1.70E+00	8.21E+00	2.31E+02	1.70E+00	2.62E-01	1.69E+01	2.62E-01	1.70E+00	1.20E+01	2.01E+02
E(Yp3)c=	7.70E-01	8.21E+00	3.02E+01	1.70E+00	8.21E+00	2.31E+02	1.70E+00	2.62E-01	1.69E+01	4.64E-02	1.70E+00	1.20E+01	2.01E+02
E(Yp4)a=	2.33E+00	2.99E+01	1.40E+02	2.33E+00	1.81E+01	1.74E+03	2.33E+00	8.45E-01	6.96E+01	2.37E-02	8.45E-01	1.81E+01	1.22E+03
E(Yp4)b=	8.45E-01	1.81E+01	1.00E+02	2.33E+00	1.81E+01	1.46E+03	2.33E+00	2.13E-01	4.66E+01	2.13E-01	2.33E+00	2.99E+01	1.22E+03
E(Yp4)c=	8.45E-01	1.81E+01	1.00E+02	2.33E+00	1.81E+01	1.46E+03	2.33E+00	2.13E-01	4.66E+01	2.37E-02	2.33E+00	2.99E+01	1.22E+03
E(Ploss)=	*206.13*	*622.89*	*883.15*	*330.97*	*788.45*	*1032.97*	*168.15*	*22.83*	*506.58*	*17.76*	*89.47*	*370.92*	*2437.81*
E(Ploss2)=	49227.24	414718.29	814307.59	123008.84	665583.41	1090647.58	31750.97	643.30	270407.06	450.97	9137.86	146615.66	6082445.11
E(Ploss)2=	42489.70	387990.59	779961.12	109539.43	621645.72	1067033.15	28274.25	521.30	256621.25	315.35	8005.51	137579.13	5942898.46
VAR(Ploss)=	6737.54	26727.70	34346.47	13469.42	43937.69	23614.42	3476.72	122.00	13785.81	135.61	1132.35	9036.52	139546.65
STD DEV =	*82.08*	*163.49*	*185.33*	*116.06*	*209.61*	*153.67*	*58.96*	*11.05*	*117.41*	*11.65*	*33.65*	*95.06*	*373.56*
Alpha(section)	5.64	13.23	20.81	7.32	12.89	41.60	7.32	3.76	17.02	1.97	6.35	13.89	39.20
Beta(section)	56	147	239	75	142	494	75	35	193	16	64	155	465
Loss[W]	794.853	1692.340	2039.216	1139.934	2163.358	1932.929	579.149	104.851	1254.316	105.884	327.920	989.510	4636.072
Total Power Loss[kW]	37.880												
Total Energy Loss[kWh]	6						Incl. Church of 15kVA						

3	3	3	3	4	4	4	4	4	4	4	4	4	4
A10/18	A10/17	A10/16	A10/15	A10/28	A10/27	A10/26	A10/25	A10/24	A10/23	A10/22	A10/21	A10/20	A10/19
25	25	25	120	35	25	25	70	25	25	95	25	35	95
0.81	0.81	0.81	0.19	0.58	0.81	0.81	0.3	0.81	0.81	0.23	0.81	0.58	0.23
50	60	50	10	90	40	60	50	50	60	45	50	50	50
0.0404	0.0484	0.0404	0.0019	0.0520	0.0323	0.0484	0.0149	0.0404	0.0484	0.0102	0.0404	0.0289	0.0113
2	6	3	13	3	1	4	11	2	6	21	2	5	30
2	6	3	13	3	2	3	10	2	6	20	2	5	29
1	5	3	12	3	2	3	10	2	6	20	3	6	30
645.6	*7592.26*	*1743.12*	*1429.68*	*2247.26*	*516.48*	*2711.52*	*7650.72*	*774.72*	*8366.98*	*2029.82*	*1162.08*	*4022.88*	*47767.36*
5.48E-01	1.65E+00	8.23E-01	3.56E+00	8.23E-01	2.74E-01	1.10E+00	3.02E+00	5.48E-01	1.65E+00	5.76E+00	5.48E-01	1.37E+00	8.23E+00
5.48E-01	1.65E+00	8.23E-01	3.56E+00	8.23E-01	5.48E-01	8.23E-01	2.74E+00	5.48E-01	1.65E+00	5.48E+00	5.48E-01	1.37E+00	7.95E+00
2.74E-01	1.37E+00	8.23E-01	3.29E+00	8.23E-01	5.48E-01	8.23E-01	2.74E+00	5.48E-01	1.65E+00	5.48E+00	8.23E-01	1.65E+00	8.23E+00
3.56E-01	2.87E+00	7.60E-01	1.31E+01	7.60E-01	1.03E-01	1.31E+00	9.40E+00	3.56E-01	2.87E+00	3.37E+01	3.56E-01	2.02E+00	6.85E+01
3.56E-01	2.87E+00	7.60E-01	1.31E+01	7.60E-01	3.56E-01	7.60E-01	7.79E+00	3.56E-01	2.87E+00	3.06E+01	3.56E-01	2.02E+00	6.40E+01
1.03E-01	2.02E+00	7.60E-01	1.12E+01	7.60E-01	3.56E-01	7.60E-01	7.79E+00	3.56E-01	2.87E+00	3.06E+01	7.60E-01	2.87E+00	6.85E+01
2.62E-01	5.29E+00	7.70E-01	4.92E+01	7.70E-01	4.64E-02	1.70E+00	3.02E+01	2.62E-01	5.29E+00	2.01E+02	2.62E-01	3.16E+00	5.77E+02
2.62E-01	5.29E+00	7.70E-01	4.92E+01	7.70E-01	2.62E-01	7.70E-01	2.29E+01	2.62E-01	5.29E+00	1.74E+02	2.62E-01	3.16E+00	5.22E+02
4.64E-02	3.16E+00	7.70E-01	3.89E+01	7.70E-01	2.62E-01	7.70E-01	2.29E+01	2.62E-01	5.29E+00	1.74E+02	7.70E-01	5.29E+00	5.77E+02
2.13E-01	1.02E+01	8.45E-01	1.90E+02	8.45E-01	2.37E-02	2.33E+00	1.00E+02	2.13E-01	1.02E+01	1.22E+03	2.13E-01	5.23E+00	4.92E+03
2.13E-01	1.02E+01	8.45E-01	1.90E+02	8.45E-01	2.13E-01	8.45E-01	6.96E+01	2.13E-01	1.02E+01	1.01E+03	2.13E-01	5.23E+00	4.31E+03
2.37E-02	5.23E+00	8.45E-01	1.40E+02	8.45E-01	2.13E-01	8.45E-01	6.96E+01	2.13E-01	1.02E+01	1.01E+03	8.45E-01	1.02E+01	4.92E+03
66.38	*643.60*	*163.17*	*113.70*	*210.36*	*53.11*	*246.68*	*615.91*	*79.66*	*706.13*	*1574.28*	*112.35*	*343.35*	*3680.18*
5801.67	451418.28	30949.02	13436.58	51439.94	3713.07	70502.30	397705.65	7849.92	539715.23	2538647.36	15576.52	129208.71	13768841.52
4406.58	414226.62	26624.83	12928.50	44252.76	2820.21	60852.93	379346.26	6345.47	498622.11	2478349.19	12622.46	117886.26	13543706.84
1395.09	37191.66	4324.19	508.08	7187.18	892.86	9649.37	18359.39	1504.45	41093.12	60298.17	2954.05	11322.45	225134.68
37.35	*192.85*	*65.76*	*22.54*	*84.78*	*29.88*	*98.23*	*135.50*	*38.79*	*202.71*	*245.56*	*54.35*	*106.41*	*474.48*
2.73	10.11	5.49	23.34	5.49	2.73	5.64	18.92	3.68	11.03	37.82	3.76	9.44	55.45
24	109	53	270	53	24	56	216	32	120	448	35	101	664
347.565	1943.045	634.310	252.715	817.764	278.052	951.230	1469.333	364.484	2058.495	3023.749	515.943	1065.842	6397.412

K.3.2 Consumer model from Data

Consumer constants			Load Current Parameter for Loss Ca	
Cb	20	2.16	Alpha=	2.165
Confidence	90%	36.79	Beta=	36.788
Vsupply[V]	230		E(Yi)=	5.56E-02
Period[hrs] (year)	720	1.7	E(Yi2)=	4.40E-03
Load Current		4.5	E(Yi3)=	4.48E-04
Mean (RMS) [A]	1.1114		E(Yi4)=	5.51E-05
Mean (AVE) [A]	0.98		E(**Yi**,Yk)=	3.09E-03
Variance (AVE)	.5254400	65.22%	E(Yi2,Yk)=	2.45E-04
Consumption			E(Yi3,Yk)=	2.49E-05
Total ma	87		E(Yi,Yj,Yk)=	1.72E-04
Total mb	85		E(Yi2,Yj,Yk)=	1.36E-05
Total mc	85		E(Yi,Yj,Yk),Yl)	9.54E-06
Total m	257		E(Yi2,Tk2)=	1.94E-05
Max Cons.[W]	1182200	W		
Alpha	1.687			
Beta	32.766			
E(Yi)	0.05			
E(Yi2)	0			
E(Yi,Yk)	0	**From Total PDF**		
E(Ptot) (mu)	57891		From alpha,beta	From data
E(Ptot2)	3358528971	**Max Loss(Scaling)**		49890.36
E(Ptot)2	3351385457	**E[PtotLoss]**	166.9370	166.94
VAR(Ptot)=(STDdev)^2	7143514	**E[PtotLoss^2]**		2976.15
STD DEV	2673	**E[PtotLoss]^2**		2879.52
Alpha	446	**VAR**	18.60	18.60
Beta	8664	**Std DEV**		4.31
Consumption[kW]	61.341	**Alpha**		1493.51
Consumption[kWh]	44165.244	**Beta**		444852.17
	3.68	**Total Loss**	172.487	
%Loss (90%) (kWh)	**0.36%**	**DEV_loss**	1966.35%	Loss
E[%loss] (VA)	**0.29%**	**DEV_total**	7.03028%	

Section:	1	1	1	1	1	1	2	2	2	2	2	2	2
Subsection	A10/6	A10/5	A10/4	A10/3	A10/2	A10/1	A10/13	A10/12	A10/11	A10/10	A10/9	A10/8	A10/7
Type:	35	35	70	25	25	120	50	50	70	70	70	95	120
Res/Km@t2	0.58	0.58	0.3	0.81	0.81	0.19	0.41	0.41	0.3	0.3	0.3	0.23	0.19
Section length[m]	70	50	60	60	50	30	60	20	50	50	50	70	80
Rphase[ohm]	0.0405	0.0289	0.0178	0.0484	0.0404	0.0056	0.0246	0.0082	0.0149	0.0149	0.0149	0.0158	0.0148
Cumulative ma	4	8	12	4	7	23	4	3	10	1	3	7	21
Cumulative mb	3	7	11	4	7	22	4	2	9	2	4	8	21
Cumulative mc	3	7	11	4	7	22	4	2	9	1	4	8	21
Max Loss[Wh]	*566.44*	*1884.28*	*2758.54*	*929.66*	*2372.58*	*3325.56*	*472.32*	*59.04*	*1562.22*	*41.58*	*249.48*	*1126.38*	*7832.16*
E(Yp)a=	2.22E-01	4.45E-01	6.67E-01	2.22E-01	3.89E-01	1.28E+00	2.22E-01	1.67E-01	5.56E-01	5.56E-02	1.67E-01	3.89E-01	1.17E+00
E(Yp)b=	1.67E-01	3.89E-01	6.11E-01	2.22E-01	3.89E-01	1.22E+00	2.22E-01	1.11E-01	5.00E-01	1.11E-01	2.22E-01	4.45E-01	1.17E+00
E(Yp)c=	1.67E-01	3.89E-01	6.11E-01	2.22E-01	3.89E-01	1.22E+00	2.22E-01	1.11E-01	5.00E-01	5.56E-02	2.22E-01	4.45E-01	1.17E+00
E(Yp2)a=	5.47E-02	2.08E-01	4.60E-01	5.47E-02	1.61E-01	1.66E+00	5.47E-02	3.17E-02	3.22E-01	4.40E-03	3.17E-02	1.61E-01	1.39E+00
E(Yp2)b=	3.17E-02	1.61E-01	3.88E-01	5.47E-02	1.61E-01	1.52E+00	5.47E-02	1.50E-02	2.62E-01	1.50E-02	5.47E-02	2.08E-01	1.39E+00
E(Yp2)c=	3.17E-02	1.61E-01	3.88E-01	5.47E-02	1.61E-01	1.52E+00	5.47E-02	1.50E-02	2.62E-01	4.40E-03	5.47E-02	2.08E-01	1.39E+00
E(Yp3)a=	1.47E-02	1.02E-01	3.29E-01	1.47E-02	7.00E-02	2.21E+00	1.47E-02	6.78E-03	1.94E-01	4.48E-04	6.78E-03	7.00E-02	1.69E+00
E(Yp3)b=	6.78E-03	7.00E-02	2.56E-01	1.47E-02	7.00E-02	1.93E+00	1.47E-02	2.36E-03	1.43E-01	2.36E-03	1.47E-02	1.02E-01	1.69E+00
E(Yp3)c=	6.78E-03	7.00E-02	2.56E-01	1.47E-02	7.00E-02	1.93E+00	1.47E-02	2.36E-03	1.43E-01	4.48E-04	1.47E-02	1.02E-01	1.69E+00
E(Yp4)a=	4.30E-03	5.27E-02	2.42E-01	4.30E-03	3.21E-02	2.97E+00	4.30E-03	1.60E-03	1.22E-01	5.51E-05	1.60E-03	3.21E-02	2.09E+00
E(Yp4)b=	1.60E-03	3.21E-02	1.74E-01	4.30E-03	3.21E-02	2.50E+00	4.30E-03	4.25E-04	8.18E-02	4.25E-04	4.30E-03	5.27E-02	2.09E+00
E(Yp4)c=	1.60E-03	3.21E-02	1.74E-01	4.30E-03	3.21E-02	2.50E+00	4.30E-03	4.25E-04	8.18E-02	5.51E-05	4.30E-03	5.27E-02	2.09E+00
E(Ploss)=	*2.17*	*6.49*	*9.16*	*3.48*	*8.22*	*10.66*	*1.77*	*0.24*	*5.26*	*0.19*	*0.94*	*3.86*	*25.17*
E(Ploss2)=	5.72	45.69	88.31	14.09	73.51	116.63	3.64	0.08	29.51	0.06	1.05	16.12	651.04
E(Ploss)2=	4.73	42.08	83.82	12.12	67.52	113.64	3.13	0.06	27.68	0.04	0.89	14.90	633.31
VAR(Ploss)=	0.99	3.62	4.50	1.97	5.99	2.99	0.51	0.02	1.83	0.02	0.16	1.22	17.72
STD DEV =	*0.99*	*1.90*	*2.12*	*1.40*	*2.45*	*1.73*	*0.71*	*0.14*	*1.35*	*0.15*	*0.41*	*1.10*	*4.21*
Alpha(section)	4.75	11.60	18.57	6.14	11.23	37.86	6.14	3.13	15.08	1.64	5.36	12.19	35.61
Beta(section)	1234	3357	5577	1633	3232	11772	1633	759	4462	357	1414	3545	11048
Loss[W]	3.507	9.012	11.956	5.356	11.468	12.928	2.721	0.426	7.052	0.388	1.485	5.325	30.689
Total Power Loss[kW]	0.219												
Total Energy Loss[kWh]	157.904							Incl. Church of 15kVA					

3	3	3	3	4	4	4	4	4	4	4	4	4	4
A10/18	A10/17	A10/16	A10/15	A10/28	A10/27	A10/26	A10/25	A10/24	A10/23	A10/22	A10/21	A10/20	A10/19
25	25	25	120	35	25	25	70	25	25	95	25	35	95
0.81	0.81	0.81	0.19	0.58	0.81	0.81	0.3	0.81	0.81	0.23	0.81	0.58	0.23
50	60	50	10	90	40	60	50	50	60	45	50	50	50
0.0404	0.0484	0.0404	0.0019	0.0520	0.0323	0.0484	0.0149	0.0404	0.0484	0.0102	0.0404	0.0289	0.0113
2	6	3	13	3	1	4	11	2	6	21	2	5	30
2	6	3	13	3	2	3	10	2	6	20	2	5	29
1	5	3	12	3	2	3	10	2	6	20	3	6	30
161.4	*1898.06*	*435.78*	*357.42*	*561.82*	*129.12*	*677.88*	*1912.68*	*193.68*	*2091.74*	*5052.46*	*290.52*	*1005.72*	*11941.84*
1.11E-01	3.33E-01	1.67E-01	7.22E-01	1.67E-01	5.56E-02	2.22E-01	6.11E-01	1.11E-01	3.33E-01	1.17E+00	1.11E-01	2.78E-01	1.67E+00
1.11E-01	3.33E-01	1.67E-01	7.22E-01	1.67E-01	1.11E-01	1.67E-01	5.56E-01	1.11E-01	3.33E-01	1.11E+00	1.11E-01	2.78E-01	1.61E+00
5.56E-02	2.78E-01	1.67E-01	6.67E-01	1.67E-01	1.11E-01	1.67E-01	5.56E-01	1.11E-01	3.33E-01	1.11E+00	1.67E-01	3.33E-01	1.67E+00
1.50E-02	1.19E-01	3.17E-02	5.39E-01	3.17E-02	4.40E-03	5.47E-02	3.88E-01	1.50E-02	1.19E-01	1.39E+00	1.50E-02	8.38E-02	2.82E+00
1.50E-02	1.19E-01	3.17E-02	5.39E-01	3.17E-02	1.50E-02	3.17E-02	3.22E-01	1.50E-02	1.19E-01	1.26E+00	1.50E-02	8.38E-02	2.64E+00
4.40E-03	8.38E-02	3.17E-02	4.60E-01	3.17E-02	1.50E-02	3.17E-02	3.22E-01	1.50E-02	1.19E-01	1.26E+00	3.17E-02	1.19E-01	2.82E+00
2.36E-03	4.53E-02	6.78E-03	4.15E-01	6.78E-03	4.48E-04	1.47E-02	2.56E-01	2.36E-03	4.53E-02	1.69E+00	2.36E-03	2.72E-02	4.83E+00
2.36E-03	4.53E-02	6.78E-03	4.15E-01	6.78E-03	2.36E-03	6.78E-03	1.94E-01	2.36E-03	4.53E-02	1.46E+00	2.36E-03	2.72E-02	4.37E+00
4.48E-04	2.72E-02	6.78E-03	3.29E-01	6.78E-03	2.36E-03	6.78E-03	1.94E-01	2.36E-03	4.53E-02	1.46E+00	6.78E-03	4.53E-02	4.83E+00
4.25E-04	1.83E-02	1.60E-03	3.29E-01	1.60E-03	5.51E-05	4.30E-03	1.74E-01	4.25E-04	1.83E-02	2.09E+00	4.25E-04	9.47E-03	8.40E+00
4.25E-04	1.83E-02	1.60E-03	3.29E-01	1.60E-03	4.25E-04	1.60E-03	1.22E-01	4.25E-04	1.83E-02	1.73E+00	4.25E-04	9.47E-03	7.35E+00
5.51E-05	9.47E-03	1.60E-03	2.42E-01	1.60E-03	4.25E-04	1.60E-03	1.22E-01	4.25E-04	1.83E-02	1.73E+00	1.60E-03	1.83E-02	8.40E+00
0.71	*6.73*	*1.73*	*1.18*	*2.23*	*0.57*	*2.60*	*6.39*	*0.85*	*7.38*	*16.25*	*1.19*	*3.59*	*37.93*
0.73	50.40	3.65	1.45	6.06	0.47	8.19	43.25	0.98	60.09	271.86	1.88	14.48	1467.23
0.50	45.24	2.98	1.39	4.96	0.32	6.77	40.83	0.73	54.39	264.19	1.43	12.90	1438.96
0.23	5.16	0.66	0.07	1.10	0.14	1.42	2.42	0.25	5.69	7.67	0.45	1.58	28.27
0.47	*2.27*	*0.81*	*0.26*	*1.05*	*0.38*	*1.19*	*1.56*	*0.50*	*2.39*	*2.77*	*0.67*	*1.26*	*5.32*
2.23	8.74	4.49	20.91	4.49	2.23	4.75	16.82	2.89	9.52	34.35	3.13	8.14	50.74
504	2458	1128	6326	1128	504	1234	5018	653	2690	10642	759	2271	15921
1.346	9.751	2.817	1.517	3.632	1.077	4.197	8.447	1.524	10.550	19.888	2.097	5.266	44.887

Appendix L

Check Meter Schematics and Components

To simplify design and in order to keep the design as modular as possible, especially with the idea to re-use this design for other applications it was decided to design two separate printed circuit boards, e.g. analog and digital.
The schematics and printer circuit boards are presented in the following pages.

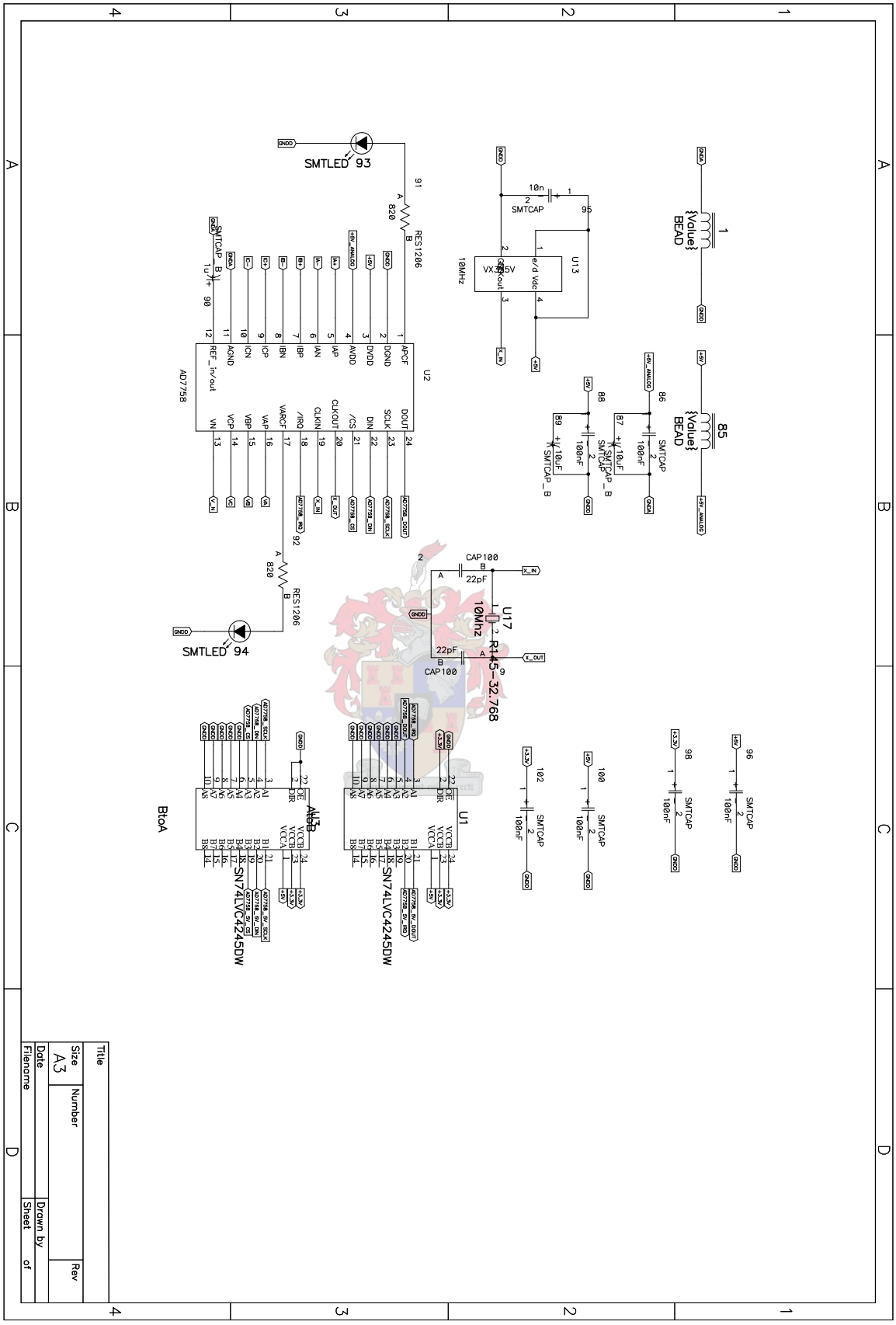


Figure L.1: *Schematic of Analog Circuitry: Energy Measurement*

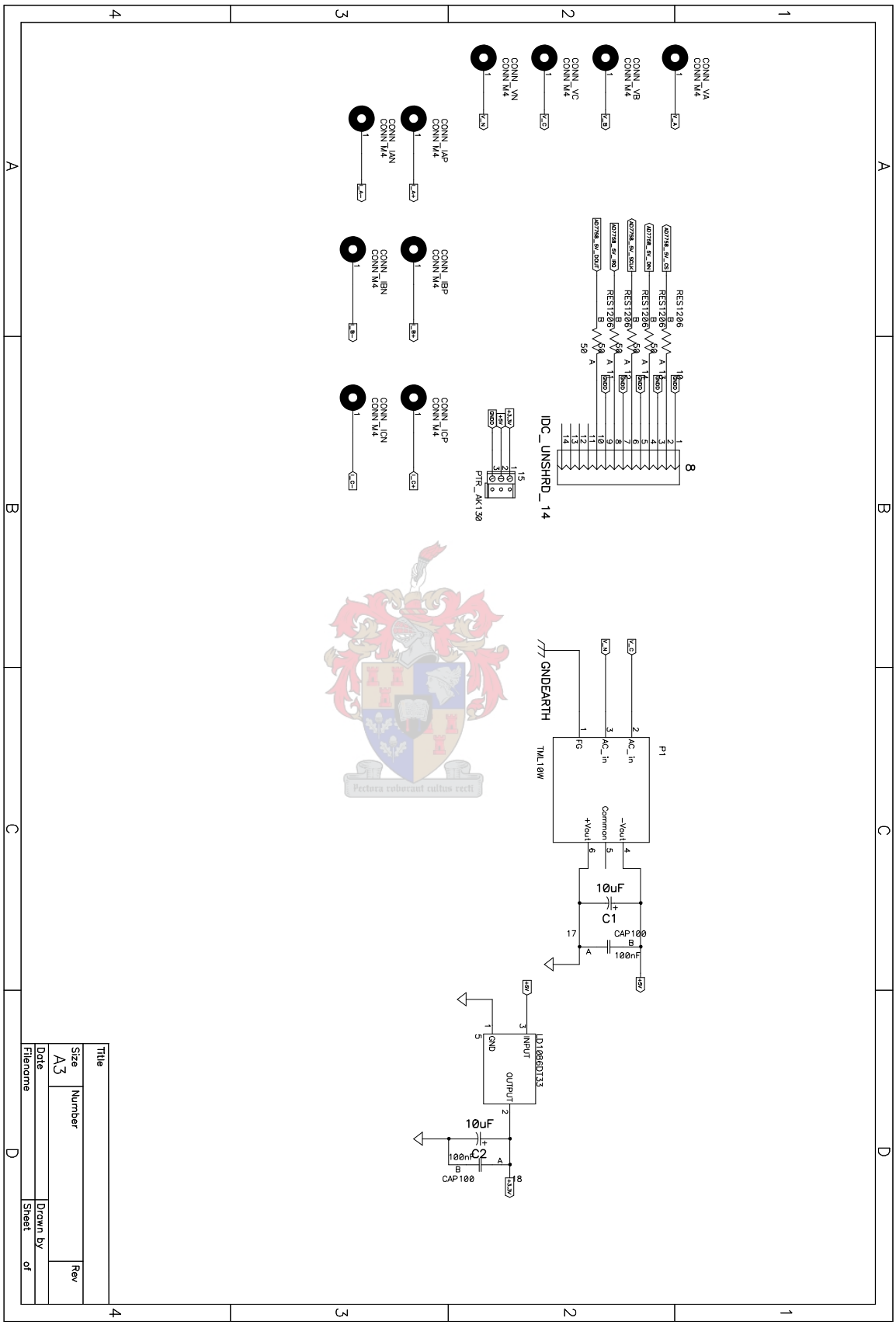


Figure L.2: *Schematic of Analog Circuitry: Power and Connectors*

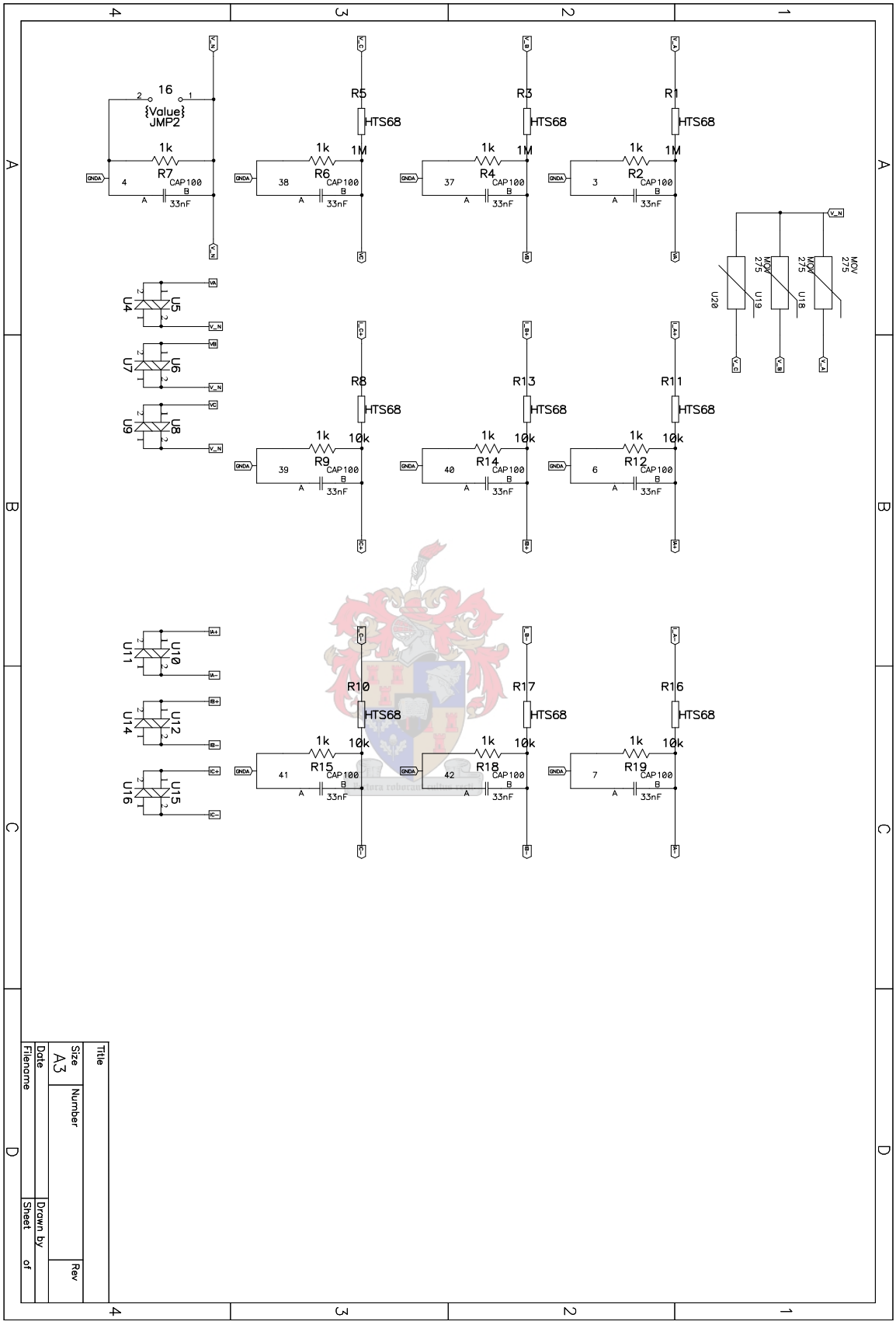


Figure L.3: *Schematic of Analog Circuitry: Signal Conditioning*

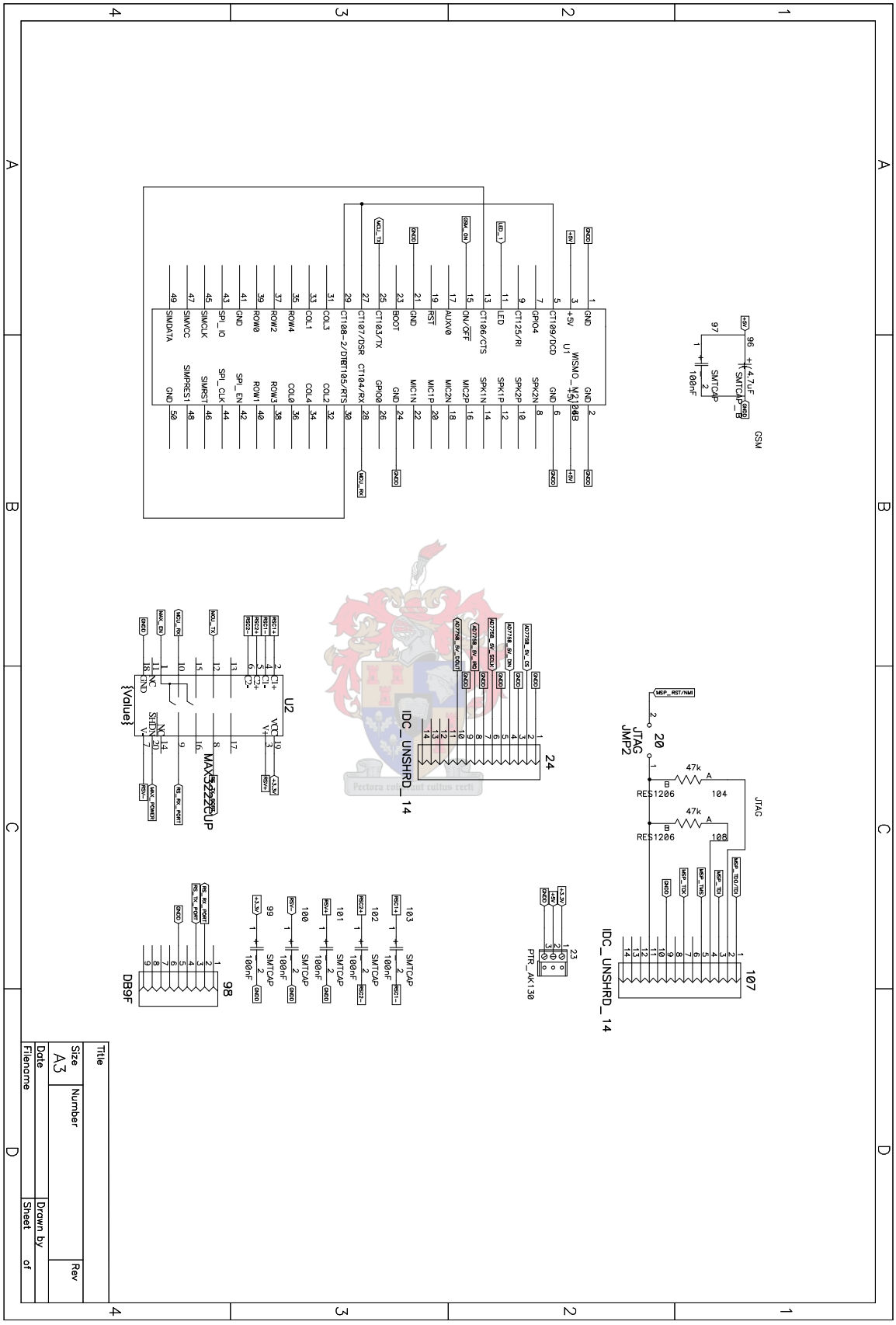


Figure L.4: *Schematic of Digital Circuitry: Communication Devices*

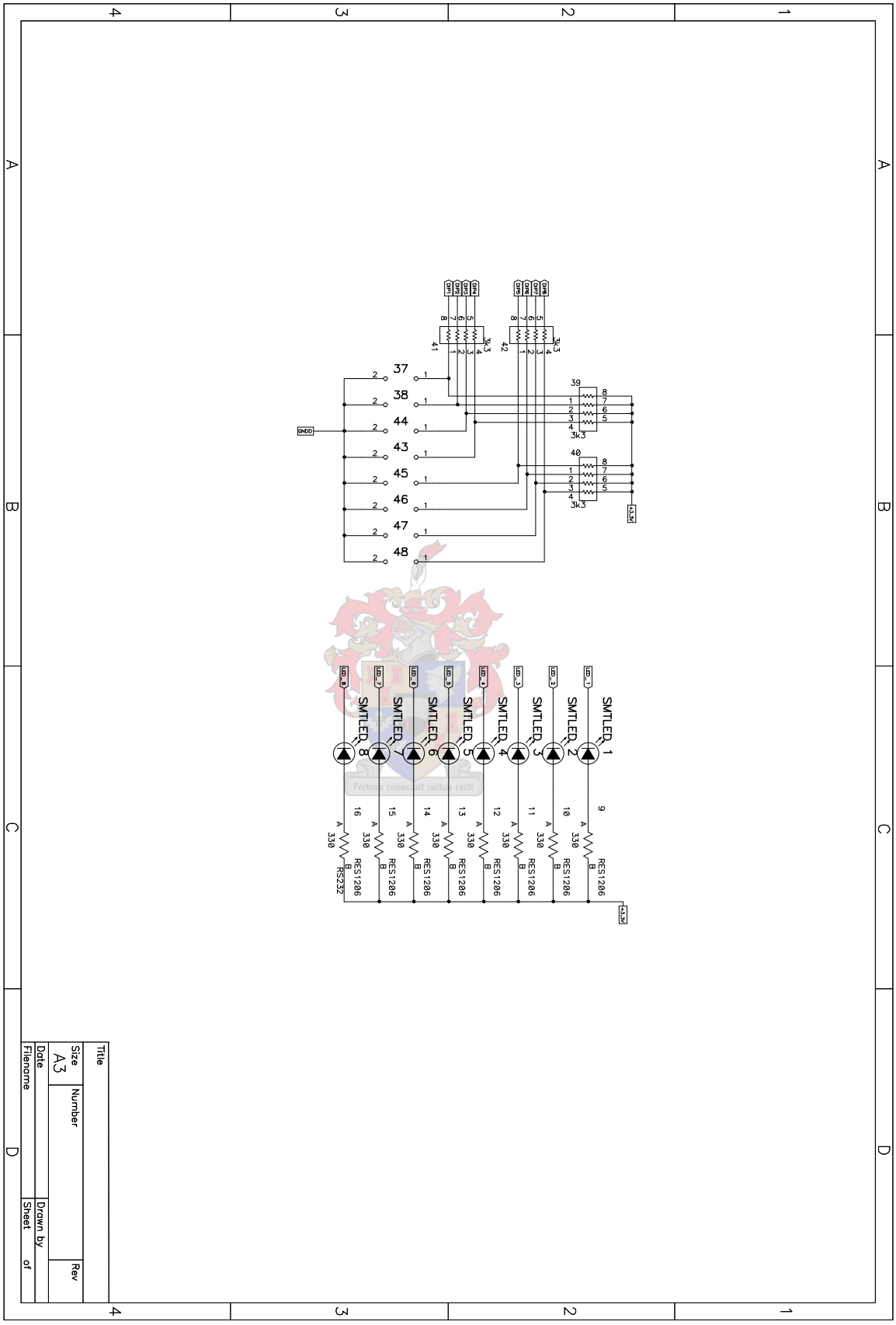


Figure L.5: *Schematic of Digital Circuitry: LED's and DIP Switches*

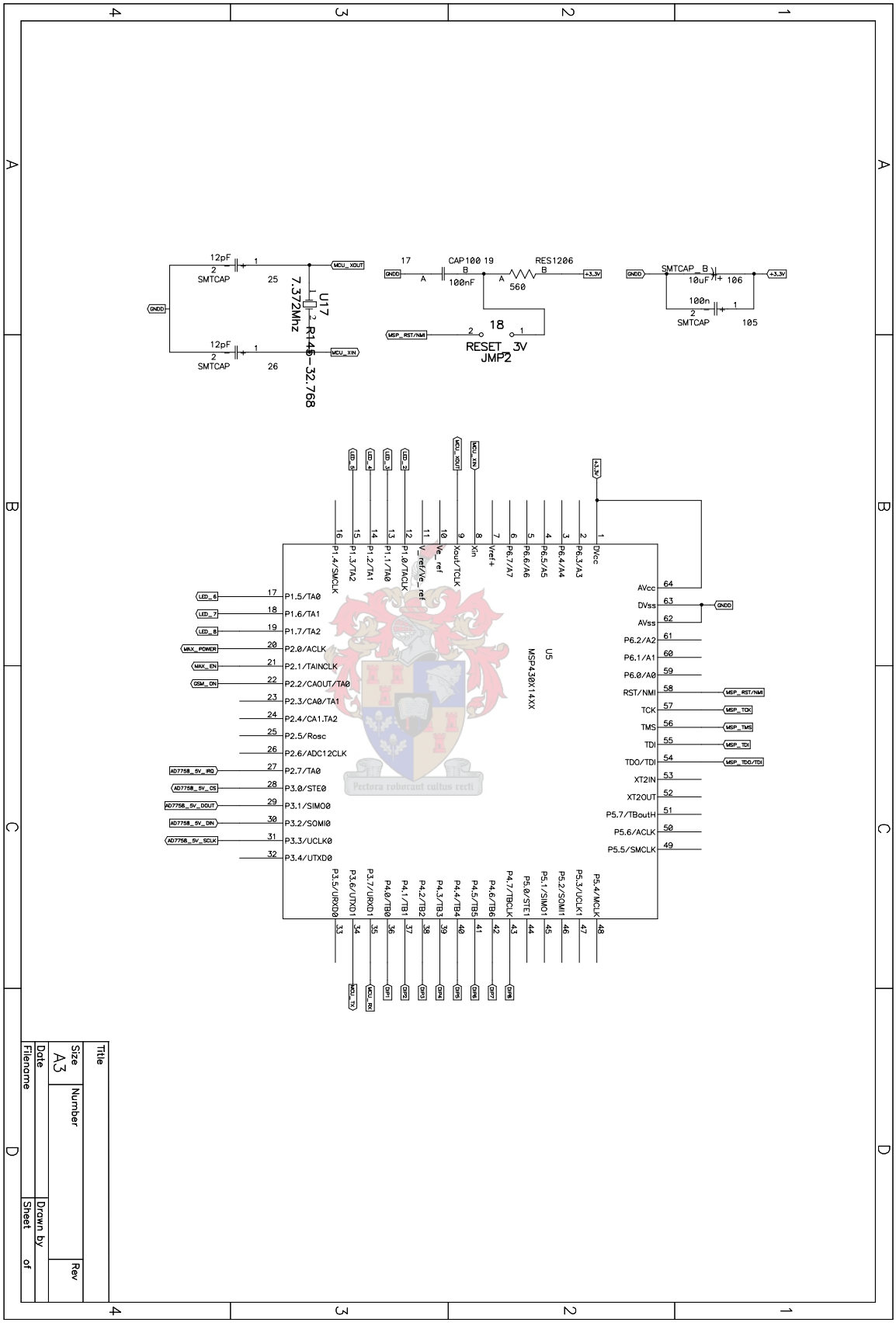


Figure L.6: *Schematic of Digital Circuitry: Micro Controller Unit*

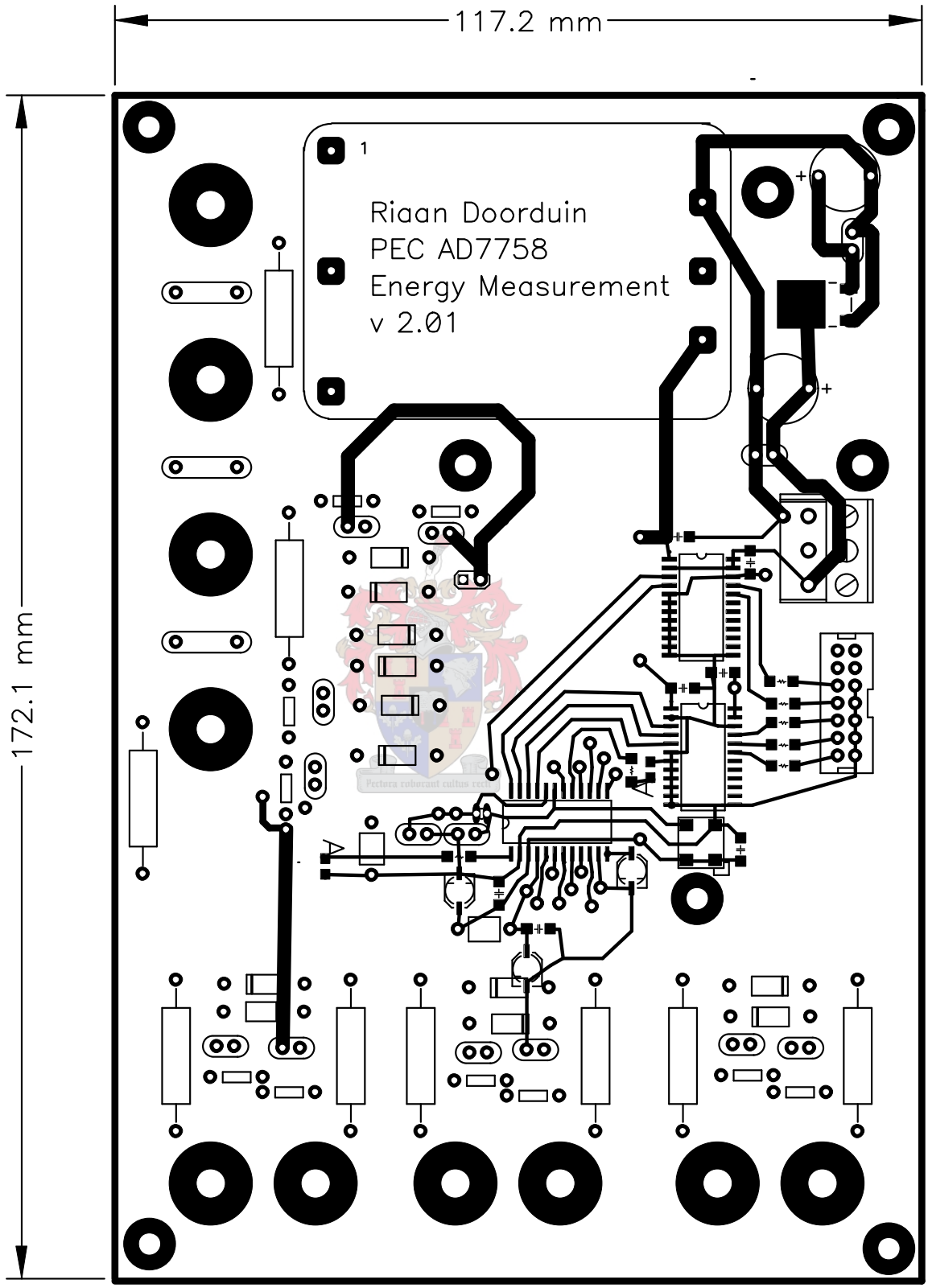


Figure L.7: *Top PCB printout of Analog Circuitry*

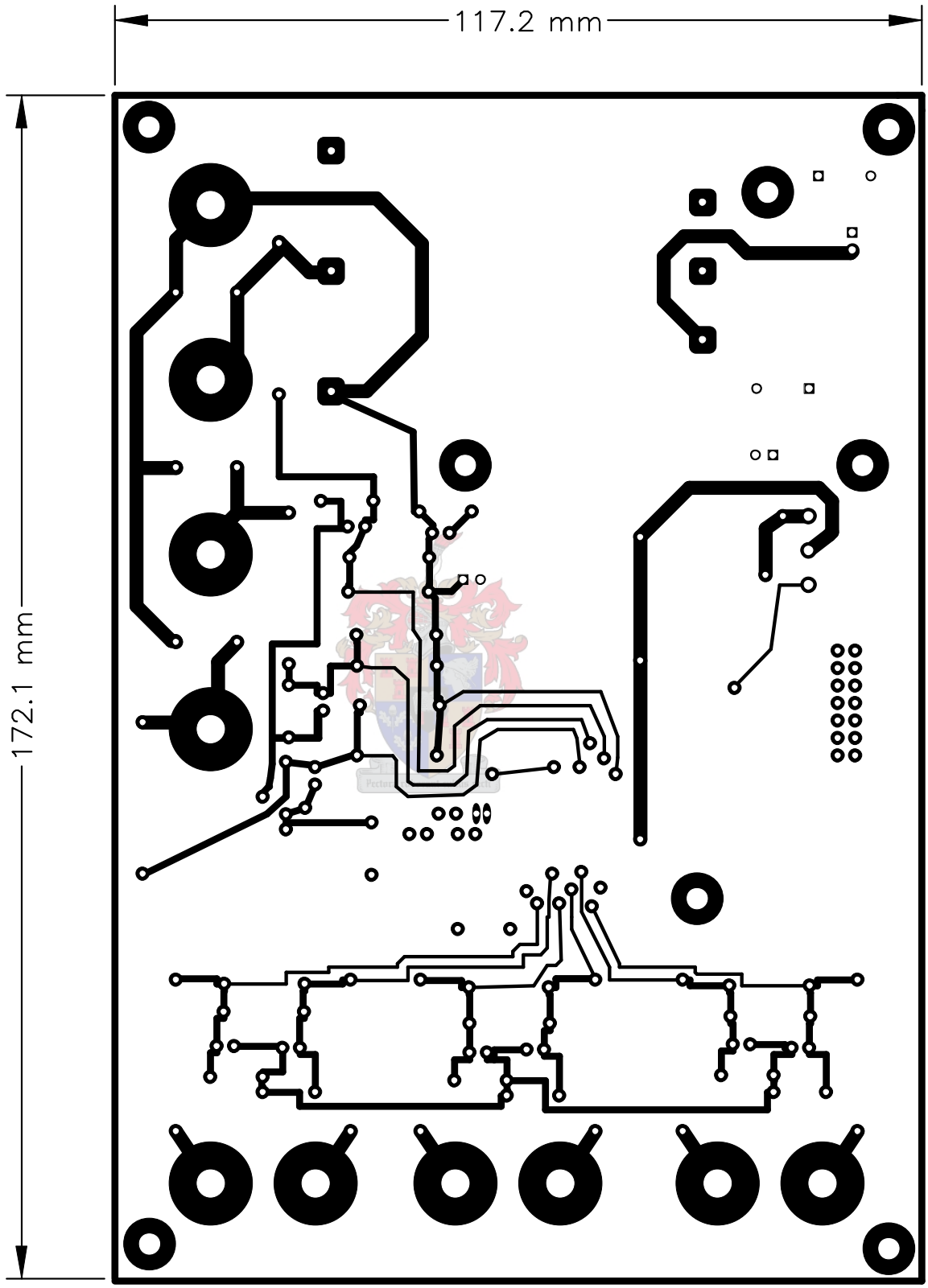


Figure L.8: *Bottom PCB printout of Analog Circuitry*

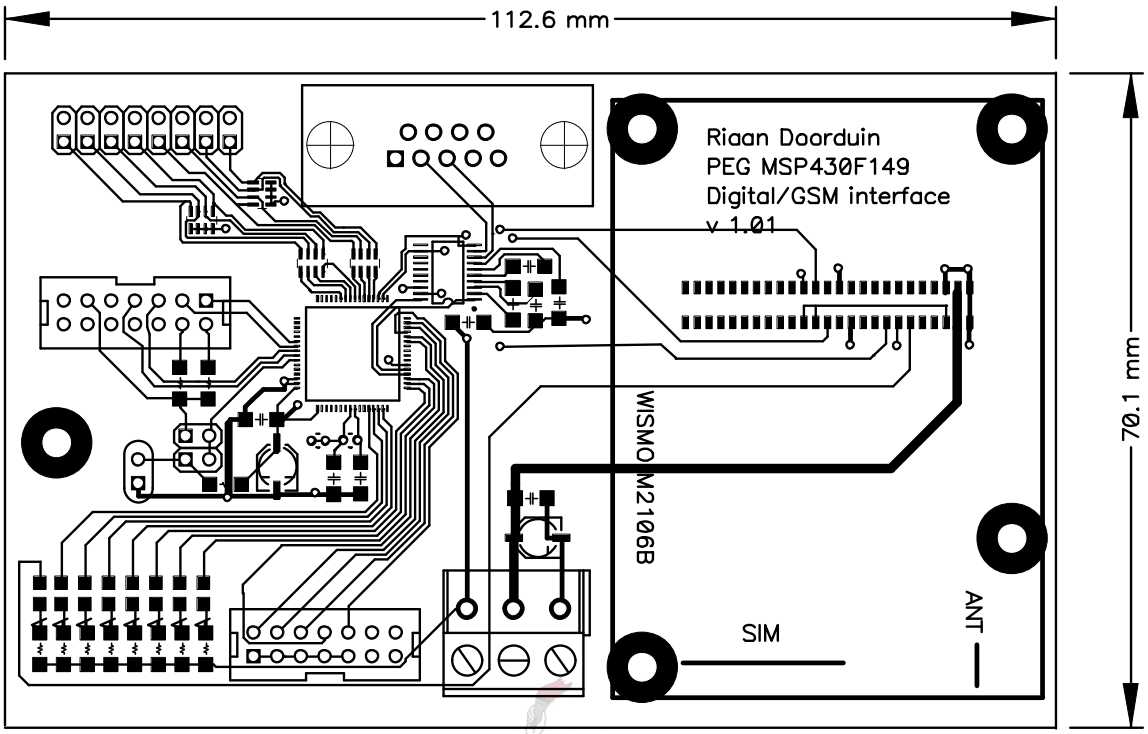


Figure L.9: *Top PCB printout of Digital Circuitry*

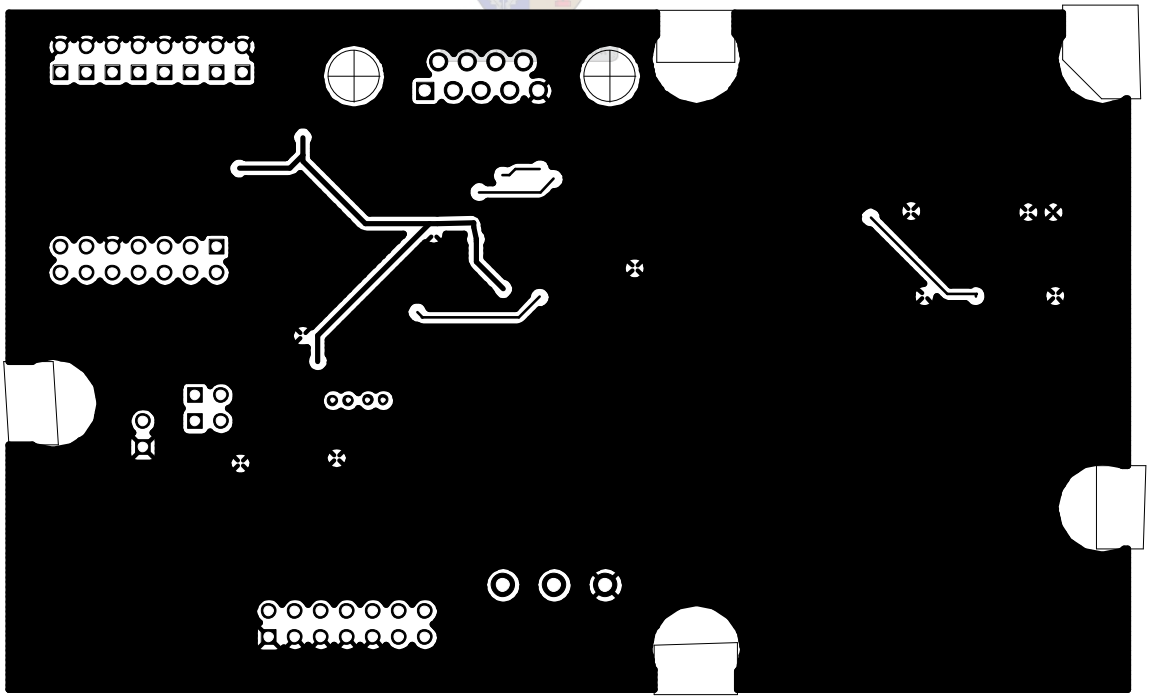


Figure L.10: *Bottom PCB printout of Digital Circuitry*

					Total System Cost:		R 9,311.33	
PCB	Ref	Description	Order Code	Manufacturer	Supplier	Quantity	Cost	Total Cost
		Box Std Slide Lid J2-S	V1ABB/YORK/J2-S	ABB	Voltex	1	R 62.59	R 62.59
		3 Phase Leads	212-988	RS	RS	1	R 512.90	R 512.90
		Spare Fuses	388-4132		RS	1	R 35.37	R 35.37
		Current Probes 1000A	SCL1000	Dent Instruments	SA elite Pro	3	R 1,608.54	R 4,825.62
		Neutral lead	206-753	RS	RS	1	R 87.73	R 87.73
	PCB1,2	Printed Circuit Boards		Northtech	Northtech	1	R 701.82	R 701.82
								R 0.00
EM	U1,3	3V to 5V buffer: SN74LVC4245DW	380-0500	Texas Instruments	RS	2	R 16.65	R 33.30
	U2	Poly Phase Energy Measurement	ADE7758ARW	Analog Devices	Avnet	1	R 65.50	R 65.50
	U4-12,14-16	Clipping Diodes	1N4007		Communica	12	R 0.09	R 1.08
	U17	10MHz Crystal	CRYSTAL			1	R 2.05	R 2.05
	U18-20	MOV 275V	238-621	BC	RS	3	R 15.68	R 47.04
	SMTLED	LEDs	SMTLED1206		Communica	2	R 0.67	R 1.34
	BEAD	EMI Leaded Filter	239-598		RS	2	R 1.49	R 2.98
								R 0.00
	CONN	Female Banana Connectors	404-200,216,250,244	RS	RS	10	R 36.51	R 365.10
	IDC14	Unshreaded IDC connector 14pin	482-121		RS	2	R 21.82	R 43.64
								R 0.00
	TML10W	Traco 10W 5V power supply	418-1917	RS	RS	1	R 498.02	R 498.02
	PTR_AK130	Power Connector Strip (3)	PTR_AK130		Communica	1	R 3.18	R 3.18
	PTR_STL130/24	Power Connector (3)	PTR_STL130/24		Communica	1	R 8.15	R 8.15
	LD1086DT33	3.3V voltage regulator	355-4768	ST	RS	1	R 22.22	R 22.22
								R 0.00
	86,88,96,98,100,102	SMT Cap 100nF	SMTCAP 100nF		Communica	6	R 1.24	R 7.44
	87,89	SMT Cap 10 uF	SMTCAP_B 10uF		Communica	2	R 3.07	R 6.14
	90	SMT Cap 1 uF	SMTCAP_B 1uF		Communica	1	R 3.50	R 3.50
	2,9	Cap 22pf	CAP 22 pF		Communica	2	R 0.40	R 0.80
	C2	Cap 10uF	CAP 10 uF		Communica	1	R 0.40	R 0.40
	18	Cap 100nF	CAP 100nF		Communica	1	R 0.40	R 0.40
	3,4,6,7,37-42	Cap 33nF	CAP 33nF		Communica	10	R 0.40	R 4.00
								R 0.00
								R 0.00
	91,92	RES1206, 820 ohm	RES1206 820ohm			2	R 0.30	R 0.60
	10-14	RES1206, 50 ohm	RES1206 50ohm			5	R 0.30	R 1.50
	R2,R4,R6,R7,R9,R12,R14,R15,R18,R19	1k ohm High Precision Resistor	165-769	Vishay	RS	10	R 13.38	R 133.80
	R1,R3,R5	1M ohm High Voltage Resistor	296-0544	Vishay	RS	3	R 76.34	R 229.02
	R8,R13,R11,R10,R17,R16	10k ohm High precision Resistor	166-728	Vishay	RS	6	R 13.63	R 81.78
								R 0.00
MCU	U1	GSM Modem	WISMO_M2106B	Wavecom	Trinity	1	R 1,140.00	R 1,140.00
	U2	MAX322CUP RS232 Driver		Texas Instruments	RS	1	R 44.00	R 44.00
	U3							R 0.00
	U4							R 0.00
	U5	MCU: MSP430x1479F	MSP430C1479F	Texas Instruments	EBV	1	R 75.22	R 75.22
	U17	7.372MHz Crystal	CRYSTAL			1	R 2.88	R 2.88
	39-41	3k3 Quad Resistor Package	241-9349		RS	2	R 0.94	R 1.88
	EM16,18,20,37,38,43-48	SIL Header, 40Way	203SSX1		Campus Elect	1	R 1.08	R 1.08
	SMTLED	LEDs	SMTLED1206		Communica	8	R 0.67	R 5.36
	U1_Conn	Connector for GSM Modem	WISMO_Connectors	Wavecom	Trinity	1	R 37.62	R 37.62
	IDC14	Unshreaded IDC connector 14pin	482-121		RS	2	R 21.82	R 43.64
	PTR_AK130	Power Connector Strip (3)	PTR_AK130		Communica	1	R 3.18	R 3.18
	PTR_STL130/24	Power Connector (3)	PTR_STL130/24		Communica	1	R 8.15	R 8.15
	DB9F	Female PCB Connector	DB9F		Communica	1	R 6.38	R 6.38
	ANT1	Wismo Antenna	WISMO_Antenna	Wavecom	Trinity	1	R 135.66	R 135.66
								R 0.00
	17,97,99-103,105	SMT Cap 100nF	SMTCAP 100nF		Communica	8	R 1.24	R 9.92
	25,26	SMT Cap 12pF	SMTCAP 12pF		Communica	2	R 1.24	R 2.48
	96	SMT Cap 4.7 uF	SMTCAP_B 10uF		Communica	1	R 3.07	R 3.07
	104,108	RES1206, 47k ohm	RES1206 47k ohm			2	R 0.35	R 0.70
	9-16	RES1206, 330 ohm	RES1206 330ohm			8	R 0.10	R 0.80
	9-16	RES1206, 560 ohm	RES1206 560ohm			1	R 0.30	R 0.30

Figure L.11: *Component List*

Appendix M

palmHHT Installation Notes and Users Guide

M.1 Package Contents

The Palm Meter Reader suite comprises the following :

- Setup File: palmHHT Setup.exe
- User Documentation: This file you are reading now.

Please take care when using the different programs bundled together and adhere to the various license agreements. **Please note that the current software and instalation procedure is still under testing.**

M.2 Pre-requisites

The system was only tested on Windows XP, although it should work on any machine able to run the Java 2 SE and Palm Desktop software.

M.2.1 Required Hardware

PC A PC workstation, with a Windows XP.

Palm A Palm with 4MB or more memory and with Palm OS 4 or greater.

M.2.2 Required Software

The following software needs to be installed on the PC prior to installation of the Palm and HotSync Software[1].

Palm Desktop The Palm desktop is supplied on the CD with the purchased palm.

Java The Java Runtime Environment must be installed, J2SE v 1.4 or later[83].

The Palm software must be installed with HotSync and Palm Desktop fully functional. A user must also be created for the purpose of synchronization. If required the network synchronization can be implemented. Please refer to the Palm HotSync documentation[64]. For the total solution to work metMeter must be installed and fully functional.

M.3 Installation

The following steps must now be executed:

1. Start the instalation program palmHHT Setup.exe.
2. Follow the instructions.
3. When the WinZip Program opens, do not “UnZip”, but use the “Run WinZip” option.
4. Click on “extract”.
5. Select “yes to all” for all the popups.
6. After install reboot if asked.
7. With the PC rebooted and ready for user interaction[2]. Select Click on the HotSync icon in the system tray.
8. Select “Custom...”.
9. Select “Palm Power Meter Reader”. (Fig. M.1)
10. Click on “Change”.
11. Ensure that “Default Data Path” is the same as the installed metMeter path. (Fig. M.2)
12. Connect the palm to the PC. Initiated a HotSync operation.

[1] Palm and HotSync is trademarks of the Palm Company

[2] After the user logged in...etc.

13. The palmHHT software should now be installed. Run the application.
14. Read the “License Agreement”. (Fig. M.3)
15. The Databases should now be generated when Fig. M.4 appears.
16. Synchronize a last time to upload the metMeter database.
17. The system is now ready for use.

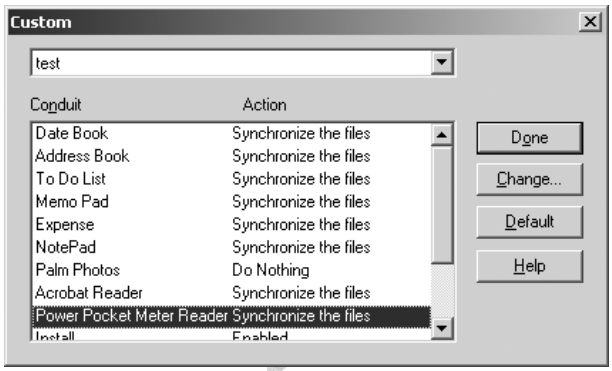


Figure M.1: *HotSync Custom Setup*

Figure M.2: *HotSync Palm Setup*

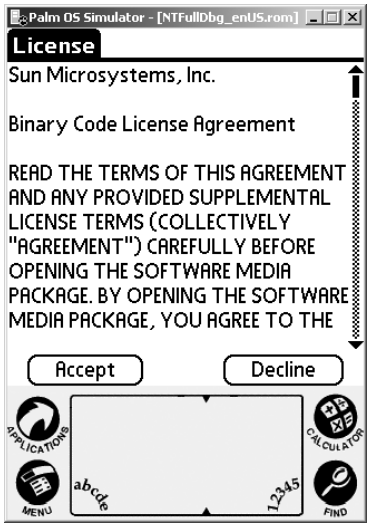


Figure M.3: *Java License Agreement for Palm MIDP*

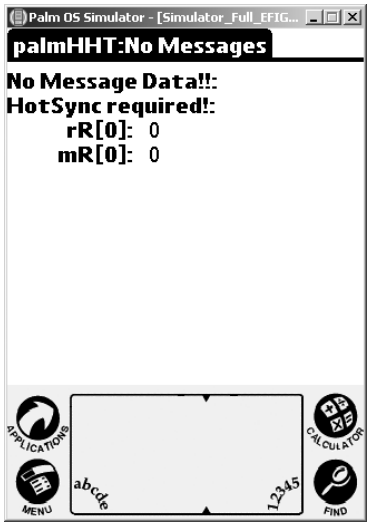


Figure M.4: *No Data uploaded for palmHHT*

Figure M.5: *First Screen of correctly setup palmHHT*

M.4 Training

For additional training or user related inquiries please refer to Appendix N.

Appendix N

Palm HHT training slides

Training was given to the Meter Readers at Theewaterskloof Municipality. The audience was a diverse group. The slides used for the training is presented in the next section.

metMeter – Palm HHT meter Reader

Training/Opleiding

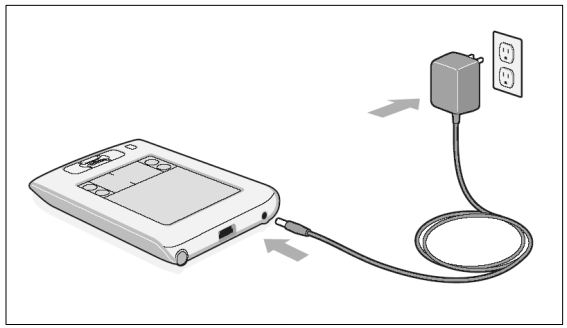
Riaan (W.A.) Doorduin
18/05/2004

Overview / Oorsig

- Palm Zire 21/71
  - Basics
  - Graffiti
- Hotsync
  - metMeter
  - palm
- palmHHT
  - 1, 2, 3...
  - Screen / Skerm
  - Datacapture / data versameling
  - Boodskappe
- ????
- Exercises / Oefeninge

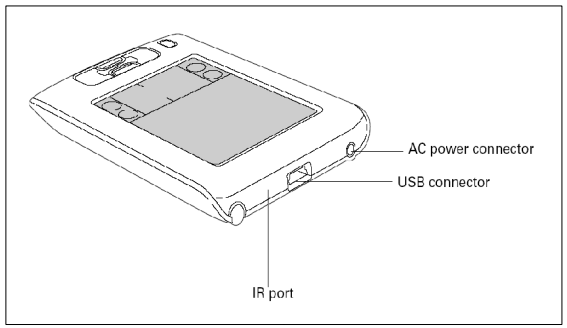
Palm - Basics

- Charge / herlaai (p.1)



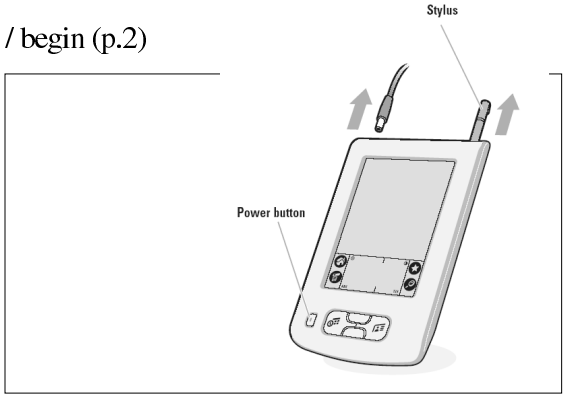
Palm - Basics

- Connectors / kabels



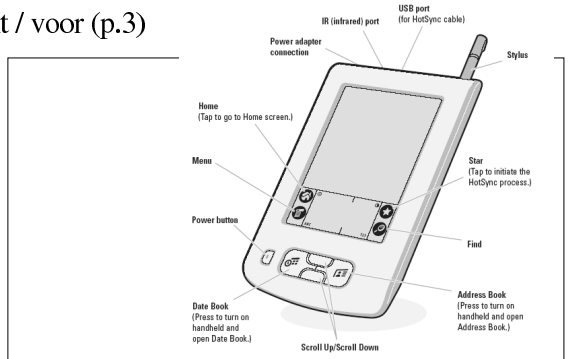
Palm – Basic

- Start / begin (p.2)



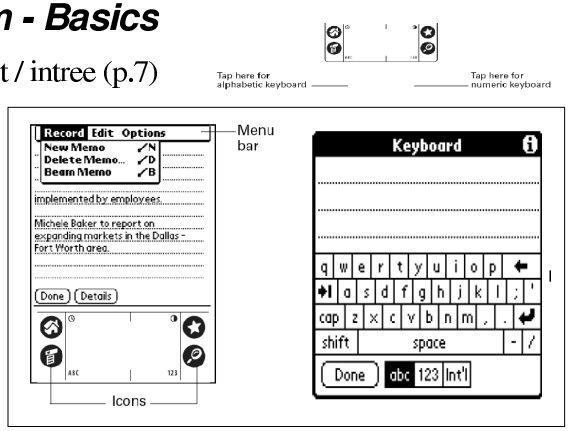
Palm - Basics

- Front / voor (p.3)



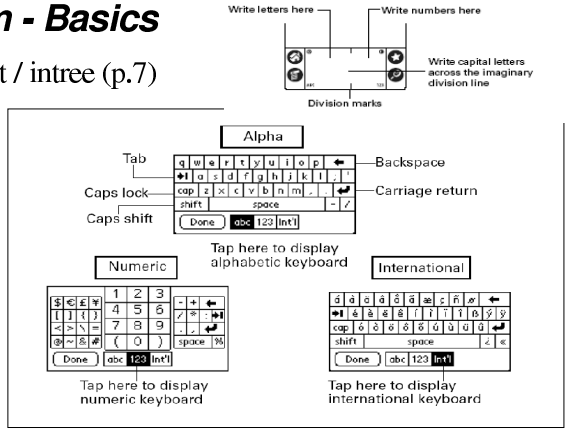
Palm - Basics

- Input / intree (p.7)

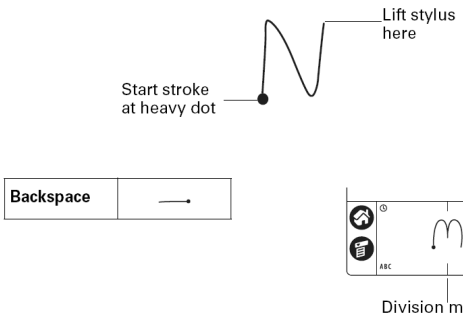


Palm - Basics

- Input / intree (p.7)



Palm - Graffiti



Palm - Graffiti

Draw letters on LEFT side of input area

Letter	Strokes	Letter	Strokes
A		B	
C		D	
E		F	
G		H	
I		J	
K		L	
M		N	

Palm - Graffiti

Draw letters on LEFT side of input area

Letter	Strokes	Letter	Strokes
O		P	
Q		R	
S		T	
U		V	
W		X	
Y		Z	

Palm - Graffiti

Draw numbers on RIGHT side of input area

Number	Strokes	Number	Strokes
0		1	
2		3	
4		5	
6		7	
8		9	

Palm - Graffiti

Draw these marks on LEFT side of input area			
Question ?		Tab	
Exclamation !			

Draw these marks on LEFT side of input area			
Mark	Stroke	Mark	Stroke
Period .		Ampersand &	
Comma ,		Carriage return	
Apostrophe '		At @	
Space		Straight quotes "	

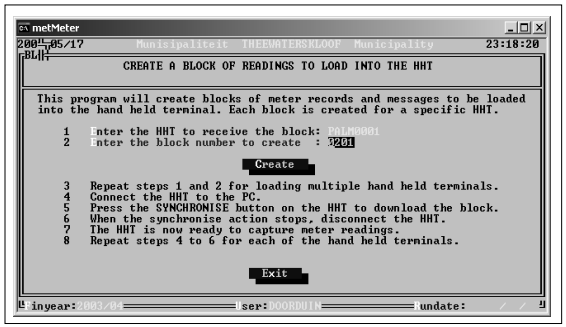
Palm - Graffiti

Draw these marks on RIGHT side of input area

Mark	Stroke	Mark	Stroke
Period .		Backslash \	
Comma ,		Slash /	
Tilde ~		Left Paren (	
Dash –		Right Paren)	
Plus +		Equal =	
Asterisk *			

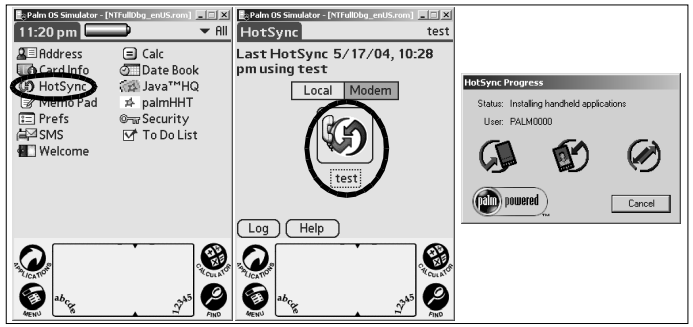
HotSync - metMeter

- Setup route / stel roete op

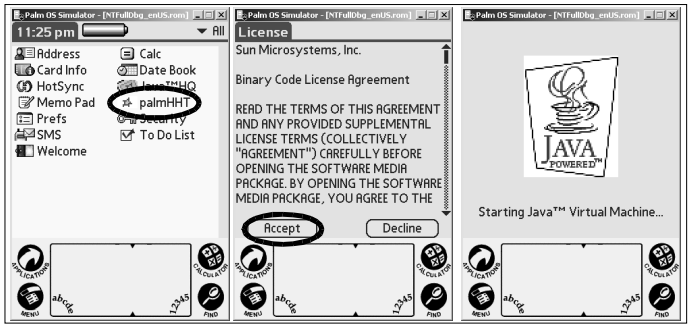


HotSync - palm

- Connect cable / koppel kabel

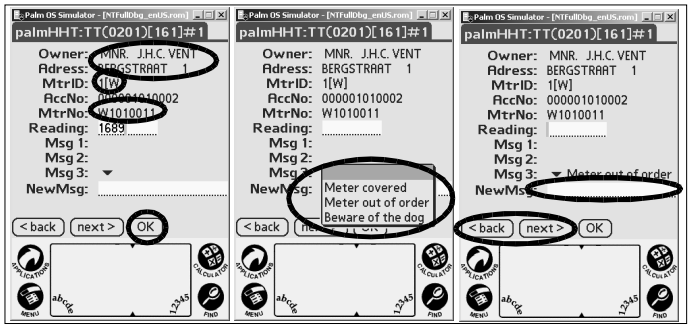


palmHHT - 1,2,3...

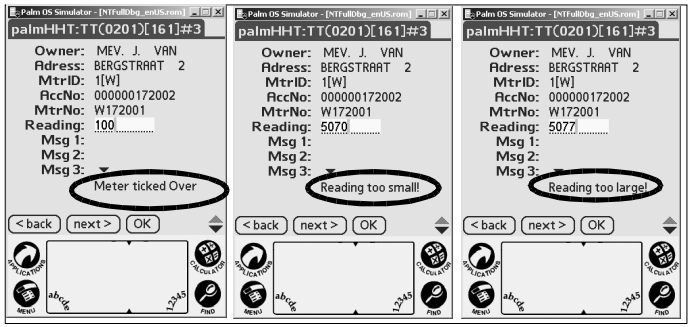


palmHHT - 1,2,3...

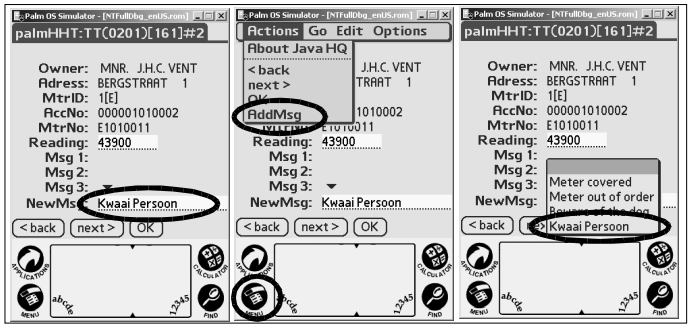
palmHHT – Screen / Skerm



palmHHT – Data capture / -versamel



palmHHT – Messages / Boodskappe



Questions / Vrae

Pause...

Exercises / Oefeninge

- Palm
  - Datum / Tyd
  - Memo Pad
- Hotsync
  - palm
- palmHHT
  - 1690
  - 43834
  - 5076
  - 223
  - “Meter onleesbaar”
  - “Beware of the dog”

The End / Einde

Thank you / Dankie

Appendix O

System Tests

O.1 Data Collection System

The tests cases and results applicable to the software can be found here:

Testcases

TESTLIST

Customer: Derick Retief
Project: Palm Handheld Meter Reader Software
Ref.: DR/HHT/P001/0001
Start: 24/06/2004
Completed: 26/06/2004

Total Tests: 49
Executed: 41
83.67%

		%	Tests
Positive	P	75.61%	31
Executed	E	2.44%	1
Negative	N	0.00%	0
Not applicable	N/A	21.95%	9

System	Subsystem	Test description	Results	Date	Complete	Results
PPST for 68k Apps 200	Palm OS	Default Setting for Palm OS Simulator	P	24/06/2004	24/06/2004	PalmOSSimulatorSetup
		Default Setting for Palm OS Emulator	P	24/06/2004	25/06/2004	Palm OS Emulator Setup
	Application Launch	Normal Launch and Exit	P	25/06/2004	25/06/2004	
		Unsported devices	E	25/06/2004	25/06/2004	Could not setup unsupported device
		Launching via Global Find	N/A			
		Launching from Memory Card	P	25/06/2004	25/06/2004	
		Handling the reset launch Code	P	25/06/2004	25/06/2004	
		Alarms	N/A			
	Alternet Sreen Config	320x480 16bit	P	25/06/2004	25/06/2004	
		240x320 8-bit	P	25/06/2004	25/06/2004	
		Additional Resolutions	P	25/06/2004	25/06/2004	
		Dynamic Input Area	N/A			
	User Interface Experie	Field Input and edit Menu	P	25/06/2004	25/06/2004	
		UI Objects	P	25/06/2004	25/06/2004	
		Modal dialog design should avoid data loss.	N/A			
		About Dialog	N/A			
	System Integration	Mask/Unmask	N/A			
		Obscuring the launcher Icon	P	25/06/2004	25/06/2004	
		Unique creator ID and database Names	P	25/06/2004	25/06/2004	Ppmr
		System Preferences	P	25/06/2004	26/06/2004	
		Application Visible in Launcher	P	26/06/2004	26/06/2004	
		Reserved Shortcuts	N/A			
		Application deletion	P	26/06/2004	26/06/2004	
		Backup Bit Set	P	26/06/2004	26/06/2004	
		Stress Test	P	26/06/2004	26/06/2004	Gremlins Test Setup
		Low Mem	N/A			Could not find MemHog
	Cobalt Compatibility	Palm OS Cobalt Simulator Compatibility	N/A			
HHTCond	GUI	Correct XML storage after Default Directory Change, e.g. Java 1.4 correctly installed.	P	26/06/2004	26/06/2004	
		When invoking Setup the correct Default Directory is displayed	P	26/06/2004	26/06/2004	
		GUI can be accessed via Conduit(HotSync) Custom Setup	P	26/06/2004	26/06/2004	
		After access via HotSync, when the GUI is closed, the HotSync process stays active	P	26/06/2004	26/06/2004	
	HotSync	Correct detection of data directory	P	26/06/2004	26/06/2004	
		Correct interpretation of PCT file	P	26/06/2004	26/06/2004	
		User correctly detected	P	26/06/2004	26/06/2004	
		Correct detection of databases	P	26/06/2004	26/06/2004	
		Upload successful	P	26/06/2004	26/06/2004	
		Download successful	P	26/06/2004	26/06/2004	
Zire 21	MIDP.prc	Java VM on Palm	P	26/06/2004	26/06/2004	
	PalmHHT.prc	Palm HHT loaded	P	26/06/2004	26/06/2004	
	Palm HHT & HotSync	1st execution correctly with no Database. Displays a zero entry into databases.				
		HotSync process now uploads data correctly				
		2nd execution shows new uploaded data.				
		HotSync process now downloads and uploads data correctly, e.g. Datafile changed				
		3rd invokation shows new database				
	PalmHHT	When a lower value than the previous value is entered for a meter reading, the error message ticked over is displayed				
		Within 10% minimum of average value added to previous reading				
		Within 10% maximum of average value added to previous reading				
		Correct display of empty and full reading	P	26/06/2004	26/06/2004	
MetMeter	Import/Export	Correct export to the Palm HHT and correct export to the metMeter system.	P	26/06/2004	26/06/2004	

Appendix P

XML Code

P.1 pInstaller

In order to package the Data Collection software, a program, pInstaller[38] used. This software requires an XML file to collect data to compile the installation package, which is supplied below.

P.1.1 xml: `settings.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<pInstaller xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://www.handx.net/devtool/pInstaller/schem
   <productName>palmHHT</productName>
   <preInstallationMessage>
      <title>HotSync Notes</title>
      <caption>Please adhere carefully to the following instructions:</caption>
      <fileName>preMessage.rtf</fileName>
   </preInstallationMessage>
   <postInstallationMessage>
      <title>Virtual Machine Update</title>
      <caption>Please run the WinZip as described below</caption>
      <fileName>postMessage.rtf</fileName>
   </postInstallationMessage>
        <jSyncInstaller>
                <fileName>jSyncInstaller.exe</fileName>
                <runHidden>true</runHidden>
        </jSyncInstaller>
   <installationFile>
      <fileName>palmHHT.prc</fileName>
      <destination>$PALM_DEVICE</destination>
      <fileGroup>0</fileGroup>
   </installationFile>
   <installationFile>
      <fileName>MIDP.prc</fileName>
      <destination>$PALM_DEVICE</destination>
      <fileGroup>0</fileGroup>
   </installationFile>
   <installationFile>
      <fileName>HHTSync.jar</fileName>
      <destination>$HOTSYNCDIR</destination>
      <fileGroup>0</fileGroup>
      <conduit>
               <fileName>jSync13.dll</fileName>
               <creatorID>Ppmr</creatorID>
               <priority>2</priority>
               <name>Palm Power Meter Reader</name>
               <JavaSettings>
                                       <ClassPath>HHTSync.jar</ClassPath>
                                       <ClassName>green.HHTSync.HHTCond</ClassName>
               </JavaSettings>
      </conduit>
```

```
   </installationFile>
   <installationFile>
      <fileName>rt.exe</fileName>
      <executeAfterInstallation>
         <doNotInstall>true</doNotInstall>
         <onlyExecuteIfSelected>true</onlyExecuteIfSelected>
         <doNotWaitForCompletion>true</doNotWaitForCompletion>
      </executeAfterInstallation>
   </installationFile>
</pInstaller>
```

P.2 Data Minor

The various tree structures used to simulate and visualize data is stored in XML files to reconstruct the environment.

P.2.1 xml: `Mooiwater_SimAllNode_ActualData.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.1_01" class="java.beans.XMLDecoder">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>188</int>
      <int>67</int>
      <int>741</int>
      <int>526</int>
     </object>
    </void>
    <void property="area">
     <string>A10</string>
    </void>
    <void property="childDataCollection">
     <object class="peg.utils.dataCollection">
      <void property="outputDateFormat">
       <string>yyyy-MM</string>
      </void>
     </object>
    </void>
    <void property="hystMax">
     <double>600.0</double>
    </void>
    <void property="hystMin">
     <double>250.0</double>
    </void>
    <void property="hystogramSource">
     <int>1</int>
    </void>
    <void property="lineAttribute">
     <void property="style">
      <int>2</int>
     </void>
    </void>
    <void property="name">
     <string>frame9</string>
    </void>
    <void property="nodeName">
     <string>MOOIWATER</string>
    </void>
    <void property="outputSource">
     <int>1</int>
    </void>
    <void property="unit">
     <string>W</string>
    </void>
   </object>
  </void>
  <void method="add">
   <object class="javax.swing.tree.DefaultMutableTreeNode">
```

```
<void property="userObject">
 <object class="peg.swing.myNode">
  <void property="bounds">
   <object class="java.awt.Rectangle">
    <int>1295</int>
    <int>123</int>
    <int>689</int>
    <int>461</int>
   </object>
  </void>
  <void property="area">
   <string>1</string>
  </void>
  <void property="childDataCollection">
   <object class="peg.utils.dataCollection">
    <void property="outputDateFormat">
     <string>yyyy-MM</string>
    </void>
   </object>
  </void>
  <void property="name">
   <string>frame2</string>
  </void>
  <void property="nodeName">
   <string>1</string>
  </void>
  <void property="outputSource">
   <int>1</int>
  </void>
  <void property="unit">
   <string>W</string>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>150</int>
      <int>22</int>
      <int>666</int>
      <int>586</int>
     </object>
    </void>
    <void property="area">
     <string>1</string>
    </void>
    <void property="deviation">
     <double>1.97</double>
    </void>
    <void property="mean">
     <double>4.69</double>
    </void>
    <void property="name">
     <string>frame3</string>
    </void>
    <void property="nodeName">
     <string>A10/6</string>
    </void>
    <void property="outputSource">
     <int>5</int>
    </void>
    <void property="scalingFactor">
     <double>4225.76</double>
    </void>
    <void property="unit">
     <string>W</string>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="size">
```

```
      <object class="java.awt.Dimension">
       <int>849</int>
       <int>643</int>
      </object>
     </void>
     <void property="area">
      <string>1</string>
     </void>
     <void property="deviation">
      <double>3.8</double>
     </void>
     <void property="mean">
      <double>14.22</double>
     </void>
     <void property="name">
      <string>frame5</string>
     </void>
     <void property="nodeName">
      <string>A10/5</string>
     </void>
     <void property="outputSource">
      <int>5</int>
     </void>
     <void property="scalingFactor">
      <double>14057.12</double>
     </void>
     <void property="unit">
      <string>W</string>
     </void>
    </object>
   </void>
  </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="size">
     <object class="java.awt.Dimension">
      <int>692</int>
      <int>583</int>
     </object>
    </void>
    <void property="area">
     <string>1</string>
    </void>
    <void property="deviation">
     <double>4.24</double>
    </void>
    <void property="mean">
     <double>20.03</double>
    </void>
    <void property="name">
     <string>frame7</string>
    </void>
    <void property="nodeName">
     <string>A10/4</string>
    </void>
    <void property="outputSource">
     <int>5</int>
    </void>
    <void property="scalingFactor">
     <double>20433.6</double>
    </void>
    <void property="unit">
     <string>W</string>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>220</int>
      <int>6</int>
```

```
      <int>716</int>
      <int>616</int>
     </object>
    </void>
    <void property="area">
     <string>1</string>
    </void>
    <void property="deviation">
     <double>2.65</double>
    </void>
    <void property="mean">
     <double>7.64</double>
    </void>
    <void property="name">
     <string>frame9</string>
    </void>
    <void property="nodeName">
     <string>A10/3</string>
    </void>
    <void property="outputSource">
     <int>5</int>
    </void>
    <void property="scalingFactor">
     <double>6865.92</double>
    </void>
    <void property="unit">
     <string>w</string>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="size">
     <object class="java.awt.Dimension">
      <int>747</int>
      <int>584</int>
     </object>
    </void>
    <void property="area">
     <string>1</string>
    </void>
    <void property="deviation">
     <double>3.34</double>
    </void>
    <void property="mean">
     <double>12.3</double>
    </void>
    <void property="name">
     <string>frame11</string>
    </void>
    <void property="nodeName">
     <string>A10/2</string>
    </void>
    <void property="outputSource">
     <int>5</int>
    </void>
    <void property="scalingFactor">
     <double>12101.4</double>
    </void>
    <void property="unit">
     <string>W</string>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="size">
     <object class="java.awt.Dimension">
      <int>856</int>
      <int>628</int>
     </object>
```

```
      </void>
      <void property="area">
       <string>1</string>
      </void>
      <void property="deviation">
       <double>3.19</double>
      </void>
      <void property="mean">
       <double>21.49</double>
      </void>
      <void property="name">
       <string>frame13</string>
      </void>
      <void property="nodeName">
       <string>A10/1</string>
      </void>
      <void property="outputSource">
       <int>5</int>
      </void>
      <void property="scalingFactor">
       <double>22577.86</double>
      </void>
      <void property="unit">
       <string>W</string>
      </void>
     </object>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>1135</int>
      <int>100</int>
      <int>628</int>
      <int>516</int>
     </object>
    </void>
    <void property="area">
     <string>2</string>
    </void>
    <void property="childDataCollection">
     <object class="peg.utils.dataCollection">
      <void property="outputDateFormat">
       <string>yyyy-MM</string>
      </void>
     </object>
    </void>
    <void property="name">
     <string>frame11</string>
    </void>
    <void property="nodeName">
     <string>2</string>
    </void>
    <void property="outputSource">
     <int>1</int>
    </void>
    <void property="unit">
     <string>W</string>
    </void>
   </object>
  </void>
  <void method="add">
   <object class="javax.swing.tree.DefaultMutableTreeNode">
    <void property="userObject">
     <object class="peg.swing.myNode">
      <void property="bounds">
       <object class="java.awt.Rectangle">
        <int>83</int>
        <int>5</int>
        <int>679</int>
        <int>552</int>
       </object>
      </void>
```

```
     <void property="area">
      <string>2</string>
     </void>
     <void property="deviation">
      <double>1.47</double>
     </void>
     <void property="mean">
      <double>3.99</double>
     </void>
     <void property="name">
      <string>frame13</string>
     </void>
     <void property="nodeName">
      <string>A10/13</string>
     </void>
     <void property="outputSource">
      <int>5</int>
     </void>
     <void property="scalingFactor">
      <double>3667.97</double>
     </void>
     <void property="unit">
      <string>W</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>122</int>
       <int>37</int>
       <int>734</int>
       <int>587</int>
      </object>
     </void>
     <void property="area">
      <string>2</string>
     </void>
     <void property="deviation">
      <double>0.28</double>
     </void>
     <void property="mean">
      <double>0.54</double>
     </void>
     <void property="name">
      <string>frame15</string>
     </void>
     <void property="nodeName">
      <string>A10/12</string>
     </void>
     <void property="outputSource">
      <int>5</int>
     </void>
     <void property="scalingFactor">
      <double>458.5</double>
     </void>
     <void property="unit">
      <string>W</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="size">
      <object class="java.awt.Dimension">
       <int>786</int>
       <int>611</int>
      </object>
     </void>
     <void property="area">
```

```
      <string>2</string>
     </void>
     <void property="deviation">
      <double>2.7</double>
     </void>
     <void property="mean">
      <double>11.49</double>
     </void>
     <void property="name">
      <string>frame17</string>
     </void>
     <void property="nodeName">
      <string>A10/11</string>
     </void>
     <void property="outputSource">
      <int>5</int>
     </void>
     <void property="scalingFactor">
      <double>11572.0</double>
     </void>
     <void property="unit">
      <string>W</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="size">
      <object class="java.awt.Dimension">
       <int>862</int>
       <int>633</int>
      </object>
     </void>
     <void property="area">
      <string>2</string>
     </void>
     <void property="deviation">
      <double>0.29</double>
     </void>
     <void property="mean">
      <double>0.4</double>
     </void>
     <void property="name">
      <string>frame19</string>
     </void>
     <void property="nodeName">
      <string>A10/10</string>
     </void>
     <void property="outputSource">
      <int>5</int>
     </void>
     <void property="scalingFactor">
      <double>308.0</double>
     </void>
     <void property="unit">
      <string>W</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>186</int>
       <int>54</int>
       <int>701</int>
       <int>600</int>
      </object>
     </void>
     <void property="area">
      <string>2</string>
```

```
     </void>
     <void property="deviation">
      <double>0.8</double>
     </void>
     <void property="mean">
      <double>2.02</double>
     </void>
     <void property="name">
      <string>frame21</string>
     </void>
     <void property="nodeName">
      <string>A10/9</string>
     </void>
     <void property="outputSource">
      <int>5</int>
     </void>
     <void property="scalingFactor">
      <double>1848.0</double>
     </void>
     <void property="unit">
      <string>W</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="size">
      <object class="java.awt.Dimension">
       <int>799</int>
       <int>636</int>
      </object>
     </void>
     <void property="area">
      <string>2</string>
     </void>
     <void property="deviation">
      <double>2.08</double>
     </void>
     <void property="mean">
      <double>7.97</double>
     </void>
     <void property="name">
      <string>frame23</string>
     </void>
     <void property="nodeName">
      <string>A10/8</string>
     </void>
     <void property="outputSource">
      <int>5</int>
     </void>
     <void property="scalingFactor">
      <double>7914.59</double>
     </void>
     <void property="unit">
      <string>W</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="size">
      <object class="java.awt.Dimension">
       <int>922</int>
       <int>587</int>
      </object>
     </void>
     <void property="area">
      <string>2</string>
     </void>
     <void property="deviation">
      <double>7.75</double>
```

```
      </void>
      <void property="mean">
       <double>50.72</double>
      </void>
      <void property="name">
       <string>frame79</string>
      </void>
      <void property="nodeName">
       <string>A10/7</string>
      </void>
      <void property="outputSource">
       <int>5</int>
      </void>
      <void property="scalingFactor">
       <double>53174.02</double>
      </void>
      <void property="unit">
       <string>W</string>
      </void>
     </object>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="size">
     <object class="java.awt.Dimension">
      <int>840</int>
      <int>622</int>
     </object>
    </void>
    <void property="area">
     <string>3</string>
    </void>
    <void property="childDataCollection">
     <object class="peg.utils.dataCollection">
      <void property="outputDateFormat">
       <string>yyyy-MM</string>
      </void>
     </object>
    </void>
    <void property="name">
     <string>frame35</string>
    </void>
    <void property="nodeName">
     <string>3</string>
    </void>
    <void property="outputSource">
     <int>1</int>
    </void>
    <void property="unit">
     <string>W</string>
    </void>
   </object>
  </void>
  <void method="add">
   <object class="javax.swing.tree.DefaultMutableTreeNode">
    <void property="userObject">
     <object class="peg.swing.myNode">
      <void property="bounds">
       <object class="java.awt.Rectangle">
        <int>93</int>
        <int>45</int>
        <int>691</int>
        <int>593</int>
       </object>
      </void>
      <void property="area">
       <string>3</string>
      </void>
      <void property="deviation">
       <double>0.92</double>
      </void>
      <void property="mean">
       <double>1.49</double>
```

```
    </void>
    <void property="name">
     <string>frame37</string>
    </void>
    <void property="nodeName">
     <string>A10/18</string>
    </void>
    <void property="outputSource">
     <int>5</int>
    </void>
    <void property="scalingFactor">
     <double>1192.0</double>
    </void>
    <void property="unit">
     <string>W</string>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>52</int>
      <int>43</int>
      <int>814</int>
      <int>649</int>
     </object>
    </void>
    <void property="area">
     <string>3</string>
    </void>
    <void property="deviation">
     <double>4.48</double>
    </void>
    <void property="mean">
     <double>14.53</double>
    </void>
    <void property="name">
     <string>frame39</string>
    </void>
    <void property="nodeName">
     <string>A10/17</string>
    </void>
    <void property="outputSource">
     <int>5</int>
    </void>
    <void property="scalingFactor">
     <double>14017.92</double>
    </void>
    <void property="unit">
     <string>W</string>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="size">
     <object class="java.awt.Dimension">
      <int>889</int>
      <int>615</int>
     </object>
    </void>
    <void property="area">
     <string>3</string>
    </void>
    <void property="deviation">
     <double>1.58</double>
    </void>
    <void property="mean">
     <double>3.67</double>
    </void>
```

```
      <void property="name">
       <string>frame41</string>
      </void>
      <void property="nodeName">
       <string>A10/16</string>
      </void>
      <void property="outputSource">
       <int>5</int>
      </void>
      <void property="scalingFactor">
       <double>3218.4</double>
      </void>
      <void property="unit">
       <string>W</string>
      </void>
     </object>
    </void>
   </object>
  </void>
  <void method="add">
   <object class="javax.swing.tree.DefaultMutableTreeNode">
    <void property="userObject">
     <object class="peg.swing.myNode">
      <void property="size">
       <object class="java.awt.Dimension">
        <int>841</int>
        <int>608</int>
       </object>
      </void>
      <void property="area">
       <string>3</string>
      </void>
      <void property="deviation">
       <double>0.47</double>
      </void>
      <void property="mean">
       <double>2.36</double>
      </void>
      <void property="name">
       <string>frame43</string>
      </void>
      <void property="nodeName">
       <string>A10/15</string>
      </void>
      <void property="outputSource">
       <int>5</int>
      </void>
      <void property="scalingFactor">
       <double>2426.59</double>
      </void>
      <void property="unit">
       <string>W</string>
      </void>
     </object>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="size">
     <object class="java.awt.Dimension">
      <int>693</int>
      <int>547</int>
     </object>
    </void>
    <void property="area">
     <string>4</string>
    </void>
    <void property="childDataCollection">
     <object class="peg.utils.dataCollection">
      <void property="outputDateFormat">
       <string>yyyy-MM</string>
      </void>
     </object>
    </void>
```

```
   <void property="name">
    <string>frame45</string>
   </void>
   <void property="nodeName">
    <string>4</string>
   </void>
   <void property="outputSource">
    <int>1</int>
   </void>
   <void property="unit">
    <string>W</string>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>64</int>
       <int>60</int>
       <int>778</int>
       <int>573</int>
      </object>
     </void>
     <void property="area">
      <string>4</string>
     </void>
     <void property="deviation">
      <double>2.06</double>
     </void>
     <void property="mean">
      <double>4.78</double>
     </void>
     <void property="name">
      <string>frame47</string>
     </void>
     <void property="nodeName">
      <string>A10/28</string>
     </void>
     <void property="outputSource">
      <int>5</int>
     </void>
     <void property="scalingFactor">
      <double>4191.26</double>
     </void>
     <void property="unit">
      <string>W</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>56</int>
       <int>53</int>
       <int>847</int>
       <int>534</int>
      </object>
     </void>
     <void property="area">
      <string>4</string>
     </void>
     <void property="deviation">
      <double>0.74</double>
     </void>
     <void property="mean">
      <double>1.19</double>
     </void>
     <void property="name">
      <string>frame49</string>
     </void>
     <void property="nodeName">
```

```
      <string>A10/27</string>
     </void>
     <void property="outputSource">
      <int>5</int>
     </void>
     <void property="scalingFactor">
      <double>953.6</double>
     </void>
     <void property="unit">
      <string>W</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>78</int>
       <int>76</int>
       <int>835</int>
       <int>521</int>
      </object>
     </void>
     <void property="area">
      <string>4</string>
     </void>
     <void property="deviation">
      <double>2.33</double>
     </void>
     <void property="mean">
      <double>5.56</double>
     </void>
     <void property="name">
      <string>frame51</string>
     </void>
     <void property="nodeName">
      <string>A10/26</string>
     </void>
     <void property="outputSource">
      <int>5</int>
     </void>
     <void property="scalingFactor">
      <double>5006.4</double>
     </void>
     <void property="unit">
      <string>W</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>139</int>
       <int>90</int>
       <int>784</int>
       <int>541</int>
      </object>
     </void>
     <void property="area">
      <string>4</string>
     </void>
     <void property="deviation">
      <double>3.1</double>
     </void>
     <void property="mean">
      <double>13.97</double>
     </void>
     <void property="name">
      <string>frame53</string>
     </void>
```

```
      <void property="nodeName">
       <string>A10/25</string>
      </void>
      <void property="outputSource">
       <int>5</int>
      </void>
      <void property="scalingFactor">
       <double>14168.0</double>
      </void>
      <void property="unit">
       <string>W</string>
      </void>
     </object>
    </void>
   </object>
  </void>
  <void method="add">
   <object class="javax.swing.tree.DefaultMutableTreeNode">
    <void property="userObject">
     <object class="peg.swing.myNode">
      <void property="bounds">
       <object class="java.awt.Rectangle">
        <int>59</int>
        <int>58</int>
        <int>832</int>
        <int>587</int>
       </object>
      </void>
      <void property="area">
       <string>4</string>
      </void>
      <void property="deviation">
       <double>0.96</double>
      </void>
      <void property="mean">
       <double>1.79</double>
      </void>
      <void property="name">
       <string>frame55</string>
      </void>
      <void property="nodeName">
       <string>A10/24</string>
      </void>
      <void property="outputSource">
       <int>5</int>
      </void>
      <void property="scalingFactor">
       <double>1430.4</double>
      </void>
      <void property="unit">
       <string>W</string>
      </void>
     </object>
    </void>
   </object>
  </void>
  <void method="add">
   <object class="javax.swing.tree.DefaultMutableTreeNode">
    <void property="userObject">
     <object class="peg.swing.myNode">
      <void property="bounds">
       <object class="java.awt.Rectangle">
        <int>51</int>
        <int>56</int>
        <int>852</int>
        <int>613</int>
       </object>
      </void>
      <void property="area">
       <string>4</string>
      </void>
      <void property="deviation">
       <double>4.71</double>
      </void>
      <void property="mean">
       <double>15.94</double>
      </void>
      <void property="name">
       <string>frame57</string>
```

```
    </void>
    <void property="nodeName">
     <string>A10/23</string>
    </void>
    <void property="outputSource">
     <int>5</int>
    </void>
    <void property="scalingFactor">
     <double>15448.32</double>
    </void>
    <void property="unit">
     <string>W</string>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>124</int>
      <int>73</int>
      <int>737</int>
      <int>514</int>
     </object>
    </void>
    <void property="area">
     <string>4</string>
    </void>
    <void property="deviation">
     <double>5.28</double>
    </void>
    <void property="mean">
     <double>33.9</double>
    </void>
    <void property="name">
     <string>frame59</string>
    </void>
    <void property="nodeName">
     <string>A10/22</string>
    </void>
    <void property="outputSource">
     <int>5</int>
    </void>
    <void property="scalingFactor">
     <double>35501.33</double>
    </void>
    <void property="unit">
     <string>W</string>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>144</int>
      <int>99</int>
      <int>655</int>
      <int>506</int>
     </object>
    </void>
    <void property="area">
     <string>4</string>
    </void>
    <void property="deviation">
     <double>1.31</double>
    </void>
    <void property="mean">
     <double>2.53</double>
    </void>
    <void property="name">
```

```
      <string>frame61</string>
     </void>
     <void property="nodeName">
      <string>A10/21</string>
     </void>
     <void property="outputSource">
      <int>5</int>
     </void>
     <void property="scalingFactor">
      <double>2145.6</double>
     </void>
     <void property="unit">
      <string>W</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>144</int>
       <int>69</int>
       <int>714</int>
       <int>515</int>
      </object>
     </void>
     <void property="area">
      <string>4</string>
     </void>
     <void property="deviation">
      <double>2.51</double>
     </void>
     <void property="mean">
      <double>7.84</double>
     </void>
     <void property="name">
      <string>frame63</string>
     </void>
     <void property="nodeName">
      <string>A10/20</string>
     </void>
     <void property="outputSource">
      <int>5</int>
     </void>
     <void property="scalingFactor">
      <double>7516.8</double>
     </void>
     <void property="unit">
      <string>W</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>167</int>
       <int>121</int>
       <int>728</int>
       <int>599</int>
      </object>
     </void>
     <void property="area">
      <string>4</string>
     </void>
     <void property="deviation">
      <double>10.15</double>
     </void>
     <void property="mean">
      <double>79.28</double>
     </void>
```

```
        <void property="name">
         <string>frame65</string>
        </void>
        <void property="nodeName">
         <string>A10/19</string>
        </void>
        <void property="outputSource">
         <int>5</int>
        </void>
        <void property="scalingFactor">
         <double>83909.92</double>
        </void>
        <void property="unit">
         <string>W</string>
        </void>
       </object>
      </void>
     </object>
    </void>
   </object>
  </void>
 </object>
</java>
```

P.2.2 xml: `CheckMeter.xml`

For this XML file there are various permutations possible. It is advised to switch the functionality of the various nodes as needed.


```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.1_01" class="java.beans.XMLDecoder">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="size">
     <object class="java.awt.Dimension">
      <int>787</int>
      <int>621</int>
     </object>
    </void>
    <void property="area">
     <string>Stellenbosch</string>
    </void>
    <void property="averagePeriod">
     <int>3</int>
    </void>
    <void property="childDataCollection">
     <object class="peg.utils.dataCollection">
      <void property="outputDateFormat">
       <string>yyyy-MM</string>
      </void>
     </object>
    </void>
    <void property="hystMax">
     <double>5000.0</double>
    </void>
    <void property="hystMin">
     <double>1000.0</double>
    </void>
    <void property="hystogramSource">
     <int>1</int>
    </void>
    <void property="nodeName">
     <string>Stellenbosch Electricity Theft Measurements</string>
    </void>
    <void property="outputSource">
     <int>1</int>
    </void>
    <void property="unit">
     <string>kWh</string>
    </void>
   </object>
  </void>
  <void method="add">
   <object class="javax.swing.tree.DefaultMutableTreeNode">
```

```
<void property="userObject">
 <object class="peg.swing.myNode">
  <void property="bounds">
   <object class="java.awt.Rectangle">
    <int>42</int>
    <int>71</int>
    <int>796</int>
    <int>568</int>
   </object>
  </void>
  <void property="area">
   <string>Mooiwater</string>
  </void>
  <void property="childDataCollection">
   <object class="peg.utils.dataCollection">
    <void property="outputDateFormat">
     <string>yyyy-MM</string>
    </void>
   </object>
  </void>
  <void property="hystMax">
   <double>37500.0</double>
  </void>
  <void property="hystMin">
   <double>37400.0</double>
  </void>
  <void property="name">
   <string>frame1</string>
  </void>
  <void property="nodeName">
   <string>Mooiwater Checkmeter</string>
  </void>
  <void property="outputSource">
   <int>1</int>
  </void>
  <void property="unit">
   <string>kWh</string>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>65</int>
      <int>67</int>
      <int>786</int>
      <int>579</int>
     </object>
    </void>
    <void property="area">
     <string>Mooiwater</string>
    </void>
    <void property="driver">
     <string>com.mysql.jdbc.Driver</string>
    </void>
    <void property="name">
     <string>frame2</string>
    </void>
    <void property="nodeName">
     <string>Mooiwater Total Meter Readings</string>
    </void>
    <void property="outputSource">
     <int>2</int>
    </void>
    <void property="protocol">
     <string>jdbc:mysql</string>
    </void>
    <void property="query">
     <string>SELECT date, sum(units) FROM prepaid, mooiwater WHERE prepaid.rdpref=mooiwater.rdpref AND date &gt; &apos;2003/03/01&apos; AN
    </void>
    <void property="unit">
     <string>kWh</string>
    </void>
    <void property="url">
     <string>//localhost:3306/stellenbosch?user=server</string>
    </void>
```

```
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>281</int>
      <int>95</int>
      <int>683</int>
      <int>555</int>
     </object>
    </void>
    <void property="area">
     <string>Mooiwater</string>
    </void>
    <void property="description">
     <string>Loss based on Beta model form Fourie</string>
    </void>
    <void property="deviation">
     <double>0.0030</double>
    </void>
    <void property="function">
     <int>2</int>
    </void>
    <void property="hystMax">
     <double>1.06</double>
    </void>
    <void property="hystMin">
     <double>1.0</double>
    </void>
    <void property="mean">
     <double>0.0058</double>
    </void>
    <void property="name">
     <string>frame12</string>
    </void>
    <void property="nodeName">
     <string>Loss Estimate</string>
    </void>
    <void property="offset">
     <double>1.0</double>
    </void>
    <void property="scalingFactor">
     <double>1.0</double>
    </void>
    <void id="GeoDate0" property="startTime">
     <void property="month">
      <int>4</int>
     </void>
     <void property="time">
      <long>1049155200000</long>
     </void>
    </void>
    <void property="startTime">
     <object idref="GeoDate0"/>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>1204</int>
      <int>108</int>
      <int>660</int>
      <int>601</int>
     </object>
    </void>
    <void property="area">
     <string>Mooiwater</string>
    </void>
```

```
       <void property="description">
        <string>Normalized loss based on Beta model form Fourie</string>
       </void>
       <void property="deviation">
        <double>4.31</double>
       </void>
       <void property="hystMax">
        <double>270.0</double>
       </void>
       <void property="hystMin">
        <double>240.0</double>
       </void>
       <void property="hystogramSource">
        <int>5</int>
       </void>
       <void property="lineAttribute">
        <void property="color">
         <object class="java.awt.Color">
          <int>0</int>
          <int>0</int>
          <int>255</int>
          <int>255</int>
         </object>
        </void>
        <void property="style">
         <int>2</int>
        </void>
       </void>
       <void property="mean">
        <double>166.937</double>
       </void>
       <void property="name">
        <string>frame7</string>
       </void>
       <void property="nodeName">
        <string>Loss Variable</string>
       </void>
       <void property="outputSource">
        <int>5</int>
       </void>
       <void property="scalingFactor">
        <double>49890.36</double>
       </void>
       <void id="GeoDate1" property="startTime">
        <void property="month">
         <int>4</int>
        </void>
        <void property="time">
         <long>1049155200000</long>
        </void>
       </void>
       <void property="startTime">
        <object idref="GeoDate1"/>
       </void>
       <void property="unit">
        <string>kWh</string>
       </void>
      </object>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>161</int>
       <int>57</int>
       <int>710</int>
       <int>611</int>
      </object>
     </void>
     <void property="area">
      <string>Mooiwater</string>
     </void>
     <void property="childDataCollection">
```

```
   <object class="peg.utils.dataCollection">
    <void property="outputDateFormat">
     <string>yyyy-MM</string>
    </void>
   </object>
  </void>
  <void property="description">
   <string>Mooiwater</string>
  </void>
  <void property="driver">
   <string>com.mysql.jdbc.Driver</string>
  </void>
  <void property="function">
   <int>1</int>
  </void>
  <void property="hystMax">
   <double>10000.0</double>
  </void>
  <void property="name">
   <string>frame8</string>
  </void>
  <void property="nodeName">
   <string>Mooiwater Readings</string>
  </void>
  <void property="outputSource">
   <int>3</int>
  </void>
  <void property="protocol">
   <string>jdbc:mysql</string>
  </void>
  <void property="query">
   <string>SELECT date, sum(units) FROM prepaid, mooiwater WHERE prepaid.rdpref=mooiwater.rdpref AND date &gt; &apos;2003/03/01&apos; AND
  </void>
  <void property="unit">
   <string>kWh</string>
  </void>
  <void property="url">
   <string>//localhost:3306/stellenbosch?user=server</string>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>162</int>
      <int>83</int>
      <int>720</int>
      <int>569</int>
     </object>
    </void>
    <void property="area">
     <string>Mooiwater</string>
    </void>
    <void property="description">
     <string>Beuacoup de l&apos;eau</string>
    </void>
    <void property="driver">
     <string>com.mysql.jdbc.Driver</string>
    </void>
    <void property="name">
     <string>frame7</string>
    </void>
    <void property="nodeName">
     <string>Electricity Theft</string>
    </void>
    <void property="protocol">
     <string>jdbc:mysql</string>
    </void>
    <void property="query">
     <string>SELECT date, sum(units) FROM prepaid, mooiwater WHERE prepaid.rdpref=mooiwater.rdpref AND date &gt; &apos;2003/03/01&apos; AN
    </void>
    <void property="unit">
     <string>kWh</string>
    </void>
    <void property="url">
     <string>//localhost:3306/stellenbosch?user=server</string>
```

```
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>62</int>
      <int>26</int>
      <int>845</int>
      <int>665</int>
     </object>
    </void>
    <void property="area">
     <string>Mooiwater</string>
    </void>
    <void property="description">
     <string>User Variance</string>
    </void>
    <void property="deviation">
     <double>119.9994</double>
    </void>
    <void property="hystMax">
     <double>600.0</double>
    </void>
    <void property="mean">
     <double>159.98</double>
    </void>
    <void property="name">
     <string>frame7</string>
    </void>
    <void property="nodeName">
     <string>Electricty theft uncertainty</string>
    </void>
    <void property="scalingFactor">
     <double>9200.0</double>
    </void>
    <void id="GeoDate2" property="startTime">
     <void property="month">
      <int>4</int>
     </void>
     <void property="time">
      <long>1049155200000</long>
     </void>
    </void>
    <void property="startTime">
     <object idref="GeoDate2"/>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>120</int>
      <int>77</int>
      <int>856</int>
      <int>609</int>
     </object>
    </void>
    <void property="area">
     <string>Mooiwater</string>
    </void>
    <void property="description">
     <string>Select users from whom no readings will be taken</string>
    </void>
    <void property="deviation">
     <double>1.0</double>
    </void>
    <void property="driver">
     <string>com.mysql.jdbc.Driver</string>
```

```
        </void>
        <void property="hystMax">
         <double>2.0</double>
        </void>
        <void property="hystMin">
         <double>-2.0</double>
        </void>
        <void property="name">
         <string>frame8</string>
        </void>
        <void property="nodeName">
         <string>Random Users</string>
        </void>
        <void property="outputSource">
         <int>2</int>
        </void>
        <void property="protocol">
         <string>jdbc:mysql</string>
        </void>
        <void property="query">
         <string>drop table tmprdp;
create table tmprdp select rdpref from mooiwater where minisub like &apos;MS A10 %&apos; GROUP BY RAND() LIMIT 19;
SELECT date, sum(units) FROM prepaid, tmprdp WHERE prepaid.rdpref=tmprdp.rdpref AND date &gt; &apos;2003/03/01&apos; GROUP BY date;</string>
        </void>
        <void property="randomEngine">
         <int>1</int>
        </void>
        <void property="scalingFactor">
         <double>1.0</double>
        </void>
        <void property="unit">
         <string>kWh</string>
        </void>
        <void property="update">
         <boolean>true</boolean>
        </void>
        <void property="url">
         <string>//localhost:3306/stellenbosch?user=server</string>
        </void>
       </object>
      </void>
     </object>
    </void>
   </object>
  </void>
  <void method="add">
   <object class="javax.swing.tree.DefaultMutableTreeNode">
    <void property="userObject">
     <object class="peg.swing.myNode">
      <void property="bounds">
       <object class="java.awt.Rectangle">
        <int>67</int>
        <int>56</int>
        <int>731</int>
        <int>611</int>
       </object>
      </void>
      <void property="area">
       <string>Mooiwater</string>
      </void>
      <void property="description">
       <string>Data of users</string>
      </void>
      <void property="name">
       <string>frame12</string>
      </void>
      <void property="nodeName">
       <string>User data</string>
      </void>
     </object>
    </void>
    <void method="add">
     <object class="javax.swing.tree.DefaultMutableTreeNode">
      <void property="userObject">
       <object class="peg.swing.myNode">
        <void property="bounds">
         <object class="java.awt.Rectangle">
          <int>19</int>
          <int>25</int>
```

```
       <int>743</int>
       <int>610</int>
      </object>
     </void>
     <void property="area">
      <string>Mooiwater</string>
     </void>
     <void property="description">
      <string>Amount of measure users</string>
     </void>
     <void property="driver">
      <string>com.mysql.jdbc.Driver</string>
     </void>
     <void property="movAvgSamples">
      <int>3</int>
     </void>
     <void property="nodeName">
      <string>Amount of users</string>
     </void>
     <void property="outputFilter">
      <int>1</int>
     </void>
     <void property="protocol">
      <string>jdbc:mysql</string>
     </void>
     <void property="query">
      <string>SELECT date, count(units) FROM prepaid, mooiwater WHERE prepaid.rdpref=mooiwater.rdpref AND date &gt; &apos;2003/01/01&apos;
     </void>
     <void property="url">
      <string>//localhost:3306/stellenbosch?user=server</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>103</int>
       <int>97</int>
       <int>740</int>
       <int>591</int>
      </object>
     </void>
     <void property="area">
      <string>Mooiwater</string>
     </void>
     <void property="description">
      <string>Average usage per meter</string>
     </void>
     <void property="driver">
      <string>com.mysql.jdbc.Driver</string>
     </void>
     <void property="movAvgSamples">
      <int>3</int>
     </void>
     <void property="name">
      <string>frame14</string>
     </void>
     <void property="nodeName">
      <string>Average Usage</string>
     </void>
     <void property="outputFilter">
      <int>1</int>
     </void>
     <void property="protocol">
      <string>jdbc:mysql</string>
     </void>
     <void property="query">
      <string>SELECT date, avg(units) FROM prepaid, mooiwater WHERE prepaid.rdpref=mooiwater.rdpref AND date &gt; &apos;2003/01/01&apos; AN
     </void>
     <void property="unit">
      <string>kWh</string>
     </void>
     <void property="url">
      <string>//localhost:3306/stellenbosch?user=server</string>
```

```
       </void>
      </object>
     </void>
    </object>
   </void>
  </object>
 </void>
</java>
```

P.2.3 xml: `LabTests100.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.1_01" class="java.beans.XMLDecoder">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>117</int>
      <int>57</int>
      <int>665</int>
      <int>572</int>
     </object>
    </void>
    <void property="area">
     <string>LAB</string>
    </void>
    <void property="childDataCollection">
     <object class="peg.utils.dataCollection">
      <void property="inputDateFormat">
       <string>yyyy-MM-dd HH:mm</string>
      </void>
      <void property="outputDateFormat">
       <string>yyyy-MM-dd HH:mm</string>
      </void>
     </object>
    </void>
    <void property="description">
     <string>Efficiency</string>
    </void>
    <void property="driver">
     <string>com.mysql.jdbc.Driver</string>
    </void>
    <void property="inputDateFormat">
     <string>yyyy-MM-dd HH:mm</string>
    </void>
    <void property="lineAttribute">
     <void property="mark">
      <int>51</int>
     </void>
     <void property="style">
      <int>2</int>
     </void>
    </void>
    <void property="movAvgSamples">
     <int>3</int>
    </void>
    <void property="nodeName">
     <string>Efficiency</string>
    </void>
    <void property="outputDateFormat">
     <string>yyyy-MM-dd HOUR</string>
    </void>
    <void property="outputFilter">
     <int>1</int>
    </void>
    <void property="outputSource">
     <int>1</int>
    </void>
    <void property="protocol">
     <string>jdbc:mysql</string>
    </void>
    <void property="query">
     <string>SELECT date, ((AW+BW+CW)/1000) FROM checkmeters WHERE date &gt; &apos;2004-08-09&apos;;</string>
    </void>
    <void property="unit">
```

```
    <string>kWh</string>
   </void>
   <void property="url">
    <string>//localhost:3306/smsdb?user=server</string>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>269</int>
       <int>69</int>
       <int>612</int>
       <int>550</int>
      </object>
     </void>
     <void property="area">
      <string>LAB</string>
     </void>
     <void property="childDataCollection">
      <object class="peg.utils.dataCollection">
       <void property="inputDateFormat">
        <string>yyyy-MM-dd HH:mm</string>
       </void>
       <void property="outputDateFormat">
        <string>yyyy-MM-dd HOUR</string>
       </void>
      </object>
     </void>
     <void property="description">
      <string>Load</string>
     </void>
     <void property="driver">
      <string>com.mysql.jdbc.Driver</string>
     </void>
     <void property="inputDateFormat">
      <string>yyyy-MM-dd HH:mm</string>
     </void>
     <void property="lineAttribute">
      <void property="mark">
       <int>48</int>
      </void>
      <void property="style">
       <int>5</int>
      </void>
     </void>
     <void property="movAvgSamples">
      <int>3</int>
     </void>
     <void property="name">
      <string>frame1</string>
     </void>
     <void property="nodeName">
      <string>Load</string>
     </void>
     <void property="outputDateFormat">
      <string>yyyy-MM-dd HOUR</string>
     </void>
     <void property="outputSource">
      <int>1</int>
     </void>
     <void property="protocol">
      <string>jdbc:mysql</string>
     </void>
     <void property="query">
      <string>SELECT date, ((AW+BW+CW)/1000) FROM checkmeters WHERE (date &gt; &apos;2004-08-12 00:00&apos; AND date &lt; &apos;2004-08-12 23
     </void>
     <void property="unit">
      <string>kWh</string>
     </void>
     <void property="update">
      <boolean>true</boolean>
     </void>
     <void property="url">
      <string>//localhost:3306/smsdb?user=server</string>
     </void>
```

```
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>152</int>
      <int>59</int>
      <int>705</int>
      <int>588</int>
     </object>
    </void>
    <void property="area">
     <string>LAB</string>
    </void>
    <void property="description">
     <string>Load 2004-08-10</string>
    </void>
    <void property="differentiate">
     <boolean>true</boolean>
    </void>
    <void property="driver">
     <string>com.mysql.jdbc.Driver</string>
    </void>
    <void property="inputDateFormat">
     <string>yyyy-MM-dd HH:mm</string>
    </void>
    <void property="nodeName">
     <string>Load</string>
    </void>
    <void property="outputDateFormat">
     <string>yyyy-MM-dd HOUR</string>
    </void>
    <void property="protocol">
     <string>jdbc:mysql</string>
    </void>
    <void property="query">
     <string>SELECT download, sum((reading)/10) FROM palmhht WHERE (download &gt; &apos;2004-08-10 00:00&apos; AND download &lt; &apos;200
    </void>
    <void property="unit">
     <string>kWh</string>
    </void>
    <void property="update">
     <boolean>true</boolean>
    </void>
    <void property="url">
     <string>//localhost:3306/palmdb?user=server</string>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>138</int>
      <int>52</int>
      <int>668</int>
      <int>586</int>
     </object>
    </void>
    <void property="area">
     <string>LAB</string>
    </void>
    <void property="description">
     <string>Load 2004-08-12</string>
    </void>
    <void property="differentiate">
     <boolean>true</boolean>
    </void>
    <void property="driver">
     <string>com.mysql.jdbc.Driver</string>
    </void>
    <void property="inputDateFormat">
```

```
       <string>yyyy-MM-dd HH:mm</string>
      </void>
      <void property="name">
       <string>frame4</string>
      </void>
      <void property="nodeName">
       <string>Load</string>
      </void>
      <void property="outputDateFormat">
       <string>yyyy-MM-dd HOUR</string>
      </void>
      <void property="outputSource">
       <int>2</int>
      </void>
      <void property="protocol">
       <string>jdbc:mysql</string>
      </void>
      <void property="query">
       <string>SELECT download, sum((reading)/10) FROM palmhht WHERE (download &gt; &apos;2004-08-12 00:00&apos; AND download &lt; &apos;200
      </void>
      <void property="unit">
       <string>kWh</string>
      </void>
      <void property="update">
       <boolean>true</boolean>
      </void>
      <void property="url">
       <string>//localhost:3306/palmdb?user=server</string>
      </void>
     </object>
    </void>
   </object>
  </void>
  <void method="add">
   <object class="javax.swing.tree.DefaultMutableTreeNode">
    <void property="userObject">
     <object class="peg.swing.myNode">
      <void property="size">
       <object class="java.awt.Dimension">
        <int>695</int>
        <int>539</int>
       </object>
      </void>
      <void property="name">
       <string>frame5</string>
      </void>
     </object>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>178</int>
      <int>105</int>
      <int>688</int>
      <int>589</int>
     </object>
    </void>
    <void property="area">
     <string>LAB</string>
    </void>
    <void property="childDataCollection">
     <object class="peg.utils.dataCollection">
      <void property="inputDateFormat">
       <string>yyyy-MM-dd HH:mm</string>
      </void>
      <void property="outputDateFormat">
       <string>yyyy-MM-dd HOUR</string>
      </void>
     </object>
    </void>
    <void property="description">
     <string>Source</string>
```

```
  </void>
  <void property="driver">
   <string>com.mysql.jdbc.Driver</string>
  </void>
  <void property="function">
   <int>3</int>
  </void>
  <void property="inputDateFormat">
   <string>yyyy-MM-dd HH:mm</string>
  </void>
  <void property="lineAttribute">
   <void property="color">
    <object class="java.awt.Color">
     <int>0</int>
     <int>0</int>
     <int>255</int>
     <int>255</int>
    </object>
   </void>
   <void property="mark">
    <int>51</int>
   </void>
   <void property="style">
    <int>5</int>
   </void>
  </void>
  <void property="movAvgSamples">
   <int>3</int>
  </void>
  <void property="name">
   <string>frame7</string>
  </void>
  <void property="nodeName">
   <string>Check Meter</string>
  </void>
  <void property="outputDateFormat">
   <string>yyyy-MM-dd HOUR</string>
  </void>
  <void property="outputSource">
   <int>1</int>
  </void>
  <void property="protocol">
   <string>jdbc:mysql</string>
  </void>
  <void property="query">
   <string>SELECT date, ((AW+BW+CW)/1000*1.13) FROM checkmeters WHERE date &gt; &apos;2004-08-09&apos;;</string>
  </void>
  <void property="unit">
   <string>kWh</string>
  </void>
  <void property="url">
   <string>//localhost:3306/smsdb?user=server</string>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>165</int>
      <int>63</int>
      <int>753</int>
      <int>598</int>
     </object>
    </void>
    <void property="area">
     <string>LAB</string>
    </void>
    <void property="description">
     <string>Source Data 2004-08-10</string>
    </void>
    <void property="differentiate">
     <boolean>true</boolean>
    </void>
    <void property="driver">
     <string>com.mysql.jdbc.Driver</string>
    </void>
```

```
    <void property="inputDateFormat">
     <string>yyyy-MM-dd HH:mm</string>
    </void>
    <void property="name">
     <string>frame4</string>
    </void>
    <void property="nodeName">
     <string>Check Meter Data</string>
    </void>
    <void property="outputDateFormat">
     <string>yyyy-MM-dd HOUR</string>
    </void>
    <void property="protocol">
     <string>jdbc:mysql</string>
    </void>
    <void property="query">
     <string>SELECT date, ((AW+BW+CW)/1000) FROM checkmeters WHERE (date &gt; &apos;2004-08-10 00:00&apos; AND date &lt; &apos;2004-08-10
    </void>
    <void property="unit">
     <string>kWh</string>
    </void>
    <void property="update">
     <boolean>true</boolean>
    </void>
    <void property="url">
     <string>//localhost:3306/smsdb?user=server</string>
    </void>
   </object>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>1172</int>
      <int>117</int>
      <int>728</int>
      <int>598</int>
     </object>
    </void>
    <void property="area">
     <string>LAB</string>
    </void>
    <void property="description">
     <string>Source Data 2004-08-12</string>
    </void>
    <void property="differentiate">
     <boolean>true</boolean>
    </void>
    <void property="driver">
     <string>com.mysql.jdbc.Driver</string>
    </void>
    <void property="inputDateFormat">
     <string>yyyy-MM-dd HH:mm</string>
    </void>
    <void property="name">
     <string>frame10</string>
    </void>
    <void property="nodeName">
     <string>Check Meter Data</string>
    </void>
    <void property="outputDateFormat">
     <string>yyyy-MM-dd HOUR</string>
    </void>
    <void property="outputSource">
     <int>2</int>
    </void>
    <void property="protocol">
     <string>jdbc:mysql</string>
    </void>
    <void property="query">
     <string>SELECT date, ((AW+BW+CW)/1000) FROM checkmeters WHERE (date &gt; &apos;2004-08-12 00:00&apos; AND date &lt; &apos;2004-08-12
    </void>
    <void property="unit">
     <string>kWh</string>
    </void>
```

```
       <void property="update">
        <boolean>true</boolean>
       </void>
       <void property="url">
        <string>//localhost:3306/smsdb?user=server</string>
       </void>
      </object>
     </void>
    </object>
   </void>
   <void method="add">
    <object class="javax.swing.tree.DefaultMutableTreeNode">
     <void property="userObject">
      <object class="peg.swing.myNode">
       <void property="bounds">
        <object class="java.awt.Rectangle">
         <int>219</int>
         <int>70</int>
         <int>679</int>
         <int>596</int>
        </object>
       </void>
       <void property="area">
        <string>LAB</string>
       </void>
       <void property="description">
        <string>Source DAta 2004-08-26</string>
       </void>
       <void property="differentiate">
        <boolean>true</boolean>
       </void>
       <void property="driver">
        <string>com.mysql.jdbc.Driver</string>
       </void>
       <void property="inputDateFormat">
        <string>yyyy-MM-dd HH:mm</string>
       </void>
       <void property="name">
        <string>frame8</string>
       </void>
       <void property="nodeName">
        <string>Check Meter Data</string>
       </void>
       <void property="outputDateFormat">
        <string>yyyy-MM-dd HOUR MINUTE</string>
       </void>
       <void property="protocol">
        <string>jdbc:mysql</string>
       </void>
       <void property="query">
        <string>SELECT date, ((AW+BW+CW)/1000*0.9715) FROM checkmeters WHERE (date &gt; &apos;2004-08-26 00:00&apos; AND date &lt; &apos;2004
       </void>
       <void property="unit">
        <string>kWh</string>
       </void>
       <void property="update">
        <boolean>true</boolean>
       </void>
       <void property="url">
        <string>//localhost:3306/smsdb?user=server</string>
       </void>
      </object>
     </void>
    </object>
   </void>
  </object>
 </void>
</object>
</java>
```

P.2.4 xml: `LabTests84.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.4.1_01" class="java.beans.XMLDecoder">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
```

```
  <void property="bounds">
   <object class="java.awt.Rectangle">
    <int>117</int>
    <int>57</int>
    <int>665</int>
    <int>572</int>
   </object>
  </void>
  <void property="area">
   <string>LAB</string>
  </void>
  <void property="childDataCollection">
   <object class="peg.utils.dataCollection">
    <void property="inputDateFormat">
     <string>yyyy-MM-dd HH:mm</string>
    </void>
    <void property="outputDateFormat">
     <string>yyyy-MM-dd HH:mm</string>
    </void>
   </object>
  </void>
  <void property="description">
   <string>Efficiency</string>
  </void>
  <void property="driver">
   <string>com.mysql.jdbc.Driver</string>
  </void>
  <void property="inputDateFormat">
   <string>yyyy-MM-dd HH:mm</string>
  </void>
  <void property="lineAttribute">
   <void property="mark">
    <int>51</int>
   </void>
   <void property="style">
    <int>2</int>
   </void>
  </void>
  <void property="movAvgSamples">
   <int>3</int>
  </void>
  <void property="name">
   <string>frame0</string>
  </void>
  <void property="nodeName">
   <string>Efficiency</string>
  </void>
  <void property="outputDateFormat">
   <string>yyyy-MM-dd HOUR</string>
  </void>
  <void property="outputFilter">
   <int>1</int>
  </void>
  <void property="outputSource">
   <int>1</int>
  </void>
  <void property="protocol">
   <string>jdbc:mysql</string>
  </void>
  <void property="query">
   <string>SELECT date, ((AW+BW+CW)/1000) FROM checkmeters WHERE date &gt; &apos;2004-08-09&apos;;</string>
  </void>
  <void property="unit">
   <string>kWh</string>
  </void>
  <void property="url">
   <string>//localhost:3306/smsdb?user=server</string>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>269</int>
      <int>69</int>
      <int>612</int>
```

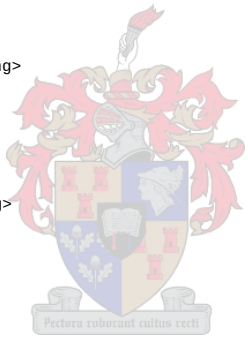
```
    <int>550</int>
   </object>
  </void>
  <void property="area">
   <string>LAB</string>
  </void>
  <void property="childDataCollection">
   <object class="peg.utils.dataCollection">
    <void property="inputDateFormat">
     <string>yyyy-MM-dd HH:mm</string>
    </void>
    <void property="outputDateFormat">
     <string>yyyy-MM-dd HOUR</string>
    </void>
   </object>
  </void>
  <void property="description">
   <string>Load</string>
  </void>
  <void property="driver">
   <string>com.mysql.jdbc.Driver</string>
  </void>
  <void property="inputDateFormat">
   <string>yyyy-MM-dd HH:mm</string>
  </void>
  <void property="lineAttribute">
   <void property="mark">
    <int>48</int>
   </void>
   <void property="style">
    <int>5</int>
   </void>
  </void>
  <void property="movAvgSamples">
   <int>3</int>
  </void>
  <void property="name">
   <string>frame2</string>
  </void>
  <void property="nodeName">
   <string>Load</string>
  </void>
  <void property="outputDateFormat">
   <string>yyyy-MM-dd HOUR</string>
  </void>
  <void property="outputSource">
   <int>1</int>
  </void>
  <void property="protocol">
   <string>jdbc:mysql</string>
  </void>
  <void property="query">
   <string>SELECT date, ((AW+BW+CW)/1000) FROM checkmeters WHERE (date &gt; &apos;2004-08-12 00:00&apos; AND date &lt; &apos;2004-08-12 23
  </void>
  <void property="unit">
   <string>kWh</string>
  </void>
  <void property="update">
   <boolean>true</boolean>
  </void>
  <void property="url">
   <string>//localhost:3306/smsdb?user=server</string>
  </void>
 </object>
</void>
<void method="add">
 <object class="javax.swing.tree.DefaultMutableTreeNode">
  <void property="userObject">
   <object class="peg.swing.myNode">
    <void property="bounds">
     <object class="java.awt.Rectangle">
      <int>152</int>
      <int>59</int>
      <int>705</int>
      <int>588</int>
     </object>
    </void>
    <void property="area">
     <string>LAB</string>
```

```
     </void>
     <void property="description">
      <string>Load 2004-08-10</string>
     </void>
     <void property="differentiate">
      <boolean>true</boolean>
     </void>
     <void property="driver">
      <string>com.mysql.jdbc.Driver</string>
     </void>
     <void property="inputDateFormat">
      <string>yyyy-MM-dd HH:mm</string>
     </void>
     <void property="name">
      <string>frame3</string>
     </void>
     <void property="nodeName">
      <string>Load</string>
     </void>
     <void property="outputDateFormat">
      <string>yyyy-MM-dd HOUR</string>
     </void>
     <void property="protocol">
      <string>jdbc:mysql</string>
     </void>
     <void property="query">
      <string>SELECT download, sum((reading)/10) FROM palmhht WHERE (download &gt; &apos;2004-08-10 00:00&apos; AND download &lt; &apos;200
     </void>
     <void property="unit">
      <string>kWh</string>
     </void>
     <void property="update">
      <boolean>true</boolean>
     </void>
     <void property="url">
      <string>//localhost:3306/palmdb?user=server</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>138</int>
       <int>52</int>
       <int>668</int>
       <int>586</int>
      </object>
     </void>
     <void property="area">
      <string>LAB</string>
     </void>
     <void property="description">
      <string>Load 2004-08-12</string>
     </void>
     <void property="differentiate">
      <boolean>true</boolean>
     </void>
     <void property="driver">
      <string>com.mysql.jdbc.Driver</string>
     </void>
     <void property="inputDateFormat">
      <string>yyyy-MM-dd HH:mm</string>
     </void>
     <void property="name">
      <string>frame5</string>
     </void>
     <void property="nodeName">
      <string>Load</string>
     </void>
     <void property="outputDateFormat">
      <string>yyyy-MM-dd HOUR</string>
     </void>
     <void property="outputSource">
      <int>2</int>
```

```
       </void>
       <void property="protocol">
        <string>jdbc:mysql</string>
       </void>
       <void property="query">
        <string>SELECT download, sum((reading)/10) FROM palmhht WHERE (download &gt; &apos;2004-08-12 00:00&apos; AND download &lt; &apos;200
       </void>
       <void property="unit">
        <string>kWh</string>
       </void>
       <void property="update">
        <boolean>true</boolean>
       </void>
       <void property="url">
        <string>//localhost:3306/palmdb?user=server</string>
       </void>
      </object>
     </void>
    </object>
   </void>
   <void method="add">
    <object class="javax.swing.tree.DefaultMutableTreeNode">
     <void property="userObject">
      <object class="peg.swing.myNode">
       <void property="size">
        <object class="java.awt.Dimension">
         <int>695</int>
         <int>539</int>
        </object>
       </void>
       <void property="name">
        <string>frame5</string>
       </void>
      </object>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>178</int>
       <int>105</int>
       <int>688</int>
       <int>589</int>
      </object>
     </void>
     <void property="area">
      <string>LAB</string>
     </void>
     <void property="childDataCollection">
      <object class="peg.utils.dataCollection">
       <void property="inputDateFormat">
        <string>yyyy-MM-dd HH:mm</string>
       </void>
       <void property="outputDateFormat">
        <string>yyyy-MM-dd HOUR</string>
       </void>
      </object>
     </void>
     <void property="description">
      <string>Source</string>
     </void>
     <void property="driver">
      <string>com.mysql.jdbc.Driver</string>
     </void>
     <void property="function">
      <int>3</int>
     </void>
     <void property="inputDateFormat">
      <string>yyyy-MM-dd HH:mm</string>
     </void>
     <void property="lineAttribute">
      <void property="color">
       <object class="java.awt.Color">
```

```
      <int>0</int>
      <int>0</int>
      <int>255</int>
      <int>255</int>
     </object>
    </void>
    <void property="mark">
     <int>51</int>
    </void>
    <void property="style">
     <int>5</int>
    </void>
   </void>
   <void property="movAvgSamples">
    <int>3</int>
   </void>
   <void property="name">
    <string>frame7</string>
   </void>
   <void property="nodeName">
    <string>Check Meter</string>
   </void>
   <void property="outputDateFormat">
    <string>yyyy-MM-dd HOUR</string>
   </void>
   <void property="outputSource">
    <int>1</int>
   </void>
   <void property="protocol">
    <string>jdbc:mysql</string>
   </void>
   <void property="query">
    <string>SELECT date, ((AW+BW+CW)/1000*1.13) FROM checkmeters WHERE date &gt; &apos;2004-08-09&apos;;</string>
   </void>
   <void property="unit">
    <string>kWh</string>
   </void>
   <void property="url">
    <string>//localhost:3306/smsdb?user=server</string>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>165</int>
       <int>63</int>
       <int>753</int>
       <int>598</int>
      </object>
     </void>
     <void property="area">
      <string>LAB</string>
     </void>
     <void property="description">
      <string>Source Data 2004-08-10</string>
     </void>
     <void property="differentiate">
      <boolean>true</boolean>
     </void>
     <void property="driver">
      <string>com.mysql.jdbc.Driver</string>
     </void>
     <void property="inputDateFormat">
      <string>yyyy-MM-dd HH:mm</string>
     </void>
     <void property="name">
      <string>frame4</string>
     </void>
     <void property="nodeName">
      <string>Check Meter Data</string>
     </void>
     <void property="outputDateFormat">
      <string>yyyy-MM-dd HOUR</string>
     </void>
     <void property="protocol">
```

```
      <string>jdbc:mysql</string>
     </void>
     <void property="query">
      <string>SELECT date, ((AW+BW+CW)/1000) FROM checkmeters WHERE (date &gt; &apos;2004-08-10 00:00&apos; AND date &lt; &apos;2004-08-10
     </void>
     <void property="unit">
      <string>kWh</string>
     </void>
     <void property="update">
      <boolean>true</boolean>
     </void>
     <void property="url">
      <string>//localhost:3306/smsdb?user=server</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
    <object class="peg.swing.myNode">
     <void property="bounds">
      <object class="java.awt.Rectangle">
       <int>1172</int>
       <int>117</int>
       <int>728</int>
       <int>598</int>
      </object>
     </void>
     <void property="area">
      <string>LAB</string>
     </void>
     <void property="description">
      <string>Source Data 2004-08-12</string>
     </void>
     <void property="differentiate">
      <boolean>true</boolean>
     </void>
     <void property="driver">
      <string>com.mysql.jdbc.Driver</string>
     </void>
     <void property="inputDateFormat">
      <string>yyyy-MM-dd HH:mm</string>
     </void>
     <void property="name">
      <string>frame9</string>
     </void>
     <void property="nodeName">
      <string>Check Meter Data</string>
     </void>
     <void property="outputDateFormat">
      <string>yyyy-MM-dd HOUR</string>
     </void>
     <void property="outputSource">
      <int>2</int>
     </void>
     <void property="protocol">
      <string>jdbc:mysql</string>
     </void>
     <void property="query">
      <string>SELECT date, ((AW+BW+CW)/1000) FROM checkmeters WHERE (date &gt; &apos;2004-08-12 00:00&apos; AND date &lt; &apos;2004-08-12
     </void>
     <void property="unit">
      <string>kWh</string>
     </void>
     <void property="update">
      <boolean>true</boolean>
     </void>
     <void property="url">
      <string>//localhost:3306/smsdb?user=server</string>
     </void>
    </object>
   </void>
  </object>
 </void>
 <void method="add">
  <object class="javax.swing.tree.DefaultMutableTreeNode">
   <void property="userObject">
```



```
      <object class="peg.swing.myNode">
       <void property="bounds">
        <object class="java.awt.Rectangle">
         <int>219</int>
         <int>70</int>
         <int>679</int>
         <int>596</int>
        </object>
       </void>
       <void property="area">
        <string>LAB</string>
       </void>
       <void property="description">
        <string>Source DAta 2004-08-26</string>
       </void>
       <void property="differentiate">
        <boolean>true</boolean>
       </void>
       <void property="driver">
        <string>com.mysql.jdbc.Driver</string>
       </void>
       <void property="inputDateFormat">
        <string>yyyy-MM-dd HH:mm</string>
       </void>
       <void property="name">
        <string>frame8</string>
       </void>
       <void property="nodeName">
        <string>Check Meter Data</string>
       </void>
       <void property="outputDateFormat">
        <string>yyyy-MM-dd HOUR MINUTE</string>
       </void>
       <void property="protocol">
        <string>jdbc:mysql</string>
       </void>
       <void property="query">
        <string>SELECT date, ((AW+BW+CW)/1000*0.9715) FROM checkmeters WHERE (date &gt; &apos;2004-08-26 00:00&apos; AND date &lt; &apos;2004
       </void>
       <void property="unit">
        <string>kWh</string>
       </void>
       <void property="update">
        <boolean>true</boolean>
       </void>
       <void property="url">
        <string>//localhost:3306/smsdb?user=server</string>
       </void>
      </object>
     </void>
    </object>
   </void>
  </object>
 </void>
</object>
</java>
```

Appendix Q

Java Code

The code are sorted according to the packages. The code were generated in Netbeans 3.5.1, and can be compiled with `javadoc` to generate documentation complient with the Java Doc specification. The original source code is shown here with the precompiled comments due to the document compiler only include public declared implementations. I found that [52] was a great help during the coding process.

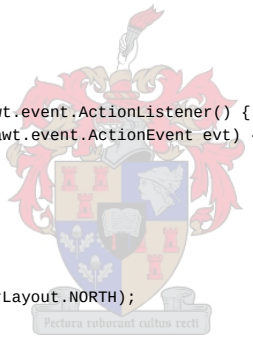
Q.1 Package: `DataMinor`

Q.1.1 Object: `DataMinor.myTree`

```
/*
 * myTree.java
 *
 * Created on November 5, 2003, 1:46 PM
 */

package DataMinor;

import java.io.*;
import java.beans.*;

import javax.swing.*;
import javax.swing.tree.*;
import java.awt.*;

import peg.sql.*;
import peg.utils.*;
import peg.swing.*;

import gov.noaa.pmel.sgt.*;
import gov.noaa.pmel.sgt.dm.*;
import gov.noaa.pmel.sgt.swing.*;
/** MyTree ties all the different nodes together using a tree structure.
 * By using the shortcut icons the contents of the nodes can be displayed.
 * The idea is to make this GUI class as interactive as possible.
 * @author Riaan Doorduin
 */
public class myTree extends JFrame {
    private DefaultMutableTreeNode rootNode = new DefaultMutableTreeNode(new myNode());
    private DefaultTreeModel treeModel = new DefaultTreeModel(rootNode);
    private myNode copyNode = null;
    private debug d = new debug("myTree",0);
    private GraphDisplay graphDisplay = null;
    final JFileChooser fc = new JFileChooser();   //For opening various setups

    /** Creates new form myTree */
    public myTree() {
```

```
        d.print(1,"myTree():InitComponents...");
        initComponents();
        d.print(1,"myTree():LoadTree...");
        LoadTree();
        d.print(1,"myTree():setExtendedState...");
        this.setExtendedState(6);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jPanel1 = new javax.swing.JPanel();
        jSplitPane1 = new javax.swing.JSplitPane();
        jPanel3 = new javax.swing.JPanel();
        jToolBar1 = new javax.swing.JToolBar();
        btnAdd = new javax.swing.JButton();
        btnRemove = new javax.swing.JButton();
        btnShow = new javax.swing.JButton();
        btnTable = new javax.swing.JButton();
        btnPlot = new javax.swing.JButton();
        btnCopy = new javax.swing.JButton();
        btnPaste = new javax.swing.JButton();
        jToolBar3 = new javax.swing.JToolBar();
        btnCalc = new javax.swing.JButton();
        lblIterations = new javax.swing.JLabel();
        txtIterations = new javax.swing.JTextField();
        btnHystogram = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        tree = new javax.swing.JTree(treeModel);
        tree.setEditable(false);
        tree.getSelectionModel().setSelectionMode(TreeSelectionModel.SINGLE_TREE_SELECTION);
        tree.setShowsRootHandles(true);
        jPanel4 = new javax.swing.JPanel();
        jScrollPane2 = new javax.swing.JScrollPane();
        jToolBar2 = new javax.swing.JToolBar();
        jLabel1 = new javax.swing.JLabel();
        lblStatus = new javax.swing.JLabel();
        jMenuBar1 = new javax.swing.JMenuBar();
        mnuMain = new javax.swing.JMenu();
        mnuNew = new javax.swing.JMenuItem();
        mnuSave = new javax.swing.JMenuItem();
        mnuLoad = new javax.swing.JMenuItem();
        jSeparator1 = new javax.swing.JSeparator();
        mnuExit = new javax.swing.JMenuItem();

        setTitle("Data Minor");
        setExtendedState(6);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });

        jPanel1.setLayout(new java.awt.BorderLayout());

        jPanel3.setLayout(new java.awt.BorderLayout());

        btnAdd.setText("Add");
        btnAdd.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnAddActionPerformed(evt);
            }
        });

        jToolBar1.add(btnAdd);

        btnRemove.setText("Remove");
        btnRemove.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnRemoveActionPerformed(evt);
            }
        });

        jToolBar1.add(btnRemove);

        btnShow.setText("Open");
```

```
btnShow.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnShowActionPerformed(evt);
    }
});

jToolBar1.add(btnShow);

btnTable.setText("Table");
btnTable.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnTableActionPerformed(evt);
    }
});

jToolBar1.add(btnTable);

btnPlot.setText("Plot");
btnPlot.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnPlotActionPerformed(evt);
    }
});

jToolBar1.add(btnPlot);

btnCopy.setText("Copy");
btnCopy.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnCopyActionPerformed(evt);
    }
});

jToolBar1.add(btnCopy);

btnPaste.setText("Paste");
btnPaste.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnPasteActionPerformed(evt);
    }
});

jToolBar1.add(btnPaste);

jPanel3.add(jToolBar1, java.awt.BorderLayout.NORTH);

btnCalc.setText("Calculate");
btnCalc.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnCalcActionPerformed(evt);
    }
});

jToolBar3.add(btnCalc);

lblIterations.setText("Itt:");
jToolBar3.add(lblIterations);

txtIterations.setText("1");
txtIterations.setToolTipText("Number of iterations to run through when calcuating");
jToolBar3.add(txtIterations);

btnHystogram.setText("Show Histogram");
btnHystogram.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnHystogramActionPerformed(evt);
    }
});

jToolBar3.add(btnHystogram);

jPanel3.add(jToolBar3, java.awt.BorderLayout.SOUTH);

tree.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        treeMouseClicked(evt);
    }
});
```



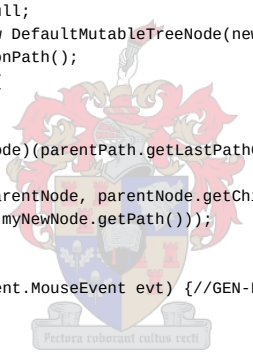
```
        jScrollPane1.setViewportView(tree);

        jPanel3.add(jScrollPane1, java.awt.BorderLayout.CENTER);

        jSplitPane1.setLeftComponent(jPanel3);

        jPanel4.setLayout(new java.awt.BorderLayout());

        jPanel4.add(jScrollPane2, java.awt.BorderLayout.CENTER);

        jSplitPane1.setRightComponent(jPanel4);

        jPanel1.add(jSplitPane1, java.awt.BorderLayout.CENTER);

        jLabel1.setText("Status:");
        jToolBar2.add(jLabel1);

        jToolBar2.add(lblStatus);

        jPanel1.add(jToolBar2, java.awt.BorderLayout.SOUTH);

        getContentPane().add(jPanel1, java.awt.BorderLayout.CENTER);

        mnuMain.setText("Main");
        mnuNew.setText("New");
        mnuNew.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnuNewActionPerformed(evt);
            }
        });

        mnuMain.add(mnuNew);

        mnuSave.setText("Save");
        mnuSave.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnuSaveActionPerformed(evt);
            }
        });

        mnuMain.add(mnuSave);

        mnuLoad.setText("Load");
        mnuLoad.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnuLoadActionPerformed(evt);
            }
        });

        mnuMain.add(mnuLoad);

        mnuMain.add(jSeparator1);

        mnuExit.setText("Exit");
        mnuExit.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnuExitActionPerformed(evt);
            }
        });

        mnuMain.add(mnuExit);

        jMenuBar1.add(mnuMain);

        setJMenuBar(jMenuBar1);

        pack();
    }//GEN-END:initComponents

    private void mnuNewActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuNewActionPerformed
        rootNode = new DefaultMutableTreeNode(new myNode());
        treeModel = new DefaultTreeModel(rootNode);
        tree.setModel(treeModel);
    }//GEN-LAST:event_mnuNewActionPerformed

    private void btnHystogramActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnHystogramActionPerformed
        d.print(3,"*btnPlotActionPerformed:...");
        DefaultMutableTreeNode parentNode = null;
        myNode myNewNode = null;
```

```
        hystogram myHG = null;
        SimpleLine mySL = null;
        TreePath parentPath = tree.getSelectionPath();

        //Get selected node
        if (tree.getSelectionPath() == null) {
            parentNode = rootNode;
        } else {
            parentNode = (DefaultMutableTreeNode)(parentPath.getLastPathComponent());
        };
        myNewNode = (myNode)(parentNode.getUserObject());

        graphDisplay = new GraphDisplay(false);
        jScrollPane2.setViewportView(graphDisplay);

        //Extract data of this node to plot
       myHG = myNewNode.getHystogram();
        if (myHG!=null) {
            myHG.print();
            graphDisplay.setTitles(myNewNode.getNodeName(),myNewNode.getArea(),myNewNode.getDescription());
            mySL = myHG.toSimpleLine();
            if (mySL != null) {
                mySL.setXMetaData(new SGTMetaData(myNewNode.getDescription(),myNewNode.getUnit(),false,false));
                mySL.setYMetaData(new SGTMetaData("Frequency","Hits",false,false));
                graphDisplay.addData(mySL,"",myNewNode.getLineAttribute());
            }
        }
        mySL = null;
        myHG = null;
        tree.updateUI();
        System.gc();
    }//GEN-LAST:event_btnHystogramActionPerformed

    private void btnCalcActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnCalcActionPerformed
        d.print(3,"*btnCalcActionPerformed://Calculate Children Functions...");
        //Node selection
        DefaultMutableTreeNode parentNode = null;
        TreePath parentPath = tree.getSelectionPath();
        if (tree.getSelectionPath() == null) {
            parentNode = rootNode;
        } else {
            parentNode = (DefaultMutableTreeNode)(parentPath.getLastPathComponent());
        };

        //Iterations
        int itt = (new java.math.BigInteger(txtIterations.getText())).intValue();
        for (int i = 0; i < itt; i++) {
           ((myNode)(rootNode.getUserObject())).setChildDataCollection(getChildrenDataCollection(parentNode)); //Calculates all the children f
           System.out.println("Itteration no:"+i);
           lblStatus.setText(""+i);
        }
        d.print(3,"btnCalcActionPerformed://tree.updateUI...");
        tree.updateUI(); //updates the tree structure to show calculations
    }//GEN-LAST:event_btnCalcActionPerformed
    /* Paste the cloned node as a child of the current selected node */
    private void btnPasteActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnPasteActionPerformed
        d.print(3,"*btnPasteActionPerformed:...");
        DefaultMutableTreeNode parentNode = null;
        DefaultMutableTreeNode myNewNode = new DefaultMutableTreeNode(copyNode);
        TreePath parentPath = tree.getSelectionPath();
        //Get selected node
        if (tree.getSelectionPath() == null) {
            parentNode = rootNode;
        } else {
            parentNode = (DefaultMutableTreeNode)(parentPath.getLastPathComponent());
        };
        treeModel.insertNodeInto(myNewNode, parentNode, parentNode.getChildCount());
        tree.scrollPathToVisible(new TreePath(myNewNode.getPath()));
    }//GEN-LAST:event_btnPasteActionPerformed
    /* Copies the current selected node */
    private void btnCopyActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnCopyActionPerformed
        d.print(3,"*btnCopyActionPerformed:...");
        DefaultMutableTreeNode parentNode = null;
        TreePath parentPath = tree.getSelectionPath();

        //Get selected node
        if (tree.getSelectionPath() == null) {
            parentNode = rootNode;
            System.out.println("ParentNode");
```

```
        } else {
            parentNode = (DefaultMutableTreeNode)(parentPath.getLastPathComponent());
        };

        copyNode = ((myNode)parentNode.getUserObject());
        copyNode = new myNode(copyNode);
    }//GEN-LAST:event_btnCopyActionPerformed
    /* Plots the current node */
    private void btnPlotActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnPlotActionPerformed
        d.print(3,"*btnPlotActionPerformed:...");
        DefaultMutableTreeNode parentNode = null;
        myNode myNewNode = null;
        dataCollection myDC = null;
        dataCollection addDC = null;
        SimpleLine mySL = null;
        GraphDisplay graphDisplay = null;
        TreePath parentPath = tree.getSelectionPath();
        System.gc();

        //Get selected node
        if (tree.getSelectionPath() == null) {
            parentNode = rootNode;
        } else {
            parentNode = (DefaultMutableTreeNode)(parentPath.getLastPathComponent());
        }
        myNewNode = (myNode)(parentNode.getUserObject());

        //Calculate Children Nodes
        myNewNode.setChildDataCollection(getChildrenDataCollection(parentNode));
        //Setup Graph environment
        graphDisplay = new GraphDisplay();
        graphDisplay.setTitles(myNewNode.getNodeName(),myNewNode.getArea(),myNewNode.getDescription());
        jScrollPane2.setViewportView(graphDisplay);

        //Extract data of this node to plot
        myDC = myNewNode.toDataCollection();
        if (myDC!=null) {
            mySL = myDC.toSimpleLine();
            if (mySL != null) {
                mySL.setXMetaData(new SGTMetaData("Time","Months",false,false));
                mySL.setYMetaData(new SGTMetaData(myNewNode.getDescription(),myNewNode.getUnit(),false,false));
                graphDisplay.addData(myDC.toSimpleLine(),myNewNode.getDescription(),myNewNode.getLineAttribute());
            }
        }

        //Plots additional data depening on the nodes sorce selection
        if (myNewNode.getOutputSource() != myNewNode.SOURCE_CHILDREN) {
            myDC = myNewNode.getChildDataCollection();
            if (myDC!=null) {
                mySL = myDC.toSimpleLine();
                if (mySL != null) {
                    mySL.setXMetaData(new SGTMetaData("Time","Months",false,false));
                    mySL.setYMetaData(new SGTMetaData(myNewNode.getDescription(),myNewNode.getUnit(),false,false));
                    graphDisplay.addData(myDC.toSimpleLine(),"(Children Output)"+myNewNode.getDescription());
                }
            }
        }
        if ((myNewNode.getOutputSource() == myNewNode.SOURCE_DB_CHILDREN) & (myNewNode.getOutputSource() == myNewNode.SOURCE_CHILDREN_DB)){
            myDC = myNewNode.getDBDataCollection();
            if (myDC!=null) {
                mySL = myDC.toSimpleLine();
                if (mySL != null) {
                    mySL.setXMetaData(new SGTMetaData("Time","Months",false,false));
                    mySL.setYMetaData(new SGTMetaData(myNewNode.getDescription(),myNewNode.getUnit(),false,false));
                    graphDisplay.addData(myDC.toSimpleLine(),"(DB Output)"+myNewNode.getDescription());
                }
            }
        }


        //Plot the direct children only
        if (parentNode.getChildCount() != 0) {
            for (int i = 0; i < parentNode.getChildCount(); i++) {
                myNewNode = (myNode)(((DefaultMutableTreeNode)(parentNode.getChildAt(i))).getUserObject());
                myDC = myNewNode.toDataCollection();
                if (myDC != null) {
                    mySL = myDC.toSimpleLine();
                    if (mySL != null) {
                        mySL.setXMetaData(new SGTMetaData("Time","Months",false,false));
```

```
                    mySL.setYMetaData(new SGTMetaData(myNewNode.getDescription(),myNewNode.getUnit(),false,false));
                    graphDisplay.addData(mySL,"(Child)"+myNewNode.getDescription(),myNewNode.getLineAttribute());
                }
            }
        }
    }
    tree.updateUI();
}//GEN-LAST:event_btnPlotActionPerformed

private void btnTableActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnTableActionPerformed
    d.print(3,"*btnTableActionPerformed:...");
    DefaultMutableTreeNode parentNode = null;
    myNode myNewNode = null;
    DBConnect myDB = null;
    TreePath parentPath = tree.getSelectionPath();

    //Get selected node
    if (tree.getSelectionPath() == null) {
        parentNode = rootNode;
    } else {
        parentNode = (DefaultMutableTreeNode)(parentPath.getLastPathComponent());
    };
    myNewNode = (myNode)(parentNode.getUserObject());
    myDB = myNewNode.getDataBaseData();
    if (!myDB.isQuerySuccess()) {
      myDB.executeSQL();
      if (myDB.getErrorString() != null)
        System.out.println("Error" + myDB.getErrorString());
    }
    if (myDB.isQuerySuccess()) {
        System.out.println("GotData");
        //jScrollPane2.setViewportView(new TableDisplay(myDB));
        jScrollPane2.setViewportView(new JTable(myNewNode.toTableModel()));
    } else {
        jScrollPane2.setViewportView(null);
    }
    System.out.flush();
}//GEN-LAST:event_btnTableActionPerformed

private void mnuExitActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuExitActionPerformed
    d.print(3,"*mnuExitActionPerformed:...");
    SaveTree();
    System.exit(0);
}//GEN-LAST:event_mnuExitActionPerformed

private void mnuLoadActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuLoadActionPerformed
    d.print(3,"mnuLoadActionPerformed:...");
    // Add your handling code here:
    int returnVal = fc.showOpenDialog(this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
        File file = fc.getSelectedFile();
        //this is where a real application would open the file.
        String fname = file.getAbsolutePath();
        d.print(3,"mnuLoadActionPerformed:File selected:"+fname);
        LoadTree(fname);
    } else {
        d.print(3,"mnuLoadActionPerformed:no File selected... e.g. canceld");
    }
}//GEN-LAST:event_mnuLoadActionPerformed

private void mnuSaveActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuSaveActionPerformed
    d.print(3,"*mnuSaveActionPerformed:...");
    int returnVal = fc.showSaveDialog(this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
        File file = fc.getSelectedFile();
        //this is where a real application would open the file.
        d.print(3,"mnuLoadActionPerformed:File selected:"+file.getName());
        String fname = file.getAbsolutePath();
        SaveTree(fname);
    } else {
        d.print(3,"mnuLoadActionPerformed:no File selected... e.g. canceld");
    }
}//GEN-LAST:event_mnuSaveActionPerformed
/* Removes the selected node */
private void btnRemoveActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnRemoveActionPerformed
    d.print(3,"*btnRemoveActionPerformed:...");
    TreePath currentSelection = tree.getSelectionPath();
    if (currentSelection != null) {
        DefaultMutableTreeNode currentNode = (DefaultMutableTreeNode)(currentSelection.getLastPathComponent());
```

```
        MutableTreeNode parent = (MutableTreeNode)(currentNode.getParent());
        if (parent != null) {
            treeModel.removeNodeFromParent(currentNode);
        }
    };
}//GEN-LAST:event_btnRemoveActionPerformed
/* Shows the current selected node, enabling the user to update the nodes data */
private void btnShowActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnShowActionPerformed
    d.print(3,"*btnShowActionPerformed:...");
    DefaultMutableTreeNode parentNode = null;
    myNode myNewNode = null;
    TreePath parentPath = tree.getSelectionPath();

    //Get selected node
    if (tree.getSelectionPath() == null) {
        parentNode = rootNode;
    } else {
        parentNode = (DefaultMutableTreeNode)(parentPath.getLastPathComponent());
    };
    myNewNode = (myNode)(parentNode.getUserObject());
    if (myNewNode != null) {
        myNewNode.show();
    }
    tree.updateUI();
    tree.scrollPathToVisible(new TreePath(parentNode.getPath()));
    tree.doLayout();
}//GEN-LAST:event_btnShowActionPerformed

private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnAddActionPerformed
    d.print(3,"*btnAddActionPerformed:...");
    // Add your handling code here:
    DefaultMutableTreeNode parentNode = null;
    DefaultMutableTreeNode myNewNode = new DefaultMutableTreeNode(new myNode());
    TreePath parentPath = tree.getSelectionPath();
    if (tree.getSelectionPath() == null) {
        parentNode = rootNode;
    } else {
        parentNode = (DefaultMutableTreeNode)(parentPath.getLastPathComponent());
    };
    treeModel.insertNodeInto(myNewNode, parentNode, parentNode.getChildCount());
    tree.scrollPathToVisible(new TreePath(myNewNode.getPath()));
}//GEN-LAST:event_btnAddActionPerformed

private void treeMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event_treeMouseClicked
    d.print(3,"*treeMouseClicked:...");
}//GEN-LAST:event_treeMouseClicked

/** Exit the Application */
private void exitForm(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_exitForm
    d.print(3,"*exitForm:...");
    SaveTree();
    System.exit(0);
}//GEN-LAST:event_exitForm

/** Starts the tree interface program
 * @param args the command line arguments
 */
public static void main(String args[]) {
    new myTree().show();
}


// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JButton btnAdd;
private javax.swing.JButton btnCalc;
private javax.swing.JButton btnCopy;
private javax.swing.JButton btnHystogram;
private javax.swing.JButton btnPaste;
private javax.swing.JButton btnPlot;
private javax.swing.JButton btnRemove;
private javax.swing.JButton btnShow;
private javax.swing.JButton btnTable;
private javax.swing.JLabel jLabel1;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel3;
private javax.swing.JPanel jPanel4;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JScrollPane jScrollPane2;
```



```
private javax.swing.JSeparator jSeparator1;
private javax.swing.JSplitPane jSplitPane1;
private javax.swing.JToolBar jToolBar1;
private javax.swing.JToolBar jToolBar2;
private javax.swing.JToolBar jToolBar3;
private javax.swing.JLabel lblIterations;
private javax.swing.JLabel lblStatus;
private javax.swing.JMenuItem mnuExit;
private javax.swing.JMenuItem mnuLoad;
private javax.swing.JMenu mnuMain;
private javax.swing.JMenuItem mnuNew;
private javax.swing.JMenuItem mnuSave;
private javax.swing.JTree tree;
private javax.swing.JTextField txtIterations;
// End of variables declaration//GEN-END:variables

//Tree help functions
private void SaveTree() {
    SaveTree("c:/treeInfo.xml");
}
private void SaveTree(String fname) {
    d.print(3,"SaveTree("+fname+"):...");
    try {
        XMLEncoder xmlE = new XMLEncoder(new BufferedOutputStream(new FileOutputStream(fname)));
        xmlE.writeObject(rootNode);
        xmlE.close();
        xmlE = new XMLEncoder(new BufferedOutputStream(new FileOutputStream("c:/temp/treeInfo.xml")));
        xmlE.writeObject(rootNode);
        xmlE.close();
        lblStatus.setText("Latest Tree structure stored!");
    } catch (Exception e) {
        lblStatus.setText("Error writing XMLfile: "+ e.getMessage());
      System.out.println("Error writing XMLfile: "+ e.getMessage());
    }
}

private void LoadTree() {
    LoadTree("c:/treeInfo.xml");
}
private void LoadTree(String fname) {
    d.print(3,"LoadTree("+fname+"):...");
    try {
      d.print(5,"LoadTree():XMLDecoder xmlD = new XMLDecoder(new...");
      XMLDecoder xmlD = new XMLDecoder(new BufferedInputStream(new FileInputStream(fname)));
      d.print(5,"LoadTree():roodNode = ...");
      rootNode = (DefaultMutableTreeNode)xmlD.readObject();
      d.print(5, "LoadTree():xmlD.Close();");
      xmlD.close();
      lblStatus.setText("Latest Tree structure Loaded!");
      d.print(5,"LoadTree():treeModel.setRoot(...");
      treeModel.setRoot(rootNode);
      d.print(5,"LoadTree():tree.setModel(...");
      tree.setModel(treeModel);
      d.print(3,"LoadTree:expandTree...");
      expandTree();
    } catch (Exception e) {
        lblStatus.setText("Error loading XMLfile: "+ e.getMessage());
        d.print(5,"LoadTree:Error loading XMLfile: "+ e.getMessage());
    }
}

/** Expand the whole tree structure */
public void expandTree() {
    d.print(3,"expandTree:...");
    int row =0;
    while (row < tree.getRowCount()) {
        if (tree.isCollapsed(row)) {
            tree.expandRow(row);
        }
        row++;
    }
}
/** This recursive method dives into all the nodes and calculates the child
 * functions. These outputs are stored in each parent node.
 * @param parentNode The parent node of the current node
 * @return the output data of this node
 */
public dataCollection getChildrenDataCollection(DefaultMutableTreeNode parentNode) {
    d.print(3,"getChildrenDataCollection:...");
```

```
        dataCollection tmpDC =null;
        dataCollection addDC = null;
        DefaultMutableTreeNode tmpChild = null;
        myNode tmpParentNode = null;
        myNode tmpChildNode = null;
        int mathFunction = 0;

        tmpParentNode = (myNode)(parentNode.getUserObject());
        d.print(5,"getChildrenDataCollection:tmpParentNode="+tmpParentNode.toString());
        d.print(5,"getChildrenDataCollection:getChildrenCount()="+parentNode.getChildCount());
        if (parentNode.getChildCount() != 0) {
            //Itterate through children
            for (int i = 0; i < parentNode.getChildCount(); i++) {
                tmpChild = (DefaultMutableTreeNode)(parentNode.getChildAt(i));
                d.print(5,"getChildrenDataCollection:tmpChild="+tmpChild.toString());
                if (tmpChild != null) {
                    tmpDC = getChildrenDataCollection(tmpChild);  //recursively evalutae children
                }
                tmpChildNode = ((myNode)tmpChild.getUserObject());
                if (tmpChildNode.toDataCollection() != null) {
                    tmpDC = tmpChildNode.toDataCollection().group(tmpParentNode.getInputDateFormat());
                } else {
                    tmpDC = null;
                }
                if (tmpDC != null) {
                    if (addDC == null) {
                        addDC = new dataCollection();
                        addDC.setInputDateFormat(tmpChildNode.getOutputDateFormat());
                        addDC.setOutputDateFormat(tmpParentNode.getInputDateFormat());
                        addDC = addDC.math(tmpDC,tmpChildNode.getFunction(),true);
                    } else {
                        addDC = addDC.math(tmpDC,tmpChildNode.getFunction());
                    }
                }
            }
            tmpDC = addDC;
            tmpParentNode.setChildDataCollection(tmpDC);
        } else {
            //Get the data from the bottom node
            d.print(5,"getChildrenDataCollection:BottomNode...");
            if (tmpParentNode != null) {
                tmpDC = tmpParentNode.toDataCollection();
                tmpParentNode.setChildDataCollection(null);
            }
        }
        return tmpDC;
    }

}
```

Q.2 Package: `energyMeasure`

Q.2.1 Object: `energyMeasure.clients`

```
/*
 * clients.java
 *
 * Created on 24 May 2004, 02:52
 */

package energyMeasure;

import javax.swing.table.AbstractTableModel;
import java.util.*;
import java.io.*;
import java.beans.*;
/**
 *
 * @author  RDoorduin
 */
public class clients extends AbstractTableModel {
    final String[] columnNames = {"Number",
                                  "Location",
                                  "Description",
```

```
                                "Minisub"};

    private static Vector clientsList = new Vector();  //Any instance of the class should refference this variable, therfor static declaration

    /** Creates a new instance of clients */
    public clients() {
        load();
    }

    public int getColumnCount() {
        return columnNames.length;
    }

    public int getRowCount() {
        return clientsList.size();
    }

    public String getColumnName(int col) {
        return columnNames[col];
    }

    public Object getValueAt(int rowIndex, int columnIndex) {
        clientsInfo cI = new clientsInfo();
        cI = (clientsInfo)clientsList.get(rowIndex);
        if (columnIndex == 0) {
            return cI.getNumber();
        } else if (columnIndex == 1) {
                return cI.getLocation();
        } else if (columnIndex == 2) {
                return cI.getDescription();
        } else if (columnIndex == 3) {
                return cI.getMinisub();
        } else {
            return new String("Test");
        }
    }

    public Class getColumnClass(int columnIndex) {
        clientsInfo cI = new clientsInfo();
        if (columnIndex == 0) {
            return cI.getNumber().getClass();
        } else if (columnIndex == 1) {
                return cI.getLocation().getClass();
        } else if (columnIndex == 2) {
                return cI.getDescription().getClass();
        } else if (columnIndex == 3) {
                return cI.getMinisub().getClass();
        } else {
            return new String("").getClass();
        }
    }

    public boolean isCellEditable(int row, int col) {
        return true;
    }

    public void setValueAt(Object value, int rowIndex, int columnIndex) {
        clientsInfo cI = new clientsInfo();
        cI = (clientsInfo)clientsList.get(rowIndex);
        if (columnIndex == 0) {
            cI.setNumber((String)value);
        } else if (columnIndex == 1) {
            cI.setLocation((String)value);
        } else if (columnIndex == 2) {
            cI.setDescription((String)value);
        } else if (columnIndex == 3) {
            cI.setMinisub((String)value);
        }
        clientsList.setElementAt(cI, rowIndex);
        save();
    }

    public static boolean addClient(clientsInfo cI) {
        clientsInfo thisCI = new clientsInfo();
        boolean duplicate = false;
        for (int i = 0;i < clientsList.size(); i++) {
            thisCI = (clientsInfo)clientsList.get(i);
            if (thisCI.getNumber().equalsIgnoreCase(cI.getNumber())) {
                duplicate = true;
```

```
            }
        }
        if (duplicate) {
            return false;
        } else {
            clientsList.add(cI);
            save();
            return true;
        }
    }

    public static void removeClient(int index) {
        clientsList.removeElementAt(index);
        save();
    }

    private static void save() {
        try {
            XMLEncoder xmlE = new XMLEncoder(new BufferedOutputStream(new FileOutputStream("c:/SMSClients.xml")));
            xmlE.writeObject(clientsList);
            xmlE.close();
        } catch (Exception e) {
          System.out.println("Error writing XMLfile: "+ e.getMessage());
        }
    }

    private static void load() {
        try {
          System.out.println("Loading SMSClients.xml...");
          XMLDecoder xmlD = new XMLDecoder(new BufferedInputStream(new FileInputStream("c:/SMSClients.xml")));
          clientsList = (Vector)xmlD.readObject();
          xmlD.close();
        } catch (Exception e) {
            System.out.println("Error loading XMLfile: "+ e.getMessage());
        }
    }
}
```

Q.2.2 Object: `energyMeasure.clientsInfo`

```
/*
 * clientsInfo.java
 *
 * Created on 21 May 2004, 10:39
 */

package energyMeasure;

/**
 *
 * @author  RDoorduin
 */
public class clientsInfo {
    private String number = "";
    private String location = "";
    private String description = "";
    private String minisub = "";

    /** Creates a new instance of clientsInfo */
    public clientsInfo() {
        number = "";
        location = "";
        description = "";
        minisub = "";
    }

    public void setNumber(String str) {
        number = str;
    }
    public String getNumber() {
        return number;
    }

    public void setLocation(String str) {
        location = str;
    }
    public String getLocation() {
```

```
        return location;
    }

    public void setDescription(String str) {
        description = str;
    }
    public String getDescription() {
        return description;
    }

    public void setMinisub(String str) {
        minisub = str;
    }
    public String getMinisub() {
        return minisub;
    }
}
```

Q.2.3 Object: `energyMeasure.energyCalibration`

```
/*
 * energyCallibration.java
 *
 * Created on 21 April 2004, 02:44
 */

package energyMeasure;

import java.lang.*;
import peg.io.*;
/**
 *
 * @author  RDoorduin
 */
public class energyCalibration {

    private int currentCallibrationChannel = 0x00;

    //Voltage and Current Offset Variables
    public int phaseAVoltageRMS1 = 0x0;
    public int phaseBVoltageRMS1 = 0x0;
    public int phaseCVoltageRMS1 = 0x0;
    public int phaseAVoltageRMS2 = 0x0;
    public int phaseBVoltageRMS2 = 0x0;
    public int phaseCVoltageRMS2 = 0x0;
    public int phaseACurrentRMS1 = 0x0;
    public int phaseBCurrentRMS1 = 0x0;
    public int phaseCCurrentRMS1 = 0x0;
    public int phaseACurrentRMS2 = 0x0;
    public int phaseBCurrentRMS2 = 0x0;
    public int phaseCCurrentRMS2 = 0x0;

    public double phaseAVoltageValue1 = 0.0;
    public double phaseBVoltageValue1 = 0.0;
    public double phaseCVoltageValue1 = 0.0;
    public double phaseAVoltageValue2 = 0.0;
    public double phaseBVoltageValue2 = 0.0;
    public double phaseCVoltageValue2 = 0.0;
    public double phaseACurrentValue1 = 0.0;
    public double phaseBCurrentValue1 = 0.0;
    public double phaseCCurrentValue1 = 0.0;
    public double phaseACurrentValue2 = 0.0;
    public double phaseBCurrentValue2 = 0.0;
    public double phaseCCurrentValue2 = 0.0;

    public double calAV_LSB = 1.0;
    public double calBV_LSB = 1.0;
    public double calCV_LSB = 1.0;
    public double calAI_LSB = 1.0;
    public double calBI_LSB = 1.0;
    public double calCI_LSB = 1.0;

    //Energy Callibration
    public int calAPCFDEN = 0x00;
    public int calVARCFDEN = 0x00;

    public int phaseAW_RMS = 0x00;
```

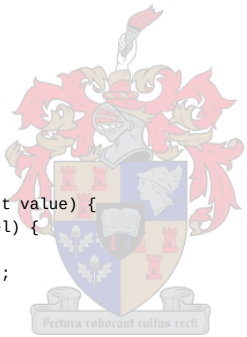
```
    public int phaseBW_RMS = 0x00;
    public int phaseCW_RMS = 0x00;
    public int phaseAVA_RMS = 0x00;
    public int phaseBVA_RMS = 0x00;
    public int phaseCVA_RMS = 0x00;
    public int phaseAVAR_RMS = 0x00;
    public int phaseBVAR_RMS = 0x00;
    public int phaseCVAR_RMS = 0x00;


    /*public int phaseAWRMS2 = 0x00;
    public int phaseBWRMS2 = 0x00;
    public int phaseCWRMS2 = 0x00;
    public int phaseAVARMS2 = 0x00;
    public int phaseBVARMS2 = 0x00;
    public int phaseCVARMS2 = 0x00;
    public int phaseAVARRMS2 = 0x00;
    public int phaseBVARRMS2 = 0x00;
    public int phaseCVARRMS2 = 0x00;*/

    public double phaseAW_Value = 0.0;
    public double phaseBW_Value = 0.0;
    public double phaseCW_Value = 0.0;
    public double phaseAVAR_Value = 0.0;
    public double phaseBVAR_Value = 0.0;
    public double phaseCVAR_Value = 0.0;
    /*public double phaseAWValue2 = 0.0;
    public double phaseBWValue2 = 0.0;
    public double phaseCWValue2 = 0.0;
    public double phaseAVARValue2 = 0.0;
    public double phaseBVARValue2 = 0.0;
    public double phaseCVARValue2 = 0.0;*/

    public int calLineCyc = 0x0;

    public int calWDiv = 0x0;
    public int calVADiv = 0x0;
    public int calVARDiv = 0x0;
/*
    public double calWhLSB = 1.0;
    public double calVAhLSB = 1.0;
    public double calVARhLSB = 1.0; */

    //Power FineTune Callibration Variables
    private int MCU_phaseAW_RMS = 0x00;
    private int MCU_phaseBW_RMS = 0x00;
    private int MCU_phaseCW_RMS = 0x00;
    private int MCU_phaseAVA_RMS = 0x00;
    private int MCU_phaseBVA_RMS = 0x00;
    private int MCU_phaseCVA_RMS = 0x00;
    private int MCU_phaseAVAR_RMS = 0x00;
    private int MCU_phaseBVAR_RMS = 0x00;
    private int MCU_phaseCVAR_RMS = 0x00;

    private double MCU_phaseAW_Value = 0.0;
    private double MCU_phaseBW_Value = 0.0;
    private double MCU_phaseCW_Value = 0.0;
    private double MCU_phaseAVAR_Value = 0.0;
    private double MCU_phaseBVAR_Value = 0.0;
    private double MCU_phaseCVAR_Value = 0.0;
    private double MCU_phaseAVA_Value = 0.0;
    private double MCU_phaseBVA_Value = 0.0;
    private double MCU_phaseCVA_Value = 0.0;

    public int MCU_phaseAW_DIV = 0;
    public int MCU_phaseBW_DIV = 0;
    public int MCU_phaseCW_DIV = 0;
    public int MCU_phaseAVAR_DIV = 0;
    public int MCU_phaseBVAR_DIV = 0;
    public int MCU_phaseCVAR_DIV = 0;
    public int MCU_phaseAVA_DIV = 0;
    public int MCU_phaseBVA_DIV = 0;
    public int MCU_phaseCVA_DIV = 0;

    //Frequency Callibration
    public double FreqValue = 0.0;
    public int FreqRMS = 0x00;
    public double calFreq = 1.0;
```

```
    //Phase shift Offset Callibration
    private double phaseCalPF1A = 1.0;
    private double phaseCalPF1B = 1.0;
    private double phaseCalPF1C = 1.0;
    private double phaseCalPF2A = 0.5;
    private double phaseCalPF2B = 0.5;
    private double phaseCalPF2C = 0.5;

    private double phaseCalW1A = 1.0;
    private double phaseCalW1B = 1.0;
    private double phaseCalW1C = 1.0;
    private double phaseCalW2A = 0.5;
    private double phaseCalW2B = 0.5;
    private double phaseCalW2C = 0.5;

    private int phaseCalWRMS1A = 0x0;
    private int phaseCalWRMS1B = 0x0;
    private int phaseCalWRMS1C = 0x0;
    private int phaseCalWRMS2A = 0x0;
    private int phaseCalWRMS2B = 0x0;
    private int phaseCalWRMS2C = 0x0;
    /** Creates a new instance of energyCallibration */
    public energyCalibration() {
    }

//Voltage and Current Offset Methods

    public void setPhaseAVoltageValue1(String strValue) {
        try {
            setPhaseAVoltageValue1(Double.valueOf(strValue).doubleValue());
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseAVoltageValue1(double tmpVal) {
           this.phaseAVoltageValue1 = tmpVal;
    }
    public double getPhaseAVoltageValue1() {
        return this.phaseAVoltageValue1;
    }

    public void setPhaseBVoltageValue1(String strValue) {
        try {
            setPhaseBVoltageValue1(Double.valueOf(strValue).doubleValue());
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseBVoltageValue1(double tmpVal) {
           this.phaseBVoltageValue1 = tmpVal;
    }
    public double getPhaseBVoltageValue1() {
        return this.phaseBVoltageValue1;
    }

    public void setPhaseCVoltageValue1(String strValue) {
        try {
            setPhaseCVoltageValue1(Double.valueOf(strValue).doubleValue());
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseCVoltageValue1(double tmpVal) {
           this.phaseCVoltageValue1 = tmpVal;
    }
    public double getPhaseCVoltageValue1() {
        return this.phaseCVoltageValue1;
    }

    public void setPhaseAVoltageValue2(String strValue) {
        try {
            setPhaseAVoltageValue2(Double.valueOf(strValue).doubleValue());
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseAVoltageValue2(double tmpVal) {
           this.phaseAVoltageValue2 = tmpVal;
    }
```

```
public double getPhaseAVoltageValue2() {
    return this.phaseAVoltageValue2;
}

public void setPhaseBVoltageValue2(String strValue) {
    try {
        setPhaseBVoltageValue2(Double.valueOf(strValue).doubleValue());
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseBVoltageValue2(double tmpVal) {
       this.phaseBVoltageValue2 = tmpVal;
}
public double getPhaseBVoltageValue2() {
    return this.phaseBVoltageValue2;
}

public void setPhaseCVoltageValue2(String strValue) {
    try {
        setPhaseCVoltageValue2(Double.valueOf(strValue).doubleValue());
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseCVoltageValue2(double tmpVal) {
       this.phaseCVoltageValue2 = tmpVal;
}
public double getPhaseCVoltageValue2() {
    return this.phaseCVoltageValue2;
}

public void setPhaseACurrentValue1(String strValue) {
    try {
        setPhaseACurrentValue1(Double.valueOf(strValue).doubleValue());
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseACurrentValue1(double tmpVal) {
       this.phaseACurrentValue1 = tmpVal;
}
public double getPhaseACurrentValue1() {
    return this.phaseACurrentValue1;
}

public void setPhaseBCurrentValue1(String strValue) {
    try {
        setPhaseBCurrentValue1(Double.valueOf(strValue).doubleValue());
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseBCurrentValue1(double tmpVal) {
       this.phaseBCurrentValue1 = tmpVal;
}
public double getPhaseBCurrentValue1() {
    return this.phaseBCurrentValue1;
}

public void setPhaseCCurrentValue1(String strValue) {
    try {
        setPhaseCCurrentValue1(Double.valueOf(strValue).doubleValue());
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseCCurrentValue1(double tmpVal) {
       this.phaseCCurrentValue1 = tmpVal;
}
public double getPhaseCCurrentValue1() {
    return this.phaseCCurrentValue1;
}

public void setPhaseACurrentValue2(String strValue) {
    try {
        setPhaseACurrentValue2(Double.valueOf(strValue).doubleValue());
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
```

```
    }
}
public void setPhaseACurrentValue2(double tmpVal) {
       this.phaseACurrentValue2 = tmpVal;
}
public double getPhaseACurrentValue2() {
    return this.phaseACurrentValue2;
}

public void setPhaseBCurrentValue2(String strValue) {
    try {
        setPhaseBCurrentValue2(Double.valueOf(strValue).doubleValue());
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseBCurrentValue2(double tmpVal) {
       this.phaseBCurrentValue2 = tmpVal;
}
public double getPhaseBCurrentValue2() {
    return this.phaseBCurrentValue2;
}

public void setPhaseCCurrentValue2(String strValue) {
    try {
        setPhaseCCurrentValue2(Double.valueOf(strValue).doubleValue());
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseCCurrentValue2(double tmpVal) {
       this.phaseCCurrentValue2 = tmpVal;
}
public double getPhaseCCurrentValue2() {
    return this.phaseCCurrentValue2;
}

public int getPhaseAVoltageRMS1() {
    return this.phaseAVoltageRMS1;
}
public int getPhaseBVoltageRMS1() {
    return this.phaseBVoltageRMS1;
}
public int getPhaseCVoltageRMS1() {
    return this.phaseCVoltageRMS1;
}
public int getPhaseAVoltageRMS2() {
    return this.phaseAVoltageRMS2;
}
public int getPhaseBVoltageRMS2() {
    return this.phaseBVoltageRMS2;
}
public int getPhaseCVoltageRMS2() {
    return this.phaseCVoltageRMS2;
}
public int getPhaseACurrentRMS1() {
    return this.phaseACurrentRMS1;
}
public int getPhaseBCurrentRMS1() {
    return this.phaseBCurrentRMS1;
}
public int getPhaseCCurrentRMS1() {
    return this.phaseCCurrentRMS1;
}
public int getPhaseACurrentRMS2() {
    return this.phaseACurrentRMS2;
}
public int getPhaseBCurrentRMS2() {
    return this.phaseBCurrentRMS2;
}
public int getPhaseCCurrentRMS2() {
    return this.phaseCCurrentRMS2;
}

public void setPhaseAVoltageRMS1(int tmpVal) {
    this.phaseAVoltageRMS1 = tmpVal;
}
public void setPhaseBVoltageRMS1(int tmpVal) {
    this.phaseBVoltageRMS1 = tmpVal;
```

```
}
public void setPhaseCVoltageRMS1(int tmpVal) {
    this.phaseCVoltageRMS1 = tmpVal;
}
public void setPhaseAVoltageRMS2(int tmpVal) {
    this.phaseAVoltageRMS2 = tmpVal;
}
public void setPhaseBVoltageRMS2(int tmpVal) {
    this.phaseBVoltageRMS2 = tmpVal;
}
public void setPhaseCVoltageRMS2(int tmpVal) {
    this.phaseCVoltageRMS2 = tmpVal;
}
public void setPhaseACurrentRMS1(int tmpVal) {
    this.phaseACurrentRMS1 = tmpVal;
}
public void setPhaseBCurrentRMS1(int tmpVal) {
    this.phaseBCurrentRMS1 = tmpVal;
}
public void setPhaseCCurrentRMS1(int tmpVal) {
    this.phaseCCurrentRMS1 = tmpVal;
}
public void setPhaseACurrentRMS2(int tmpVal) {
    this.phaseACurrentRMS2 = tmpVal;
}
public void setPhaseBCurrentRMS2(int tmpVal) {
    this.phaseBCurrentRMS2 = tmpVal;
}
public void setPhaseCCurrentRMS2(int tmpVal) {
    this.phaseCCurrentRMS2 = tmpVal;
}

public void setCurrentChannel(int cc) {
    currentCallibrationChannel = cc;
    switch (cc) {
        case protocol.VALUE_MCU_FREQ:
            calculateFreq();
            break;
    }
}
public void setCurrentCallibrationValue(int value) {
    switch (this.currentCallibrationChannel) {
        case protocol.VALUE_MCU_V1_A:
            this.phaseAVoltageRMS1 = value;
            break;
        case protocol.VALUE_MCU_V2_A:
            this.phaseAVoltageRMS2 = value;
            break;
        case protocol.VALUE_MCU_V1_B:
            this.phaseBVoltageRMS1 = value;
            break;
        case protocol.VALUE_MCU_V2_B:
            this.phaseBVoltageRMS2 = value;
            break;
        case protocol.VALUE_MCU_V1_C:
            this.phaseCVoltageRMS1 = value;
            break;
        case protocol.VALUE_MCU_V2_C:
            this.phaseCVoltageRMS2 = value;
            break;
        case protocol.VALUE_MCU_I1_A:
            this.phaseACurrentRMS1 = value;
            break;
        case protocol.VALUE_MCU_I2_A:
            this.phaseACurrentRMS2 = value;
            break;
        case protocol.VALUE_MCU_I1_B:
            this.phaseBCurrentRMS1 = value;
            break;
        case protocol.VALUE_MCU_I2_B:
            this.phaseBCurrentRMS2 = value;
            break;
        case protocol.VALUE_MCU_I1_C:
            this.phaseCCurrentRMS1 = value;
            break;
        case protocol.VALUE_MCU_I2_C:
            this.phaseCCurrentRMS2 = value;
            break;
    }
```



```
}

public int calculateVoltageOffset(double V1, double V2, int RMS1, int RMS2) {
    double tmpDouble;
    int tmpInt;
    tmpDouble = ((V1*RMS2) - (V2*RMS1))/(V2-V1);  //V2-V1 instead of V1-V2 to subtract instead of accumulate yet another offset.
    tmpDouble = tmpDouble / 64;
    tmpInt = (int)tmpDouble;
    if (tmpInt > 0x07FF) {
        tmpInt = 0x07FF;    //Maximum for 2's compliment 12bit register
    }
    if (tmpInt < (-0x07FF)) {
        tmpInt = -0x07FF;   //Minimum for 2's compliment 12bit register
    }
    System.out.println("V1  :" + V1);
    System.out.println("V2  :" + V2);
    System.out.println("RMS1:" + RMS1);
    System.out.println("RMS2:" + RMS2);
    System.out.println("Offset(doub): " + tmpDouble);
    System.out.println("Offset(int) : " + tmpInt);
    return  tmpInt;
}

public int calculateCurrentOffset(double I1, double I2, int RMS1, int RMS2) {
    double tmpDouble =0.0;
    int tmpInt;
    tmpDouble = ((I1*I1*RMS2*RMS2) - (I2*I2*RMS1*RMS1))/((I2*I2)-(I1*I1))/(16384);
    tmpInt = (int)tmpDouble;
    if (tmpInt > 0x07FF) {
        tmpInt = 0x07FF;
    }
    if (tmpInt < (-0x07FF)) {
        tmpInt = -0x07FF;
    }
    System.out.println("I1  :" + I1);
    System.out.println("I2  :" + I2);
    System.out.println("RMS1:" + RMS1);
    System.out.println("RMS2:" + RMS2);
    System.out.println("Offset(doub): " + tmpDouble);
    System.out.println("Offset(int) : " + tmpInt);
    return  tmpInt;
}

public double calculateVoltageCurrentLSB(double V1, int RMS1) {
    return (double)(V1/RMS1);
}

//Beans
public void setCalAV_LSB(double value) {
    calAV_LSB = value;
}
public double getCalAV_LSB() {
    return calAV_LSB;
}

public void setCalBV_LSB(double value) {
    calBV_LSB = value;
}
public double getCalBV_LSB() {
    return calBV_LSB;
}

public void setCalCV_LSB(double value){
    calCV_LSB = value;
}
public double getCalCV_LSB(){
    return calCV_LSB;
}

public double getCalAI_LSB() {
    return calAI_LSB;
}
public void setCalAI_LSB(double value) {
    calAI_LSB = value;
}

public double getCalBI_LSB() {
    return calBI_LSB;
}
```

```
    public void setCalBI_LSB(double value) {
        calBI_LSB = value;
    }

    public void setCalCI_LSB(double value) {
        calCI_LSB = value;
    }
    public double getCalCI_LSB() {
        return calCI_LSB;
    }

// CF and Meter Constant methods
    /*public int calculateMeterConstantDen(int MC, double Energy, double freq) {
        int tmpValue = (int)((freq*1000*3600)/(MC*Energy))*calAPCFDEN;
        if (tmpValue == 0) {
            return 63;
        } else {
            return tmpValue;
        }
    }*/

//Gain Callibration
    public void setPhaseAW_Value(String strValue) {
        try {
            this.phaseAW_Value = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseBW_Value(String strValue) {
        try {
            this.phaseBW_Value = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseCW_Value(String strValue) {
        try {
            this.phaseCW_Value = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseAVAR_Value(String strValue) {
        try {
            this.phaseAVAR_Value = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseBVAR_Value(String strValue) {
        try {
            this.phaseBVAR_Value = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseCVAR_Value(String strValue) {
        try {
            this.phaseCVAR_Value = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
   /* public void setPhaseAW_RMS(String strValue) {
        try {
            this.phaseAW_RMS = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseBW_RMS(String strValue) {
        try {
            this.phaseBW_RMS = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseCW_RMS(String strValue) {
```

```
        try {
            this.phaseCW_RMS = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseAVAR_RMS(String strValue) {
        try {
            this.phaseAVAR_RMS = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseBVAR_RMS(String strValue) {
        try {
            this.phaseBVAR_RMS = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseCVAR_RMS(String strValue) {
        try {
            this.phaseCVAR_RMS = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }*/

    // power - power measured on external meter
    // energy - measured by AD7758
    //  - see p.65 AD7758_datasheet_Rev.0
    public int calculatePowerGain(double power,int energy) {
        double accumTime = 0.0;
        double tmpVal = 0.0;
        if (this.FreqRMS == 0) {
            this.FreqRMS = 800;
        }
        if (power < 0) {
            power = power * (-1);
        }
        if (energy < 0) {
            energy = energy * (-1);
        }
        accumTime = this.calLineCyc/(2*this.FreqRMS*this.calFreq);
        tmpVal = (energy*3600)/(accumTime*power);
        System.out.println("Power :" + power);
        System.out.println("Energy:" + energy);
        System.out.println("Double:" + (double)tmpVal);
        System.out.println("Int   :" + (int)tmpVal);
        return (int)tmpVal;
    }

//Power Fine Tune Callibration
    public void setMCU_phaseAW_RMS(int tmpVal) {
       this.MCU_phaseAW_RMS = tmpVal;
    }
    public int getMCU_phaseAW_RMS() {
       return this.MCU_phaseAW_RMS;
    }

    //MCU_phaseBW_RMS
    public void setMCU_phaseBW_RMS(int tmpVal) {
       this.MCU_phaseBW_RMS = tmpVal;
    }
    public int getMCU_phaseBW_RMS() {
       return this.MCU_phaseBW_RMS;
    }

    //MCU_phaseCW_RMS
    public void setMCU_phaseCW_RMS(int tmpVal) {
       this.MCU_phaseCW_RMS = tmpVal;
    }
    public int getMCU_phaseCW_RMS() {
       return this.MCU_phaseCW_RMS;
    }

    //MCU_phaseAVA_RMS
    public void setMCU_phaseAVA_RMS(int tmpVal) {
       this.MCU_phaseAVA_RMS = tmpVal;
```

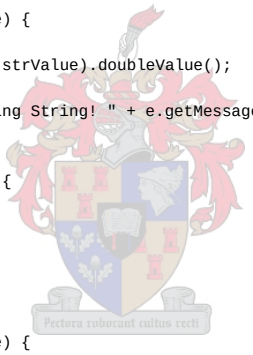
```
}
public int getMCU_phaseAVA_RMS() {
   return this.MCU_phaseAVA_RMS;
}

//MCU_phaseBVA_RMS
public void setMCU_phaseBVA_RMS(int tmpVal) {
   this.MCU_phaseBVA_RMS = tmpVal;
}
public int getMCU_phaseBVA_RMS() {
   return this.MCU_phaseBVA_RMS;
}

//MCU_phaseCVA_RMS
public void setMCU_phaseCVA_RMS(int tmpVal) {
   this.MCU_phaseCVA_RMS = tmpVal;
}
public int getMCU_phaseCVA_RMS() {
   return this.MCU_phaseCVA_RMS;
}

//MCU_phaseAVAR_RMS
public void setMCU_phaseAVAR_RMS(int tmpVal) {
   this.MCU_phaseAVAR_RMS = tmpVal;
}
public int getMCU_phaseAVAR_RMS() {
   return this.MCU_phaseAVAR_RMS;
}

//MCU_phaseBVAR_RMS
public void setMCU_phaseBVAR_RMS(int tmpVal) {
   this.MCU_phaseBVAR_RMS = tmpVal;
}
public int getMCU_phaseBVAR_RMS() {
   return this.MCU_phaseBVAR_RMS;
}

//MCU_phaseCVAR_RMS
public void setMCU_phaseCVAR_RMS(int tmpVal) {
   this.MCU_phaseCVAR_RMS = tmpVal;
}
public int getMCU_phaseCVAR_RMS() {
   return this.MCU_phaseCVAR_RMS;
}

//MCU_phaseAW_Value
public void setMCU_phaseAW_Value(String strValue) {
    try {
        this.MCU_phaseAW_Value = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setMCU_phaseAW_Value(double tmpVal) {
   this.MCU_phaseAW_Value = tmpVal;
}
public double getMCU_phaseAW_Value() {
   return this.MCU_phaseAW_Value;
}

//MCU_phaseBW_Value
public void setMCU_phaseBW_Value(String strValue) {
    try {
        this.MCU_phaseBW_Value = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setMCU_phaseBW_Value(double tmpVal) {
   this.MCU_phaseBW_Value = tmpVal;
}
public double getMCU_phaseBW_Value() {
   return this.MCU_phaseBW_Value;
}

//MCU_phaseCW_Value
public void setMCU_phaseCW_Value(String strValue) {
    try {
        this.MCU_phaseCW_Value = Double.valueOf(strValue).doubleValue();
```

```
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setMCU_phaseCW_Value(double tmpVal) {
   this.MCU_phaseCW_Value = tmpVal;
}
public double getMCU_phaseCW_Value() {
   return this.MCU_phaseCW_Value;
}

//MCU_phaseAVAR_Value
public void setMCU_phaseAVAR_Value(String strValue) {
    try {
        this.MCU_phaseAVAR_Value = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setMCU_phaseAVAR_Value(double tmpVal) {
   this.MCU_phaseAVAR_Value = tmpVal;
}
public double getMCU_phaseAVAR_Value() {
   return this.MCU_phaseAVAR_Value;
}

//MCU_phaseBVAR_Value
public void setMCU_phaseBVAR_Value(String strValue) {
    try {
        this.MCU_phaseBVAR_Value = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setMCU_phaseBVAR_Value(double tmpVal) {
   this.MCU_phaseBVAR_Value = tmpVal;
}
public double getMCU_phaseBVAR_Value() {
   return this.MCU_phaseBVAR_Value;
}

//MCU_phaseCVAR_Value
public void setMCU_phaseCVAR_Value(String strValue) {
    try {
        this.MCU_phaseCVAR_Value = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setMCU_phaseCVAR_Value(double tmpVal) {
   this.MCU_phaseCVAR_Value = tmpVal;
}
public double getMCU_phaseCVAR_Value() {
   return this.MCU_phaseCVAR_Value;
}

//MCU_phaseAVA_Value
public void setMCU_phaseAVA_Value(String strValue) {
    try {
        this.MCU_phaseAVA_Value = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setMCU_phaseAVA_Value(double tmpVal) {
   this.MCU_phaseAVA_Value = tmpVal;
}
public double getMCU_phaseAVA_Value() {
   return this.MCU_phaseAVA_Value;
}

//MCU_phaseBVA_Value
public void setMCU_phaseBVA_Value(String strValue) {
    try {
        this.MCU_phaseBVA_Value = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
```

```
    public void setMCU_phaseBVA_Value(double tmpVal) {
       this.MCU_phaseBVA_Value = tmpVal;
    }
    public double getMCU_phaseBVA_Value() {
       return this.MCU_phaseBVA_Value;
    }

    //MCU_phaseCVA_Value
    public void setMCU_phaseCVA_Value(String strValue) {
        try {
            this.MCU_phaseCVA_Value = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setMCU_phaseCVA_Value(double tmpVal) {
       this.MCU_phaseCVA_Value = tmpVal;
    }
    public double getMCU_phaseCVA_Value() {
       return this.MCU_phaseCVA_Value;
    }


    public int reCalcPowerGain(double currentVal, int measuredVal, int calibratedDiv) {
        double newCal = 0.0;
        newCal = measuredVal*calibratedDiv/currentVal;
        System.out.println("Reffernence:" + currentVal);
        System.out.println("Meter      :" + measuredVal);
        System.out.println("OldCal     :" + calibratedDiv);
        System.out.println("newCal     :" + (int)newCal);
        return (int)newCal;
    }
/*
    public double calculatePowerLSB(double VA,int energy) {
        double accumTime = 0.0;
        double tmpVal = 0.0;
        accumTime = this.calLineCyc/(2*this.FreqRMS*this.calFreq);
        tmpVal = (VA*accumTime)/(3600*energy);
        return tmpVal;
    }

    public void setCalWhLSB(double value) {
        calWhLSB = value;
    }
    public double getCalWhLSB() {
        return calWhLSB;
    }*/
    //Beans
    /*
    public void setCalVAhLSB(double value) {
        calVAhLSB = value;
    }
    public double getCalVAhLSB() {
        return calVAhLSB;
    }

    public void setCalVARhLSB(double value) {
        calVARhLSB = value;
    }
    public double getCalVARhLSB() {
        return calVARhLSB;
    }*/

    //Frequency Calibration
    public void setFreqValue(String strValue) {
        try {
            this.FreqValue = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }

    public void calculateFreq() {
        this.calFreq = this.FreqValue/this.FreqRMS;
    }

    public void setCalFreq(double value) {
        calFreq = value;
    }
```

```
    public double getCalFreq() {
        return calFreq;
    }

//Phase Calibration
// storagae methods

    //power factor
    public void setPhaseCalPF1A(String strValue) {
        try {
            this.phaseCalPF1A = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseCalPF1A(double powerFactor) {
        this.phaseCalPF1A = powerFactor;
    }

    public double getPhaseCalPF1A() {
        return this.phaseCalPF1A;
    }

    public void setPhaseCalPF1B(String strValue) {
        try {
            this.phaseCalPF1B = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseCalPF1B(double powerFactor) {
        this.phaseCalPF1B = powerFactor;
    }

    public double getPhaseCalPF1B() {
        return this.phaseCalPF1B;
    }

    public void setPhaseCalPF1C(String strValue) {
        try {
            this.phaseCalPF1C = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseCalPF1C(double powerFactor) {
        this.phaseCalPF1C = powerFactor;
    }
    public double getPhaseCalPF1C() {
        return this.phaseCalPF1C;
    }

    public void setPhaseCalPF2A(String strValue) {
        try {
            this.phaseCalPF2A = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseCalPF2A(double powerFactor) {
        this.phaseCalPF2A = powerFactor;
    }

    public double getPhaseCalPF2A() {
        return this.phaseCalPF2A;
    }

    public void setPhaseCalPF2B(String strValue) {
        try {
            this.phaseCalPF2B = Double.valueOf(strValue).doubleValue();
        } catch (NumberFormatException e) {
            System.out.println("Error Converting String! " + e.getMessage());
        }
    }
    public void setPhaseCalPF2B(double powerFactor) {
        this.phaseCalPF2B = powerFactor;
    }
    public double getPhaseCalPF2B() {
        return this.phaseCalPF2B;
```

```
}

public void setPhaseCalPF2C(String strValue) {
    try {
        this.phaseCalPF2C = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseCalPF2C(double powerFactor) {
    this.phaseCalPF2C = powerFactor;
}
public double getPhaseCalPF2C() {
    return this.phaseCalPF2C;
}

//Watts
public void setPhaseCalW1A(String strValue) {
    try {
        this.phaseCalW1A = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseCalW1A(double tmpVal) {
    this.phaseCalW1A = tmpVal;
}

public double getPhaseCalW1A() {
    return this.phaseCalW1A;
}

public void setPhaseCalW1B(String strValue) {
    try {
        this.phaseCalW1B = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseCalW1B(double tmpVal) {
    this.phaseCalW1B = tmpVal;
}
public double getPhaseCalW1B() {
    return this.phaseCalW1B;
}

public void setPhaseCalW1C(String strValue) {
    try {
        this.phaseCalW1C = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseCalW1C(double tmpVal) {
    this.phaseCalW1C = tmpVal;
}
public double getPhaseCalW1C() {
    return this.phaseCalW1C;
}

public void setPhaseCalW2A(String strValue) {
    try {
        this.phaseCalW2A = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseCalW2A(double tmpVal) {
    this.phaseCalW2A = tmpVal;
}

public double getPhaseCalW2A() {
    return this.phaseCalW2A;
}

public void setPhaseCalW2B(String strValue) {
    try {
        this.phaseCalW2B = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
```



```
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseCalW2B(double tmpVal) {
    this.phaseCalW2B = tmpVal;
}
public double getPhaseCalW2B() {
    return this.phaseCalW2B;
}

public void setPhaseCalW2C(String strValue) {
    try {
        this.phaseCalW2C = Double.valueOf(strValue).doubleValue();
    } catch (NumberFormatException e) {
        System.out.println("Error Converting String! " + e.getMessage());
    }
}
public void setPhaseCalW2C(double tmpVal) {
    this.phaseCalW2C = tmpVal;
}
public double getPhaseCalW2C() {
    return this.phaseCalW2C;
}

//Measure Values
public void setPhaseCalWRMS1A(int whRMS) {
    this.phaseCalWRMS1A = whRMS;
}
public int getPhaseCalWRMS1A() {
    return this.phaseCalWRMS1A;
}

public void setPhaseCalWRMS1B(int whRMS) {
    this.phaseCalWRMS1B = whRMS;
}
public int getPhaseCalWRMS1B() {
    return this.phaseCalWRMS1B;
}

public void setPhaseCalWRMS1C(int whRMS) {
    this.phaseCalWRMS1C = whRMS;
}
public int getPhaseCalWRMS1C() {
    return this.phaseCalWRMS1C;
}

public void setPhaseCalWRMS2A(int whRMS) {
    this.phaseCalWRMS2A = whRMS;
}
public int getPhaseCalWRMS2A() {
    return this.phaseCalWRMS2A;
}

public void setPhaseCalWRMS2B(int whRMS) {
    this.phaseCalWRMS2B = whRMS;
}
public int getPhaseCalWRMS2B() {
    return this.phaseCalWRMS2B;
}

public void setPhaseCalWRMS2C(int whRMS) {
    this.phaseCalWRMS2C = whRMS;
}
public int getPhaseCalWRMS2C() {
    return this.phaseCalWRMS2C;
}

//Callibration procedure:
public byte calPhaseOffset(double PF1, double W1, int WRMS1, double PF2, double W2, int WRMS2) {
    //java.lang.Math
    double error = 0.0;
    double VA1 = 0.0;
    double VA2 = 0.0;
    double phase = 0.0;
    double phaseError = 0.0;
    double phCal = 0.0;
    byte bytePhCal = 0;
    if (PF1 > PF2) {
        VA2 = W1/WRMS1;
```

```
            VA1 = W2/WRMS2;
            phase = (java.lang.Math.acos(PF2)-java.lang.Math.acos(PF1));
        } else {
            VA1 = W1/WRMS1;
            VA2 = W2/WRMS2;
            phase = (java.lang.Math.acos(PF1)-java.lang.Math.acos(PF2));
        }
        error = ((VA2-VA1)/VA1);
        phaseError = java.lang.Math.toDegrees(java.lang.Math.asin(error/java.lang.Math.sin(phase)));
        if (phaseError < 0) {
            phCal = phaseError/(2.4*java.lang.Math.pow(10,-6)*360*50);
        } else {
            phCal = phaseError/(4.8*java.lang.Math.pow(10,-6)*360*50);
        }
        bytePhCal = (byte)phCal;
        System.out.println(phCal);
        System.out.println(bytePhCal);
        return bytePhCal;
    }
}
```

Q.2.4 Object: `energyMeasure.energyServer`

```
/*
 * energyServer.java
 *
 * Created on 21 May 2004, 10:33
 */

package energyMeasure;

import gov.noaa.pmel.util.*;
import javax.comm.*;
import java.util.*;
import java.text.*;
import java.io.*;
import peg.io.*;
import peg.sql.*;
import peg.utils.*;
/**
 *
 * @author  RDoorduin
 */
public class energyServer extends Thread implements SerialPortEventListener {
    private clients clientsList = new clients();
    private serialInterface serial = new serialInterface("Port1");
    private gsmATinterface gsmAT = new gsmATinterface();
    private debug db = new debug("energyServer",9);

    private boolean maybeActive = false;

    /** Creates a new instance of energyServer */
    public energyServer(int currentPort) {
        db.print(currentPort,"currentPort="+currentPort);
        serial.setCurrentPort(currentPort);
        serial.openCurrentPort();
        //db.print(2,"energyServer():openingPort(COM2)...");
        //serial.openPort("COM2");
        if (serial.isOpen()) {
            System.out.println("Serial Port Open");
            serial.addListener(this);
            this.setupGSM();
            maybeActive = true;
        } else {
            System.out.println("Port not open!:" + energyServerSetup.getSetupString(energyServerSetup.COMPort));
            System.out.println("Please ensure correct port setup and try to run program again!");
            maybeActive = false;
            System.exit(0);
        }
    }

    public energyServer() {
        this(0);// 0  for RAN, 1 for DOORDUIN

    }

    public void setupGSM() {
```

```
    int res;
    gsmAT = new gsmATinterface(serial.out);
    gsmAT.commandWait("ATE0");
        //gsmAT.commandWait("AT","OK");
    //Setup modem
    res = gsmAT.commandWait("AT+CPIN=" + energyServerSetup.getSetupString(energyServerSetup.ServerPIN), "+CREG: 1", 1,6000, "+CME ERROR: 3
    res =0;
    System.out.println("Res:" + res);
    if ((res == 0) | (res == -1)) { //Either PIN OK, or PIN already entered...
        //Setup SMS setup and CLIP
        gsmAT.commandWait("AT+CNMI=0,1,1,1,0");
        gsmAT.commandWait("AT+CMGF=1");
        gsmAT.commandWait("AT+CMGS=\"" + energyServerSetup.getSetupString(energyServerSetup.SMSCenter) + "\"");
        gsmAT.commandWait("AT+CLIP=1");
        gsmAT.commandWait("AT+CMEE=1");
        gsmAT.commandWait("AT+WIND=63");

        //Check SMS on network and card
        gsmAT.clearData();
        gsmAT.clearDecodeBuffer();
        gsmAT.command("AT+CMGL=\"ALL\""); //Dump all SMS'es
    } else {
        System.out.println("Please ensure correct operation of modem!");
        System.exit(0);
    }
}

public void run() {
    GeoDate alarmDate = new GeoDate();
    GeoDate alarmTime = new GeoDate();
    GeoDate now = new GeoDate();
    int mon;
    int day;
    int year;
    int hour;
    int min;
    int sec;
    int msec;
    int compVal;
    boolean serviced = false;

    if (!maybeActive) {
        System.exit(0); //if the serial resources are used by other instances,
                        //there is no use for this thread to run
    }

    while (true) {

        //Set correct alarm trigger
        alarmDate.now();
        alarmTime.now();
        try {
            alarmTime = new GeoDate(energyServerSetup.getSetupString(energyServerSetup.UpdateTime), "HH:mm");
        } catch (IllegalTimeValue e) {
            System.out.println("Error with Time Parsing: " + e.getMessage());
        }
        mon = alarmDate.getGMTMonth();
        day = alarmDate.getGMTDay();
        year = alarmDate.getGMTYear();
        hour = alarmTime.getGMTHours();
        min = alarmTime.getGMTMinutes();
        sec = 0;
        msec = 0;

        try {
            alarmDate.set(mon, day, year, hour, min, sec, msec);
        } catch (IllegalTimeValue e) {
            System.out.println("Error with Alarm Setting: " + e.getMessage());
        }
        System.out.println(alarmDate.toString() + " [" + now.toString() + "] ");
        //The current time
        now.now();
        now.increment(2,GeoDate.HOURS);    //2 Hours west of Greenwich

        //Compare the time
        compVal = now.compareTo(alarmDate);
        if (compVal > 0) {
            if (!serviced) {
                sendSMStoClients();
```

```
                serviced = true;
            }
        } else if (compVal < 0) {
            serviced = false;
        } else {
            System.out.println(" Equal!" + compVal);
        }

        //General event signalled from the energyServerSystem class
        if (energyServerSetup.getSetupString(energyServerSetup.DumpAllSMSes).compareTo("True") == 0) {
            energyServerSetup.setSetupString(energyServerSetup.DumpAllSMSes,"");
            System.out.println("Dumping all SMS'es");
            gsmAT.command("Test"+(char)0x01A);
            try {
                sleep(4000);   //To ensure each minute is catched...
            } catch (InterruptedException e) {
                System.out.println("Thread Interupted:"+e.getMessage());
            }
            gsmAT.command("AT+CMGL=\"ALL\""); //Dump all SMS'es
        } else if (energyServerSetup.getSetupString(energyServerSetup.UpdateNow).compareTo("True") == 0) {
            energyServerSetup.setSetupString(energyServerSetup.UpdateNow,"");
            System.out.println("Sending Update SMS'es:");
            sendSMStoClients();
        }
        try {
            sleep(10000);   //To ensure each minute is catched...
        } catch (InterruptedException e) {
            System.out.println("Thread Interupted:"+e.getMessage());
        }
    }
}

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    energyServer eS = new energyServer();
    eS.start();
}

public void storeSMSdata() {
    String driver = energyServerSetup.getSetupString(energyServerSetup.DBDriver);
    String protocol = energyServerSetup.getSetupString(energyServerSetup.DBProtocol);
    String url = energyServerSetup.getSetupString(energyServerSetup.DBurl);
    String user = energyServerSetup.getSetupString(energyServerSetup.DBuser);
    String password = energyServerSetup.getSetupString(energyServerSetup.DBpassword);

    String tmpStr = "";

    if ((gsmAT.getATDataCount() == 9) & ((gsmAT.getATResultParametersCount() == 6) | (gsmAT.getATResultParametersCount() == 5))) {
            //Print for debugging purposes
            if (gsmAT.getATResultParametersCount() == 6) {
                tmpStr = ""+gsmAT.getATResultParameter(2)+",'"+gsmAT.getATResultParameter(4).substring(1)+" "+gsmAT.getATResultParameter(5
            } else if (gsmAT.getATResultParametersCount() == 5) {
                tmpStr = ""+gsmAT.getATResultParameter(1)+",'"+gsmAT.getATResultParameter(3).substring(1)+" "+gsmAT.getATResultParameter(4
            }

            for (int i = 0; i < gsmAT.getATDataCount(); i++) {
                tmpStr = tmpStr + ("," + gsmAT.getATDataAt(i));
            }
            System.out.println(tmpStr);
            System.out.flush();

            //Setup database
            DBConnect myDB = new DBConnect(driver,protocol,url,user,password);
            myDB.executeSQL("CREATE TABLE IF NOT EXISTS CheckMeters (checkMeterNumber VARCHAR(20) NOT NULL, date DATETIME NOT NULL, AW INT
            myDB.executeSQL("INSERT INTO CheckMeters VALUES(" + tmpStr + ")");
            if (myDB.isQuerySuccess()) {
                System.out.println("DBUpdated!");
            } else {
                System.out.println("Error extracting data from database:" + myDB.getErrorString());
            }
            /*
            if (myDB.isQuerySuccess()) {
                javax.swing.JFrame myTempFrame = new javax.swing.JFrame();
                myTempFrame.getContentPane().add(new peg.swing.TableDisplay(myDB));
                myTempFrame.setBounds(10,10,300,450);
                myTempFrame.show();
            } else {
```

```
                System.out.println("Error extracting data from database:" + myDB.getErrorString());
            }*/

        }

    }

    public void sendSMStoClients() {
        for (int i = 0; i < clientsList.getRowCount(); i++) {
            System.out.println("Client number: "+clientsList.getValueAt(i,0));
            gsmAT.sendSMS((String)clientsList.getValueAt(i,0), "Hello World!");
            try {
                sleep(4000);   //To ensure each minute is catched...
            } catch (InterruptedException e) {
                System.out.println("Thread Interupted:"+e.getMessage());
            }
            gsmAT.command("Test"+(char)0x01A);
        }
    }
    public void serialEvent(javax.comm.SerialPortEvent serialPortEvent) {
        String thisResult = "";
        String tmpStr = "";
        byte[] tmpByteArray;
        byte[] b = null;
        if (serialPortEvent.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
            gsmAT.decode(serial.in);
            if (gsmAT.isResultReady() & !gsmAT.isInWaitState()) {
                stateEventMachine(gsmATinterface.EV_GSMnewResultAvailable);
            }
        }
    }

    public void stateEventMachine(int event) {
        System.out.print("("+gsmAT.ATState+")");
        switch (gsmAT.ATState) {
            case (gsmATinterface.AT_idle):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.getNextState();
                if (gsmAT.ATState == gsmATinterface.AT_CMTI) {
                    gsmAT.command("AT+CMGR="+gsmAT.getATResultParameter(1));
                    gsmAT.ATState = gsmATinterface.AT_idle;
                }
                break;
            case (gsmATinterface.AT_delete_all_SMS):
                gsmAT.ATState = gsmATinterface.AT_idle;
                break;
            case (gsmATinterface.AT_CMGR):
                gsmAT.HexDataDecoder(gsmAT.getNextResult());
                gsmAT.printHexData();
                storeSMSdata();
                gsmAT.command("AT+CMGD="+gsmAT.currentSMS);
                gsmAT.ATState = gsmAT.ATState = gsmATinterface.AT_idle;;
                break;
            case (gsmATinterface.AT_CMGL):
                gsmAT.HexDataDecoder(gsmAT.getNextResult());
                gsmAT.printHexData();
                storeSMSdata();
                gsmAT.ATState = gsmATinterface.AT_wait_for_next_SMS_listing;
                break;
            case (gsmATinterface.AT_wait_for_next_SMS_listing):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                if (gsmAT.getATResultCode().equalsIgnoreCase("OK")) {
                    //gsmAT.ATState = gsmAT.ATState = gsmATinterface.AT_idle;
                    gsmAT.command("AT+CMGD=1");
                    gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_1;
                } else {
                    gsmAT.getNextState();
                }
                break;
            case (gsmATinterface.AT_delete_all_SMSes_1):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=2");
                gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_2;
                break;
            case (gsmATinterface.AT_delete_all_SMSes_2):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
```

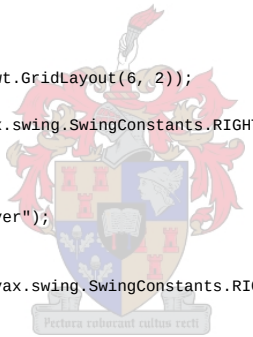
```
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=3");
                gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_3;
                break;
            case (gsmATinterface.AT_delete_all_SMSes_3):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=4");
                gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_4;
                break;
            case (gsmATinterface.AT_delete_all_SMSes_4):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=5");
                gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_5;
                break;
            case (gsmATinterface.AT_delete_all_SMSes_5):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=6");
                gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_6;
                break;
            case (gsmATinterface.AT_delete_all_SMSes_6):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=7");
                gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_7;
                break;
            case (gsmATinterface.AT_delete_all_SMSes_7):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=8");
                gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_8;
                break;
            case (gsmATinterface.AT_delete_all_SMSes_8):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=9");
                gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_9;
                break;
            case (gsmATinterface.AT_delete_all_SMSes_9):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=10");
                gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_10;
                break;
            case (gsmATinterface.AT_delete_all_SMSes_10):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=11");
                gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_11;
                break;
            case (gsmATinterface.AT_delete_all_SMSes_11):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=12");
                gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_12;
                break;
            case (gsmATinterface.AT_delete_all_SMSes_12):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=13");
                gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_13;
                break;
            case (gsmATinterface.AT_delete_all_SMSes_13):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=14");
                gsmAT.ATState = gsmATinterface.AT_delete_all_SMSes_14;
                break;
            case (gsmATinterface.AT_delete_all_SMSes_14):
                gsmAT.SMSdecoder(gsmAT.getNextResult());
                gsmAT.printATResultParameters();
                gsmAT.command("AT+CMGD=15");
                gsmAT.ATState = gsmATinterface.AT_idle;
                break;
        }
    }
}
```

Q.2.5 Object: `energyMeasure.energyServerGUI`

```
/*
 * energyServerGUI.java
 *
 * Created on 24 May 2004, 10:49
 */

package energyMeasure;

/**
 *
 * @author  RDoorduin
 */
public class energyServerGUI extends javax.swing.JFrame {

    /** Creates new form energyServerGUI */
    public energyServerGUI() {
        initComponents();
        clearPanels();
        this.setSize(600,600);
    }

    private void clearPanels() {
        pnlClientsAdd.setVisible(false);
        pnlClientsList.setVisible(false);
        pnlDataBaseSetup.setVisible(false);
        pnlServerSetup.setVisible(false);
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jToolBar1 = new javax.swing.JToolBar();
        lblMessage = new javax.swing.JLabel();
        pnlMain = new javax.swing.JPanel();
        pnlClientsList = new javax.swing.JPanel();
        spnClientsList = new javax.swing.JScrollPane();
        tblClientsList = new javax.swing.JTable(new clients());
        pnlClientsListButtons = new javax.swing.JPanel();
        btnClientsListDelete = new javax.swing.JButton();
        btnClientsListAdd = new javax.swing.JButton();
        btnClientsListCancel = new javax.swing.JButton();
        pnlClientsAdd = new javax.swing.JPanel();
        lblNumber = new javax.swing.JLabel();
        txtNumber = new javax.swing.JTextField();
        lblLocation = new javax.swing.JLabel();
        txtLocation = new javax.swing.JTextField();
        lblDescription = new javax.swing.JLabel();
        txtDescription = new javax.swing.JTextField();
        lblMinisub = new javax.swing.JLabel();
        txtMinisub = new javax.swing.JTextField();
        btnClientsAddCancel = new javax.swing.JButton();
        btnClientsAddOK = new javax.swing.JButton();
        pnlDataBaseSetup = new javax.swing.JPanel();
        lblDriver = new javax.swing.JLabel();
        txtDriver = new javax.swing.JTextField();
        lblProtocol = new javax.swing.JLabel();
        txtProtocol = new javax.swing.JTextField();
        lblURL = new javax.swing.JLabel();
        txtURL = new javax.swing.JTextField();
        lblUser = new javax.swing.JLabel();
        txtUser = new javax.swing.JTextField();
        lblPassword = new javax.swing.JLabel();
        txtPassword = new javax.swing.JPasswordField();
        btnDataBaseSetupCancel = new javax.swing.JButton();
        btnDataBaseSetupOK = new javax.swing.JButton();
        pnlServerSetup = new javax.swing.JPanel();
        lblSMSCenter = new javax.swing.JLabel();
        txtSMSCenter = new javax.swing.JTextField();
        lblServerNumber = new javax.swing.JLabel();
        txtServerNumber = new javax.swing.JTextField();
        lblServerPIN = new javax.swing.JLabel();
        txtServerPIN = new javax.swing.JTextField();
        lblCOMPort = new javax.swing.JLabel();
        txtCOMPort = new javax.swing.JTextField();
```

```
lblUpdateTime = new javax.swing.JLabel();
btnUpdateTime = new javax.swing.JButton();
btnServerSetupCancel = new javax.swing.JButton();
btnServerSetupOK = new javax.swing.JButton();
btnDumpSMS = new javax.swing.JButton();
btnServerUpdate = new javax.swing.JButton();
jMenuBar1 = new javax.swing.JMenuBar();
mnuClients = new javax.swing.JMenu();
mnuClientsAdd = new javax.swing.JMenuItem();
mnuClientsList = new javax.swing.JMenuItem();
mnuSettings = new javax.swing.JMenu();
mnuSettingsDataBase = new javax.swing.JMenuItem();
mnuSettingsServer = new javax.swing.JMenuItem();

addWindowListener(new java.awt.event.WindowAdapter() {
    public void windowClosing(java.awt.event.WindowEvent evt) {
        exitForm(evt);
    }
});

lblMessage.setText("Msg:");
jToolBar1.add(lblMessage);

getContentPane().add(jToolBar1, java.awt.BorderLayout.SOUTH);

pnlClientsList.setLayout(new java.awt.BorderLayout());

spnClientsList.setViewportView(tblClientsList);

pnlClientsList.add(spnClientsList, java.awt.BorderLayout.CENTER);

pnlClientsListButtons.setLayout(new java.awt.GridLayout(1, 0));

btnClientsListDelete.setText("Remove");
btnClientsListDelete.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnClientsListDeleteActionPerformed(evt);
    }
});

pnlClientsListButtons.add(btnClientsListDelete);

btnClientsListAdd.setText("Add");
btnClientsListAdd.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnClientsListAddActionPerformed(evt);
    }
});

pnlClientsListButtons.add(btnClientsListAdd);

btnClientsListCancel.setText("Cancel");
btnClientsListCancel.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnClientsListCancelActionPerformed(evt);
    }
});

pnlClientsListButtons.add(btnClientsListCancel);

pnlClientsList.add(pnlClientsListButtons, java.awt.BorderLayout.SOUTH);

pnlMain.add(pnlClientsList);

pnlClientsAdd.setLayout(new java.awt.GridLayout(5, 2));

lblNumber.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblNumber.setText("Cell Number:");
pnlClientsAdd.add(lblNumber);

txtNumber.setText("Enter Cell Number Here...\ne.g. 0848163790");
pnlClientsAdd.add(txtNumber);

lblLocation.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblLocation.setText("Location:");
pnlClientsAdd.add(lblLocation);

txtLocation.setText("Enter Location Here...");
pnlClientsAdd.add(txtLocation);
```

```
lblDescription.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblDescription.setText("Description:");
pnlClientsAdd.add(lblDescription);

txtDescription.setText("Enter Meter Description Here... e.g. SL7000");
pnlClientsAdd.add(txtDescription);

lblMinisub.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblMinisub.setText("Minisub:");
pnlClientsAdd.add(lblMinisub);

txtMinisub.setText("Enter Minisub Number here and feeder info");
pnlClientsAdd.add(txtMinisub);

btnClientsAddCancel.setText("Cancel");
btnClientsAddCancel.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnClientsAddCancelActionPerformed(evt);
    }
});

pnlClientsAdd.add(btnClientsAddCancel);

btnClientsAddOK.setText("OK");
btnClientsAddOK.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnClientsAddOKActionPerformed(evt);
    }
});

pnlClientsAdd.add(btnClientsAddOK);

pnlMain.add(pnlClientsAdd);

pnlDataBaseSetup.setLayout(new java.awt.GridLayout(6, 2));

lblDriver.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblDriver.setText("Driver:");
pnlDataBaseSetup.add(lblDriver);

txtDriver.setText("com.mysql.jdbc.Driver");
pnlDataBaseSetup.add(txtDriver);

lblProtocol.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblProtocol.setText("Protocol:");
pnlDataBaseSetup.add(lblProtocol);

txtProtocol.setText("jdbc:mysql");
pnlDataBaseSetup.add(txtProtocol);

lblURL.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblURL.setText("URL:");
pnlDataBaseSetup.add(lblURL);

txtURL.setText("//localhost:3306/SMSdb?user=blah&password=blah");
pnlDataBaseSetup.add(txtURL);

lblUser.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblUser.setText("User:");
pnlDataBaseSetup.add(lblUser);

pnlDataBaseSetup.add(txtUser);

lblPassword.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblPassword.setText("Password:");
pnlDataBaseSetup.add(lblPassword);

pnlDataBaseSetup.add(txtPassword);

btnDataBaseSetupCancel.setText("Cancel");
btnDataBaseSetupCancel.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnDataBaseSetupCancelActionPerformed(evt);
    }
});

pnlDataBaseSetup.add(btnDataBaseSetupCancel);

btnDataBaseSetupOK.setText("OK");
```

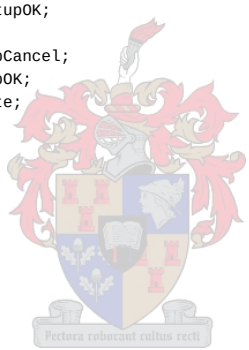


```
btnDataBaseSetupOK.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnDataBaseSetupOKActionPerformed(evt);
    }
});

pnlDataBaseSetup.add(btnDataBaseSetupOK);

pnlMain.add(pnlDataBaseSetup);

pnlServerSetup.setLayout(new java.awt.GridLayout(0, 2));

lblSMSCenter.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblSMSCenter.setText("SMS Center:");
pnlServerSetup.add(lblSMSCenter);

txtSMSCenter.setText("+27841000000");
pnlServerSetup.add(txtSMSCenter);

lblServerNumber.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblServerNumber.setText("Server Number:");
pnlServerSetup.add(lblServerNumber);

txtServerNumber.setText("0846845946");
pnlServerSetup.add(txtServerNumber);

lblServerPIN.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblServerPIN.setText("Server PIN:");
pnlServerSetup.add(lblServerPIN);

txtServerPIN.setText("1172");
pnlServerSetup.add(txtServerPIN);

lblCOMPort.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblCOMPort.setText("COM Port:");
pnlServerSetup.add(lblCOMPort);

txtCOMPort.setText("COM1");
pnlServerSetup.add(txtCOMPort);

lblUpdateTime.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblUpdateTime.setText("Update Time:");
pnlServerSetup.add(lblUpdateTime);

btnUpdateTime.setText("...");
btnUpdateTime.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnUpdateTimeActionPerformed(evt);
    }
});

pnlServerSetup.add(btnUpdateTime);

btnServerSetupCancel.setText("Cancel");
btnServerSetupCancel.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnServerSetupCancelActionPerformed(evt);
    }
});

pnlServerSetup.add(btnServerSetupCancel);

btnServerSetupOK.setText("OK");
btnServerSetupOK.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnServerSetupOKActionPerformed(evt);
    }
});

pnlServerSetup.add(btnServerSetupOK);

btnDumpSMS.setText("Dump SMS");
btnDumpSMS.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnDumpSMSActionPerformed(evt);
    }
});

pnlServerSetup.add(btnDumpSMS);
```

```
        btnServerUpdate.setText("Run Update");
        btnServerUpdate.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnServerUpdateActionPerformed(evt);
            }
        });

        pnlServerSetup.add(btnServerUpdate);

        pnlMain.add(pnlServerSetup);

        getContentPane().add(pnlMain, java.awt.BorderLayout.CENTER);

        mnuClients.setText("Clients");
        mnuClientsAdd.setText("Add");
        mnuClientsAdd.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnuClientsAddActionPerformed(evt);
            }
        });

        mnuClients.add(mnuClientsAdd);

        mnuClientsList.setText("View");
        mnuClientsList.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnuClientsListActionPerformed(evt);
            }
        });

        mnuClients.add(mnuClientsList);

        jMenuBar1.add(mnuClients);

        mnuSettings.setText("Settings");
        mnuSettingsDataBase.setText("Data Base");
        mnuSettingsDataBase.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnuSettingsDataBaseActionPerformed(evt);
            }
        });

        mnuSettings.add(mnuSettingsDataBase);

        mnuSettingsServer.setText("Server Setup");
        mnuSettingsServer.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnuSettingsServerActionPerformed(evt);
            }
        });

        mnuSettings.add(mnuSettingsServer);

        jMenuBar1.add(mnuSettings);

        setJMenuBar(jMenuBar1);

        pack();
    }//GEN-END:initComponents

    private void btnDumpSMSActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnDumpSMSActionPerformed
        energyServerSetup.setSetupString(energyServerSetup.DumpAllSMSes, "True");
    }//GEN-LAST:event_btnDumpSMSActionPerformed

    private void btnServerUpdateActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnServerUpdateActionPerformed
        energyServerSetup.setSetupString(energyServerSetup.UpdateNow, "True");
    }//GEN-LAST:event_btnServerUpdateActionPerformed

    private void btnUpdateTimeActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnUpdateTimeActionPerformed
        gov.noaa.pmel.sgt.swing.prop.GeoDateDialog dateTimeDialog = new gov.noaa.pmel.sgt.swing.prop.GeoDateDialog();
        gov.noaa.pmel.util.GeoDate thisDate = new gov.noaa.pmel.util.GeoDate();
        thisDate.now();
        try {
            thisDate = new gov.noaa.pmel.util.GeoDate(btnUpdateTime.getText(), "HH:mm");
        } catch (gov.noaa.pmel.util.IllegalTimeValue e) {
            thisDate.now();
        }
        if (dateTimeDialog.showDialog(new gov.noaa.pmel.util.GeoDate(thisDate),20,20) == 1) {
            btnUpdateTime.setText(dateTimeDialog.getDate().toString("HH:mm"));
```

```
    }
}//GEN-LAST:event_btnUpdateTimeActionPerformed

private void btnServerSetupOKActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnServerSetupOKActionPerformed
    energyServerSetup.setSetupString(energyServerSetup.ServerNumber, txtServerNumber.getText());
    energyServerSetup.setSetupString(energyServerSetup.ServerPIN, txtServerPIN.getText());
    energyServerSetup.setSetupString(energyServerSetup.COMPort, txtCOMPort.getText());
    energyServerSetup.setSetupString(energyServerSetup.UpdateTime, btnUpdateTime.getText());
    energyServerSetup.setSetupString(energyServerSetup.SMSCenter, txtSMSCenter.getText());
    pnlServerSetup.setVisible(false);
}//GEN-LAST:event_btnServerSetupOKActionPerformed

private void btnServerSetupCancelActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnServerSetupCancelActionPerformed
    pnlServerSetup.setVisible(false);
}//GEN-LAST:event_btnServerSetupCancelActionPerformed

private void mnuSettingsServerActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuSettingsServerActionPerformed
    txtServerNumber.setText(energyServerSetup.getSetupString(energyServerSetup.ServerNumber));
    txtServerPIN.setText(energyServerSetup.getSetupString(energyServerSetup.ServerPIN));
    txtCOMPort.setText(energyServerSetup.getSetupString(energyServerSetup.COMPort));
    btnUpdateTime.setText(energyServerSetup.getSetupString(energyServerSetup.UpdateTime));
    txtSMSCenter.setText(energyServerSetup.getSetupString(energyServerSetup.SMSCenter));
    pnlServerSetup.setVisible(true);
}//GEN-LAST:event_mnuSettingsServerActionPerformed

private void btnDataBaseSetupOKActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnDataBaseSetupOKActionPerformed
    energyServerSetup.setSetupString(energyServerSetup.DBDriver, txtDriver.getText());
    energyServerSetup.setSetupString(energyServerSetup.DBProtocol, txtProtocol.getText());
    energyServerSetup.setSetupString(energyServerSetup.DBurl, txtURL.getText());
    energyServerSetup.setSetupString(energyServerSetup.DBpassword, new String(txtPassword.getPassword()));
    energyServerSetup.setSetupString(energyServerSetup.DBuser, txtUser.getText());
    pnlDataBaseSetup.setVisible(false);
}//GEN-LAST:event_btnDataBaseSetupOKActionPerformed

private void btnDataBaseSetupCancelActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnDataBaseSetupCancelActionPerforme
    pnlDataBaseSetup.setVisible(false);
}//GEN-LAST:event_btnDataBaseSetupCancelActionPerformed

private void mnuSettingsDataBaseActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuSettingsDataBaseActionPerformed
    txtDriver.setText(energyServerSetup.getSetupString(energyServerSetup.DBDriver));
    txtProtocol.setText(energyServerSetup.getSetupString(energyServerSetup.DBProtocol));
    txtURL.setText(energyServerSetup.getSetupString(energyServerSetup.DBurl));
    txtPassword.setText(energyServerSetup.getSetupString(energyServerSetup.DBpassword));
    txtUser.setText(energyServerSetup.getSetupString(energyServerSetup.DBuser));
    clearPanels();
    pnlDataBaseSetup.setVisible(true);
}//GEN-LAST:event_mnuSettingsDataBaseActionPerformed

private void btnClientsListDeleteActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnClientsListDeleteActionPerformed
    clients.removeClient(tblClientsList.getSelectedRow());
    tblClientsList.updateUI();
}//GEN-LAST:event_btnClientsListDeleteActionPerformed

private void btnClientsListAddActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnClientsListAddActionPerformed
    clearPanels();
    pnlClientsAdd.setVisible(true);
}//GEN-LAST:event_btnClientsListAddActionPerformed

private void btnClientsListCancelActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnClientsListCancelActionPerformed
    pnlClientsList.setVisible(false);
}//GEN-LAST:event_btnClientsListCancelActionPerformed

private void mnuClientsListActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuClientsListActionPerformed
    tblClientsList.updateUI();
    clearPanels();
    pnlClientsList.setVisible(true);
}//GEN-LAST:event_mnuClientsListActionPerformed

private void mnuClientsAddActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuClientsAddActionPerformed
    clearPanels();
    pnlClientsAdd.setVisible(true);
}//GEN-LAST:event_mnuClientsAddActionPerformed

private void btnClientsAddOKActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnClientsAddOKActionPerformed
    clientsInfo cI = new clientsInfo();
    cI.setNumber(txtNumber.getText());
    cI.setLocation(txtLocation.getText());
    cI.setDescription(txtDescription.getText());
    cI.setMinisub(txtMinisub.getText());
```

```
    if (!clients.addClient(cI)) {
        lblMessage.setText("Error adding client: Check duplicate phone number");
    } else {
        pnlClientsAdd.setVisible(false);
    }
}//GEN-LAST:event_btnClientsAddOKActionPerformed

private void btnClientsAddCancelActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnClientsAddCancelActionPerformed
    pnlClientsAdd.setVisible(false);
}//GEN-LAST:event_btnClientsAddCancelActionPerformed

/** Exit the Application */
private void exitForm(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_exitForm
    System.exit(0);
}//GEN-LAST:event_exitForm

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    new energyServerGUI().show();
}


// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JButton btnClientsAddCancel;
private javax.swing.JButton btnClientsAddOK;
private javax.swing.JButton btnClientsListAdd;
private javax.swing.JButton btnClientsListCancel;
private javax.swing.JButton btnClientsListDelete;
private javax.swing.JButton btnDataBaseSetupCancel;
private javax.swing.JButton btnDataBaseSetupOK;
private javax.swing.JButton btnDumpSMS;
private javax.swing.JButton btnServerSetupCancel;
private javax.swing.JButton btnServerSetupOK;
private javax.swing.JButton btnServerUpdate;
private javax.swing.JButton btnUpdateTime;
private javax.swing.JMenuBar jMenuBar1;
private javax.swing.JToolBar jToolBar1;
private javax.swing.JLabel lblCOMPort;
private javax.swing.JLabel lblDescription;
private javax.swing.JLabel lblDriver;
private javax.swing.JLabel lblLocation;
private javax.swing.JLabel lblMessage;
private javax.swing.JLabel lblMinisub;
private javax.swing.JLabel lblNumber;
private javax.swing.JLabel lblPassword;
private javax.swing.JLabel lblProtocol;
private javax.swing.JLabel lblSMSCenter;
private javax.swing.JLabel lblServerNumber;
private javax.swing.JLabel lblServerPIN;
private javax.swing.JLabel lblURL;
private javax.swing.JLabel lblUpdateTime;
private javax.swing.JLabel lblUser;
private javax.swing.JMenu mnuClients;
private javax.swing.JMenuItem mnuClientsAdd;
private javax.swing.JMenuItem mnuClientsList;
private javax.swing.JMenu mnuSettings;
private javax.swing.JMenuItem mnuSettingsDataBase;
private javax.swing.JMenuItem mnuSettingsServer;
private javax.swing.JPanel pnlClientsAdd;
private javax.swing.JPanel pnlClientsList;
private javax.swing.JPanel pnlClientsListButtons;
private javax.swing.JPanel pnlDataBaseSetup;
private javax.swing.JPanel pnlMain;
private javax.swing.JPanel pnlServerSetup;
private javax.swing.JScrollPane spnClientsList;
private javax.swing.JTable tblClientsList;
private javax.swing.JTextField txtCOMPort;
private javax.swing.JTextField txtDescription;
private javax.swing.JTextField txtDriver;
private javax.swing.JTextField txtLocation;
private javax.swing.JTextField txtMinisub;
private javax.swing.JTextField txtNumber;
private javax.swing.JPasswordField txtPassword;
private javax.swing.JTextField txtProtocol;
private javax.swing.JTextField txtSMSCenter;
private javax.swing.JTextField txtServerNumber;
private javax.swing.JTextField txtServerPIN;
```



```
    private javax.swing.JTextField txtURL;
    private javax.swing.JTextField txtUser;
    // End of variables declaration//GEN-END:variables
}
```

Q.2.6 Object: `energyMeasure.energyServerSetup`

```
/*
 * energyServerSetup.java
 *
 * Created on 24 May 2004, 09:07
 */

package energyMeasure;

import java.util.*;
import java.io.*;
import java.beans.*;

/**
 *
 * @author  RDoorduin
 */
public class energyServerSetup {

    static Vector SetupInfo = new Vector();

    public final static int DBDriver = 0,
                            DBProtocol = 1,
                            DBurl = 2,
                            DBuser = 3,
                            DBpassword = 4,
                            ServerNumber = 5,
                            ServerPIN = 6,
                            COMPort = 7,
                            UpdateTime = 8,
                            UpdateNow = 9,
                            DumpAllSMSes = 10,
                            SMSCenter = 11,
                            TotalVariables = 12;

    /** Creates a new instance of energyServerSetup */
    public energyServerSetup() {
    }

    public static void setSetupString(int index, String str) {
        SetupInfo.setSize(TotalVariables);
        SetupInfo.set(index,str);
        save();
    }
    public static String getSetupString(int index) {
        String tmpStr = new String("");
        load();
        SetupInfo.setSize(TotalVariables);
        tmpStr = (String)SetupInfo.get(index);
        if (tmpStr == null) {
            return "";
        } else {
            return tmpStr;
        }
    }

    public static void setSetupDate(int index, java.util.Date date) {
        SetupInfo.setSize(TotalVariables);
        SetupInfo.set(index,date);
        save();
    }
    public static java.util.Date getSetupDate(int index) {
        load();
        SetupInfo.setSize(TotalVariables);
        return (java.util.Date)SetupInfo.get(index);
    }

    private static void save() {
        try {
            XMLEncoder xmlE = new XMLEncoder(new BufferedOutputStream(new FileOutputStream("c:/SMSServerSetup.xml")));
            xmlE.writeObject(SetupInfo);
```

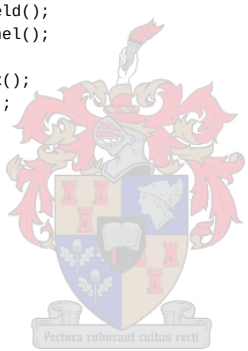
```
            xmlE.close();
        } catch (Exception e) {
          System.out.println("Error writing XMLfile: "+ e.getMessage());
        }
    }

    private static void load() {
        try {
          //System.out.println("Loading checkMeterInfo.xml...");
          XMLDecoder xmlD = new XMLDecoder(new BufferedInputStream(new FileInputStream("c:/SMSServerSetup.xml")));
          SetupInfo = (Vector)xmlD.readObject();
          xmlD.close();
        } catch (Exception e) {
            System.out.println("Error loading XMLfile: "+ e.getMessage());
        }
    }
}
```

Q.2.7 Object: `energyMeasure.myEnergy`

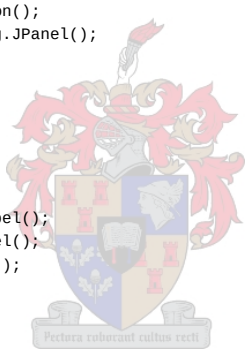
```
/*
 * myEnergy.java
 *
 * Created on 30 March 2004, 11:30
 */

package energyMeasure;

import java.io.*;
import java.beans.*;
import java.lang.*;
import javax.comm.*;
import java.text.*;
import peg.io.*;
/**
 *
 * @author  RDoorduin
 */
public class myEnergy extends javax.swing.JFrame implements SerialPortEventListener {
    private serialInterface serial = new serialInterface("Port1");
    private protocol p = new protocol();
    private energyCalibration eCal = new energyCalibration();
    private flatFile ff = new flatFile("C:/log.csv",flatFile.TYPE_CSV);

    /** Creates new form myEnergy */
    public myEnergy() {
        initComponents();
        serial.setCurrentPort(1);
        serial.openCurrentPort();
        if (serial.isOpen()) {
            serial.addListener(this);
        } else {
            System.out.println("Port not open!");
        }
        mnuOpenActionPerformed(null);

        this.pnlCalibrateVIRMSOFFSET.setVisible(false);
        //this.pnlEnergyMeasureMent.setVisible(false);
        this.pnlFrequencyCalibration.setVisible(false);
        this.pnlEnergyGainCalibration.setVisible(false);
        this.pnlMCCalibration.setVisible(false);
        //this.pnlMeasureMents.setVisible(false);
        this.pnlOtherMeasurements.setVisible(false);
        this.pnlRegisters.setVisible(false);
        this.pnlPhaseCallibration.setVisible(false);
    }


    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        pnlCalibrateVIRMSOFFSET = new javax.swing.JPanel();
        btnOffPhaseA = new javax.swing.JButton();
        btnOffPhaseB = new javax.swing.JButton();
```

```
btnOffPhaseC = new javax.swing.JButton();
btnCalVIOff = new javax.swing.JButton();
btnV_1CC = new javax.swing.JButton();
txtV_1CC = new javax.swing.JTextField();
btnV_2CC = new javax.swing.JButton();
txtV_2CC = new javax.swing.JTextField();
btnI_1CC = new javax.swing.JButton();
txtI_1CC = new javax.swing.JTextField();
btnI_2CC = new javax.swing.JButton();
txtI_2CC = new javax.swing.JTextField();
jLabel25 = new javax.swing.JLabel();
lblVaRMSOffsetRegister = new javax.swing.JLabel();
jLabel28 = new javax.swing.JLabel();
lblIaRMSOffsetRegister = new javax.swing.JLabel();
jLabel26 = new javax.swing.JLabel();
lblVbRMSOffsetRegister = new javax.swing.JLabel();
jLabel30 = new javax.swing.JLabel();
lblIbRMSOffsetRegister = new javax.swing.JLabel();
jLabel27 = new javax.swing.JLabel();
lblVcRMSOffsetRegister = new javax.swing.JLabel();
jLabel32 = new javax.swing.JLabel();
lblIcRMSOffsetRegister = new javax.swing.JLabel();
pnlFrequencyCalibration = new javax.swing.JPanel();
jLabel31 = new javax.swing.JLabel();
txtFreqValue = new javax.swing.JTextField();
jLabel33 = new javax.swing.JLabel();
txtFreqMeasure = new javax.swing.JTextField();
btnCalFreq = new javax.swing.JButton();
lblFreqMeasure = new javax.swing.JLabel();
pnlPhaseCallibration = new javax.swing.JPanel();
btnPhaseUpdateForm = new javax.swing.JButton();
jLabel47 = new javax.swing.JLabel();
jLabel53 = new javax.swing.JLabel();
jLabel54 = new javax.swing.JLabel();
btnUpdatePhaseRegister = new javax.swing.JButton();
txtPhaseACallibration = new javax.swing.JTextField();
txtPhaseBCallibration = new javax.swing.JTextField();
txtPhaseCCallibration = new javax.swing.JTextField();
jLabel56 = new javax.swing.JLabel();
txtPhase1APF = new javax.swing.JTextField();
txtPhase1BPF = new javax.swing.JTextField();
txtPhase1CPF = new javax.swing.JTextField();
jLabel57 = new javax.swing.JLabel();
txtPhase1AW = new javax.swing.JTextField();
txtPhase1BW = new javax.swing.JTextField();
txtPhase1CW = new javax.swing.JTextField();
jLabel61 = new javax.swing.JLabel();
lblPhase1AWRMS = new javax.swing.JLabel();
lblPhase1BWRMS = new javax.swing.JLabel();
lblPhase1CWRMS = new javax.swing.JLabel();
txtPhaseCycles = new javax.swing.JTextField();
btnPhaseMeasurement1A = new javax.swing.JButton();
btnPhaseMeasurement1B = new javax.swing.JButton();
btnPhaseMeasurement1C = new javax.swing.JButton();
jLabel58 = new javax.swing.JLabel();
txtPhase2APF = new javax.swing.JTextField();
txtPhase2BPF = new javax.swing.JTextField();
txtPhase2CPF = new javax.swing.JTextField();
jLabel59 = new javax.swing.JLabel();
txtPhase2AW = new javax.swing.JTextField();
txtPhase2BW = new javax.swing.JTextField();
txtPhase2CW = new javax.swing.JTextField();
jLabel62 = new javax.swing.JLabel();
lblPhase2AWRMS = new javax.swing.JLabel();
lblPhase2BWRMS = new javax.swing.JLabel();
lblPhase2CWRMS = new javax.swing.JLabel();
btnPhaseCallibrate = new javax.swing.JButton();
btnPhaseMeasurement2A = new javax.swing.JButton();
btnPhaseMeasurement2B = new javax.swing.JButton();
btnPhaseMeasurement2C = new javax.swing.JButton();
pnlEnergyGainCalibration = new javax.swing.JPanel();
btnReset = new javax.swing.JButton();
btnGainCalPhaseA = new javax.swing.JButton();
btnGainCalPhaseB = new javax.swing.JButton();
btnGainCalPhaseC = new javax.swing.JButton();
jLabel35 = new javax.swing.JLabel();
txtGainCalW_A = new javax.swing.JTextField();
txtGainCalW_B = new javax.swing.JTextField();
txtGainCalW_C = new javax.swing.JTextField();
```

```
jLabel36 = new javax.swing.JLabel();
txtGainCalVAR_A = new javax.swing.JTextField();
txtGainCalVAR_B = new javax.swing.JTextField();
txtGainCalVAR_C = new javax.swing.JTextField();
jLabel39 = new javax.swing.JLabel();
lblAWG = new javax.swing.JLabel();
lblBWG = new javax.swing.JLabel();
lblCWG = new javax.swing.JLabel();
jLabel45 = new javax.swing.JLabel();
lblAVAG = new javax.swing.JLabel();
lblBVAG = new javax.swing.JLabel();
lblCVAG = new javax.swing.JLabel();
jLabel49 = new javax.swing.JLabel();
lblAVARG = new javax.swing.JLabel();
lblBVARG = new javax.swing.JLabel();
lblCVARG = new javax.swing.JLabel();
jLabel41 = new javax.swing.JLabel();
lblMCUWDIVA = new javax.swing.JLabel();
lblMCUWDIVB = new javax.swing.JLabel();
lblMCUWDIVC = new javax.swing.JLabel();
jLabel51 = new javax.swing.JLabel();
lblMCUVARDIVA = new javax.swing.JLabel();
lblMCUVARDIVB = new javax.swing.JLabel();
lblMCUVARDIVC = new javax.swing.JLabel();
jLabel55 = new javax.swing.JLabel();
lblMCUVADIVA = new javax.swing.JLabel();
lblMCUVADIVB = new javax.swing.JLabel();
lblMCUVADIVC = new javax.swing.JLabel();
jLabel44 = new javax.swing.JLabel();
txtGainCycles = new javax.swing.JTextField();
chkFreq = new javax.swing.JCheckBox();
jTextField1 = new javax.swing.JTextField();
pnlMeasureMents = new javax.swing.JPanel();
jPanel1 = new javax.swing.JPanel();
chkMonitor = new javax.swing.JCheckBox();
chkScale = new javax.swing.JCheckBox();
chkLog = new javax.swing.JCheckBox();
jLabel4 = new javax.swing.JLabel();
jLabel5 = new javax.swing.JLabel();
jLabel7 = new javax.swing.JLabel();
jLabel34 = new javax.swing.JLabel();
jLabel8 = new javax.swing.JLabel();
jLabel1 = new javax.swing.JLabel();
lblAVRMS = new javax.swing.JLabel();
lblAIRMS = new javax.swing.JLabel();
lblAWATTHR = new javax.swing.JLabel();
lblAVARHR = new javax.swing.JLabel();
lblAVAHR = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();
lblBVRMS = new javax.swing.JLabel();
lblBIRMS = new javax.swing.JLabel();
lblBWATTHR = new javax.swing.JLabel();
lblBVARHR = new javax.swing.JLabel();
lblBVAHR = new javax.swing.JLabel();
jLabel3 = new javax.swing.JLabel();
lblCVRMS = new javax.swing.JLabel();
lblCIRMS = new javax.swing.JLabel();
lblCWATTHR = new javax.swing.JLabel();
lblCVARHR = new javax.swing.JLabel();
lblCVAHR = new javax.swing.JLabel();
pnlEnergyMeasureMent = new javax.swing.JPanel();
btnMeasureStart = new javax.swing.JButton();
btnMeasureStop = new javax.swing.JButton();
jLabel29 = new javax.swing.JLabel();
jLabel50 = new javax.swing.JLabel();
jLabel52 = new javax.swing.JLabel();
jLabel43 = new javax.swing.JLabel();
lblEnergyW = new javax.swing.JLabel();
lblEnergyWA = new javax.swing.JLabel();
lblEnergyWB = new javax.swing.JLabel();
lblEnergyWC = new javax.swing.JLabel();
btnEnergyUpdateValues = new javax.swing.JButton();
jLabel63 = new javax.swing.JLabel();
txtUpdateWhA = new javax.swing.JTextField();
txtUpdateWhB = new javax.swing.JTextField();
txtUpdateWhC = new javax.swing.JTextField();
jLabel70 = new javax.swing.JLabel();
jLabel71 = new javax.swing.JLabel();
btnUpdateWhA = new javax.swing.JButton();
```



```
btnUpdateWhB = new javax.swing.JButton();
btnUpdateWhC = new javax.swing.JButton();
jLabel46 = new javax.swing.JLabel();
lblEnergyVAR = new javax.swing.JLabel();
lblEnergyVARA = new javax.swing.JLabel();
lblEnergyVARB = new javax.swing.JLabel();
lblEnergyVARC = new javax.swing.JLabel();
jLabel64 = new javax.swing.JLabel();
jLabel65 = new javax.swing.JLabel();
txtUpdatevarhA = new javax.swing.JTextField();
txtUpdatevarhB = new javax.swing.JTextField();
txtUpdatevarhC = new javax.swing.JTextField();
jLabel68 = new javax.swing.JLabel();
jLabel69 = new javax.swing.JLabel();
btnUpdatevarhA = new javax.swing.JButton();
btnUpdatevarhB = new javax.swing.JButton();
btnUpdatevarhC = new javax.swing.JButton();
jLabel48 = new javax.swing.JLabel();
lblEnergyVA = new javax.swing.JLabel();
lblEnergyVAA = new javax.swing.JLabel();
lblEnergyVAB = new javax.swing.JLabel();
lblEnergyVAC = new javax.swing.JLabel();
jLabel66 = new javax.swing.JLabel();
jLabel67 = new javax.swing.JLabel();
txtUpdateVAhA = new javax.swing.JTextField();
txtUpdateVAhB = new javax.swing.JTextField();
txtUpdateVAhC = new javax.swing.JTextField();
jLabel72 = new javax.swing.JLabel();
jLabel73 = new javax.swing.JLabel();
btnUpdateVAhA = new javax.swing.JButton();
btnUpdateVAhB = new javax.swing.JButton();
btnUpdateVAhC = new javax.swing.JButton();
pnlOtherMeasurements = new javax.swing.JPanel();
jLabel6 = new javax.swing.JLabel();
lblTEMP = new javax.swing.JLabel();
jLabel9 = new javax.swing.JLabel();
jLabel10 = new javax.swing.JLabel();
lblFREQ = new javax.swing.JLabel();
jLabel11 = new javax.swing.JLabel();
jLabel24 = new javax.swing.JLabel();
lblTotalPackets = new javax.swing.JLabel();
lblAcknowledge = new javax.swing.JLabel();
pnlRegisters = new javax.swing.JPanel();
jLabel12 = new javax.swing.JLabel();
lblOPMODE = new javax.swing.JLabel();
jLabel13 = new javax.swing.JLabel();
lblMMODE = new javax.swing.JLabel();
jLabel14 = new javax.swing.JLabel();
lblWAVMODE = new javax.swing.JLabel();
jLabel15 = new javax.swing.JLabel();
lblCOMPMODE = new javax.swing.JLabel();
jLabel16 = new javax.swing.JLabel();
lblLCYCMODE = new javax.swing.JLabel();
jLabel17 = new javax.swing.JLabel();
lblMASK = new javax.swing.JLabel();
jLabel18 = new javax.swing.JLabel();
lblRSTATUS = new javax.swing.JLabel();
jLabel37 = new javax.swing.JLabel();
lblLINECYC = new javax.swing.JLabel();
jLabel38 = new javax.swing.JLabel();
lblWDIV = new javax.swing.JLabel();
jLabel40 = new javax.swing.JLabel();
lblVADIV = new javax.swing.JLabel();
jLabel42 = new javax.swing.JLabel();
lblVARDIV = new javax.swing.JLabel();
pnlMCCalibration = new javax.swing.JPanel();
jLabel19 = new javax.swing.JLabel();
txtMC = new javax.swing.JTextField();
jLabel20 = new javax.swing.JLabel();
txtMCMeasurePowerW = new javax.swing.JTextField();
jLabel21 = new javax.swing.JLabel();
txtMCMeasureWFreq = new javax.swing.JTextField();
jLabel22 = new javax.swing.JLabel();
txtMCMeasurePowerVAR = new javax.swing.JTextField();
jLabel23 = new javax.swing.JLabel();
txtMCMeasureVARFreq = new javax.swing.JTextField();
btnCalWMC = new javax.swing.JButton();
btnCalVAMC = new javax.swing.JButton();
lblAPCFNum = new javax.swing.JLabel();
```

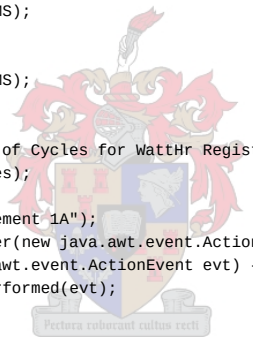


```
lblVARCFNum = new javax.swing.JLabel();
lblAPCFDen = new javax.swing.JLabel();
lblVARCFDen = new javax.swing.JLabel();
jMenuBar1 = new javax.swing.JMenuBar();
mnuFile = new javax.swing.JMenu();
mnuOpen = new javax.swing.JMenuItem();
mnuSave = new javax.swing.JMenuItem();
mnuCallibration = new javax.swing.JMenu();
mnuCFLEDsOn = new javax.swing.JMenuItem();
mnuCFLEDsOFF = new javax.swing.JMenuItem();
jSeparator1 = new javax.swing.JSeparator();
mnuWriteToFlash = new javax.swing.JMenuItem();
mnuReadFlash = new javax.swing.JMenuItem();
mnuMCU = new javax.swing.JMenu();
mnuALLREGISTERS = new javax.swing.JMenuItem();
mnuWindows = new javax.swing.JMenu();
mnuWinEnergyMeasurements = new javax.swing.JMenuItem();
mnuWinMeasurements = new javax.swing.JMenuItem();
mnuWinOtherMeasurememets = new javax.swing.JMenuItem();
mnuWinRegisters = new javax.swing.JMenuItem();
jSeparator2 = new javax.swing.JSeparator();
mnuWinVIOffsetCalibration = new javax.swing.JMenuItem();
mnuWinFrequencyCalibration = new javax.swing.JMenuItem();
mnuWinPhaseCalibration = new javax.swing.JMenuItem();
mnuWinEnergyGainCalibration = new javax.swing.JMenuItem();
mnuWinMCCalibration = new javax.swing.JMenuItem();

getContentPane().setLayout(new javax.swing.BoxLayout(getContentPane(), javax.swing.BoxLayout.Y_AXIS));

setTitle("Energy Measurement");
setName("EnergyMeasurementFrame");
addWindowListener(new java.awt.event.WindowAdapter() {
    public void windowClosing(java.awt.event.WindowEvent evt) {
        exitForm(evt);
    }
});

pnlCalibrateVIRMSOFFSET.setLayout(new java.awt.GridLayout(6, 4));

pnlCalibrateVIRMSOFFSET.setBorder(new javax.swing.border.TitledBorder("V and I Offset Calibration"));
btnOffPhaseA.setText("Phase A");
btnOffPhaseA.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnOffPhaseAActionPerformed(evt);
    }
});

pnlCalibrateVIRMSOFFSET.add(btnOffPhaseA);

btnOffPhaseB.setText("Phase B");
btnOffPhaseB.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnOffPhaseBActionPerformed(evt);
    }
});

pnlCalibrateVIRMSOFFSET.add(btnOffPhaseB);

btnOffPhaseC.setText("Phase C");
btnOffPhaseC.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnOffPhaseCActionPerformed(evt);
    }
});

pnlCalibrateVIRMSOFFSET.add(btnOffPhaseC);

btnCalVIOff.setText("Cal: A");
btnCalVIOff.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnCalVIOffActionPerformed(evt);
    }
});

pnlCalibrateVIRMSOFFSET.add(btnCalVIOff);

btnV_1CC.setText("V_1[V]:");
btnV_1CC.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        btnV_1CCActionPerformed(evt);
    }
});

pnlCalibrateVIRMSOFFSET.add(btnV_1CC);

txtV_1CC.setText("200");
pnlCalibrateVIRMSOFFSET.add(txtV_1CC);

btnV_2CC.setText("V_2[V]:");
btnV_2CC.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnV_2CCActionPerformed(evt);
    }
});

pnlCalibrateVIRMSOFFSET.add(btnV_2CC);

txtV_2CC.setText("20");
pnlCalibrateVIRMSOFFSET.add(txtV_2CC);

btnI_1CC.setText("I_1[A]:");
btnI_1CC.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnI_1CCActionPerformed(evt);
    }
});

pnlCalibrateVIRMSOFFSET.add(btnI_1CC);

txtI_1CC.setText("10");
pnlCalibrateVIRMSOFFSET.add(txtI_1CC);

btnI_2CC.setText("I_2[A]:");
btnI_2CC.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnI_2CCActionPerformed(evt);
    }
});

pnlCalibrateVIRMSOFFSET.add(btnI_2CC);

txtI_2CC.setText("1");
pnlCalibrateVIRMSOFFSET.add(txtI_2CC);

jLabel25.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel25.setText("VaRMS:");
pnlCalibrateVIRMSOFFSET.add(jLabel25);

lblVaRMSOffsetRegister.setText("0x0");
pnlCalibrateVIRMSOFFSET.add(lblVaRMSOffsetRegister);

jLabel28.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel28.setText("IaRMS:");
pnlCalibrateVIRMSOFFSET.add(jLabel28);

lblIaRMSOffsetRegister.setText("0x0");
pnlCalibrateVIRMSOFFSET.add(lblIaRMSOffsetRegister);

jLabel26.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel26.setText("VbRMS:");
pnlCalibrateVIRMSOFFSET.add(jLabel26);

lblVbRMSOffsetRegister.setText("0x0");
pnlCalibrateVIRMSOFFSET.add(lblVbRMSOffsetRegister);

jLabel30.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel30.setText("IbRMS:");
pnlCalibrateVIRMSOFFSET.add(jLabel30);

lblIbRMSOffsetRegister.setText("0x0");
pnlCalibrateVIRMSOFFSET.add(lblIbRMSOffsetRegister);

jLabel27.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel27.setText("VcRMS:");
pnlCalibrateVIRMSOFFSET.add(jLabel27);

lblVcRMSOffsetRegister.setText("0x0");
pnlCalibrateVIRMSOFFSET.add(lblVcRMSOffsetRegister);
```

```
jLabel32.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel32.setText("IcRMS:");
pnlCalibrateVIRMSOFFSET.add(jLabel32);

lblIcRMSOffsetRegister.setText("0x0");
pnlCalibrateVIRMSOFFSET.add(lblIcRMSOffsetRegister);

getContentPane().add(pnlCalibrateVIRMSOFFSET);

pnlFrequencyCalibration.setLayout(new java.awt.GridLayout(3, 0));

pnlFrequencyCalibration.setBorder(new javax.swing.border.TitledBorder("Frequency Calibration"));
jLabel31.setText("Freq [Hz]:");
pnlFrequencyCalibration.add(jLabel31);

txtFreqValue.setText("50");
pnlFrequencyCalibration.add(txtFreqValue);

jLabel33.setText("Freq Register:");
pnlFrequencyCalibration.add(jLabel33);

txtFreqMeasure.setText("0");
pnlFrequencyCalibration.add(txtFreqMeasure);

btnCalFreq.setText("Calibrate");
btnCalFreq.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnCalFreqActionPerformed(evt);
    }
});

pnlFrequencyCalibration.add(btnCalFreq);

lblFreqMeasure.setText("0x0");
pnlFrequencyCalibration.add(lblFreqMeasure);

getContentPane().add(pnlFrequencyCalibration);

pnlPhaseCallibration.setLayout(new java.awt.GridLayout(0, 4));

pnlPhaseCallibration.setBorder(new javax.swing.border.TitledBorder("Phase Callibration"));
btnPhaseUpdateForm.setText("Update Form");
btnPhaseUpdateForm.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnPhaseUpdateFormActionPerformed(evt);
    }
});

pnlPhaseCallibration.add(btnPhaseUpdateForm);

jLabel47.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel47.setText("Phase A");
pnlPhaseCallibration.add(jLabel47);

jLabel53.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel53.setText("Phase B");
pnlPhaseCallibration.add(jLabel53);

jLabel54.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel54.setText("Phase C");
pnlPhaseCallibration.add(jLabel54);

btnUpdatePhaseRegister.setText("Update Registers Directly");
btnUpdatePhaseRegister.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnUpdatePhaseRegisterActionPerformed(evt);
    }
});

pnlPhaseCallibration.add(btnUpdatePhaseRegister);

txtPhaseACallibration.setText("0");
pnlPhaseCallibration.add(txtPhaseACallibration);

txtPhaseBCallibration.setText("0");
pnlPhaseCallibration.add(txtPhaseBCallibration);

txtPhaseCCallibration.setText("0");
pnlPhaseCallibration.add(txtPhaseCCallibration);
```

```
jLabel56.setText("Power Factor 1");
pnlPhaseCallibration.add(jLabel56);

txtPhase1APF.setText("1.0");
pnlPhaseCallibration.add(txtPhase1APF);

txtPhase1BPF.setText("1.0");
pnlPhaseCallibration.add(txtPhase1BPF);

txtPhase1CPF.setText("1.0");
pnlPhaseCallibration.add(txtPhase1CPF);

jLabel57.setText("Watt 1");
pnlPhaseCallibration.add(jLabel57);

txtPhase1AW.setText("500");
pnlPhaseCallibration.add(txtPhase1AW);

txtPhase1BW.setText("500");
pnlPhaseCallibration.add(txtPhase1BW);

txtPhase1CW.setText("500");
pnlPhaseCallibration.add(txtPhase1CW);

jLabel61.setText("Cycles:");
pnlPhaseCallibration.add(jLabel61);

lblPhase1AWRMS.setText("0x0");
pnlPhaseCallibration.add(lblPhase1AWRMS);

lblPhase1BWRMS.setText("0x0");
pnlPhaseCallibration.add(lblPhase1BWRMS);

lblPhase1CWRMS.setText("0x0");
pnlPhaseCallibration.add(lblPhase1CWRMS);

txtPhaseCycles.setText("250");
txtPhaseCycles.setToolTipText("Number of Cycles for WattHr Registers to accumulate");
pnlPhaseCallibration.add(txtPhaseCycles);

btnPhaseMeasurement1A.setText("Measurement 1A");
btnPhaseMeasurement1A.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnPhaseMeasurement1AActionPerformed(evt);
    }
});

pnlPhaseCallibration.add(btnPhaseMeasurement1A);

btnPhaseMeasurement1B.setText("Measurement 1B");
btnPhaseMeasurement1B.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnPhaseMeasurement1BActionPerformed(evt);
    }
});

pnlPhaseCallibration.add(btnPhaseMeasurement1B);

btnPhaseMeasurement1C.setText("Measurement 1C");
btnPhaseMeasurement1C.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnPhaseMeasurement1CActionPerformed(evt);
    }
});

pnlPhaseCallibration.add(btnPhaseMeasurement1C);

jLabel58.setText("Power Factor 2");
pnlPhaseCallibration.add(jLabel58);

txtPhase2APF.setText("0.5");
pnlPhaseCallibration.add(txtPhase2APF);

txtPhase2BPF.setText("0.5");
pnlPhaseCallibration.add(txtPhase2BPF);

txtPhase2CPF.setText("0.5");
pnlPhaseCallibration.add(txtPhase2CPF);
```



```
jLabel59.setText("Watt 2");
pnlPhaseCallibration.add(jLabel59);

txtPhase2AW.setText("500");
pnlPhaseCallibration.add(txtPhase2AW);

txtPhase2BW.setText("500");
pnlPhaseCallibration.add(txtPhase2BW);

txtPhase2CW.setText("500");
pnlPhaseCallibration.add(txtPhase2CW);

pnlPhaseCallibration.add(jLabel62);

lblPhase2AWRMS.setText("0x0");
pnlPhaseCallibration.add(lblPhase2AWRMS);

lblPhase2BWRMS.setText("0x0");
pnlPhaseCallibration.add(lblPhase2BWRMS);

lblPhase2CWRMS.setText("0x0");
pnlPhaseCallibration.add(lblPhase2CWRMS);

btnPhaseCallibrate.setText("Callibrate");
btnPhaseCallibrate.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnPhaseCallibrateActionPerformed(evt);
    }
});

pnlPhaseCallibration.add(btnPhaseCallibrate);

btnPhaseMeasurement2A.setText("Measurement 2A");
btnPhaseMeasurement2A.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnPhaseMeasurement2AActionPerformed(evt);
    }
});

pnlPhaseCallibration.add(btnPhaseMeasurement2A);

btnPhaseMeasurement2B.setText("Measurement 2B");
btnPhaseMeasurement2B.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnPhaseMeasurement2BActionPerformed(evt);
    }
});

pnlPhaseCallibration.add(btnPhaseMeasurement2B);

btnPhaseMeasurement2C.setText("Measurement 2C");
btnPhaseMeasurement2C.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnPhaseMeasurement2CActionPerformed(evt);
    }
});

pnlPhaseCallibration.add(btnPhaseMeasurement2C);

getContentPane().add(pnlPhaseCallibration);

pnlEnergyGainCalibration.setLayout(new java.awt.GridLayout(10, 4));

pnlEnergyGainCalibration.setBorder(new javax.swing.border.TitledBorder("Power Gain Calibration (Line Cycle)"));
btnReset.setText("Reset");
btnReset.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnResetActionPerformed(evt);
    }
});

pnlEnergyGainCalibration.add(btnReset);

btnGainCalPhaseA.setText("Cal Phase A");
btnGainCalPhaseA.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnGainCalPhaseAActionPerformed(evt);
    }
});
```

```
pnlEnergyGainCalibration.add(btnGainCalPhaseA);

btnGainCalPhaseB.setText("Cal Phase B");
btnGainCalPhaseB.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnGainCalPhaseBActionPerformed(evt);
    }
});

pnlEnergyGainCalibration.add(btnGainCalPhaseB);

btnGainCalPhaseC.setText("Cal Phase C");
btnGainCalPhaseC.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnGainCalPhaseCActionPerformed(evt);
    }
});

pnlEnergyGainCalibration.add(btnGainCalPhaseC);

jLabel35.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel35.setText("Active [W]:");
pnlEnergyGainCalibration.add(jLabel35);

txtGainCalW_A.setText("0");
pnlEnergyGainCalibration.add(txtGainCalW_A);

txtGainCalW_B.setText("0");
pnlEnergyGainCalibration.add(txtGainCalW_B);

txtGainCalW_C.setText("0");
pnlEnergyGainCalibration.add(txtGainCalW_C);

jLabel36.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel36.setText("Reactive  [var]:");
pnlEnergyGainCalibration.add(jLabel36);

txtGainCalVAR_A.setText("0");
pnlEnergyGainCalibration.add(txtGainCalVAR_A);

txtGainCalVAR_B.setText("0");
pnlEnergyGainCalibration.add(txtGainCalVAR_B);

txtGainCalVAR_C.setText("0");
pnlEnergyGainCalibration.add(txtGainCalVAR_C);

jLabel39.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel39.setText("xWG:");
pnlEnergyGainCalibration.add(jLabel39);

lblAWG.setText("0x0");
pnlEnergyGainCalibration.add(lblAWG);

lblBWG.setText("0x0");
pnlEnergyGainCalibration.add(lblBWG);

lblCWG.setText("0x0");
pnlEnergyGainCalibration.add(lblCWG);

jLabel45.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel45.setText("xVAG:");
pnlEnergyGainCalibration.add(jLabel45);

lblAVAG.setText("0x0");
pnlEnergyGainCalibration.add(lblAVAG);

lblBVAG.setText("0x0");
pnlEnergyGainCalibration.add(lblBVAG);

lblCVAG.setText("0x0");
pnlEnergyGainCalibration.add(lblCVAG);

jLabel49.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel49.setText("xVARG:");
pnlEnergyGainCalibration.add(jLabel49);

lblAVARG.setText("0x0");
pnlEnergyGainCalibration.add(lblAVARG);
```

```
lblBVARG.setText("0x0");
pnlEnergyGainCalibration.add(lblBVARG);

lblCVARG.setText("0x0");
pnlEnergyGainCalibration.add(lblCVARG);

jLabel41.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel41.setText("MCU xWDIV:");
pnlEnergyGainCalibration.add(jLabel41);

lblMCUWDIVA.setText("0x0");
pnlEnergyGainCalibration.add(lblMCUWDIVA);

lblMCUWDIVB.setText("0x0");
pnlEnergyGainCalibration.add(lblMCUWDIVB);

lblMCUWDIVC.setText("0x0");
pnlEnergyGainCalibration.add(lblMCUWDIVC);

jLabel51.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel51.setText("MCU xVARDIV:");
pnlEnergyGainCalibration.add(jLabel51);

lblMCUVARDIVA.setText("0x0");
pnlEnergyGainCalibration.add(lblMCUVARDIVA);

lblMCUVARDIVB.setText("0x0");
pnlEnergyGainCalibration.add(lblMCUVARDIVB);

lblMCUVARDIVC.setText("0x0");
pnlEnergyGainCalibration.add(lblMCUVARDIVC);

jLabel55.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel55.setText("MCU xVADIV:");
pnlEnergyGainCalibration.add(jLabel55);

lblMCUVADIVA.setText("0x0");
pnlEnergyGainCalibration.add(lblMCUVADIVA);

lblMCUVADIVB.setText("0x0");
pnlEnergyGainCalibration.add(lblMCUVADIVB);

lblMCUVADIVC.setText("0x0");
pnlEnergyGainCalibration.add(lblMCUVADIVC);

jLabel44.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel44.setText("Number of Cycles:");
pnlEnergyGainCalibration.add(jLabel44);

txtGainCycles.setText("250");
pnlEnergyGainCalibration.add(txtGainCycles);

chkFreq.setText("Freq[Hz]");
chkFreq.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
pnlEnergyGainCalibration.add(chkFreq);

jTextField1.setText("50");
pnlEnergyGainCalibration.add(jTextField1);

getContentPane().add(pnlEnergyGainCalibration);

pnlMeasureMents.setLayout(new java.awt.GridLayout(4, 5));

pnlMeasureMents.setBorder(new javax.swing.border.TitledBorder("Measurements"));
chkMonitor.setText("Monitor");
chkMonitor.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        chkMonitorActionPerformed(evt);
    }
});

jPanel1.add(chkMonitor);

chkScale.setText("Scale");
jPanel1.add(chkScale);

chkLog.setText("Log");
chkLog.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        chkLogActionPerformed(evt);
    }
});

jPanel1.add(chkLog);

pnlMeasureMents.add(jPanel1);

jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel4.setText("VRMS [V]");
pnlMeasureMents.add(jLabel4);

jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel5.setText("IRMS [A]");
pnlMeasureMents.add(jLabel5);

jLabel7.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel7.setText("E [Wh]");
pnlMeasureMents.add(jLabel7);

jLabel34.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel34.setText("E [var]");
pnlMeasureMents.add(jLabel34);

jLabel8.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel8.setText("E [VA]");
pnlMeasureMents.add(jLabel8);

jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel1.setText("A");
pnlMeasureMents.add(jLabel1);

lblAVRMS.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblAVRMS.setText("0x0");
pnlMeasureMents.add(lblAVRMS);

lblAIRMS.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblAIRMS.setText("0x0");
pnlMeasureMents.add(lblAIRMS);

lblAWATTHR.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblAWATTHR.setText("0x0");
pnlMeasureMents.add(lblAWATTHR);

lblAVARHR.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblAVARHR.setText("0x0");
pnlMeasureMents.add(lblAVARHR);

lblAVAHR.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblAVAHR.setText("0x0");
pnlMeasureMents.add(lblAVAHR);

jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel2.setText("B");
pnlMeasureMents.add(jLabel2);

lblBVRMS.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblBVRMS.setText("0x0");
pnlMeasureMents.add(lblBVRMS);

lblBIRMS.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblBIRMS.setText("0x0");
pnlMeasureMents.add(lblBIRMS);

lblBWATTHR.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblBWATTHR.setText("0x0");
pnlMeasureMents.add(lblBWATTHR);

lblBVARHR.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblBVARHR.setText("0x0");
pnlMeasureMents.add(lblBVARHR);

lblBVAHR.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblBVAHR.setText("0x0");
pnlMeasureMents.add(lblBVAHR);

jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel3.setText("C");
jLabel3.setToolTipText("");
```

```
pnlMeasureMents.add(jLabel3);

lblCVRMS.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblCVRMS.setText("0x0");
pnlMeasureMents.add(lblCVRMS);

lblCIRMS.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblCIRMS.setText("0x0");
pnlMeasureMents.add(lblCIRMS);

lblCWATTHR.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblCWATTHR.setText("0x0");
pnlMeasureMents.add(lblCWATTHR);

lblCVARHR.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblCVARHR.setText("0x0");
pnlMeasureMents.add(lblCVARHR);

lblCVAHR.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblCVAHR.setText("0x0");
pnlMeasureMents.add(lblCVAHR);

getContentPane().add(pnlMeasureMents);

pnlEnergyMeasureMent.setLayout(new java.awt.GridLayout(0, 5));

pnlEnergyMeasureMent.setBorder(new javax.swing.border.TitledBorder("Energy Measurements"));
btnMeasureStart.setText("Start");
btnMeasureStart.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnMeasureStartActionPerformed(evt);
    }
});

pnlEnergyMeasureMent.add(btnMeasureStart);

btnMeasureStop.setText("Stop");
btnMeasureStop.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnMeasureStopActionPerformed(evt);
    }
});

pnlEnergyMeasureMent.add(btnMeasureStop);

jLabel29.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel29.setText("Phase A");
pnlEnergyMeasureMent.add(jLabel29);

jLabel50.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel50.setText("Phase B");
pnlEnergyMeasureMent.add(jLabel50);

jLabel52.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel52.setText("Phase C");
pnlEnergyMeasureMent.add(jLabel52);

jLabel43.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel43.setText("Active [Wh]:");
pnlEnergyMeasureMent.add(jLabel43);

lblEnergyW.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblEnergyW.setText("0");
pnlEnergyMeasureMent.add(lblEnergyW);

lblEnergyWA.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblEnergyWA.setText("0");
pnlEnergyMeasureMent.add(lblEnergyWA);

lblEnergyWB.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblEnergyWB.setText("0");
pnlEnergyMeasureMent.add(lblEnergyWB);

lblEnergyWC.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblEnergyWC.setText("0");
pnlEnergyMeasureMent.add(lblEnergyWC);

btnEnergyUpdateValues.setText("Update Values");
btnEnergyUpdateValues.addActionListener(new java.awt.event.ActionListener() {
```

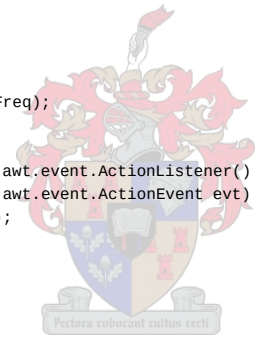
```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnEnergyUpdateValuesActionPerformed(evt);
    }
});

pnlEnergyMeasureMent.add(btnEnergyUpdateValues);

jLabel63.setText("Measured:");
pnlEnergyMeasureMent.add(jLabel63);

txtUpdateWhA.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
txtUpdateWhA.setText("0");
pnlEnergyMeasureMent.add(txtUpdateWhA);

txtUpdateWhB.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
txtUpdateWhB.setText("0");
pnlEnergyMeasureMent.add(txtUpdateWhB);

txtUpdateWhC.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
txtUpdateWhC.setText("0");
pnlEnergyMeasureMent.add(txtUpdateWhC);

pnlEnergyMeasureMent.add(jLabel70);

jLabel71.setText("Update:");
pnlEnergyMeasureMent.add(jLabel71);

btnUpdateWhA.setText("Wh A");
btnUpdateWhA.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnUpdateWhAActionPerformed(evt);
    }
});

pnlEnergyMeasureMent.add(btnUpdateWhA);

btnUpdateWhB.setText("Wh B");
btnUpdateWhB.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnUpdateWhBActionPerformed(evt);
    }
});

pnlEnergyMeasureMent.add(btnUpdateWhB);

btnUpdateWhC.setText("Wh C");
btnUpdateWhC.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnUpdateWhCActionPerformed(evt);
    }
});

pnlEnergyMeasureMent.add(btnUpdateWhC);

jLabel46.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel46.setText("React. [varh]:");
pnlEnergyMeasureMent.add(jLabel46);

lblEnergyVAR.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblEnergyVAR.setText("0");
pnlEnergyMeasureMent.add(lblEnergyVAR);

lblEnergyVARA.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblEnergyVARA.setText("0");
pnlEnergyMeasureMent.add(lblEnergyVARA);

lblEnergyVARB.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblEnergyVARB.setText("0");
pnlEnergyMeasureMent.add(lblEnergyVARB);

lblEnergyVARC.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblEnergyVARC.setText("0");
pnlEnergyMeasureMent.add(lblEnergyVARC);

pnlEnergyMeasureMent.add(jLabel64);

jLabel65.setText("Measured:");
pnlEnergyMeasureMent.add(jLabel65);
```

```
txtUpdatevarhA.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
txtUpdatevarhA.setText("0");
pnlEnergyMeasureMent.add(txtUpdatevarhA);

txtUpdatevarhB.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
txtUpdatevarhB.setText("0");
pnlEnergyMeasureMent.add(txtUpdatevarhB);

txtUpdatevarhC.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
txtUpdatevarhC.setText("0");
pnlEnergyMeasureMent.add(txtUpdatevarhC);

pnlEnergyMeasureMent.add(jLabel68);

jLabel69.setText("Update:");
pnlEnergyMeasureMent.add(jLabel69);

btnUpdatevarhA.setText("varh A");
btnUpdatevarhA.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnUpdatevarhAActionPerformed(evt);
    }
});

pnlEnergyMeasureMent.add(btnUpdatevarhA);

btnUpdatevarhB.setText("varh B");
btnUpdatevarhB.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnUpdatevarhBActionPerformed(evt);
    }
});

pnlEnergyMeasureMent.add(btnUpdatevarhB);

btnUpdatevarhC.setText("varh C");
btnUpdatevarhC.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnUpdatevarhCActionPerformed(evt);
    }
});

pnlEnergyMeasureMent.add(btnUpdatevarhC);

jLabel48.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel48.setText("Compl. [VA]:");
pnlEnergyMeasureMent.add(jLabel48);

lblEnergyVA.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblEnergyVA.setText("0");
pnlEnergyMeasureMent.add(lblEnergyVA);

lblEnergyVAA.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblEnergyVAA.setText("0");
pnlEnergyMeasureMent.add(lblEnergyVAA);

lblEnergyVAB.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblEnergyVAB.setText("0");
pnlEnergyMeasureMent.add(lblEnergyVAB);

lblEnergyVAC.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblEnergyVAC.setText("0");
pnlEnergyMeasureMent.add(lblEnergyVAC);

pnlEnergyMeasureMent.add(jLabel66);

jLabel67.setText("Measured:");
pnlEnergyMeasureMent.add(jLabel67);

txtUpdateVAhA.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
txtUpdateVAhA.setText("0");
pnlEnergyMeasureMent.add(txtUpdateVAhA);

txtUpdateVAhB.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
txtUpdateVAhB.setText("0");
pnlEnergyMeasureMent.add(txtUpdateVAhB);

txtUpdateVAhC.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
txtUpdateVAhC.setText("0");
```

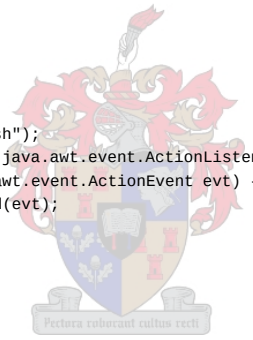
```
pnlEnergyMeasureMent.add(txtUpdateVAhC);

pnlEnergyMeasureMent.add(jLabel72);

jLabel73.setText("Update:");
pnlEnergyMeasureMent.add(jLabel73);

btnUpdateVAhA.setText("VAh A");
btnUpdateVAhA.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnUpdateVAhAActionPerformed(evt);
    }
});

pnlEnergyMeasureMent.add(btnUpdateVAhA);

btnUpdateVAhB.setText("VAh B");
btnUpdateVAhB.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnUpdateVAhBActionPerformed(evt);
    }
});

pnlEnergyMeasureMent.add(btnUpdateVAhB);

btnUpdateVAhC.setText("VAh C");
btnUpdateVAhC.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnUpdateVAhCActionPerformed(evt);
    }
});

pnlEnergyMeasureMent.add(btnUpdateVAhC);

getContentPane().add(pnlEnergyMeasureMent);

pnlOtherMeasurements.setLayout(new java.awt.GridLayout(3, 3));

pnlOtherMeasurements.setBorder(new javax.swing.border.TitledBorder("Other Measurements"));
jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel6.setText("Temp:");
pnlOtherMeasurements.add(jLabel6);

lblTEMP.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblTEMP.setText("0");
pnlOtherMeasurements.add(lblTEMP);

jLabel9.setText("degC");
pnlOtherMeasurements.add(jLabel9);

jLabel10.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel10.setText("Freq:");
pnlOtherMeasurements.add(jLabel10);

lblFREQ.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblFREQ.setText("0");
pnlOtherMeasurements.add(lblFREQ);

jLabel11.setText("Hz");
pnlOtherMeasurements.add(jLabel11);

jLabel24.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel24.setText("Protocol:");
pnlOtherMeasurements.add(jLabel24);

lblTotalPackets.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblTotalPackets.setText("0");
pnlOtherMeasurements.add(lblTotalPackets);

lblAcknowledge.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblAcknowledge.setText("NO");
pnlOtherMeasurements.add(lblAcknowledge);

getContentPane().add(pnlOtherMeasurements);

pnlRegisters.setLayout(new java.awt.GridLayout(11, 2));

pnlRegisters.setBorder(new javax.swing.border.TitledBorder("Registers"));
jLabel12.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
```

```
jLabel12.setText("OPMODE:");
pnlRegisters.add(jLabel12);

lblOPMODE.setText("0x000");
pnlRegisters.add(lblOPMODE);

jLabel13.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel13.setText("MMODE:");
pnlRegisters.add(jLabel13);

lblMMODE.setText("0x000");
pnlRegisters.add(lblMMODE);

jLabel14.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel14.setText("WAVMODE:");
pnlRegisters.add(jLabel14);

lblWAVMODE.setText("0x000");
pnlRegisters.add(lblWAVMODE);

jLabel15.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel15.setText("COMPMODE:");
pnlRegisters.add(jLabel15);

lblCOMPMODE.setText("0x000");
pnlRegisters.add(lblCOMPMODE);

jLabel16.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel16.setText("LCYCMODE:");
pnlRegisters.add(jLabel16);

lblLCYCMODE.setText("0x000");
pnlRegisters.add(lblLCYCMODE);

jLabel17.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel17.setText("MASK:");
pnlRegisters.add(jLabel17);

lblMASK.setText("0x000");
pnlRegisters.add(lblMASK);

jLabel18.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel18.setText("RSTATUS:");
pnlRegisters.add(jLabel18);

lblRSTATUS.setText("0x000");
pnlRegisters.add(lblRSTATUS);

jLabel37.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel37.setText("LINECYC:");
pnlRegisters.add(jLabel37);

lblLINECYC.setText("0x000");
pnlRegisters.add(lblLINECYC);

jLabel38.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel38.setText("WDIV:");
pnlRegisters.add(jLabel38);

lblWDIV.setText("0x000");
pnlRegisters.add(lblWDIV);

jLabel40.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel40.setText("VADIV:");
pnlRegisters.add(jLabel40);

lblVADIV.setText("0x000");
pnlRegisters.add(lblVADIV);

jLabel42.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel42.setText("VARDIV:");
pnlRegisters.add(jLabel42);

lblVARDIV.setText("0x000");
pnlRegisters.add(lblVARDIV);

getContentPane().add(pnlRegisters);

pnlMCCalibration.setLayout(new java.awt.GridLayout(8, 2));
```

```
pnlMCCalibration.setBorder(new javax.swing.border.TitledBorder("Meter Constant (MC) Calibration"));
jLabel19.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel19.setText("Meter Constant[imp/kWh]:");
pnlMCCalibration.add(jLabel19);

txtMC.setText("1000");
pnlMCCalibration.add(txtMC);

jLabel20.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel20.setText("Measured Power[kW]:");
pnlMCCalibration.add(jLabel20);

txtMCMeasurePowerW.setText("0");
pnlMCCalibration.add(txtMCMeasurePowerW);

jLabel21.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel21.setText("Measure W CF Freq [Hz]:");
pnlMCCalibration.add(jLabel21);

txtMCMeasureWFreq.setText("0");
pnlMCCalibration.add(txtMCMeasureWFreq);

jLabel22.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel22.setText("Measured VAR [VAr]:");
pnlMCCalibration.add(jLabel22);

txtMCMeasurePowerVAR.setText("0");
pnlMCCalibration.add(txtMCMeasurePowerVAR);

jLabel23.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel23.setText("MEasured VAR CF Freq [Hz]:");
jLabel23.setToolTipText("");
pnlMCCalibration.add(jLabel23);

txtMCMeasureVARFreq.setText("0");
pnlMCCalibration.add(txtMCMeasureVARFreq);

btnCalWMC.setText("Calibrate W MC");
btnCalWMC.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnCalWMCActionPerformed(evt);
    }
});

pnlMCCalibration.add(btnCalWMC);

btnCalVAMC.setText("Calibrate VA MC");
btnCalVAMC.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnCalVAMCActionPerformed(evt);
    }
});

pnlMCCalibration.add(btnCalVAMC);

lblAPCFNum.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblAPCFNum.setText("APCFnum");
lblAPCFNum.setToolTipText("");
pnlMCCalibration.add(lblAPCFNum);

lblVARCFNum.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblVARCFNum.setText("VARCFnum");
pnlMCCalibration.add(lblVARCFNum);

lblAPCFDen.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblAPCFDen.setText("APCFden");
pnlMCCalibration.add(lblAPCFDen);

lblVARCFDen.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
lblVARCFDen.setText("VARCFden");
pnlMCCalibration.add(lblVARCFDen);

getContentPane().add(pnlMCCalibration);

mnuFile.setText("File");
mnuOpen.setText("Open");
mnuOpen.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuOpenActionPerformed(evt);
```



```
    }
});

mnuFile.add(mnuOpen);

mnuSave.setText("Save");
mnuSave.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuSaveActionPerformed(evt);
    }
});

mnuFile.add(mnuSave);

jMenuBar1.add(mnuFile);

mnuCallibration.setText("Calibration");
mnuCFLEDsOn.setText("Callibration LEDs ON");
mnuCFLEDsOn.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuCFLEDsOnActionPerformed(evt);
    }
});

mnuCallibration.add(mnuCFLEDsOn);

mnuCFLEDsOFF.setText("Callibration LEDs OFF");
mnuCFLEDsOFF.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuCFLEDsOFFActionPerformed(evt);
    }
});

mnuCallibration.add(mnuCFLEDsOFF);

mnuCallibration.add(jSeparator1);

mnuWriteToFlash.setText("Write to Flash");
mnuWriteToFlash.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuWriteToFlashActionPerformed(evt);
    }
});

mnuCallibration.add(mnuWriteToFlash);

mnuReadFlash.setText("Read Flash");
mnuReadFlash.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuReadFlashActionPerformed(evt);
    }
});

mnuCallibration.add(mnuReadFlash);

jMenuBar1.add(mnuCallibration);

mnuMCU.setText("MCU");
mnuALLREGISTERS.setText("Dump All Registers");
mnuALLREGISTERS.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuALLREGISTERSActionPerformed(evt);
    }
});

mnuMCU.add(mnuALLREGISTERS);

jMenuBar1.add(mnuMCU);

mnuWindows.setText("Windows");
mnuWinEnergyMeasurements.setText("Energy Measurements");
mnuWinEnergyMeasurements.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuWinEnergyMeasurementsActionPerformed(evt);
    }
});

mnuWindows.add(mnuWinEnergyMeasurements);
```



```
mnuWinMeasurements.setText("Measurements");
mnuWinMeasurements.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuWinMeasurementsActionPerformed(evt);
    }
});

mnuWindows.add(mnuWinMeasurements);

mnuWinOtherMeasurememets.setText("Other Measurements");
mnuWinOtherMeasurememets.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuWinOtherMeasurememetsActionPerformed(evt);
    }
});

mnuWindows.add(mnuWinOtherMeasurememets);

mnuWinRegisters.setText("Registers");
mnuWinRegisters.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuWinRegistersActionPerformed(evt);
    }
});

mnuWindows.add(mnuWinRegisters);

mnuWindows.add(jSeparator2);

mnuWinVIOffsetCalibration.setText("Offset Calibration (V&I)");
mnuWinVIOffsetCalibration.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuWinVIOffsetCalibrationActionPerformed(evt);
    }
});

mnuWindows.add(mnuWinVIOffsetCalibration);

mnuWinFrequencyCalibration.setText("Frequency Calibration");
mnuWinFrequencyCalibration.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuWinFrequencyCalibrationActionPerformed(evt);
    }
});

mnuWindows.add(mnuWinFrequencyCalibration);

mnuWinPhaseCalibration.setText("Phase Calibration");
mnuWinPhaseCalibration.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuWinPhaseCalibrationActionPerformed(evt);
    }
});

mnuWindows.add(mnuWinPhaseCalibration);

mnuWinEnergyGainCalibration.setText("Energy Gain Calibration");
mnuWinEnergyGainCalibration.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuWinEnergyGainCalibrationActionPerformed(evt);
    }
});

mnuWindows.add(mnuWinEnergyGainCalibration);

mnuWinMCCalibration.setText("Meter Constant Calibration");
mnuWinMCCalibration.setEnabled(false);
mnuWinMCCalibration.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        mnuWinMCCalibrationActionPerformed(evt);
    }
});

mnuWindows.add(mnuWinMCCalibration);

jMenuBar1.add(mnuWindows);

setJMenuBar(jMenuBar1);
```

```
        pack();
    }//GEN-END:initComponents

    private void btnEnergyUpdateValuesActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnEnergyUpdateValuesActionPerformed
        txtUpdateWhA.setText(""+eCal.getMCU_phaseAW_Value());
        txtUpdateWhB.setText(""+eCal.getMCU_phaseBW_Value());
        txtUpdateWhC.setText(""+eCal.getMCU_phaseCW_Value());
        txtUpdatevarhA.setText(""+eCal.getMCU_phaseAVAR_Value());
        txtUpdatevarhB.setText(""+eCal.getMCU_phaseBVAR_Value());
        txtUpdatevarhC.setText(""+eCal.getMCU_phaseCVAR_Value());
        txtUpdateVAhB.setText(""+eCal.getMCU_phaseBVA_Value());
        txtUpdateVAhA.setText(""+eCal.getMCU_phaseAVA_Value());
        txtUpdateVAhC.setText(""+eCal.getMCU_phaseCVA_Value());

        lblEnergyWA.setText(""+eCal.getMCU_phaseAW_RMS());
        lblEnergyWB.setText(""+eCal.getMCU_phaseBW_RMS());
        lblEnergyWC.setText(""+eCal.getMCU_phaseCW_RMS());
        lblEnergyVARA.setText(""+eCal.getMCU_phaseAVAR_RMS());
        lblEnergyVARB.setText(""+eCal.getMCU_phaseBVAR_RMS());
        lblEnergyVARC.setText(""+eCal.getMCU_phaseCVAR_RMS());
        lblEnergyVAA.setText(""+eCal.getMCU_phaseAVA_RMS());
        lblEnergyVAB.setText(""+eCal.getMCU_phaseBVA_RMS());
        lblEnergyVAC.setText(""+eCal.getMCU_phaseCVA_RMS());
    }//GEN-LAST:event_btnEnergyUpdateValuesActionPerformed

    private void btnUpdateVAhCActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnUpdateVAhCActionPerformed
        int tmpVal = 0;
        eCal.setMCU_phaseCVA_Value(txtUpdateVAhC.getText());
        tmpVal = eCal.reCalcPowerGain(eCal.getMCU_phaseCVA_Value(), eCal.getMCU_phaseCVA_RMS(), eCal.MCU_phaseCVA_DIV);
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VADIVC,p.dataInt4(tmpVal)));
    }//GEN-LAST:event_btnUpdateVAhCActionPerformed

    private void btnUpdateVAhBActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnUpdateVAhBActionPerformed
        int tmpVal = 0;
        eCal.setMCU_phaseBVA_Value(txtUpdateVAhB.getText());
        tmpVal = eCal.reCalcPowerGain(eCal.getMCU_phaseBVA_Value(), eCal.getMCU_phaseBVA_RMS(), eCal.MCU_phaseBVA_DIV);
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VADIVB,p.dataInt4(tmpVal)));
    }//GEN-LAST:event_btnUpdateVAhBActionPerformed

    private void btnUpdateVAhAActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnUpdateVAhAActionPerformed
        int tmpVal = 0;
        eCal.setMCU_phaseAVA_Value(txtUpdateVAhA.getText());
        tmpVal = eCal.reCalcPowerGain(eCal.getMCU_phaseAVA_Value(), eCal.getMCU_phaseAVA_RMS(), eCal.MCU_phaseAVA_DIV);
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VADIVA,p.dataInt4(tmpVal)));
    }//GEN-LAST:event_btnUpdateVAhAActionPerformed

    private void btnUpdatevarhCActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnUpdatevarhCActionPerformed
        int tmpVal = 0;
        eCal.setMCU_phaseCVAR_Value(txtUpdatevarhC.getText());
        tmpVal = eCal.reCalcPowerGain(eCal.getMCU_phaseCVAR_Value(), eCal.getMCU_phaseCVAR_RMS(), eCal.MCU_phaseCVAR_DIV);
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VARDIVC,p.dataInt4(tmpVal)));
    }//GEN-LAST:event_btnUpdatevarhCActionPerformed

    private void btnUpdatevarhBActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnUpdatevarhBActionPerformed
        int tmpVal = 0;
        eCal.setMCU_phaseBVAR_Value(txtUpdatevarhB.getText());
        tmpVal = eCal.reCalcPowerGain(eCal.getMCU_phaseBVAR_Value(), eCal.getMCU_phaseBVAR_RMS(), eCal.MCU_phaseBVAR_DIV);
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VARDIVB,p.dataInt4(tmpVal)));
    }//GEN-LAST:event_btnUpdatevarhBActionPerformed

    private void btnUpdatevarhAActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnUpdatevarhAActionPerformed
        int tmpVal = 0;
        eCal.setMCU_phaseAVAR_Value(txtUpdatevarhA.getText());
        tmpVal = eCal.reCalcPowerGain(eCal.getMCU_phaseAVAR_Value(), eCal.getMCU_phaseAVAR_RMS(), eCal.MCU_phaseAVAR_DIV);
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VARDIVA,p.dataInt4(tmpVal)));
    }//GEN-LAST:event_btnUpdatevarhAActionPerformed

    private void btnUpdateWhCActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnUpdateWhCActionPerformed
        int tmpVal = 0;
        eCal.setMCU_phaseCW_Value(txtUpdateWhC.getText());
        tmpVal = eCal.reCalcPowerGain(eCal.getMCU_phaseCW_Value(), eCal.getMCU_phaseCW_RMS(), eCal.MCU_phaseCW_DIV);
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_WDIVC,p.dataInt4(tmpVal)));
    }//GEN-LAST:event_btnUpdateWhCActionPerformed

    private void btnUpdateWhBActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnUpdateWhBActionPerformed
        int tmpVal = 0;
        eCal.setMCU_phaseBW_Value(txtUpdateWhB.getText());
        tmpVal = eCal.reCalcPowerGain(eCal.getMCU_phaseBW_Value(), eCal.getMCU_phaseBW_RMS(), eCal.MCU_phaseBW_DIV);
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_WDIVB,p.dataInt4(tmpVal)));
```

```
}//GEN-LAST:event_btnUpdateWhBActionPerformed

private void mnuWinPhaseCalibrationActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuWinPhaseCalibrationActionPerforme
    this.pnlPhaseCallibration.setVisible(!pnlPhaseCallibration.isVisible());
}//GEN-LAST:event_mnuWinPhaseCalibrationActionPerformed

private void btnUpdateWhAActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnUpdateWhAActionPerformed
    int tmpVal = 0;
    eCal.setMCU_phaseAW_Value(txtUpdateWhA.getText());
    tmpVal = eCal.reCalcPowerGain(eCal.getMCU_phaseAW_Value(), eCal.getMCU_phaseAW_RMS(), eCal.MCU_phaseAW_DIV);
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_WDIVA,p.dataInt4(tmpVal)));
}//GEN-LAST:event_btnUpdateWhAActionPerformed

private void btnPhaseCallibrateActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnPhaseCallibrateActionPerformed
    byte tmpVal = 0;
    System.out.println("PhaseA");
    tmpVal = eCal.calPhaseOffset(eCal.getPhaseCalPF1A(),eCal.getPhaseCalW1A(), eCal.getPhaseCalWRMS1A(), eCal.getPhaseCalPF2A(),eCal.getPh
    serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_APHCAL,p.data(tmpVal)));

    System.out.println("PhaseB");
    tmpVal = eCal.calPhaseOffset(eCal.getPhaseCalPF1B(),eCal.getPhaseCalW1B(), eCal.getPhaseCalWRMS1B(), eCal.getPhaseCalPF2B(),eCal.getPh
    serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_BPHCAL,p.data(tmpVal)));

    System.out.println("PhaseC");
    tmpVal = eCal.calPhaseOffset(eCal.getPhaseCalPF1C(),eCal.getPhaseCalW1C(), eCal.getPhaseCalWRMS1C(), eCal.getPhaseCalPF2C(),eCal.getPh
    serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_CPHCAL,p.data(tmpVal)));

}//GEN-LAST:event_btnPhaseCallibrateActionPerformed

private void btnPhaseUpdateFormActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnPhaseUpdateFormActionPerformed
    lblPhase1AWRMS.setText(""+eCal.getPhaseCalWRMS1A());
    txtPhase1APF.setText(""+eCal.getPhaseCalPF1A());
    txtPhase1AW.setText(""+eCal.getPhaseCalW1A());
    lblPhase1BWRMS.setText(""+eCal.getPhaseCalWRMS1B());
    txtPhase1BPF.setText(""+eCal.getPhaseCalPF1B());
    txtPhase1BW.setText(""+eCal.getPhaseCalW1B());
    lblPhase1CWRMS.setText(""+eCal.getPhaseCalWRMS1C());
    txtPhase1CPF.setText(""+eCal.getPhaseCalPF1C());
    txtPhase1CW.setText(""+eCal.getPhaseCalW1C());
    lblPhase2AWRMS.setText(""+eCal.getPhaseCalWRMS2A());
    txtPhase2APF.setText(""+eCal.getPhaseCalPF2A());
    txtPhase2AW.setText(""+eCal.getPhaseCalW2A());
    lblPhase2BWRMS.setText(""+eCal.getPhaseCalWRMS2B());
    txtPhase2BPF.setText(""+eCal.getPhaseCalPF2B());
    txtPhase2BW.setText(""+eCal.getPhaseCalW2B());
    lblPhase2CWRMS.setText(""+eCal.getPhaseCalWRMS2C());
    txtPhase2CPF.setText(""+eCal.getPhaseCalPF2C());
    txtPhase2CW.setText(""+eCal.getPhaseCalW2C());
}//GEN-LAST:event_btnPhaseUpdateFormActionPerformed

private void btnPhaseMeasurement2CActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnPhaseMeasurement2CActionPerformed
    int LineCyc = Integer.parseInt(txtPhaseCycles.getText());
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_LINECYC, p.dataInt2(LineCyc)));
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_CPHCAL, p.data((byte)0x0)));

    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION,p.data(p.VALUE_MCU_PHASE2C)));
    eCal.setPhaseCalPF2C(txtPhase2CPF.getText());
    eCal.setPhaseCalW2C(txtPhase2CW.getText());
}//GEN-LAST:event_btnPhaseMeasurement2CActionPerformed

private void btnPhaseMeasurement2BActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnPhaseMeasurement2BActionPerformed
    int LineCyc = Integer.parseInt(txtPhaseCycles.getText());
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_LINECYC, p.dataInt2(LineCyc)));
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_BPHCAL, p.data((byte)0x0)));

    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION,p.data(p.VALUE_MCU_PHASE2B)));
    eCal.setPhaseCalPF2B(txtPhase2BPF.getText());
    eCal.setPhaseCalW2B(txtPhase2BW.getText());
}//GEN-LAST:event_btnPhaseMeasurement2BActionPerformed

private void btnPhaseMeasurement2AActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnPhaseMeasurement2AActionPerformed
    int LineCyc = Integer.parseInt(txtPhaseCycles.getText());
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_LINECYC, p.dataInt2(LineCyc)));
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_APHCAL, p.data((byte)0x0)));

    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION,p.data(p.VALUE_MCU_PHASE2A)));
    eCal.setPhaseCalPF2A(txtPhase2APF.getText());
    eCal.setPhaseCalW2A(txtPhase2AW.getText());
}//GEN-LAST:event_btnPhaseMeasurement2AActionPerformed
```

```
private void btnPhaseMeasurement1CActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnPhaseMeasurement1CActionPerformed
    int LineCyc = Integer.parseInt(txtPhaseCycles.getText());
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_LINECYC, p.dataInt2(LineCyc)));
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_CPHCAL, p.data((byte)0x0)));

    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION,p.data(p.VALUE_MCU_PHASE1C)));
    eCal.setPhaseCalPF1C(txtPhase1CPF.getText());
    eCal.setPhaseCalW1C(txtPhase1CW.getText());
}//GEN-LAST:event_btnPhaseMeasurement1CActionPerformed

private void btnPhaseMeasurement1BActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnPhaseMeasurement1BActionPerformed
    int LineCyc = Integer.parseInt(txtPhaseCycles.getText());
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_LINECYC, p.dataInt2(LineCyc)));
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_BPHCAL, p.data((byte)0x0)));

    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION,p.data(p.VALUE_MCU_PHASE1B)));
    eCal.setPhaseCalPF1B(txtPhase1BPF.getText());
    eCal.setPhaseCalW1B(txtPhase1BW.getText());
}//GEN-LAST:event_btnPhaseMeasurement1BActionPerformed

private void btnPhaseMeasurement1AActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnPhaseMeasurement1AActionPerformed
    int LineCyc = Integer.parseInt(txtPhaseCycles.getText());
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_LINECYC, p.dataInt2(LineCyc)));
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_APHCAL, p.data((byte)0x0)));

    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION,p.data(p.VALUE_MCU_PHASE1A)));
    eCal.setPhaseCalPF1A(txtPhase1APF.getText());
    eCal.setPhaseCalW1A(txtPhase1AW.getText());
}//GEN-LAST:event_btnPhaseMeasurement1AActionPerformed

private void btnUpdatePhaseRegisterActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnUpdatePhaseRegisterActionPerforme
    serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_APHCAL,p.data(Byte.valueOf(txtPhaseACallibration.getText()).byteValue())));
    serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_BPHCAL,p.data(Byte.valueOf(txtPhaseBCallibration.getText()).byteValue())));
    serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_CPHCAL,p.data(Byte.valueOf(txtPhaseCCallibration.getText()).byteValue())));
}//GEN-LAST:event_btnUpdatePhaseRegisterActionPerformed

private void chkLogActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_chkLogActionPerformed
    if (chkLog.isSelected()) {
        ff.open(true);
    } else {
        ff.close();
    }
}//GEN-LAST:event_chkLogActionPerformed

private void mnuReadFlashActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuReadFlashActionPerformed
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_FLASH,p.data(p.VALUE_MCU_FLASH_READ)));
}//GEN-LAST:event_mnuReadFlashActionPerformed

private void mnuWriteToFlashActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuWriteToFlashActionPerformed
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_FLASH,p.data(p.VALUE_MCU_FLASH_WRITE)));
}//GEN-LAST:event_mnuWriteToFlashActionPerformed

private void mnuWinMCCalibrationActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuWinMCCalibrationActionPerformed
    this.pnlMCCalibration.setVisible(!pnlMCCalibration.isVisible());
}//GEN-LAST:event_mnuWinMCCalibrationActionPerformed

private void mnuWinEnergyGainCalibrationActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuWinEnergyGainCalibrationActi
    this.pnlEnergyGainCalibration.setVisible(!pnlEnergyGainCalibration.isVisible());
}//GEN-LAST:event_mnuWinEnergyGainCalibrationActionPerformed

private void mnuWinFrequencyCalibrationActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuWinFrequencyCalibrationAction
    this.pnlFrequencyCalibration.setVisible(!pnlFrequencyCalibration.isVisible());
}//GEN-LAST:event_mnuWinFrequencyCalibrationActionPerformed

private void mnuWinVIOffsetCalibrationActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuWinVIOffsetCalibrationActionPe
    this.pnlCalibrateVIRMSOFFSET.setVisible(!pnlCalibrateVIRMSOFFSET.isVisible());
}//GEN-LAST:event_mnuWinVIOffsetCalibrationActionPerformed

private void mnuWinRegistersActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuWinRegistersActionPerformed
    this.pnlRegisters.setVisible(!pnlRegisters.isVisible());
}//GEN-LAST:event_mnuWinRegistersActionPerformed

private void mnuWinOtherMeasurememetsActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuWinOtherMeasurememetsActionPerf
    this.pnlOtherMeasurements.setVisible(!pnlOtherMeasurements.isVisible());
}//GEN-LAST:event_mnuWinOtherMeasurememetsActionPerformed

private void mnuWinMeasurementsActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuWinMeasurementsActionPerformed
    this.pnlMeasureMents.setVisible(!pnlMeasureMents.isVisible());
}//GEN-LAST:event_mnuWinMeasurementsActionPerformed
```

```
private void mnuWinEnergyMeasurementsActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuWinEnergyMeasurementsActionPerf
    this.pnlEnergyMeasureMent.setVisible(!pnlEnergyMeasureMent.isVisible());
}//GEN-LAST:event_mnuWinEnergyMeasurementsActionPerformed

private void btnResetActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnResetActionPerformed
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_WDIVA,p.dataInt4(0x01)));
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_WDIVB,p.dataInt4(0x01)));
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_WDIVC,p.dataInt4(0x01)));
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VARDIVA,p.dataInt4(0x01)));
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VARDIVB,p.dataInt4(0x01)));
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VARDIVC,p.dataInt4(0x01)));
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VADIVA,p.dataInt4(0x01)));
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VADIVB,p.dataInt4(0x01)));
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VADIVC,p.dataInt4(0x01)));
}//GEN-LAST:event_btnResetActionPerformed

private void mnuSaveActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuSaveActionPerformed
    try {
        System.out.println("Saving checkMeterInfo.xml...");
        XMLEncoder xmlE = new XMLEncoder(new BufferedOutputStream(new FileOutputStream("c:/checkMeterInfo.xml")));
        xmlE.writeObject(eCal);
        xmlE.close();
    } catch (Exception e) {
      System.out.println("Error writing XMLfile: "+ e.getMessage());
    }
}//GEN-LAST:event_mnuSaveActionPerformed

private void mnuOpenActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuOpenActionPerformed
    try {
      System.out.println("Loading checkMeterInfo.xml...");
      XMLDecoder xmlD = new XMLDecoder(new BufferedInputStream(new FileInputStream("c:/checkMeterInfo.xml")));
      eCal = (energyCalibration)xmlD.readObject();
      xmlD.close();
      if (eCal == null) {
          eCal = new energyCalibration();
      }
    } catch (Exception e) {
        System.out.println("Error loading XMLfile: "+ e.getMessage());
    }
}//GEN-LAST:event_mnuOpenActionPerformed

private void btnMeasureStopActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnMeasureStopActionPerformed
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_MEASURE_ENERGY,p.data(p.VALUE_MCU_STOP)));
}//GEN-LAST:event_btnMeasureStopActionPerformed

private void btnMeasureStartActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnMeasureStartActionPerformed
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_REGISTERDUMP,p.data(p.VALUE_MCU_MAINSCREEN)));
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_MEASURE_ENERGY,p.data(p.VALUE_MCU_START)));
}//GEN-LAST:event_btnMeasureStartActionPerformed

private void btnCalFreqActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnCalFreqActionPerformed
    eCal.setFreqValue(txtFreqValue.getText());
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION,p.data(p.VALUE_MCU_FREQ)));
}//GEN-LAST:event_btnCalFreqActionPerformed

private void btnGainCalPhaseCActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnGainCalPhaseCActionPerformed
    int LineCyc = Integer.parseInt(txtGainCycles.getText());
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_LINECYC, p.dataInt2(LineCyc)));
    eCal.setPhaseCW_Value(txtGainCalW_C.getText());
    eCal.setPhaseCVAR_Value(txtGainCalVAR_C.getText());
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION,p.data(p.VALUE_MCU_GAIN_C)));
}//GEN-LAST:event_btnGainCalPhaseCActionPerformed

private void btnGainCalPhaseBActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnGainCalPhaseBActionPerformed
    int LineCyc = Integer.parseInt(txtGainCycles.getText());
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_LINECYC, p.dataInt2(LineCyc)));
    eCal.setPhaseBW_Value(txtGainCalW_B.getText());
    eCal.setPhaseBVAR_Value(txtGainCalVAR_B.getText());
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION,p.data(p.VALUE_MCU_GAIN_B)));
}//GEN-LAST:event_btnGainCalPhaseBActionPerformed

private void btnGainCalPhaseAActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnGainCalPhaseAActionPerformed
    int LineCyc = Integer.parseInt(txtGainCycles.getText());
    serial.sendPacket(p.encode(p.COMP_AD7758, p.AD7758_LINECYC, p.dataInt2(LineCyc)));
    eCal.setPhaseAW_Value(txtGainCalW_A.getText());
    eCal.setPhaseAVAR_Value(txtGainCalVAR_A.getText());
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION,p.data(p.VALUE_MCU_GAIN_A)));
}//GEN-LAST:event_btnGainCalPhaseAActionPerformed
```

```
    private void btnCalVAMCActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnCalVAMCActionPerformed
        serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_VARCFNUM,p.dataInt2(0x00)));
        try {
 //           serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_VARCFDEN,p.dataInt2(eCal.calculateMeterConstantDen(Integer.parseInt(txtMC.getT
        } catch (NumberFormatException e) {
            System.out.println(e.getMessage());
        }
    }//GEN-LAST:event_btnCalVAMCActionPerformed

    private void btnCalWMCActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnCalWMCActionPerformed
        serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_APCFNUM,p.dataInt2(0x00)));
        try {
//            serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_APCFDEN,p.dataInt2(eCal.calculateMeterConstantDen(Integer.parseInt(txtMC.getTe
        } catch (NumberFormatException e) {
            System.out.println(e.getMessage());
        }
    }//GEN-LAST:event_btnCalWMCActionPerformed

    private void btnCalVIOffActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnCalVIOffActionPerformed
        int tmpVVal = 0x0;
        int tmpIVal = 0x0;
        System.out.println(btnCalVIOff.getText()+":");
        if (btnCalVIOff.getText() == "Cal: A") {
            tmpVVal = eCal.calculateVoltageOffset(eCal.phaseAVoltageValue1, eCal.phaseAVoltageValue2, eCal.phaseAVoltageRMS1, eCal.phaseAVolta
            serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_AVRMSOS,p.dataInt2(tmpVVal)));
            tmpIVal = eCal.calculateCurrentOffset(eCal.phaseACurrentValue1, eCal.phaseACurrentValue2, eCal.phaseACurrentRMS1, eCal.phaseACurre
            serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_AIRMSOS,p.dataInt2(tmpIVal)));
            eCal.calAV_LSB = eCal.calculateVoltageCurrentLSB(eCal.phaseAVoltageValue1, eCal.phaseAVoltageRMS1);
            eCal.calAI_LSB = eCal.calculateVoltageCurrentLSB(eCal.phaseACurrentValue1, eCal.phaseACurrentRMS1);
            serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION,p.data(p.VALUE_MCU_FLASH)));
        } else if (btnCalVIOff.getText() == "Cal: B") {
            tmpVVal = eCal.calculateVoltageOffset(eCal.phaseBVoltageValue1, eCal.phaseBVoltageValue2, eCal.phaseBVoltageRMS1, eCal.phaseBVolta
            serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_BVRMSOS,p.dataInt2(tmpVVal)));
            tmpIVal = eCal.calculateCurrentOffset(eCal.phaseBCurrentValue1, eCal.phaseBCurrentValue2, eCal.phaseBCurrentRMS1, eCal.phaseBCurre
            serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_BIRMSOS,p.dataInt2(tmpIVal)));
            eCal.calBV_LSB = eCal.calculateVoltageCurrentLSB(eCal.phaseBVoltageValue1, eCal.phaseBVoltageRMS1);
            eCal.calBI_LSB = eCal.calculateVoltageCurrentLSB(eCal.phaseBCurrentValue1, eCal.phaseBCurrentRMS1);
            serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION,p.data(p.VALUE_MCU_FLASH)));
        } else if (btnCalVIOff.getText() == "Cal: C") {
            tmpVVal = eCal.calculateVoltageOffset(eCal.phaseCVoltageValue1, eCal.phaseCVoltageValue2, eCal.phaseCVoltageRMS1, eCal.phaseCVolta
            serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_CVRMSOS,p.dataInt2(tmpVVal)));
            tmpIVal = eCal.calculateCurrentOffset(eCal.phaseCCurrentValue1, eCal.phaseCCurrentValue2, eCal.phaseCCurrentRMS1, eCal.phaseCCurre
            serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_CIRMSOS,p.dataInt2(tmpIVal)));
            eCal.calCV_LSB = eCal.calculateVoltageCurrentLSB(eCal.phaseCVoltageValue1, eCal.phaseCVoltageRMS1);
            eCal.calCI_LSB = eCal.calculateVoltageCurrentLSB(eCal.phaseCCurrentValue1, eCal.phaseCCurrentRMS1);
            serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION,p.data(p.VALUE_MCU_FLASH)));
        }
    }//GEN-LAST:event_btnCalVIOffActionPerformed

    private void btnI_2CCActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnI_2CCActionPerformed
        if (btnCalVIOff.getText() == "Cal: A") {
            eCal.setPhaseACurrentValue2(txtI_2CC.getText());
            serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION, p.data(p.VALUE_MCU_I2_A)));
        } else if (btnCalVIOff.getText() == "Cal: B") {
            eCal.setPhaseBCurrentValue2(txtI_2CC.getText());
            serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION, p.data(p.VALUE_MCU_I2_B)));
        } else if (btnCalVIOff.getText() == "Cal: C") {
            eCal.setPhaseCCurrentValue2(txtI_2CC.getText());
            serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION, p.data(p.VALUE_MCU_I2_C)));
        }
    }//GEN-LAST:event_btnI_2CCActionPerformed

    private void btnI_1CCActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnI_1CCActionPerformed
        if (btnCalVIOff.getText() == "Cal: A") {
            eCal.setPhaseACurrentValue1(txtI_1CC.getText());
            serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION, p.data(p.VALUE_MCU_I1_A)));
        } else if (btnCalVIOff.getText() == "Cal: B") {
            eCal.setPhaseBCurrentValue1(txtI_1CC.getText());
            serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION, p.data(p.VALUE_MCU_I1_B)));
        } else if (btnCalVIOff.getText() == "Cal: C") {
            eCal.setPhaseCCurrentValue1(txtI_1CC.getText());
            serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION, p.data(p.VALUE_MCU_I1_C)));
        }
    }//GEN-LAST:event_btnI_1CCActionPerformed

    private void btnV_2CCActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnV_2CCActionPerformed
        if (btnCalVIOff.getText() == "Cal: A") {
            eCal.setPhaseAVoltageValue2(txtV_2CC.getText());
            serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION, p.data(p.VALUE_MCU_V2_A)));
```

```
    } else if (btnCalVIOff.getText() == "Cal: B") {
        eCal.setPhaseBVoltageValue2(txtV_2CC.getText());
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION, p.data(p.VALUE_MCU_V2_B)));
    } else if (btnCalVIOff.getText() == "Cal: C") {
        eCal.setPhaseCVoltageValue2(txtV_2CC.getText());
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION, p.data(p.VALUE_MCU_V2_C)));
    }
}//GEN-LAST:event_btnV_2CCActionPerformed

private void btnV_1CCActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnV_1CCActionPerformed
    if (btnCalVIOff.getText() == "Cal: A") {
        eCal.setPhaseAVoltageValue1(txtV_1CC.getText());
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION, p.data(p.VALUE_MCU_V1_A)));
    } else if (btnCalVIOff.getText() == "Cal: B") {
        eCal.setPhaseBVoltageValue1(txtV_1CC.getText());
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION, p.data(p.VALUE_MCU_V1_B)));
    } else if (btnCalVIOff.getText() == "Cal: C") {
        eCal.setPhaseCVoltageValue1(txtV_1CC.getText());
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALIBRATION, p.data(p.VALUE_MCU_V1_C)));
    }
}//GEN-LAST:event_btnV_1CCActionPerformed

private void btnOffPhaseCActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnOffPhaseCActionPerformed
    btnCalVIOff.setText("Cal: C");
}//GEN-LAST:event_btnOffPhaseCActionPerformed

private void btnOffPhaseBActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnOffPhaseBActionPerformed
    btnCalVIOff.setText("Cal: B");
}//GEN-LAST:event_btnOffPhaseBActionPerformed

private void btnOffPhaseAActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnOffPhaseAActionPerformed
    btnCalVIOff.setText("Cal: A");
}//GEN-LAST:event_btnOffPhaseAActionPerformed

private void chkMonitorActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_chkMonitorActionPerformed
    if (chkMonitor.isSelected()) {
        serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_REGISTERDUMP,p.data(p.VALUE_MCU_MONITOR)));
    }
}//GEN-LAST:event_chkMonitorActionPerformed

private void mnuALLREGISTERSActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuALLREGISTERSActionPerformed
    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_REGISTERDUMP,p.data(p.VALUE_MCU_MAINSCREEN)));
}//GEN-LAST:event_mnuALLREGISTERSActionPerformed

private void mnuCFLEDsOFFActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuCFLEDsOFFActionPerformed
    serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_OPMODE,p.data((byte)0x04)));
}//GEN-LAST:event_mnuCFLEDsOFFActionPerformed

private void mnuCFLEDsOnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_mnuCFLEDsOnActionPerformed
    serial.sendPacket(p.encode(p.COMP_AD7758,p.AD7758_OPMODE,p.data((byte)0x00)));
}//GEN-LAST:event_mnuCFLEDsOnActionPerformed

/** Exit the Application */
private void exitForm(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_exitForm
    mnuSaveActionPerformed(null);
    System.exit(0);
}//GEN-LAST:event_exitForm

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    new myEnergy().show();
}

public void serialEvent(javax.comm.SerialPortEvent serialPortEvent) {
    boolean scale = true;
    int tmpVal = 0;
    int MC = 0;
    double VA = 0.0;
    scale = chkScale.isSelected();
    if (serialPortEvent.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
        p.decode(serial.in);
        lblTotalPackets.setText(""+p.getTotalPackets()+":"+p.getComponent()+":"+p.getRegister());

        if (p.getComponent() == p.COMP_AD7758) {
            tmpVal = p.getLatestValue();
            switch (p.getRegister()) {
                case protocol.AD7758_AWATTHR:
```

```
    if ((tmpVal & 0x08000) == 0x08000) {
        tmpVal = tmpVal | 0x0FFFF0000;
    }
    if (scale) {
        lblAWATTHR.setText(""+NumberFormat.getInstance().format(tmpVal));//*eCal.calWhLSB));
    } else {
        lblAWATTHR.setText(""+tmpVal);//*eCal.calWhLSB));
    }
    eCal.phaseAW_RMS = tmpVal;
    break;
case protocol.AD7758_BWATTHR:
    if ((tmpVal & 0x08000) == 0x08000) {
        tmpVal = tmpVal | 0x0FFFF0000;
    }
    lblBWATTHR.setText(""+tmpVal);//*eCal.calWhLSB);
    eCal.phaseBW_RMS = tmpVal;
    break;
case protocol.AD7758_CWATTHR:
    if ((tmpVal & 0x08000) == 0x08000) {
        tmpVal = tmpVal | 0x0FFFF0000;
    }
    lblCWATTHR.setText(""+tmpVal);//*eCal.calWhLSB);
    eCal.phaseCW_RMS = tmpVal;
    break;
case protocol.AD7758_AVARHR:
    if ((tmpVal & 0x08000) == 0x08000) {
        tmpVal = tmpVal | 0x0FFFF0000;
    }
    lblAVARHR.setText(""+tmpVal);//*eCal.calVARhLSB);
    eCal.phaseAVAR_RMS = tmpVal;
    break;
case protocol.AD7758_BVARHR:
    if ((tmpVal & 0x08000) == 0x08000) {
        tmpVal = tmpVal | 0x0FFFF0000;
    }
    lblBVARHR.setText(""+tmpVal);//*eCal.calVARhLSB);
    eCal.phaseBVAR_RMS = tmpVal;
    break;
case protocol.AD7758_CVARHR:
    if ((tmpVal & 0x08000) == 0x08000) {
        tmpVal = tmpVal | 0x0FFFF0000;
    }
    lblCVARHR.setText(""+tmpVal);//*eCal.calVARhLSB);
    eCal.phaseCVAR_RMS = tmpVal;
    break;
case protocol.AD7758_AVAHR:
    if ((tmpVal & 0x08000) == 0x08000) {
        tmpVal = tmpVal | 0x0FFFF0000;
    }
    lblAVAHR.setText(""+p.getLatestValue());//*eCal.calVAhLSB);
    eCal.phaseAVA_RMS = p.getLatestValue();
    break;
case protocol.AD7758_BVAHR:
    if ((tmpVal & 0x08000) == 0x08000) {
        tmpVal = tmpVal | 0x0FFFF0000;
    }
    lblBVAHR.setText(""+p.getLatestValue());//*eCal.calVAhLSB);
    eCal.phaseBVA_RMS = p.getLatestValue();
    break;
case protocol.AD7758_CVAHR:
    if ((tmpVal & 0x08000) == 0x08000) {
        tmpVal = tmpVal | 0x0FFFF0000;
    }
    lblCVAHR.setText(""+p.getLatestValue());//*eCal.calVAhLSB);
    eCal.phaseCVA_RMS = p.getLatestValue();
    break;
case protocol.AD7758_AIRMS:
    if (scale) {
        lblAIRMS.setText(""+NumberFormat.getInstance().format(p.getLatestValue()*eCal.calAI_LSB));
    } else {
        lblAIRMS.setText(""+p.getLatestHexValue());
    }
    if (chkLog.isSelected()) ff.appendNextColumn(lblAIRMS.getText());
    break;
case protocol.AD7758_BIRMS:
    if (scale) {
        lblBIRMS.setText(""+NumberFormat.getInstance().format(p.getLatestValue()*eCal.calBI_LSB));
    } else {
        lblBIRMS.setText(""+p.getLatestHexValue());
```

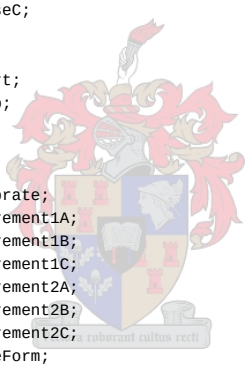
```
    }
    if (chkLog.isSelected()) ff.appendNextColumn(lblBIRMS.getText());
    break;
case protocol.AD7758_CIRMS:
    if (scale) {
        lblCIRMS.setText(""+NumberFormat.getInstance().format(p.getLatestValue()*eCal.calCI_LSB));
    } else {
        lblCIRMS.setText(""+p.getLatestHexValue());
    }
    if (chkLog.isSelected()) ff.appendNextColumn(lblCIRMS.getText());
    break;
case protocol.AD7758_AVRMS:
    if (scale) {
        lblAVRMS.setText(""+NumberFormat.getInstance().format(p.getLatestValue()*eCal.calAV_LSB));
    } else {
        lblAVRMS.setText(""+p.getLatestHexValue());
    }
    if (chkLog.isSelected()) ff.appendNextColumn(lblAVRMS.getText());
    break;
case protocol.AD7758_BVRMS:
    if (scale) {
        lblBVRMS.setText(""+NumberFormat.getInstance().format(p.getLatestValue()*eCal.calBV_LSB));
    } else {
        lblBVRMS.setText(""+p.getLatestHexValue());
    }
    if (chkLog.isSelected()) ff.appendNextColumn(lblBVRMS.getText());
    break;
case protocol.AD7758_CVRMS:
    if (scale) {
        lblCVRMS.setText(""+NumberFormat.getInstance().format(p.getLatestValue()*eCal.calCV_LSB));
    } else {
        lblCVRMS.setText(""+p.getLatestHexValue());
    }
    if (chkLog.isSelected()) ff.appendNextColumn(lblCVRMS.getText());
    break;
case protocol.AD7758_TEMP:
    lblTEMP.setText(""+p.getLatestValue());
    break;
case protocol.AD7758_FREQ:
    lblFREQ.setText("" + (int)(p.getLatestValue()*eCal.calFreq));
    eCal.FreqRMS = p.getLatestValue();
    lblFreqMeasure.setText("" + p.getLatestHexValue());
    txtFreqMeasure.setText("" + p.getLatestValue());
    break;
case protocol.AD7758_OPMODE:
    lblOPMODE.setText(""+p.getLatestHexValue());
    break;
case protocol.AD7758_MMODE:
    lblMMODE.setText(""+p.getLatestHexValue());
    break;
case protocol.AD7758_WAVMODE:
    lblWAVMODE.setText(""+p.getLatestHexValue());
    break;
case protocol.AD7758_COMPMODE:
    lblCOMPMODE.setText(""+p.getLatestHexValue());
    break;
case protocol.AD7758_LCYCMODE:
    lblLCYCMODE.setText(""+p.getLatestHexValue());
    break;
case protocol.AD7758_MASK:
    lblMASK.setText(""+p.getLatestHexValue());
    break;
case protocol.AD7758_RSTATUS:
    lblRSTATUS.setText(""+p.getLatestHexValue());
    break;
case protocol.AD7758_AVRMSOS:
    lblVaRMSOffsetRegister.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_BVRMSOS:
    lblVbRMSOffsetRegister.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_CVRMSOS:
    lblVcRMSOffsetRegister.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_AIRMSOS:
    lblIaRMSOffsetRegister.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_BIRMSOS:
    lblIbRMSOffsetRegister.setText(p.getLatestHexValue());
```

```
    break;
case protocol.AD7758_CIRMSOS:
    lblIcRMSOffsetRegister.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_APCFNUM:
    lblAPCFNum.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_APCFDEN:
    lblAPCFDen.setText(p.getLatestHexValue());
    eCal.calAPCFDEN = p.getLatestValue();
    break;
case protocol.AD7758_VARCFNUM:
    lblVARCFNum.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_VARCFDEN:
    lblVARCFDen.setText(p.getLatestHexValue());
    eCal.calVARCFDEN = p.getLatestValue();
    break;
case protocol.AD7758_LINECYC:
    lblLINECYC.setText(p.getLatestHexValue());
    eCal.calLineCyc = p.getLatestValue();
    break;
case protocol.AD7758_WDIV:
    lblWDIV.setText(p.getLatestHexValue());
    eCal.calWDiv = p.getLatestValue();
    break;
case protocol.AD7758_VADIV:
    lblVADIV.setText(p.getLatestHexValue());
    eCal.calVADiv = p.getLatestValue();
    break;
case protocol.AD7758_VARDIV:
    lblVARDIV.setText(p.getLatestHexValue());
    eCal.calVARDiv = p.getLatestValue();
    break;
case protocol.AD7758_AWG:
    lblAWG.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_BWG:
    lblBWG.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_CWG:
    lblCWG.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_AVAG:
    lblAVAG.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_BVAG:
    lblBVAG.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_CVAG:
    lblCVAG.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_AVARG:
    lblAVARG.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_BVARG:
    lblBVARG.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_CVARG:
    lblCVARG.setText(p.getLatestHexValue());
    break;
case protocol.AD7758_APHCAL:
    if ((tmpVal & 0x040) == 0x040) {
        tmpVal = tmpVal | 0x0FFFFFF00;
    }
    txtPhaseACallibration.setText(""+tmpVal);
    break;
case protocol.AD7758_BPHCAL:
    if ((tmpVal & 0x040) == 0x040) {
        tmpVal = tmpVal | 0x0FFFFFF00;
    }
    txtPhaseBCallibration.setText(""+tmpVal);
    break;
case protocol.AD7758_CPHCAL:
    if ((tmpVal & 0x040) == 0x040) {
        tmpVal = tmpVal | 0x0FFFFFF00;
    }
    txtPhaseCCallibration.setText(""+tmpVal);
    break;
```

```
    }
} else if (p.getComponent() == p.COMP_MCU) {
    switch (p.getRegister()) {
        case protocol.MCU_ACKNOWLEDGE:
            lblAcknowledge.setText("ACK("+p.getLatestHexValue()+")");
            break;
        case protocol.MCU_REGISTERDUMP:
            if (p.getLatestValue() == p.VALUE_MCU_MONITOR) {
                chkMonitorActionPerformed(null);
                if (chkLog.isSelected()) ff.appendNextRow();
            }
            break;
        case protocol.MCU_CALIBRATION:
            eCal.setCurrentCallibrationValue(p.getLatestValue());
            break;
        case protocol.MCU_CURRENTCAL:
            eCal.setCurrentChannel(p.getLatestValue());
            switch (p.getLatestValue()) {
                case protocol.VALUE_MCU_GAIN_A:
                    try {
                        MC = Integer.parseInt(txtMC.getText());
                    } catch (NumberFormatException e) {
                        System.out.println(e.getMessage());
                    }
                    tmpVal = eCal.calculatePowerGain(eCal.phaseAW_Value,eCal.phaseAW_RMS);
                    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_WDIVA,p.dataInt4(tmpVal)));
                    tmpVal = eCal.calculatePowerGain(eCal.phaseAVAR_Value,eCal.phaseAVAR_RMS);
                    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VARDIVA,p.dataInt4(tmpVal)));

                    VA = Math.sqrt((eCal.phaseAVAR_Value*eCal.phaseAVAR_Value)+(eCal.phaseAW_Value*eCal.phaseAW_Value));
                    tmpVal = eCal.calculatePowerGain(VA,eCal.phaseAVA_RMS);
                    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VADIVA,p.dataInt4(tmpVal)));
                    break;
                case protocol.VALUE_MCU_GAIN_B:
                    try {
                        MC = Integer.parseInt(txtMC.getText());
                    } catch (NumberFormatException e) {
                        System.out.println(e.getMessage());
                    }
                    tmpVal = eCal.calculatePowerGain(eCal.phaseBW_Value,eCal.phaseBW_RMS);
                    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_WDIVB,p.dataInt4(tmpVal)));
                    tmpVal = eCal.calculatePowerGain(eCal.phaseBVAR_Value,eCal.phaseBVAR_RMS);
                    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VARDIVB,p.dataInt4(tmpVal)));

                    VA = Math.sqrt((eCal.phaseBVAR_Value*eCal.phaseBVAR_Value)+(eCal.phaseBW_Value*eCal.phaseBW_Value));
                    tmpVal = eCal.calculatePowerGain(VA,eCal.phaseBVA_RMS);
                    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VADIVB,p.dataInt4(tmpVal)));
                    break;
                case protocol.VALUE_MCU_GAIN_C:
                    try {
                        MC = Integer.parseInt(txtMC.getText());
                    } catch (NumberFormatException e) {
                        System.out.println(e.getMessage());
                    }
                    tmpVal = eCal.calculatePowerGain(eCal.phaseCW_Value,eCal.phaseCW_RMS);
                    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_WDIVC,p.dataInt4(tmpVal)));
                    tmpVal = eCal.calculatePowerGain(eCal.phaseCVAR_Value,eCal.phaseCVAR_RMS);
                    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VARDIVC,p.dataInt4(tmpVal)));

                    VA = Math.sqrt((eCal.phaseCVAR_Value*eCal.phaseCVAR_Value)+(eCal.phaseCW_Value*eCal.phaseCW_Value));
                    tmpVal = eCal.calculatePowerGain(VA,eCal.phaseCVA_RMS);
                    serial.sendPacket(p.encode(p.COMP_MCU,p.MCU_CALLIBRATION_VADIVC,p.dataInt4(tmpVal)));
                    break;
            }
            break;
        case protocol.MCU_MEASURE_ENERGY_W:
            lblEnergyW.setText(""+p.getLatestValue());//*eCal.calWhLSB);
            break;
        case protocol.MCU_MEASURE_ENERGY_VA:
            lblEnergyVA.setText(""+p.getLatestValue());//*eCal.calWhLSB);
            break;
        case protocol.MCU_MEASURE_ENERGY_VAR:
            lblEnergyVAR.setText(""+p.getLatestValue());//*eCal.calWhLSB);
            break;
        case protocol.MCU_MEASURE_ENERGY_W_A:
            lblEnergyWA.setText(""+p.getLatestValue());//*eCal.calWhLSB);
            eCal.setMCU_phaseAW_RMS(p.getLatestValue());
            break;
        case protocol.MCU_MEASURE_ENERGY_VA_A:
```

```
            lblEnergyVAA.setText(""+p.getLatestValue());//*eCal.calWhLSB);
            eCal.setMCU_phaseAVA_RMS(p.getLatestValue());
            break;
        case protocol.MCU_MEASURE_ENERGY_VAR_A:
            lblEnergyVARA.setText(""+p.getLatestValue());//*eCal.calWhLSB);
            eCal.setMCU_phaseAVAR_RMS(p.getLatestValue());
            break;
        case protocol.MCU_MEASURE_ENERGY_W_B:
            lblEnergyWB.setText(""+p.getLatestValue());//*eCal.calWhLSB);
            eCal.setMCU_phaseBW_RMS(p.getLatestValue());
            break;
        case protocol.MCU_MEASURE_ENERGY_VA_B:
            lblEnergyVAB.setText(""+p.getLatestValue());//*eCal.calWhLSB);
            eCal.setMCU_phaseBVA_RMS(p.getLatestValue());
            break;
        case protocol.MCU_MEASURE_ENERGY_VAR_B:
            lblEnergyVARB.setText(""+p.getLatestValue());//*eCal.calWhLSB);
            eCal.setMCU_phaseBVAR_RMS(p.getLatestValue());
            break;
        case protocol.MCU_MEASURE_ENERGY_W_C:
            lblEnergyWC.setText(""+p.getLatestValue());//*eCal.calWhLSB);
            eCal.setMCU_phaseCW_RMS(p.getLatestValue());
            break;
        case protocol.MCU_MEASURE_ENERGY_VA_C:
            lblEnergyVAC.setText(""+p.getLatestValue());//*eCal.calWhLSB);
            eCal.setMCU_phaseCVA_RMS(p.getLatestValue());
            break;
        case protocol.MCU_MEASURE_ENERGY_VAR_C:
            lblEnergyVARC.setText(""+p.getLatestValue());//*eCal.calWhLSB);
            eCal.setMCU_phaseCVAR_RMS(p.getLatestValue());
            break;
        case protocol.MCU_MESSAGE:
            switch (p.getLatestValue()) {
                case protocol.VALUE_MCU_OVERFLOW:
                    System.out.println("Overflow occured during callibration!");
            }
            break;
        case protocol.MCU_CALLIBRATION_WDIVA:
            lblMCUWDIVA.setText(p.getLatestHexValue());
            eCal.MCU_phaseAW_DIV = p.getLatestValue();
            break;
        case protocol.MCU_CALLIBRATION_WDIVB:
            lblMCUWDIVB.setText(p.getLatestHexValue());
            eCal.MCU_phaseBW_DIV = p.getLatestValue();
            break;
        case protocol.MCU_CALLIBRATION_WDIVC:
            lblMCUWDIVC.setText(p.getLatestHexValue());
            eCal.MCU_phaseCW_DIV = p.getLatestValue();
            break;
        case protocol.MCU_CALLIBRATION_VARDIVA:
            lblMCUVARDIVA.setText(p.getLatestHexValue());
            eCal.MCU_phaseAVAR_DIV = p.getLatestValue();
            break;
        case protocol.MCU_CALLIBRATION_VARDIVB:
            lblMCUVARDIVB.setText(p.getLatestHexValue());
            eCal.MCU_phaseBVAR_DIV = p.getLatestValue();
            break;
        case protocol.MCU_CALLIBRATION_VARDIVC:
            lblMCUVARDIVC.setText(p.getLatestHexValue());
            eCal.MCU_phaseCVAR_DIV = p.getLatestValue();
            break;
        case protocol.MCU_CALLIBRATION_VADIVA:
            lblMCUVADIVA.setText(p.getLatestHexValue());
            eCal.MCU_phaseAVA_DIV = p.getLatestValue();
            break;
        case protocol.MCU_CALLIBRATION_VADIVB:
            lblMCUVADIVB.setText(p.getLatestHexValue());
            eCal.MCU_phaseBVA_DIV = p.getLatestValue();
            break;
        case protocol.MCU_CALLIBRATION_VADIVC:
            lblMCUVADIVC.setText(p.getLatestHexValue());
            eCal.MCU_phaseCVA_DIV = p.getLatestValue();
            break;
        case protocol.MCU_MEASURE_PHASE_W1A:
            eCal.setPhaseCalWRMS1A(p.getLatestValue());
            lblPhase1AWRMS.setText(p.getLatestHexValue());
            break;
        case protocol.MCU_MEASURE_PHASE_W1B:
            eCal.setPhaseCalWRMS1B(p.getLatestValue());
```

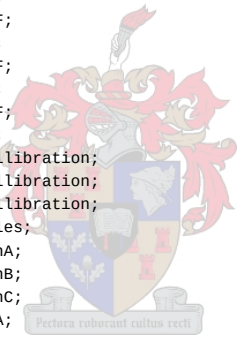
```
                    lblPhase1BWRMS.setText(p.getLatestHexValue());
                    break;
                case protocol.MCU_MEASURE_PHASE_W1C:
                    eCal.setPhaseCalWRMS1C(p.getLatestValue());
                    lblPhase1CWRMS.setText(p.getLatestHexValue());
                    break;
                case protocol.MCU_MEASURE_PHASE_W2A:
                    eCal.setPhaseCalWRMS2A(p.getLatestValue());
                    lblPhase2AWRMS.setText(p.getLatestHexValue());
                    break;
                case protocol.MCU_MEASURE_PHASE_W2B:
                    eCal.setPhaseCalWRMS2B(p.getLatestValue());
                    lblPhase2BWRMS.setText(p.getLatestHexValue());
                    break;
                case protocol.MCU_MEASURE_PHASE_W2C:
                    eCal.setPhaseCalWRMS2C(p.getLatestValue());
                    lblPhase2CWRMS.setText(p.getLatestHexValue());
                    break;
            }
        }
    }
}

// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JButton btnCalFreq;
private javax.swing.JButton btnCalVAMC;
private javax.swing.JButton btnCalVIOff;
private javax.swing.JButton btnCalWMC;
private javax.swing.JButton btnEnergyUpdateValues;
private javax.swing.JButton btnGainCalPhaseA;
private javax.swing.JButton btnGainCalPhaseB;
private javax.swing.JButton btnGainCalPhaseC;
private javax.swing.JButton btnI_1CC;
private javax.swing.JButton btnI_2CC;
private javax.swing.JButton btnMeasureStart;
private javax.swing.JButton btnMeasureStop;
private javax.swing.JButton btnOffPhaseA;
private javax.swing.JButton btnOffPhaseB;
private javax.swing.JButton btnOffPhaseC;
private javax.swing.JButton btnPhaseCallibrate;
private javax.swing.JButton btnPhaseMeasurement1A;
private javax.swing.JButton btnPhaseMeasurement1B;
private javax.swing.JButton btnPhaseMeasurement1C;
private javax.swing.JButton btnPhaseMeasurement2A;
private javax.swing.JButton btnPhaseMeasurement2B;
private javax.swing.JButton btnPhaseMeasurement2C;
private javax.swing.JButton btnPhaseUpdateForm;
private javax.swing.JButton btnReset;
private javax.swing.JButton btnUpdatePhaseRegister;
private javax.swing.JButton btnUpdateVAhA;
private javax.swing.JButton btnUpdateVAhB;
private javax.swing.JButton btnUpdateVAhC;
private javax.swing.JButton btnUpdateWhA;
private javax.swing.JButton btnUpdateWhB;
private javax.swing.JButton btnUpdateWhC;
private javax.swing.JButton btnUpdatevarhA;
private javax.swing.JButton btnUpdatevarhB;
private javax.swing.JButton btnUpdatevarhC;
private javax.swing.JButton btnV_1CC;
private javax.swing.JButton btnV_2CC;
private javax.swing.JCheckBox chkFreq;
private javax.swing.JCheckBox chkLog;
private javax.swing.JCheckBox chkMonitor;
private javax.swing.JCheckBox chkScale;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel14;
private javax.swing.JLabel jLabel15;
private javax.swing.JLabel jLabel16;
private javax.swing.JLabel jLabel17;
private javax.swing.JLabel jLabel18;
private javax.swing.JLabel jLabel19;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel20;
private javax.swing.JLabel jLabel21;
private javax.swing.JLabel jLabel22;
```



```
    private javax.swing.JLabel jLabel23;
    private javax.swing.JLabel jLabel24;
    private javax.swing.JLabel jLabel25;
    private javax.swing.JLabel jLabel26;
    private javax.swing.JLabel jLabel27;
    private javax.swing.JLabel jLabel28;
    private javax.swing.JLabel jLabel29;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel30;
    private javax.swing.JLabel jLabel31;
    private javax.swing.JLabel jLabel32;
    private javax.swing.JLabel jLabel33;
    private javax.swing.JLabel jLabel34;
    private javax.swing.JLabel jLabel35;
    private javax.swing.JLabel jLabel36;
    private javax.swing.JLabel jLabel37;
    private javax.swing.JLabel jLabel38;
    private javax.swing.JLabel jLabel39;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel40;
    private javax.swing.JLabel jLabel41;
    private javax.swing.JLabel jLabel42;
    private javax.swing.JLabel jLabel43;
    private javax.swing.JLabel jLabel44;
    private javax.swing.JLabel jLabel45;
    private javax.swing.JLabel jLabel46;
    private javax.swing.JLabel jLabel47;
    private javax.swing.JLabel jLabel48;
    private javax.swing.JLabel jLabel49;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel50;
    private javax.swing.JLabel jLabel51;
    private javax.swing.JLabel jLabel52;
    private javax.swing.JLabel jLabel53;
    private javax.swing.JLabel jLabel54;
    private javax.swing.JLabel jLabel55;
    private javax.swing.JLabel jLabel56;
    private javax.swing.JLabel jLabel57;
    private javax.swing.JLabel jLabel58;
    private javax.swing.JLabel jLabel59;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel61;
    private javax.swing.JLabel jLabel62;
    private javax.swing.JLabel jLabel63;
    private javax.swing.JLabel jLabel64;
    private javax.swing.JLabel jLabel65;
    private javax.swing.JLabel jLabel66;
    private javax.swing.JLabel jLabel67;
    private javax.swing.JLabel jLabel68;
    private javax.swing.JLabel jLabel69;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel70;
    private javax.swing.JLabel jLabel71;
    private javax.swing.JLabel jLabel72;
    private javax.swing.JLabel jLabel73;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JSeparator jSeparator2;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JLabel lblAIRMS;
    private javax.swing.JLabel lblAPCFDen;
    private javax.swing.JLabel lblAPCFNum;
    private javax.swing.JLabel lblAVAG;
    private javax.swing.JLabel lblAVAHR;
    private javax.swing.JLabel lblAVARG;
    private javax.swing.JLabel lblAVARHR;
    private javax.swing.JLabel lblAVRMS;
    private javax.swing.JLabel lblAWATTHR;
    private javax.swing.JLabel lblAWG;
    private javax.swing.JLabel lblAcknowledge;
    private javax.swing.JLabel lblBIRMS;
    private javax.swing.JLabel lblBVAG;
    private javax.swing.JLabel lblBVAHR;
    private javax.swing.JLabel lblBVARG;
    private javax.swing.JLabel lblBVARHR;
    private javax.swing.JLabel lblBVRMS;
```

```
    private javax.swing.JLabel lblBWATTHR;
    private javax.swing.JLabel lblBWG;
    private javax.swing.JLabel lblCIRMS;
    private javax.swing.JLabel lblCOMPMODE;
    private javax.swing.JLabel lblCVAG;
    private javax.swing.JLabel lblCVAHR;
    private javax.swing.JLabel lblCVARG;
    private javax.swing.JLabel lblCVARHR;
    private javax.swing.JLabel lblCVRMS;
    private javax.swing.JLabel lblCWATTHR;
    private javax.swing.JLabel lblCWG;
    private javax.swing.JLabel lblEnergyVA;
    private javax.swing.JLabel lblEnergyVAA;
    private javax.swing.JLabel lblEnergyVAB;
    private javax.swing.JLabel lblEnergyVAC;
    private javax.swing.JLabel lblEnergyVAR;
    private javax.swing.JLabel lblEnergyVARA;
    private javax.swing.JLabel lblEnergyVARB;
    private javax.swing.JLabel lblEnergyVARC;
    private javax.swing.JLabel lblEnergyW;
    private javax.swing.JLabel lblEnergyWA;
    private javax.swing.JLabel lblEnergyWB;
    private javax.swing.JLabel lblEnergyWC;
    private javax.swing.JLabel lblFREQ;
    private javax.swing.JLabel lblFreqMeasure;
    private javax.swing.JLabel lblIaRMSOffsetRegister;
    private javax.swing.JLabel lblIbRMSOffsetRegister;
    private javax.swing.JLabel lblIcRMSOffsetRegister;
    private javax.swing.JLabel lblLCYCMODE;
    private javax.swing.JLabel lblLINECYC;
    private javax.swing.JLabel lblMASK;
    private javax.swing.JLabel lblMCUVADIVA;
    private javax.swing.JLabel lblMCUVADIVB;
    private javax.swing.JLabel lblMCUVADIVC;
    private javax.swing.JLabel lblMCUVARDIVA;
    private javax.swing.JLabel lblMCUVARDIVB;
    private javax.swing.JLabel lblMCUVARDIVC;
    private javax.swing.JLabel lblMCUWDIVA;
    private javax.swing.JLabel lblMCUWDIVB;
    private javax.swing.JLabel lblMCUWDIVC;
    private javax.swing.JLabel lblMMODE;
    private javax.swing.JLabel lblOPMODE;
    private javax.swing.JLabel lblPhase1AWRMS;
    private javax.swing.JLabel lblPhase1BWRMS;
    private javax.swing.JLabel lblPhase1CWRMS;
    private javax.swing.JLabel lblPhase2AWRMS;
    private javax.swing.JLabel lblPhase2BWRMS;
    private javax.swing.JLabel lblPhase2CWRMS;
    private javax.swing.JLabel lblRSTATUS;
    private javax.swing.JLabel lblTEMP;
    private javax.swing.JLabel lblTotalPackets;
    private javax.swing.JLabel lblVADIV;
    private javax.swing.JLabel lblVARCFDen;
    private javax.swing.JLabel lblVARCFNum;
    private javax.swing.JLabel lblVARDIV;
    private javax.swing.JLabel lblVaRMSOffsetRegister;
    private javax.swing.JLabel lblVbRMSOffsetRegister;
    private javax.swing.JLabel lblVcRMSOffsetRegister;
    private javax.swing.JLabel lblWAVMODE;
    private javax.swing.JLabel lblWDIV;
    private javax.swing.JMenuItem mnuALLREGISTERS;
    private javax.swing.JMenuItem mnuCFLEDsOFF;
    private javax.swing.JMenuItem mnuCFLEDsOn;
    private javax.swing.JMenu mnuCallibration;
    private javax.swing.JMenu mnuFile;
    private javax.swing.JMenu mnuMCU;
    private javax.swing.JMenuItem mnuOpen;
    private javax.swing.JMenuItem mnuReadFlash;
    private javax.swing.JMenuItem mnuSave;
    private javax.swing.JMenuItem mnuWinEnergyGainCalibration;
    private javax.swing.JMenuItem mnuWinEnergyMeasurements;
    private javax.swing.JMenuItem mnuWinFrequencyCalibration;
    private javax.swing.JMenuItem mnuWinMCCalibration;
    private javax.swing.JMenuItem mnuWinMeasurements;
    private javax.swing.JMenuItem mnuWinOtherMeasurememets;
    private javax.swing.JMenuItem mnuWinPhaseCalibration;
    private javax.swing.JMenuItem mnuWinRegisters;
    private javax.swing.JMenuItem mnuWinVIOffsetCalibration;
    private javax.swing.JMenu mnuWindows;
```

```
    private javax.swing.JMenuItem mnuWriteToFlash;
    private javax.swing.JPanel pnlCalibrateVIRMSOFFSET;
    private javax.swing.JPanel pnlEnergyGainCalibration;
    private javax.swing.JPanel pnlEnergyMeasureMent;
    private javax.swing.JPanel pnlFrequencyCalibration;
    private javax.swing.JPanel pnlMCCalibration;
    private javax.swing.JPanel pnlMeasureMents;
    private javax.swing.JPanel pnlOtherMeasurements;
    private javax.swing.JPanel pnlPhaseCallibration;
    private javax.swing.JPanel pnlRegisters;
    private javax.swing.JTextField txtFreqMeasure;
    private javax.swing.JTextField txtFreqValue;
    private javax.swing.JTextField txtGainCalVAR_A;
    private javax.swing.JTextField txtGainCalVAR_B;
    private javax.swing.JTextField txtGainCalVAR_C;
    private javax.swing.JTextField txtGainCalW_A;
    private javax.swing.JTextField txtGainCalW_B;
    private javax.swing.JTextField txtGainCalW_C;
    private javax.swing.JTextField txtGainCycles;
    private javax.swing.JTextField txtI_1CC;
    private javax.swing.JTextField txtI_2CC;
    private javax.swing.JTextField txtMC;
    private javax.swing.JTextField txtMCMeasurePowerVAR;
    private javax.swing.JTextField txtMCMeasurePowerW;
    private javax.swing.JTextField txtMCMeasureVARFreq;
    private javax.swing.JTextField txtMCMeasureWFreq;
    private javax.swing.JTextField txtPhase1APF;
    private javax.swing.JTextField txtPhase1AW;
    private javax.swing.JTextField txtPhase1BPF;
    private javax.swing.JTextField txtPhase1BW;
    private javax.swing.JTextField txtPhase1CPF;
    private javax.swing.JTextField txtPhase1CW;
    private javax.swing.JTextField txtPhase2APF;
    private javax.swing.JTextField txtPhase2AW;
    private javax.swing.JTextField txtPhase2BPF;
    private javax.swing.JTextField txtPhase2BW;
    private javax.swing.JTextField txtPhase2CPF;
    private javax.swing.JTextField txtPhase2CW;
    private javax.swing.JTextField txtPhaseACallibration;
    private javax.swing.JTextField txtPhaseBCallibration;
    private javax.swing.JTextField txtPhaseCCallibration;
    private javax.swing.JTextField txtPhaseCycles;
    private javax.swing.JTextField txtUpdateVAhA;
    private javax.swing.JTextField txtUpdateVAhB;
    private javax.swing.JTextField txtUpdateVAhC;
    private javax.swing.JTextField txtUpdateWhA;
    private javax.swing.JTextField txtUpdateWhB;
    private javax.swing.JTextField txtUpdateWhC;
    private javax.swing.JTextField txtUpdatevarhA;
    private javax.swing.JTextField txtUpdatevarhB;
    private javax.swing.JTextField txtUpdatevarhC;
    private javax.swing.JTextField txtV_1CC;
    private javax.swing.JTextField txtV_2CC;
    // End of variables declaration//GEN-END:variables

}
```



Q.3 Package: `green.palmHHT`

For coding the palm device the following refferences were used:

- `http://java.sun.com/j2me`
- `http://www.developer.com/java/j2me/article.php/1561591`
- `http://developers.sun.com/techtopics/mobility/midp/articles/databaserms/`

Q.3.1 Object: `green.palmHHT.palmHHT`

```
/*
 * palmHHT.java
```

```
 *
 * Created on 02 February 2004, 10:36
 */

package green.palmHHT;

import java.util.*;
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

/** <B>Palm HHT Software Description: </B><BR>
 *
 * The palmHHT software forms part of the MET suite of programs to facilitate the
 * capture of meter readings on the spot. The process minizes erreors and give
 * management information about the complete reading process and adds value to the
 * customer care process.<BR>
 * The palmHHT software is developed for the use on a Palm Handheld Device, with
 * Palm OS 5 or later. MIDP4PALM1.0 Java Virtual Machine should be used.
 * The software was developed on the Java platform to promote device independant
 * operation.<BR><BR>
 *
 * <B>System flow:</B><BR>
 * The folowing steps give a short overview of the flow of the process:<BR>
 * <UL>
 *    <LI>A route and message file is created and stored in a default directory.
 *    <LI>These files are transmitted to the palm Handheld device via the Palm HotSync
 *          software. This is done by pressing the HotSync button on the Palm Cradle.
 *    <LI>After the synchronization the hand held is ready for opration in the field and
 *           can be given to a meter reader.
 *    <LI>The palmHHT software can be started by tapping on the star or on the palmHHT
 *          icon.
 *    <LI>Once started up a screen will all the required fields appear. The fields can
 *           be edited using the stylus. A record is stored after the OK button is clicked.
 *           Error messages guide the operator through the process.
 *    <LI>Once all the readings are completed the Palm Handheld is placed in the cradle.
 *    <LI>Pressing the HotSync button will initiate the synchronizing process and the
 *           data from the field will be transfered to the database on the computor. The
 *           new route is also uploaded during this data transfer.
 * </UL>
 *
 * <B>Terminal program flow:</B><BR>
 * Once synchronized the capturing program can be started by tapping on the palmHHT
 * icon.<BR>
 *
 * The following fileds are displayed on the screen:<BR>
 * <UL>
 *    <LI>Situation adress
 *    <LI>Meter ID and type
 *    <LI>Accound Number
 *    <LI>Meter Number
 *    <LI>Previous Reading
 *    <LI>Message 1
 *    <LI>Message 2
 *    <LI>Message 3
 *    <LI>Error messages
 *    <LI>New message
 * </UL>
 *
 * The NEXT and PREV buttons can be used to scroll to the next measurement data or
 * back. The OK button stores the current data and does error checking on the
 * information. To add a new message to the list can be done by clicking on the
 * title bar and selecting Actions | addMsg.<BR>
 *
 * The meter number is a field that is parameter controlled and can be a display field or
 * an entry required field.<BR>
 * The previous reading is a field that is parameter controlled and can be displayed or
 * hidden.<BR><BR>
 *
 * The reading field has the following attributes:<BR>
 * The input length of this field extextis equal to the number of digits as supplied by the debtor
 * system.<BR><BR>
 *
 * A number of tries parameter can be set for the reading field. The default value is 3. The number of
 * times the meter reader enters a reading is recorded and checked against the number of tries parameter.
 * When the number of tries is exhausted, the program accepts the current values and move to the next record.<BR>
 * The reading entered must be within the range 90% to 110% of the sum of the previous reading and the average
 * usage.<BR>
 * If the reading entered is less than the previous reading a warning is given and the reading must be
 * re-entered. This also acts as a safety check for meters that ticked over.<BR>
 * Each record may have three messages that can be downloaded from the debtor system. The messages linked to
```

```
 * the current record will be displayed.<BR><BR>
 *
 * The meter reader may use the free message areas to send messages back to the office. The third message area
 * will always be used if no free message area exists. When activated, the messaging section works as follows:
 * <UL>
 *    <LI>A message is selected using the drop down list provided on
 *           screen.
 *    <LI>The correct message is selected when required.
 *    <LI>If no message describing the situation a new message can be entered on the
 *        space provided. This message is added to the list using the Actions|AddMsg
 *        action. The Actions menu is activated by tapping on the top left corner on the
 *        title of the application.
 * </UL>
 *
 * <B>General information:</B><BR>
 * All records are date and time stamped.<BR>
 * For writing and adding information onto the handheld refer to the palm
 * instructions, or the Graffiti 2 Application on the handheld.<BR><BR>
 *
 * <B>SYSTEM COMPONENTS</B><BR>
 * The following components must be present before the system can function properly:<BR>
 * <UL>
 *    <LI>A debtor system that can output a meter route list to an ASCII file according to a
 *     specified record layout.
 *    <LI>A microcomputer that have been setup with Palm Desktop, HotSync and Palm
 *    <LI>Conduit Development Kit.
 *    <LI>Java 2 SDK v 1.4. or later
 *    <LI>A CD with the handheld software and conduit software.
 * </UL>
 *
 * <B>SETTING UP THE SYSTEM</B><BR>
 * Do the following steps to setup a microcomputer to use the meter reading
 * system:<BR>
 * <UL>
 *    <LI>Install the above mentioned software, except the handheld terminal and conduit software.
 *    <LI>Create a sub-directory called HHT.
 *    <LI>Copy all the files on the CD in the directory HHT into the HHT directory on the
 *           PC.
 *    <LI>Install the MIDP.prc and palmHHT.prc file onto the each palm handheld device using the Palm
 *           Desktop Quick Install software.
 *    <LI>Use the Conduit Setup program provided on the CD to install the java class for
 *  the conduit.
 * </UL>
 *
 * <B>SETTING UP THE HAND HELD TERMINAL</B><BR>
 * The following steps are to setup a "blank" or new hand held terminal:<BR>
 * <UL>
 *    <LI>Make sure that the battery is fully charged and the Palm Device is placed in
 *           the cradle.
 *    <LI>Ensure that the conduits is configure correctly as explained in the above
 *           section.
 *    <LI>Press the HotSync button at the bottom of the handheld cradle.
 *    <LI>Run the palmHHT software on the palm to setup the empty databases.
 *    <LI>Create a route from the debtor system using the MET software.
 *    <LI>Ensure that the default directory is correctly setup.
 *    <LI>Press the HotSync button to upload the new route.
 * </UL>
 *
 * <B>TECHNICAL INFORMATION</B><BR>
 * The following additional memeroy is required for the system:<BR>
 * <UL>
 *    <LI>JAVA MIDP - 600k
 *    <LI>PALM HHT  - 60k
 *    <LI>PER RECCORD - 500bytes
 * </UL>
 *
 * => 4Meg = According to MIDP recomendation
 * @author Riaan Doorduin
 * @version 1.0
 */
public class palmHHT extends MIDlet implements CommandListener {

    /*
     * Variable and class declerations
     */

    //The records, giving access to the recordStores and datstructures implemented
    // in the recordStores
    private MessageRecords mR = null;
```

```
private RouteRecords rR = null;

//The commands and actions available to users.
private Command prevCommand;
private Command nextCommand;
private Command OKCommand;
//private Command addCommand;
private Command addMsgCommand;
private Command toFirstCommand;
private Command toLastCommand;

//The main form
private Form frmMain;

//Items available for adding to the form. Unused Items are commented out, to
//  create more compact and less memory intensive code.
private StringItem lblInfo = new StringItem("","");
private StringItem lblAccountNumber = new StringItem("AccNo","0");
private StringItem lblMeterID = new StringItem("MtrID","0");
private StringItem lblSituationAdress = new StringItem("Adress","0");
private StringItem lblStreetNumber = new StringItem("StreetNumber","0");
private StringItem lblMeterType = new StringItem("MeterType","0");
private StringItem lblOwnerName = new StringItem("Owner","0");
private StringItem lblInitials = new StringItem("Initials","0");
private StringItem lblMeterNumber = new StringItem("MtrNo","0");
private TextField txtMeterNumber = new TextField("MtrNo","",8,TextField.ANY);
private StringItem lblOldReading = new StringItem("Prv rd","0");
private StringItem lblAverageUsage = new StringItem("Average Usage","0");
private StringItem lblPercentage = new StringItem("Percentage","0");
private StringItem lblMNM = new StringItem("MNM","0");
private StringItem lblNewReading = new StringItem("Reading","0");
private TextField txtNewReading = new TextField("Reading","",8,TextField.NUMERIC);
private StringItem lblReadingCode = new StringItem("Reading Code","0");
private StringItem lblReadingDate = new StringItem("Reading Date","0");
private StringItem lblNumberOfDigits = new StringItem("Number Of Digits","0");
private StringItem lblTime = new StringItem("Time","0");
private StringItem lblNumberOfTries = new StringItem("Number Of Tries","0");
private ChoiceGroup lstMessage1 = new ChoiceGroup("Msg 1", ChoiceGroup.EXCLUSIVE);
private StringItem lblMessage1 = new StringItem("Msg 1","");
private ChoiceGroup lstMessage2 = new ChoiceGroup("Msg 2", ChoiceGroup.EXCLUSIVE);
private StringItem lblMessage2 = new StringItem("Msg 2","");
private ChoiceGroup lstMessage3 = new ChoiceGroup("Msg 3", ChoiceGroup.EXCLUSIVE);
private StringItem lblMessage3 = new StringItem("Msg 3","");
private StringItem lblFactor = new StringItem("Factor","0");
private StringItem lblError = new StringItem("","");
private TextField txtNewMessage = new TextField("NewMsg","",20,TextField.ANY);

private HeaderInfo header = new HeaderInfo();
private int i = 0;
private int retry = 0;
private int msgretry = 0;
private int tries = 3;
private int percentage = 90;
private int prevReading = 999999999;
private String prevMeterNumber = "ZZZZZZZZ";
private int inputNumberState = 0;  // 0 = MtrNumber Input, 1 = Reading Input
private Display display;    // The display for this MIDlet

//Current info, to save old messages.
//private boolean keepInfo = false;
//private String meterReadingOld = "";
//private String meterNumberOld = "";
//private String meterMessageOld = "";

private static boolean inEvent = false; //To ensure that the last event is serviced prior
                                        // to a new envent being serviced.
/**
 * set and forms the NewReading field for input data
 */
private void setNewReadingLabel(String text,int length) {
    txtNewReading.setString("");
    try {
        if (text.length() >= length) {
          txtNewReading.setMaxSize(text.length());
        } else if (length != 0) {
            txtNewReading.setMaxSize(length);
        } else {
            txtNewReading.setMaxSize(8);
        }
```

```
    } catch (IllegalArgumentException e) {
        txtNewReading.setMaxSize(8);
    }
    try {
      txtNewReading.setString(text);
    } catch (IllegalArgumentException e) {
      txtNewReading.setString("");
    }
}

/**
 * Updates the title bar of the form with the current values and status
 */
private void updateTitle() {
    if (rR != null) {
      frmMain.setTitle(rR.getMeterInfo().getSituationAdress()+" "+rR.getMeterInfo().getStreetNumber());
      rR.writeCurrentMeterNumber();
    }
}

/**
 * This method populates the field with the required values
 */
private void updateForm() {
    MessageInfo message = new MessageInfo();
    MeterInfo meter = rR.getMeterInfo();
    String tmpStr = new String();
    int tmpVal = 0;
    byte i = 0;
    if (meter != null) {
        //Set current info of record
        lblInfo.setText("("+header.getRouteNo()+")["+rR.getNumberOfMeters()+"]#"+rR.getCurrentMeterNumber());

        //Check if initials of owner is present, to ensure that no leading space occur.
        lblOwnerName.setText((meter.getInitials().trim()+" "+meter.getOwnerName().trim()).trim());
        lblMeterID.setText(meter.getMeterID()+"["+rR.getMeterInfo().getMeterType()+"]");
        lblAccountNumber.setText(meter.getAccountNumber());
        lblMeterNumber.setText(meter.getMeterNumber());
        lblOldReading.setText(meter.getOldReading());
        lblAverageUsage.setText(meter.getAverageUsage());
        try {
            tmpVal = new Integer(Integer.parseInt(meter.getNewReading().trim())).intValue();
        } catch (NumberFormatException e) {
            tmpVal = 0;
        }
        if (tmpVal == 0) {
            setNewReadingLabel("",meter.getNumberOfDigitsInt());
        } else {
            setNewReadingLabel(""+tmpVal,meter.getNumberOfDigitsInt());
        }
        lblMessage1.setText(lstMessage3.getString(meter.getMessage1Int()));
        lblMessage2.setText(lstMessage3.getString(meter.getMessage2Int()));
        lstMessage3.setSelectedIndex(meter.getMessage3Int(),true);
    } else {
        lblAccountNumber.setText("Null");
        lblMeterID.setText("Null");
        lblMeterType.setText("Null");
        lblMeterNumber.setText("Null");
        lblOldReading.setText("Null");
        txtNewReading.setString("0000");
    }
}

/** Upon creation and initialization of the software after a synchronization process
 * this routine setups the default screen and creates all the correct labels:
 *
 * This method is responsible for the correct screen layout.
 *
 * Before the screen can be initialized the databases are checked.
 * The system uses two databases namely a route and a message database.
 * These databases are used to populate the screens and capture the field data.
 */
public palmHHT() {
    // to delay the debugger for setting up the databases
    mR = new MessageRecords();
    rR = new RouteRecords();
    //Load data:
    display = Display.getDisplay(this);
    prevCommand = new Command("< back",Command.SCREEN,3);
```

```
nextCommand = new Command("next >",Command.SCREEN,5);
OKCommand = new Command("OK",Command.SCREEN,6);
addMsgCommand = new Command("AddMsg",Command.SCREEN,8);
toFirstCommand = new Command("ToFirst", Command.SCREEN,9);
toLastCommand = new Command("ToLast", Command.SCREEN,10);

frmMain = new Form("NewForm");


if (mR.getNumRecords() != 0) {        //Check if message records exist.
    frmMain.setTitle("Messages != 0");

    //Populate the Message List
    lstMessage3.append(" ",null);
    for (int i=0; i < mR.getNumberOfMessages(); i++) {
        MessageInfo message = mR.getNextMessage();
        lstMessage3.append(message.getMessage(),null);
    }

    frmMain.setTitle("Messages Done!");
    if (rR == null) {
        frmMain.setTitle("Route = null!");
    }
    if (rR.getNumRecords() != 0) {   //Check if meter records is present
        frmMain.setTitle("Routes != 0");

        if (rR.getNumRecords() > 7) {   //Check if meter records is present
            frmMain.setTitle("Routes > 7");
            if (rR.getNumRecords() < 13000) {   //Check if enumeration was successfull
                frmMain.setTitle("Routes < 13000");
                // Decode Header
                header = rR.readHeaderInfo();
                try {
                   // The number of Tries allowed per meter reading
                   tries = new Integer(Integer.parseInt(header.getTries().trim())).intValue();
                } catch (NumberFormatException e) {
                    tries = 3;
                }
                try {
                   // Update Position form Last Good Value Load
                   rR.advanceToMeterInfo(new Integer(Integer.parseInt(header.getLastRecordNumber().trim())).intValue());
                } catch (NumberFormatException e) {}

                //Display Setup
                //---------------------------------------
                //  Commands
                frmMain.setTitle("Route Ready");
                frmMain.addCommand(prevCommand);
                frmMain.addCommand(OKCommand);
                frmMain.addCommand(nextCommand);
                frmMain.addCommand(addMsgCommand);
                frmMain.addCommand(toFirstCommand);
                frmMain.addCommand(toLastCommand);

                //  Values Fields
                if (header.getDisplayOwnerName().toUpperCase().equals("Y")) {
                    frmMain.append(lblOwnerName);
                }
                //frmMain.append(lblSituationAdress);
                frmMain.append(lblMeterID);
                frmMain.append(lblAccountNumber);
                if (header.getAskMeterNumber().toUpperCase().equals("Y")) {
                    frmMain.append(txtMeterNumber);
                } else {
                    frmMain.append(lblMeterNumber);
                }
                if (header.getDisplayPreviousReading().toUpperCase().equals("Y")) {
                    frmMain.append(lblOldReading);
                }
                frmMain.append(txtNewReading);
                frmMain.append(lblMessage1);
                //frmMain.append(lblMessage2);
                frmMain.append(lstMessage3);
                frmMain.append(lblError);
                frmMain.append(txtNewMessage);
                frmMain.append(lblInfo);

                lblError.setText("Total Records: "+rR.getNumRecords());
                updateTitle();
```

```
                    updateForm();
                } else {
                    frmMain.setTitle("Maximum Route Size!");
                    frmMain.append(new StringItem("Route Data is over",""));
                    frmMain.append(new StringItem("the allowed limit!!",""));
                    frmMain.append(new StringItem("Check Route File!!",""));
                    frmMain.append(new StringItem("HotSync required!",""));
                    frmMain.append(new StringItem("rR["+rR.getNumRecords()+"]",""+rR.getNextRecordID()));
                    frmMain.append(new StringItem("mR["+mR.getNumRecords()+"]",""+mR.getNextRecordID()));
                }
            } else {
                frmMain.setTitle("No Route");
                frmMain.append(new StringItem("Route Data Empty!!",""));
                frmMain.append(new StringItem("Check Route File!!",""));
                frmMain.append(new StringItem("HotSync required!",""));
                frmMain.append(new StringItem("rR["+rR.getNumRecords()+"]",""+rR.getNextRecordID()));
                frmMain.append(new StringItem("mR["+mR.getNumRecords()+"]",""+mR.getNextRecordID()));
            }
        } else {
            frmMain.setTitle("No Route");
            frmMain.append(new StringItem("No Route Data!!",""));
            frmMain.append(new StringItem("HotSync required!",""));
            frmMain.append(new StringItem("rR["+rR.getNumRecords()+"]",""+rR.getNextRecordID()));
            frmMain.append(new StringItem("mR["+mR.getNumRecords()+"]",""+mR.getNextRecordID()));
        }
    } else {
        frmMain.setTitle("No Messages");
        frmMain.append(new StringItem("No Message Data!!",""));
        frmMain.append(new StringItem("HotSync required!",""));
        frmMain.append(new StringItem("rR["+rR.getNumRecords()+"]",""+rR.getNextRecordID()));
        frmMain.append(new StringItem("mR["+mR.getNumRecords()+"]",""+mR.getNextRecordID()));
    }
}

/** Start up the this MIDlet by setting the current display and associating
 * the command and listener.
 */
public void startApp() {
    frmMain.setCommandListener(this);
    display.setCurrent(frmMain);
}

/**
 * Pause is a no-op since there are no background activities or
 * record stores that need to be closed.
 */
public void pauseApp() {
}

/** Destroy must cleanup everything not handled by the garbage collector.
 * In this case there is nothing to cleanup.
 * @param unconditional If unconditional destroy must occur
 */
public void destroyApp(boolean unconditional) {
}

/* Saves the current record without any checking into the database
 */
private void saveRecord() {
    if (header.getAskMeterNumber().toUpperCase().equals("Y")) {
        rR.writeMeterNumber(txtMeterNumber.getString());
    }
    rR.writeMessage3(""+(lstMessage3.getSelectedIndex()));
    rR.writeCurrentDate();
    rR.writeCurrentTime();
    rR.writeNewReading(txtNewReading.getString());
    rR.writeNumberOfTries(""+(++retry));
    msgretry = 0;
    prevMeterNumber = "ZZZZZZZZ";
    retry = 0;
    prevReading = 999999999;
    rR.getNextMeterInfo();
    lblError.setText("Value Accepted!");
    //new Alert("Value Accepted!").setType(AlertType.playSound());
}
/** Respond to commands, including exit
 * On the exit command, cleanup and notify that the MIDlet has been destroyed.
 * @param c The command which called the event
 * @param s Which display the event was called from
```

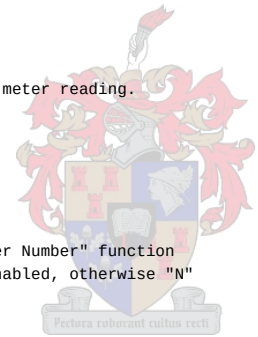
```
 */
public void commandAction(Command c, Displayable s) {

    int average = 0;
    int newReading = 0;
    int oldReading = 0;
    String tmpStr = "";
    boolean meterNumberCorrect = true;

    if (!inEvent) {
        inEvent = true;

        /* The OK command
         * This command is used to verify and store the values entered into the system
         * This command validates the that the new reading is correct and complies with the
         * 10% critea.
         * If required the current Meter Number must also be entered into the system.
         */
        lblError.setText("");

        if (c == OKCommand) {
            lblError.setText("");
            try {
                average = Integer.parseInt(rR.getMeterInfo().getAverageUsage().trim());
            } catch (NumberFormatException e) {
                average = 0;
                tmpStr = e.getMessage();
            }
            try {
                newReading = Integer.parseInt(txtNewReading.getString().trim());
            } catch (NumberFormatException e) {
                newReading = 0;
                tmpStr = e.getMessage();
            }
            try {
                oldReading = new Integer(Integer.parseInt(rR.getMeterInfo().getOldReading().trim())).intValue();
            } catch (NumberFormatException e) {
                oldReading = 0;
                tmpStr = e.getMessage();
            }

            //Test to check Meter Number

            meterNumberCorrect = true;
            if (header.getAskMeterNumber().toUpperCase().equals("Y")) {
                if (rR.getMeterInfo().getMeterNumber().trim().toUpperCase().equals(txtMeterNumber.getString().toUpperCase().trim()) || //M
                    (prevMeterNumber.trim().toUpperCase().equals(txtMeterNumber.getString().trim().toUpperCase())) || //Same Meter Number
                    (msgretry >= tries)) {
                } else {
                    lblError.setText("Incorrect Meter Number!");
                    msgretry++;
                    prevMeterNumber = txtMeterNumber.getString().trim();
                    //meterNumberOld = "";
                    meterNumberCorrect = false;
                }
            }
            if (meterNumberCorrect) {
                //Test Meter Reading
                if ((retry >= tries) || (prevReading == newReading)) {
                    saveRecord();
                } else if (txtNewMessage.getString().length() != 0) {
                    lstMessage3.append(txtNewMessage.getString(),null);
                    lstMessage3.setSelectedIndex(lstMessage3.size()-1,true);
                    mR.setMessage(txtNewMessage.getString());
                    mR.addNewMessageInfo();
                    txtNewMessage.setString("");
                    saveRecord();
                } else if (newReading <= oldReading) {
                    lblError.setText("Meter ticked Over");
                    retry++;
                    prevReading = newReading;
                } else if (newReading == oldReading) {
                    lblError.setText("Meter stopped");
                    retry++;
                    prevReading = newReading;
                } else if ((newReading*100) <= ((oldReading*100) + (average*(100-percentage)))) {
                    lblError.setText("Reading too small!");
                    retry++;
                    prevReading = newReading;
```

```
                } else if ((newReading*100) >= ((oldReading*100) + (average*(100+percentage)))) {
                    lblError.setText("Reading too large!");
                    retry++;
                    prevReading = newReading;
                } else {
                    saveRecord();
                }
            }
            updateTitle();
            updateForm();

        // Return to previous Meter Info
        } else if (c == prevCommand) {
            msgretry = 0;
            prevMeterNumber = "ZZZZZZZZ";
            retry = 0;
            prevReading = 999999999;
            rR.getPreviousMeterInfo();
            updateTitle();
            updateForm();

        //Advance to next Meter Info
        } else if (c == nextCommand) {
            msgretry = 0;
            prevMeterNumber = "ZZZZZZZZ";
            retry = 0;
            prevReading = 999999999;
            rR.getNextMeterInfo();
            updateTitle();
            updateForm();

        // Adds a message to the message list if a new message was entered.
        } else if (c == addMsgCommand) {
            if (txtNewMessage.getString().length() != 0) {
                lstMessage3.append(txtNewMessage.getString(),null);
                mR.setMessage(txtNewMessage.getString());
                mR.addNewMessageInfo();
                txtNewMessage.setString("");
            }
        //Skips to first Meter Info
        } else if (c == toFirstCommand) {
            rR.advanceToMeterInfo(0);
            updateTitle();
            updateForm();

        //Skips to last Meter Info
        } else if (c == toLastCommand) {
            rR.advanceToMeterInfo(rR.getNumberOfMeters());
            updateTitle();
            updateForm();
        }

        inEvent = false;
    }
  }
}
```

Q.3.2 Object: `green.palmHHT.HeaderInfo`

```
/*
 * headerInfo.java
 *
 * Created on 03 February 2004, 08:14
 */

package green.palmHHT;

import java.util.*;
/** The header contains information for the functioning of the handheld terminal,
 * e.g. route number and parameters for correct setup of routes and display.
 * @author RDoorduin
 */
public class HeaderInfo{

    private String RouteNo = "0000";
    private String Tries = "3";
    private String AskMeterNumber = "Y";
```

```
private String DisplayPreviousReading = "Y";
private String DisplayOwnerName = "Y";
private String LastRecordNumber = "0";
private final int HEADER_FIELDS = 6;
/** Creates a new instance of headerInfo */
public HeaderInfo() {
}

/** Easy way to allow scalability for fields.
 * @return the number of fields in the
 */
public int getNumberOfFields() {
    return HEADER_FIELDS;
}

//RouteNo bean
/** Get the current route number
 * @return the route number
 */
public String getRouteNo() {
    return RouteNo;
}
/** Set the current route number
 * @param txt the route number
 */
public void setRouteNo(String txt) {
    RouteNo = txt;
}

/** Gets the number of tries allowed per meter reading.
 * @return the number of tries
 */
public String getTries() {
    return Tries;
}
/** Sets the number of tries allowed per meter reading.
 * @param Tries the number of tries
 */
public void setTries(String Tries) {
    this.Tries = Tries;
}

/** Get the field to enable the "Ask Meter Number" function
 * @return "Y" if the function must be enabled, otherwise "N"
 */
public String getAskMeterNumber() {
    return AskMeterNumber;
}
/** Set the field to enable the "Ask Meter Number" function
 * @param AskMeterNumber "Y" if the function must be enabled, otherwise "N"
 */
public void setAskMeterNumber(String AskMeterNumber) {
    this.AskMeterNumber = AskMeterNumber;
}

/** Get the field to enable the "Display Previous Reading" function
 * @return "Y" if the function must be enabled, otherwise "N"
 */
public String getDisplayPreviousReading() {
    return DisplayPreviousReading;
}
/** Set the field to enable the "Display Previous Reading" function
 * @param DisplayPreviousReading "Y" if the function must be enabled, otherwise "N"
 */
public void setDisplayPreviousReading(String DisplayPreviousReading) {
    this.DisplayPreviousReading = DisplayPreviousReading;
}

/** Get the field to enable the "Dipslay Owner Name" function
 * @return "Y" if the function must be enabled, otherwise "N"
 */
public String getDisplayOwnerName() {
    return DisplayOwnerName;
}
/** Set the field to enable the "Dipslay Owner Name" function
 * @param DisplayOwnerName "Y" if the function must be enabled, otherwise "N"
 */
public void setDisplayOwnerName(String DisplayOwnerName) {
    this.DisplayOwnerName = DisplayOwnerName;
```



```
    }

    /** Get the last MeterInfo field that was accessed.
     * @return the last MeterInfo number
     */
    public String getLastRecordNumber() {
        return LastRecordNumber;
    }
    /** S
     * et the last MeterInfo field that was accessed.
     * @param LastRecordNumber the last MeterInfo number
     */
    public void setLastRecordNumber(String LastRecordNumber) {
        this.LastRecordNumber = LastRecordNumber;
    }
}
```

Q.3.3 Object: `green.palmHHT.MeterInfo`

```
/*
 * MeterInfo.java
 *
 * Created on 02 February 2004, 11:00
 */

package green.palmHHT;

/** This class contains all the beans for the to access the MeterInfo object.
 * @author RDoorduin
 */
public class MeterInfo {
    //Fields of used
    private String AccountNumber = "123456789012";
    private String MeterID = "1";
    private String SituationAdress = "12345678901234";
    private String StreetNumber = "1234";
    private String MeterType = "1";
    private String OwnerName = "1234567890123456789";
    private String Initials = "12";
    private String MeterNumber = "12345678";
    private String OldReading = "12345678";
    private String AverageUsage = "12345678";
    private String Percentage = "123";
    private String MNM = "1234";
    private String NewReading = "12345678";
    private String ReadingCode = "12";
    private String ReadingDate = "12345678";
    private String NumberOfDigits = "1";
    private String Time = "123456";
    private String NumberOfTries = "1";
    private String Message1 = "123";
    private String Message2 = "123";
    private String Message3 = "123";
    private String Factor = "12345678";

    private final int METERINFO_FIELDS = 22;

    /** Creates a new instance of MeterInfo */
    public MeterInfo() {
    }

    /** The number of fields contained within the class.
     * @return Number of Fields in class
     */
    public int getNumberOfFields() {
        return METERINFO_FIELDS;
    }

    /** Sets the Account Number, e.g. the account to be billed for the usage.
     * @param txt the account number
     */
    public void setAccountNumber(String txt) {
        AccountNumber = txt;
    }
    /** Gets the Account Number, e.g. the account to be billed for the usage.
     * @return the account number
     */
```

```
public String getAccountNumber() {
    return AccountNumber;
}

/** Sets the Meter ID, e.g. in one physicle adress, more than one meter of can be
 * contained. This points to the meter within the specific location or meter box.
 * @param txt the meter ID
 */
public void setMeterID(String txt) {
    MeterID = txt;
}
/** Gets the Meter ID, e.g. in one physicle adress, more than one meter of can be
 * contained. This points to the meter within the specific location or meter box.
 * @return the meter ID
 */
public String getMeterID() {
    return MeterID;
}
/** Sets the physicle locaton adress of the meter.
 * @param txt The location or adress
 */
public void setSituationAdress(String txt) {
    SituationAdress = txt;
}
/** Gets the physicle locaton adress of the meter.
 * @return the location or address
 */
public String getSituationAdress(){
    return SituationAdress;
}

/** Sets the street number of the adress.
 * @param txt the street number
 */
public void setStreetNumber(String txt) {
     StreetNumber = txt;
}
/** Sets the street number of the adress.
 * @return teh street number
 */
public String getStreetNumber(){
    return StreetNumber;
}

/** Sets the meter type of the meter, e.g. W-water, K-kVA, E-electricity
 * @param txt the meter type
 */
public void setMeterType(String txt) {
     MeterType = txt;
}
/** Gets the meter type of the meter, e.g. W-water, K-kVA, E-electricity
 * @return the meter type
 */
public String getMeterType(){
    return MeterType;
}

/** Sets the owner name.
 * @param txt the owner name
 */
public void setOwnerName(String txt) {
     OwnerName = txt;
}
/** Gets the owner name
 * @return the owner name
 */
public String getOwnerName(){
    return OwnerName;
}

/** Sets the owner initials
 * @param txt the owner initials
 */
public void setInitials(String txt) {
     Initials= txt;
}
/** Gets the owner initials
 * @return the owner initials
 */
```

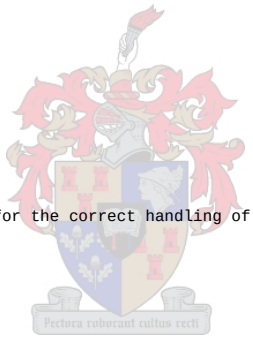
```
public String getInitials(){
    return Initials;
}

/** Sets the Meter Number, e.g. the serial number of the specific meter.
 * @param txt meter serial number
 */
public void setMeterNumber(String txt) {
     MeterNumber= txt;
}
/** Gets the Meter Number, e.g. the serial number of the specific meter.
 * @return the meter serial number
 */
public String getMeterNumber(){
    return MeterNumber;
}

/** Sets the reading prior to the current reading
 * @param txt the old reading
 */
public void setOldReading(String txt) {
     OldReading = txt;
}
/** Gets the reading prior to the current reading
 * @return the old reading
 */
public String getOldReading(){
    return OldReading;
}

/** Sets the average usage of the meter.
 * @param txt the average usage
 */
public void setAverageUsage(String txt) {
     AverageUsage = txt;
}
/** Gets the average usage of the meter.
 * @return the average usage
 */
public String getAverageUsage(){
    return AverageUsage;
}

/** Sets the percentage of the usage.
 * @param txt the percentage
 */
public void setPercentage(String txt) {
     Percentage = txt;
}
/** Gets the percentage of the usage.
 * @return the percentage
 */
public String getPercentage(){
    return Percentage;
}

/** Sets teh MNM field
 * @param txt MNM
 */
public void setMNM(String txt) {
     MNM = txt;
}
/** Gets the MNM field
 * @return MNM
 */
public String getMNM(){
    return MNM;
}

/** Sets the new or current reading.
 * @param txt The new reading
 */
public void setNewReading(String txt) {
     NewReading = txt;
}
/** Gets the new or current reading.
 * @return the new reading
 */
public String getNewReading(){
```

```
    return NewReading;
}

/** Set the reading code
 * @param txt the reading code
 */
public void setReadingCode(String txt) {
      ReadingCode= txt;
}
/** Gets the reading code
 * @return the reading code
 */
public String getReadingCode(){
    return ReadingCode;
}

/** Set the reading date, e.g. the current system date when the reading was made in
 * YYMMDD format.
 * @param txt the reading date in YYMMDD format
 */
public void setReadingDate(String txt) {
      ReadingDate= txt;
}
/** Gets the reading date, e.g. the current system date when the reading was made YYMMDD
 * format.
 * @return the reading date YYMMDD format
 */
public String getReadingDate(){
    return ReadingDate;
}

/** Sets the number of digits used for the current meter
 * @param txt the number of digits
 */
public void setNumberOfDigits(String txt) {
      NumberOfDigits= txt;
}
/** Gets the number of digits used for the current meter
 * @return the number of digits
 */
public String getNumberOfDigits(){
    return NumberOfDigits;
}
/** Gets the number of digits used for the current meter
 * @return the number of digits
 */
public int getNumberOfDigitsInt(){
    int i = 0;
    try {
        i = new Integer(Integer.parseInt(getNumberOfDigits().trim())).intValue();
    } catch (NumberFormatException e) {
    }
    return i;
}

/** Sets the current system time when the reading was made in HHMMSS format.
 * @param txt the current system time in HHMMSS format
 */
public void setTime(String txt) {
     Time = txt;
}
/** Gets the current system time when the reading was made in HHMMSS format.
 * @return the current system time in HHMMSS format
 */
public String getTime(){
    return Time;
}

/** Sets the number of attempts the user made to store the data correctly.
 * @param txt the number of tries
 */
public void setNumberOfTries(String txt) {
     NumberOfTries = txt;
}
/** Gets the number of attempts the user made to store the data correctly.
 * @return the number of tries
 */
public String getNumberOfTries(){
    return NumberOfTries;
```

```
}
/** Gets the number of attempts the user made to store the data correctly.
 * @return the number of tries
 */
public int getNumberOfTriesInt(){
    int i = 0;
    try {
        i = new Integer(Integer.parseInt(getNumberOfTries().trim())).intValue();
    } catch (NumberFormatException e) {
    }
    return i;
}
/** Set message#1. This message contains either information from the backoffice
 * software or containg messages intended for the backoffice.
 * @param txt the message index
 */
public void setMessage1(String txt) {
     Message1 = txt;
}
/** Get message#1. This message contains either information from the backoffice
 * software or containg messages intended for the backoffice.
 * @return the message index
 */
public String getMessage1(){
    return Message1;
}
/** Get message#1. This message contains either information from the backoffice
 * software or containg messages intended for the backoffice.
 * @return the message index
 */
public int getMessage1Int(){
    int i = 0;
    try {
        i = new Integer(Integer.parseInt(getMessage1().trim())).intValue();
    } catch (NumberFormatException e) {
    }
    return i;
}

/** Set message #2. This message contains either information from the backoffice
 * software or containg messages intended for the backoffice.
 * @param txt the message index
 */
public void setMessage2(String txt) {
     Message2= txt;
}
/** Get message#2. This message contains either information from the backoffice
 * software or containg messages intended for the backoffice.
 * @return the message index
 */
public String getMessage2(){
    return Message2;
}
/** Get message#2. This message contains either information from the backoffice
 * software or containg messages intended for the backoffice.
 * @return the message index
 */
public int getMessage2Int(){
    int i = 0;
    try {
        i = new Integer(Integer.parseInt(getMessage2().trim())).intValue();
    } catch (NumberFormatException e) {
    }
    return i;
}

//Message3 bean
/** Set message #3. This message contains either information from the backoffice
 * software or containg messages intended for the backoffice.
 * @param txt the message index
 */
public void setMessage3(String txt) {
     Message3 = txt;
}
/** Get message#3. This message contains either information from the backoffice
 * software or containg messages intended for the backoffice.
 * @return the message index
 */
public String getMessage3(){
```

```
        return Message3;
    }
    /** Get message#3. This message contains either information from the backoffice
     * software or containg messages intended for the backoffice.
     * @return the message index
     */
    public int getMessage3Int(){
        int i = 0;
        try {
            i = new Integer(Integer.parseInt(getMessage3().trim())).intValue();
        } catch (NumberFormatException e) {
        }
        return i;
    }
    /** Sets the factor field
     * @param txt the factor
     */
    public void setFactor(String txt) {
         Factor= txt;
    }
    /** Gets the factor field
     * @return the factor
     */
    public String getFactor(){
        return Factor;
    }
}
```

Q.3.4 Object: `green.palmHHT.MessageInfo`




```
/*
 * MessageInfo.java
 *
 * Created on 15 March 2004, 11:06
 */

package green.palmHHT;

/** MessageInfo contains the fields required for the correct handling of messages
 * within the palmHHT system.
 * @author RDoorduin
 */
public class MessageInfo {

    private String message = "";
    private String status = "OLD";
    private final int MESSAGEINFO_FIELDS = 2;

    /** Creates a new instance of MessageInfo */
    public MessageInfo() {
    }

    /** Easy way of allowing scalability in the recordstucture, by providing the number
     * of fields used in the class.
     * @return number of fields
     */
    public int getNumberOfFields() {
        return MESSAGEINFO_FIELDS;
    }

    /** Gets the message stored in the object.
     * @return the message text
     */
    public String getMessage() {
        return message;
    }
    /** Gets the message stored in the object.
     * @param message the message text
     */
    public void setMessage(String message) {
        this.message = message;
        setStatus("NEW");
    }

    /** Gets the status of the message, e.g "OLD" or "NEW"
     * @return the status
     */
```

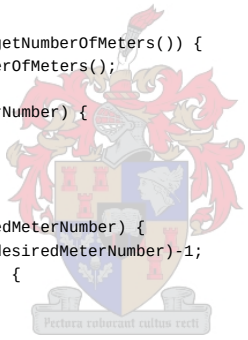
```
    public String getStatus() {
        return status;
    }
    /** Sets the status of the message, e.g "OLD" or "NEW"
     * @param status the status
     */
    public void setStatus(String status) {
        this.status = status;
    }

    /** Sets the message and status of the object.
     * @param message the message
     * @param status the status
     */
    public void setMessage(String message, String status) {
        setMessage(message);
        setStatus(status);
    }
}
```

Q.3.5 Object: `green.palmHHT.RouteRecords`

```
/*
 * routeRecords.java
 *
 * Created on 02 February 2004, 11:02
 */

package green.palmHHT;

import javax.microedition.rms.*;
/** The RouteRecords class is responsible for the interfacing with the PalmHHT with
 * the record structure in terms of organition of header and MeterInfo data.
 *
 * The RecordManager class is used extensively through this class.<BR>
 *
 * Please note that the currentMeterNumber is not the current RecordID. The
 * RecordID refers to the position of the particular field. The currentMeter number
 * refers to the position within the Route loaded into the recordset.<BR>
 *
 * The Header info is also included in these records. This is done for historic
 * reasons whene initially porting the software and making it compatible with the
 * backoffice software.
 * @author RDoorduin
 */
public class RouteRecords extends RecordManager {
    //Data Cariers
    private MeterInfo m = new MeterInfo();
    private HeaderInfo h = new HeaderInfo();

    //Position Control
    private int RecordIDlastPositon = 0;
    private int RecordIDstartMeterInfoID = 0;   //For offset calculations
    private int currentMeterNumber = -1;        //To keep track of current MeterInfo Record Position

    //Specific Record Variables
    private int posMeterNumber = 0;
    private int posNewReading = 0;
    private int posReadingDate = 0;
    private int posTime = 0;
    private int posNumberOfTries = 0;
    private int posMessage3 = 0;

    /** Creates a new instance of routeRecords */
    public RouteRecords() {
        super.setRecordStoreID("HHT");      //Points to correct database
        super.openRecords();                //Open database
        super.getNextRecord();              //Point to first record in database
    }

/*
 * Record Handling ---------------------------------------------------------------------------
 */
    // New with Enumeration included

    /** The number of meters in a route is calculated with this method
     * @return The number of meters in the recordStore
```

```
 */
public int getNumberOfMeters() {
    return (super.getNumRecords()-RecordIDstartMeterInfoID)/m.getNumberOfFields();
}

/** The current meter number is stored in the class and is updated continuesly. This
 * method returns the value of the current Meter Info contained in the MeterInfo
 * object.
 * @return The current Meter number
 */
public int getCurrentMeterNumber() {
    return currentMeterNumber;
}

/** Advances to the next MeterInfo in the recordsStore
 * The positions of the field which should be to are stored in approriate variables
 * for use with the writeXXXX methods.
 * @return The MeterInfo is returned
 */
public MeterInfo getNextMeterInfo() {
    if (this.getCurrentMeterNumber() == -1) {
        this.readHeaderInfo();
    }
    if (this.getCurrentMeterNumber() < this.getNumberOfMeters()) {
        m.setAccountNumber(new String(super.getNextRecord()));
        m.setMeterID(new String(super.getNextRecord()));
        m.setSituationAdress(new String(super.getNextRecord()));
        m.setStreetNumber(new String(super.getNextRecord()));
        m.setMeterType(new String(super.getNextRecord()));
        m.setOwnerName(new String(super.getNextRecord()));
        m.setInitials(new String(super.getNextRecord()));
        posMeterNumber = super.getNextRecordID();
        m.setMeterNumber(new String(super.getRecord(posMeterNumber)));
        m.setOldReading(new String(super.getNextRecord()));
        m.setAverageUsage(new String(super.getNextRecord()));
        m.setPercentage(new String(super.getNextRecord()));
        m.setMNM(new String(super.getNextRecord()));
        posNewReading = super.getNextRecordID();
        m.setNewReading(new String(super.getRecord(posNewReading)));
        m.setReadingCode(new String(super.getNextRecord()));
        posReadingDate = super.getNextRecordID();
        m.setReadingDate(new String(super.getRecord(posReadingDate)));
        m.setNumberOfDigits(new String(super.getNextRecord()));
        posTime = super.getNextRecordID();
        m.setTime(new String(super.getRecord(posTime)));
        posNumberOfTries = super.getNextRecordID();
        m.setNumberOfTries(new String(super.getRecord(posNumberOfTries)));
        m.setMessage1(new String(super.getNextRecord()));
        m.setMessage2(new String(super.getNextRecord()));
        posMessage3 = super.getNextRecordID();
        m.setMessage3(new String(super.getRecord(posMessage3)));
        m.setFactor(new String(super.getNextRecord()));
        currentMeterNumber++;
    }
    return m;
}

/*
 * Reading the previous recordSet requires revers order to store the data correctly
 */

/** Retrieves the previous MeterInformation
 * @return The MeterInfo is returned
 */
public MeterInfo getPreviousMeterInfo() {
    this.skipPreviousMeterInfo(); //current record
    this.skipPreviousMeterInfo(); //prevrecord
    m = this.getNextMeterInfo();
    return m;
}

/** Skips onto the next MeterInfo segment. The thinking behind this method is that
 * it should execute faster, because no instructions is wasted on copying the data
 * to the correct memory fields.
 */
public void skipNextMeterInfo() {
    if (this.getCurrentMeterNumber() == -1) {
        this.readHeaderInfo();
    }
```

```
        if (this.getCurrentMeterNumber() < this.getNumberOfMeters()) {
            for (int i = 0; i < m.getNumberOfFields(); i++) {
                super.getNextRecordID();
            }
            currentMeterNumber++;
        }
    }

    /** Skips onto the previous MeterInfo segment.
     * @see skipNextMeterInfo()
     */
    public void skipPreviousMeterInfo() {
        if (this.getCurrentMeterNumber() < 0) {
            this.readHeaderInfo();
        } else {
            if (this.getCurrentMeterNumber() != 0) {
                for (int i = 0; i < m.getNumberOfFields(); i++) {
                    super.getPreviousRecordID();
                }
                currentMeterNumber--;
            }
        }
    }

    /** For skipping to a specific MeterInfo section.
     * @param desiredMeterNumber The meterNumber which MeterInfo is required. The numbering starts at
     * <CODE>0</CODE> and
     * contiuues to <CODE>getNumberOfMetere()-1</CODE>
     */
    public void advanceToMeterInfo(int desiredMeterNumber) {
        int tmpCount = 0;
        if (currentMeterNumber < 0) {
            desiredMeterNumber = 0;
        } else if (currentMeterNumber > this.getNumberOfMeters()) {
            desiredMeterNumber = this.getNumberOfMeters();
        }
        if (currentMeterNumber == desiredMeterNumber) {
            this.getNextMeterInfo();
        } else if (desiredMeterNumber == 0) {
            this.readHeaderInfo();
            this.getNextMeterInfo();
        } else if (currentMeterNumber > desiredMeterNumber) {
            tmpCount = (currentMeterNumber - desiredMeterNumber)-1;
            for (int i = 0; i < tmpCount; i++) {
                this.skipPreviousMeterInfo();
            }
            this.getPreviousMeterInfo();
        } else if (currentMeterNumber < desiredMeterNumber) {
            tmpCount = (desiredMeterNumber - currentMeterNumber)-1;
            for (int i = 0; i < tmpCount; i++) {
                this.skipNextMeterInfo();
            }
            this.getNextMeterInfo();
        }
    }

/*
 * METER INFO ---------------------------------------------------------------------------------
 */

    /** Presents the MeterInfo contained in the current MeterInfo object in the object.
     * @return The current MeterInfo
     */
    public MeterInfo getMeterInfo() {
        return m;
    }

    /** Writes the MeterNumber of the current MeterInfo to the recordsStore.
     * The MeterNumber is the number labeled on the meter to be read.
     * @param str The Meter Number.
     */
    public void writeMeterNumber(String str) {
        super.setRecord(posMeterNumber,str.getBytes());
    }
    /** Writes the New Reading of the current MeterInfo to the recordsStore.
     * The New Reading is the actual reading from the display of the currentMeter.
     * @param str The Current Reading on the display of the meter.
     */
    public void writeNewReading(String str) {
```

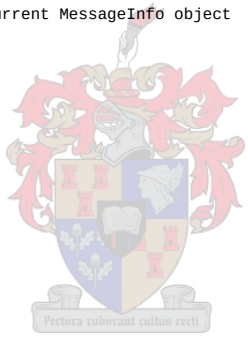


```
        super.setRecord(posNewReading,str.getBytes());
    }
    /** Writes the Date of the current MeterInfo to the recordsStore.
     * The Date is the current system time of the device.
     */
    public void writeCurrentDate() {
        super.setRecord(posReadingDate,green.palmHHT.utils.getCurrentDate().getBytes());
    }
    /** Writes the Time of the current MeterInfo to the recordsStore.
     * The Time is the current system time of the device.
     */
    public void writeCurrentTime() {
        super.setRecord(posTime,green.palmHHT.utils.getCurrentTime().getBytes());
    }

    /** Writes the Number of Tries of the current MeterInfo to the recordsStore.
     * The Number of Tries represents the amount of times the user tried to enter the
     * correct value into the record.
     * @param str The amount of times tried to enter the correct records.
     */
    public void writeNumberOfTries(String str) {
        super.setRecord(posNumberOfTries,str.getBytes());
    }
    /** Writes the Message Index of a message which the user would like to send to the
     * backoffice software.
     * This number coresponds to the message number in the MessageInfo field.
     * @param str The message index
     */
    public void writeMessage3(String str) {
        super.setRecord(posMessage3,str.getBytes());
    }
/*
 * HEADER INFO ---------------------------------------------------------------------------
 */
    /** Returns the HeaderInfo contained in the unit.
     * @return The headerinfo of the route.
     */
    public HeaderInfo getHeaderInfo() {
        return h;
    }

    /** Reads the HeaderInfo from the recordsStore.
     * @return The HeaderInfo
     */
    public HeaderInfo readHeaderInfo() {
        super.resetRecords();
        if (super.hasNextRecord()) {
            super.getNextRecord();
            h.setRouteNo(new String(super.getNextRecord()));
            h.setTries(new String(super.getNextRecord()));
            h.setAskMeterNumber(new String(super.getNextRecord()));
            h.setDisplayPreviousReading(new String(super.getNextRecord()));
            h.setDisplayOwnerName(new String(super.getNextRecord()));
            RecordIDlastPositon=this.getNextRecordID();
            h.setLastRecordNumber(new String(super.getRecord(RecordIDlastPositon)));
            this.RecordIDstartMeterInfoID = super.getThisRecordID();
            this.currentMeterNumber = 0;
            super.lastErrorMessage = ""+this.RecordIDstartMeterInfoID;
            return h;
        } else {
            return null;
        }
    }

    /** Writes the MeterNumber of the current MeterInfo to the Header of the
     * recordsStore. This is to ensure that when the palm is switched of and on again,
     * the latest accessed MeterInfo records appear.
     */
    public void writeCurrentMeterNumber() {
        super.setRecord(RecordIDlastPositon,(""+this.currentMeterNumber).getBytes());
    }
}
```

Q.3.6 Object: `green.palmHHT.MessageRecords`

```
/*
 * MessageRecords.java
```

```
 *
 * Created on 13 March 2004, 10:04
 */

package green.palmHHT;

/** The MessageRecords class is responsible for the interfacing with the PalmHHT with
 * the record structure in terms of organition of the MessageInfo data. The RecordManager
 * class is used extensively through this class.
 * @author RDoorduin
 */
public class MessageRecords extends RecordManager {
    private MessageInfo m = new MessageInfo();
    private final int offset = 1;

    /** Creates a new instance of MessageRecords */
    public MessageRecords() {
        setRecordStoreID("HHTMes");
        openRecords();
        super.getNextRecord();
    }
/*
 * Message handling
 */

    /** Returns the current MessageInfo which is obtained in the object.
     * @return The MessageInfo object
     */
    public MessageInfo getMessageInfo() {
        return m;
    }
    /** Set the Message of the current MessageInfo object.
     * @param txt The Message to set in the current MessageInfo object
     */
    public void setMessage(String txt) {
        m.setMessage(txt);
    }
    /** Set the status of the message, e.g.
     * "OLD" or
     * "NEW"
     * @param txt The Status of the message
     */
    public void setStatus(String txt) {
        m.setStatus(txt);
    }

/*
 * Record Handling ---------------------------------------------------------------------------
 */
    //New Routing for use with enumeration
    /** The number of message stored in the recordsStore
     * @return The number of Messages in the recordsStore
     */
    public int getNumberOfMessages() {
        return (super.getNumRecords()-1)/m.getNumberOfFields();
    }

    /** Advances to the next message in the recordsStore
     * @return The MessageInfo of the current Message
     */
    public MessageInfo getNextMessage() {
        String tmpString = new String(super.getNextRecord());
        m.setMessage(new String(super.getNextRecord()));
        m.setStatus(tmpString);
        return m;
    }

    /** Reads the message prior to the current message from the recordsStore
     * @return The MessageInfo of the current Message
     */
    public MessageInfo getPreviousMessage() {
        m.setMessage(new String(super.getPreviousRecord()));
        m.setStatus(new String(super.getPreviousRecord()));
        return m;
    }

    /** Resets the current recordsStore and message index. */
    public void resetMessages() {
        super.resetRecords();
        super.getNextRecordID();
```



```
    }

    /** Add a new message to the current recordStore. The values of this message must be
     * set up with the setMessage and setStatus, prior to invoking this method.
     */
    public void addNewMessageInfo() {
        super.addRecord(m.getStatus().getBytes());
        super.addRecord(m.getMessage().getBytes());
        super.rebuildRecords();
    }
}
```

Q.3.7 Object: `green.palmHHT.RecordManager`

```
/*
 * RecordManager.java
 *
 * Created on 13 March 2004, 09:49
 * @version 2.0
 *
 * Change History:
 * v2.0 Uses the RecordEnumeration to implement a sequencial record structure.
 *          This required a great deal of reprogramming, but access to the records have proven
 *          more accurate than previous method.
 * v1.0 First design based on direct record access, assuming sequencial record structure.
 */

package green.palmHHT;

import javax.microedition.rms.*;
/** Record manager is a top layer onto the recordStore class, provided by the
 * javax.microedition.rms class. This class provides a easy interface to the record
 * structure on the palm.<BR><BR>
 *
 * A good explenation of the record structure van be found at
 * <HTML>
 *  <a href="http://developers.sun.com/techtopics/mobility/midp/articles/databaserms">
 *    Databases and MIDP, Part 1
 *  </a>
 * </HTML>
 * <BR><BR>
 * A Quick Summary and Guide to Records and RecordStores:<BR>
 *
 * A <b>record</b> is an individual item in the record store. It is made up of a array of
 * bytes. One can only store and retrieve the information in this format. A record
 * in this context does not contain any fields.<BR>
 *
 * A <b>recordStore</b> is a collection of records. Each record store contains a
 * name or ID. In the recordStore the records are added sequencially. Although
 * implied that recordsStore will be stored with the first record at 1 and the next
 * at 2, it was found that recordsStores does not always behave as such. To access
 * the records sequencially the use of the <I>enumeration</I> is required.<BR><BR>
 *
 * In this code physicle refers to the index of the records store associated with
 * the record. This assumes that the enumerated index and the physical index are not
 * the same.
 * @author RDoorduin
 */
public class RecordManager {
    private RecordStore readingsStore = null; //Create storage space
    private String readingStoreID;
    /** This provides a means for communication of possible errors occuring during
     * implementation.
     */
    public String lastErrorMessage = "";
    private RecordEnumeration recEnum = null;

    /** Creates a new instance of routeRecords */
    public RecordManager() {
    }

    /** Each record store or database is accessed using a name or ID. This method sets
     * the name/ID for this specific RecordStore which will be used.
     * @param readingStoreID The name of the database to be accessed by the recordStore
     */
    public void setRecordStoreID(String readingStoreID){
        this.readingStoreID = readingStoreID;
```

```
}

/** Inidicates the available space available in memory for records.
 * @return Available memory in bytes
 */
public int getAvailableSpace() {
    if (readingsStore != null) {
        try {
          openRecords();
          int tmpI = readingsStore.getSizeAvailable();
          return tmpI;
        } catch (RecordStoreException ex) {
            return 0;
        }
    } else {
        return 0;
    }
}

/** Indicates the number of records in the current recordsStore
 * @return Number of records in recordsStore
 */
public int getNumRecords() {
    if (this.recEnum == null) {
        return 15000;
    } else {
        return this.recEnum.numRecords();
    }
}

/** Opens the recordsStore for the specified database/name or recordStoreID.
 * @return if successefull, true
 */
protected boolean openRecords() {
    try {
        if (readingStoreID != null) {
            if (readingsStore == null) {
                readingsStore = RecordStore.openRecordStore(readingStoreID, true);
                if (readingsStore.getNumRecords() < 13000) {
                    try {
                        recEnum = readingsStore.enumerateRecords(null,null,false); //false => no memory problems
                    } catch (java.lang.OutOfMemoryError e) {
                        recEnum = null;
                        lastErrorMessage = e.getMessage();
                    }
                } else {
                    recEnum = null;
                    lastErrorMessage = "Records size too large!";
                }
            }
            return true;  // Return true if open
        }
    } catch (RecordStoreException e) {
        lastErrorMessage = e.getMessage();
    }
    return false;
}

/** Gets the contents of the next record in the sequence.
 * If no record is available, or an exception was thrown, the contents will reflect
 * the error.
 * @return Content of the next record
 */
protected byte[] getNextRecord() {
    if (recEnum != null) {
        try {
            if (this.recEnum.hasNextElement()) {
                byte[] tmpB = this.recEnum.nextRecord();
                if (tmpB == null) {
                    System.out.println("getNextRecord:Null");
                    System.out.flush();
                    return new String("Null").getBytes();
                } else {
                    return tmpB;
                }
            } else {
                return new String("NoNewRecord").getBytes();
            }
        } catch (RecordStoreNotOpenException e) {
```

```
            return new String("getNextRec:RSnOE:"+e.getMessage()).getBytes();
        } catch (InvalidRecordIDException e) {
            return new String("getNextRec:IRIDE:"+e.getMessage()).getBytes();
        } catch (RecordStoreException e) {
            return new String("getNextRec:RSE:"+e.getMessage()).getBytes();
        }
    } else {
        System.out.println("getNextRecord:recEnum=Null");
        System.out.flush();
        return new String("recEnum=null").getBytes();
    }
}

/** Gets the contents of the previous record in the sequence.
 * If no record is available, or an exception was thrown, the contents will reflect
 * the error.
 * @return Content of the previous record
 */
protected byte[] getPreviousRecord() {
    if (recEnum != null) {
        try {
            if (this.recEnum.hasPreviousElement()) {
                byte[] tmpB = this.recEnum.previousRecord();
                return tmpB;
            } else {
                return new String("NoNewRecord").getBytes();
            }
        } catch (RecordStoreNotOpenException e) {
            return new String("getNextRec:RSnOE:"+e.getMessage()).getBytes();
        } catch (InvalidRecordIDException e) {
            return new String("getNextRec:IRIDE:"+e.getMessage()).getBytes();
        } catch (RecordStoreException e) {
            return new String("getNextRec:RSE:"+e.getMessage()).getBytes();
        }
    } else {
        return new String("recEnum=null").getBytes();
    }
}

/** Resets the current record to the initial state, after initialization of the
 * enumeration or after a rebuild.
 */
protected void resetRecords() {
    this.recEnum.reset();
}
/** Rebuilds the enumeration. This operation should be performed after a change or
 * addition to the recordStore.
 */
protected void rebuildRecords() {
    this.recEnum.rebuild();
}

/** Verifies if there is another record available in the secuence.
 * @return True if a record is available
 */
protected boolean hasNextRecord() {
    return this.recEnum.hasNextElement();
}
/** Verifies if there is another record prior to the current record is available in the secuence.
 * @return True, if a record is available
 */
protected boolean hasPreviousRecord() {
    return this.recEnum.hasPreviousElement();
}

/** This method determines the last record read's physicle record index in the
 * recordsStore structure
 * @return The last read record ID.
 */
protected int getThisRecordID() {
    int thisID = 0;
    try {
        if (this.recEnum.hasNextElement()) {
            thisID = this.recEnum.nextRecordId();
            this.recEnum.previousRecordId();
        } else if (this.recEnum.hasPreviousElement()) {
            thisID = this.recEnum.previousRecordId();
            this.recEnum.nextRecordId();
        } else {
```

```
            thisID = 0;
        }
    } catch (InvalidRecordIDException e) {
        thisID = 0;
    }
    return thisID;
}

/** Determines the next record ID.
 * @return The ID of the next record.
 */
protected int getNextRecordID() {
    try {
        return this.recEnum.nextRecordId();
    } catch (InvalidRecordIDException e) {
        return 0;
    }
}
/** Determines the previous record's ID in the sequence.
 * @return The previous record's ID
 */
protected int getPreviousRecordID() {
    try {
        return this.recEnum.previousRecordId();
    } catch (InvalidRecordIDException e) {
        return 0;
    }
}

/** Adds a record to the record sequence. <b>The enumeration is not rebuild
 * automaticaly when calling this method.</b>
 * @return The physical Record ID of the new record.
 * @param b The data to be added and associated with this record.
 */
protected int addRecord(byte[] b) {
    if (readingsStore != null) {
        try {
            openRecords();
            int tmpI = readingsStore.addRecord(b,0,b.length);
            return tmpI;
        } catch (InvalidRecordIDException ridex) {
            return 0;
        } catch (RecordStoreException ex) {
            return 0;
        }
    } else {
        return 0;
    }
}

/** Sets the value of a physicle record in the recordStore.
 * @return The changed record ID.
 * @param recordID The physicle record ID
 * @param b The data to be entered into the record
 */
protected int setRecord(int recordID, byte[] b) {
    if (readingsStore != null) {
        try {
            openRecords();
            readingsStore.setRecord(recordID,b,0,b.length);
            return recordID;
        } catch (InvalidRecordIDException ridex) {
            return addRecord(b);
        } catch (RecordStoreException ex) {
            return 0;
        }
    } else {
        return 0;
    }
}

/** Retrieves the data of a physicle record.
 * @param recordID The record ID or physicle index of the record structure
 * @return The data stored in the record specified
 */
protected byte[] getRecord(int recordID) {
    if (readingsStore != null) {
        try {
            openRecords();
```

```
                byte[] tmpB = readingsStore.getRecord(recordID);
                if (tmpB == null) {
                    System.out.println("getRecord:Null");
                    System.out.flush();
                    return new String("Null").getBytes();
                } else {
                    return tmpB;
                }
            } catch (InvalidRecordIDException ridex) {
                return new String("InvRecID:["+recordID+"]"+getNextRecordID()).getBytes();
            } catch (RecordStoreException ex) {
                return new String("RecordStoreEx").getBytes();
            }
        } else {
            System.out.println("getRecord:Null");
            System.out.flush();
            return new String("Null").getBytes();
        }
    }
}
```

Q.3.8 Object: `green.palmHHT.utils`

```
/*
 * utils.java
 *
 * Created on 19 June 2004, 02:18
 */

package green.palmHHT;

import java.util.*;
/** This class gives additional support by providing special methods to other
 * classes.
 * @author RDoorduin
 */
public class utils {

    /** Creates a new instance of utils */
    public utils() {
    }
/*
 * DATE/TIME INFO ----------------------------------------------------------------------------
 */

    /** Returns the current system time in a HHMMSS string format.
     * @return The time in String format
     */
    public static String getCurrentTime() {
        String timeString = new String("");
        String tmpStr = new String();
        //Hour
        tmpStr = ""+Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
        if (tmpStr.length() == 1) {
            tmpStr = "0"+tmpStr;
        }
        timeString = timeString + tmpStr;
        //Minute
        tmpStr = ""+Calendar.getInstance().get(Calendar.MINUTE);
        if (tmpStr.length() == 1) {
            tmpStr = "0"+tmpStr;
        }
        timeString = timeString + tmpStr;
        //Seconds
        tmpStr = ""+Calendar.getInstance().get(Calendar.SECOND);
        if (tmpStr.length() == 1) {
            tmpStr = "0"+tmpStr;
        }
        timeString = timeString + tmpStr;
        return timeString;
    }

    /** Returns the current system date in a DDMMYYYY string format.
     * @return The date in string format as specified
     */
    public static String getCurrentDate() {
        String timeString = new String("");
```

```
        String tmpStr = new String();

        //Day
        tmpStr = ""+Calendar.getInstance().get(Calendar.DAY_OF_MONTH);
        if (tmpStr.length() == 1) {
            tmpStr = "0"+tmpStr;
        }
        timeString = timeString + tmpStr;

        //Month
        tmpStr = ""+(Calendar.getInstance().get(Calendar.MONTH)+1);
        if (tmpStr.length() == 1) {
            tmpStr = "0"+tmpStr;
        }
        timeString = timeString + tmpStr;

        //Year
        tmpStr = ""+(Calendar.getInstance().get(Calendar.YEAR));
        if (tmpStr.length() == 1) {
            tmpStr = "200"+tmpStr;
        } else if (tmpStr.length() == 2) {
            tmpStr = "20"+tmpStr;
        } else if (tmpStr.length() == 3) {
            tmpStr = "2"+tmpStr;
        }
        timeString = timeString + tmpStr;
        return timeString;
    }
}
```

Q.4 Package: `green.HHTSync`

Q.4.1 Object: `green.HHTSync.HHTCond`


```
/*
 * HHTCond.java
 *
 * Created on 03 February 2004, 02:59
 */

package green.HHTSync;

import palm.conduit.*;

import green.palmHHT.*;

import peg.io.*;
import peg.utils.*;

import java.util.*;
import java.io.*;
import javax.swing.*;

/** HHTCond interfaces with the JSync class provided by Palm. This classs interacts with
 * the HotSync process. Only one synchronization process may be active, therefore
 * only one HotSync process can call the Conduit at a time. It is also important
 * that the other methods which are not used by the HotSync process be declared as
 * private.
 * @author RDoorduin
 */
public class HHTCond implements Conduit {
    private HHTConduitSetupData data = new HHTConduitSetupData();

    private String defaultPath = "";
    private String userName = "";
    private String oldRouteNumber = "";
    private String newRouteNumber = "";
    private final String creatorID = "-Ppmr";
    private String routeDataBase = "HHT"+creatorID;
    private String messageDataBase = "HHTMes"+creatorID;

    private int db; //Current palm database handle
    private int[] headerInfoStructure = {0};
    private String[] meterInfoHeaders = {"AccountNo",
```

```
                                "ID",
                                "Address",
                                "StreetNo",
                                "MeterType",
                                "OwnerName",
                                "Initials",
                                "MeterNo",
                                "OldReading",
                                "AvgUsage",
                                "Percentage",
                                "MNM",
                                "NewReading",
                                "ReadingCode",
                                "ReadingDate",
                                "NumberOfDigits",
                                "Time",
                                "NumberOfTries",
                                "Message1",
                                "Message2",
                                "Message3",
                                "Factor"}; //FlatFile columnHeaders
private int[] meterInfoStructure = {12,3,14,4,1,18,2,8,8,8,3,4,8,2,8,1,6,1,3,3,3,8};
                                                                    //FlatFile field spacing
private String[] messageInfoHeaders = {"ID",
                                "O/N",
                                "Message"}; //FlatFile columnHeaders
private int[] messageInfoStructure = {3,3,0}; //FlatFile field spacing

private final boolean LOGON = true;  //To log all debug data
private debug debugThis = new debug("HHTConduit",8);

private peg.swing.progressBar pb = null;

/** Creates a new instance of HHTCond */
public HHTCond() {
}

/** Main class, if the class is invoked by the Java Virtual Machine as a exectable
 * program. This method was added to allow easy access to the configuration of the
 * conduit to the user.
 * @param args any arguments that may be passed
 */
public static void main(String args[]) {
    System.getProperties().list(System.out);
    new HHTSyncGUI().show();    //When invoked the Setup GUI is initiated.
}

/** This method is invoked by the HotSync | Custom... setup program.
 * @param info any information sent from the HotSync program
 * @return any message intended for the HotSync program, defaulted to 0.
 */
public int configure(ConfigureConduitInfo info) {
    new HHTSyncGUI(true).show();    //The Setup GUI is started.
    return 0;
}

/** To specify the name of the conduit. This is also required by the HotSync | Custom
 * configuratuion program and Syncronization process.
 * @return the name of the conduit
 */
public String name() {
    return "metMeter Conduit";
}

/** This method is invoked by the HotSync Syncronization process. Properties are
 * passed to the method to retrieve user and system information.
 * @param props the properties, e.g. user info
 */
public void open(SyncProperties props) {
    try {
        pb = new peg.swing.progressBar("palmHHT Synchronization",0,6);

        Log.startSync();
        debugThis.print(1,"open():Log Started!");

        debugThis.print(1,"open():SetupConduitData()...");
        SetupConduitData(props);

        debugThis.print(1,"open():downloadRouteRata()");
```

```
        downloadRouteData();

        debugThis.print(1,"open():downloadMessages()");
        downloadMessages(this.oldRouteNumber);

        debugThis.print(1,"open():uploadRouteData("+props.userName+"):...");
        uploadRouteData(props.userName);

        debugThis.print(1,"open():uploadRouteData("+props.userName+"):...");
        uploadMessagesData(this.newRouteNumber);

        pb.dispose();
        pb = null;
        Log.out("OK Power Pocket Meter Reader");
    } catch (Throwable t) {
        t.printStackTrace();
        Log.abortSync();
    }
    Log.endSync();
}

//Private decleration follow below...

/** ConduitData is setup in this method. From both the Syncproperties and
 * the properties stored in HHTConduitSetup object.
 * @param props the properties, e.g. user info
 */
private void SetupConduitData(SyncProperties props) {
    pb.setMin(0);
    pb.setMax(1);
    pb.setValue(0,"Setting up data...");
    this.userName = props.userName;
    debugThis.print(3, "SetupConduitData():userName="+this.userName);
    this.defaultPath = props.pathName;
    if (this.data != null) {
        if (this.data.getDataPath() != null) {
            if (this.data.getDataPath().length() != 0) {
                this.defaultPath = this.data.getDataPath()+"\\";
            }
        }
    }
    if (props.remoteNames.length == 0) {
        this.routeDataBase = "HHT"+creatorID;
        this.messageDataBase = "HHTMes"+creatorID;
    } else {
        this.routeDataBase = props.remoteNames[0]+creatorID;
        this.messageDataBase = props.remoteNames[0]+creatorID;
    }
    debugThis.print(3, "SetupConduitData():routeDataBase="+this.routeDataBase);
    debugThis.print(3, "SetupConduitData():messageDataBase="+this.messageDataBase);
    debugThis.print(3, "SetupConduitData():defaultPath="+defaultPath);
    pb.setValue(1);
}

/** This method is responsible for downloading the all RouteData from the palm
 * onto a file which can be processed by the backoffice software.
 */
private void downloadRouteData() {
    String tmpStr = new String();
    StringRecord sR = new StringRecord();
    String fileName = new String();
    peg.io.flatFile ff = null;
    peg.sql.DBConnect myDB = null;
    String dbQueryStr = new String();
    int columnCount = 0;
    boolean noDBData = true;
    int recId = 0;
    String dateTimeString = new String();
    String downloadTimeString = null;

    pb.setValue(0,"Downloading Route Information...");
    if (openSyncDB(this.routeDataBase)) {
        try {
            sR.setIndex(1);    //Route Number
            SyncManager.readRecordByIndex(db,sR);
            this.oldRouteNumber = sR.getString();
            debugThis.print(3,"downloadRouteData():oldRouteNumber = "+this.oldRouteNumber);

            //FlatFile Implementation
```

```
fileName = defaultPath+"READ"+this.oldRouteNumber;//+".dat";
debugThis.print(4, "downloadRouteData():filename="+fileName);
ff = new peg.io.flatFile(fileName,this.meterInfoStructure);

//SQL Database Implementation
if (data.getDBActive()) {
    //Setup database
    myDB = new peg.sql.DBConnect(data.getDBDriver(),data.getDBProtocol(),data.getDBSubname(),data.getDBUser(),data.getDBPasswo
    myDB.executeSQL("CREATE TABLE IF NOT EXISTS palmHHT (accountNo VARCHAR(12) NOT NULL, id VARCHAR(3) NOT NULL, meterNo VARCH
    if (myDB.isQuerySuccess()) {
        debugThis.print(3,"downloadRouteData():DBUpdated!");
    } else {
        debugThis.print(3,"downloadRouteData():DB:Tabele not created... already exists:" + myDB.getErrorString());
    }
}

recId = new green.palmHHT.HeaderInfo().getNumberOfFields(); //point to first record after the header.
debugThis.print(5,"downloadRouteData():recId = "+recId);

ff.open(true); //for writing to the stream
pb.setMin(0);
pb.setMax(SyncManager.getDBRecordCount(db));
while ((recId+1) < SyncManager.getDBRecordCount(db)) {
    sR.setIndex(++recId);
    SyncManager.readRecordByIndex(db, sR);
    tmpStr = sR.getString();
    ff.appendNextColumn(tmpStr); //Add next value to output file.
    pb.setValue(recId);
    if (myDB != null) {
        switch (columnCount++) {
            case 0:
                dbQueryStr = dbQueryStr + tmpStr + ", ";
                break;
            case 1:
                dbQueryStr = dbQueryStr + tmpStr + ", ";
                break;
            case 7:
                dbQueryStr = dbQueryStr + tmpStr + ", ";
                break;
            case 12:
                dbQueryStr = dbQueryStr + tmpStr + ", ";
                break;
            case 14:
                debugThis.print(3, "downloadRouteData():date: "+tmpStr);
                if (tmpStr.length() == 8) {
                    dateTimeString = "'"+ tmpStr.substring(4,8)+"/"+tmpStr.substring(2,4)+"/"+tmpStr.substring(0,2) + " ";
                }
                break;
            case 16:
                if (tmpStr.length() == 6) {
                    dateTimeString = dateTimeString + tmpStr.substring(0,2)+":"+tmpStr.substring(2,4)+":"+tmpStr.substring(4,6
                    if (downloadTimeString == null) {
                        downloadTimeString = dateTimeString;
                    }
                    dbQueryStr = dbQueryStr + dateTimeString + "," + downloadTimeString;
                    noDBData = false;
                }
                break;
        }
        if (columnCount >= this.meterInfoStructure.length) {
            debugThis.print(3,"downloadRouteData():INSERT INTO palmhht VALUES("+ dbQueryStr +");");
            if (!noDBData) {
                myDB.executeSQL("INSERT INTO palmhht VALUES("+ dbQueryStr+ ");");
                if (myDB.isQuerySuccess()) {
                    debugThis.print(3,"downloadRouteData():DBUpdated!");
                } else {
                    debugThis.print(3,"downloadRouteData():DB:Error inserting data:" + myDB.getErrorString());
                }
            }
            dbQueryStr = new String("");
            noDBData = true;
            columnCount = 0;
        }
    }
}
if (myDB != null) {
    myDB.close();
}
debugThis.print(3,"downloadRouteData():Closing Bufferd Output...");
```

```
        } catch (SyncException e) {
            debugThis.print(2,"downloadRouteData():SyncException:" +e.getMessage());
            debugThis.print(2,"downloadRouteData():SyncException:recId="+recId);
        } catch (IOException e) {
            debugThis.print(2,"downloadRouteData():IOException:" +e.getMessage());
        }
        this.closeSyncDB();
        debugThis.print(4,"downloadRouteData():closedSyncDB!");
        if (ff != null) {
            ff.close();
        }
    } else {
        debugThis.print(2,"downloadRouteData():Could not open dataBase on Palm");
    }
}

/** This method loads all the messages from the palm onto a flatFile which can be processed by
 * backoffice software.
 * @param routeNumber the number of the current route which should is being stored.
 */
private void downloadMessages(String routeNumber) {
    String tmpStr = new String();
    StringRecord sR = new StringRecord();
    String fileName = new String();
    peg.io.flatFile ff = null;
    int messageRecordNumber = 0;
    int recId = 0;

    pb.setValue(0,"Downloading Stored Messages...");
    if (openSyncDB(this.messageDataBase)) {
        try {
            sR.setIndex(1);    //Route Number
            SyncManager.readRecordByIndex(db,sR);
            this.oldRouteNumber = sR.getString();
            debugThis.print(3,"downloadMessages():oldRouteNumber = "+this.oldRouteNumber);

            fileName = defaultPath+"MESS"+routeNumber;//+".dat";
            debugThis.print(4, "downloadMessages():filename="+fileName);
            ff = new peg.io.flatFile(fileName,this.messageInfoStructure);

            ff.open(true); //for writing to the stream
            pb.setMin(0);
            pb.setMax(SyncManager.getDBRecordCount(db));
            while ((recId+1) < SyncManager.getDBRecordCount(db)) {
                tmpStr = ""+(++messageRecordNumber);
                ff.appendNextColumn(tmpStr); //Add next value to output file.
                debugThis.print(9,"downloadMessages():MessageNumber="+tmpStr);

                sR.setIndex(++recId);
                SyncManager.readRecordByIndex(db, sR);
                tmpStr = sR.getString();
                ff.appendNextColumn(tmpStr); //Add next value to output file.
                debugThis.print(9,"downloadMessages():Status="+tmpStr);
                pb.setValue(recId);

                sR.setIndex(++recId);
                SyncManager.readRecordByIndex(db, sR);
                tmpStr = sR.getString();
                ff.appendNextColumn(tmpStr); //Add next value to output file.
                debugThis.print(9,"downloadMessages():Message="+tmpStr);
                pb.setValue(recId);
            }
            ff.appendNextRow(); //Ensure that file end with a CR and LF  2004_08_17
            debugThis.print(3,"downloadMessages():Closing Bufferd Output...");
        } catch (SyncException e) {
            debugThis.print(2,"downloadMessages():SyncException:" +e.getMessage());
            debugThis.print(2,"downloadMessages():SyncException:recId="+recId);
        } catch (IOException e) {
            debugThis.print(2,"downloadMessages():IOException:" +e.getMessage());
        }
        this.closeSyncDB();
        if (ff != null) {
            ff.close();
        }
    } else {
        debugThis.print(2,"downloadMessages():Could not open dataBase on Palm");
    }
}
```

```
/** This method loads the Header Info onto the palm. The file from which the
 * data is read is is prepared by the backoffice software.
 * The username provided points to the specific file which stores the header information.
 * @param username the username registered on the palm
 */
private void uploadHeaderInfo(String username) {
    String tmpLine = new String();
    String tmpStr = new String();
    StringRecord sR = new StringRecord();
    String fileName = new String();
    peg.io.flatFile ff = null;

    debugThis.print(0, "uploadHeaderInfo():Buffering Filename...");
    if (username.length() > 8) {
        fileName = defaultPath+username.substring(0,8)+".PCT";
    } else {
        fileName = defaultPath+username + ".PCT";
    }
    debugThis.print(3, "uploadHeaderInfo():filename="+fileName);
    ff = new peg.io.flatFile(fileName,this.headerInfoStructure);

    try {
        sR.setString(0,"dummyRouteRec");
        SyncManager.writeRec(db, sR);

        if (ff.open()) {
            this.newRouteNumber = ff.getNextRow();
            debugThis.print(3, "uploadHeaderInfo():newRouteNumber="+this.newRouteNumber);
            sR.addString(this.newRouteNumber);
            SyncManager.writeRec(db, sR);

            sR.addString(ff.getNextRow());
            SyncManager.writeRec(db, sR);

            sR.addString(ff.getNextRow());
            SyncManager.writeRec(db, sR);

            sR.addString(ff.getNextRow());
            SyncManager.writeRec(db, sR);

            sR.addString(ff.getNextRow());
            SyncManager.writeRec(db, sR);
        } else {
            debugThis.print(3,"uploadHeaderInfo():file("+fileName+") could not open");
        }
        sR.addString("0");  // Record Number to be shown when starting application after reload.
        SyncManager.writeRec(db, sR);
    } catch (SyncException e) {
        debugThis.print(3,"uploadHeaderInfo():SyncException:" + e.getMessage());
    } catch (IOException e) {
        debugThis.print(3,"uploadHeaderInfo():IOException:" + e.getMessage());
    }
    ff.close();
}

/** The Route Data consist of a Header and Meter part. The Header method is called from
 * this method, and the Meter Data is uploaded here.
 * The data of the route is prepared by the backoffice software.
 * @param username the username registered on the palm
 */
private void uploadRouteData(String userName) {
    StringRecord sR = new StringRecord();
    peg.io.flatFile ff = null;

    pb.setValue(0,"Uploading New Route Information...");
    if (openSyncDB(this.routeDataBase)) {
        if (clearSyncDB()) {

            //Upload HeaderInfo...
            debugThis.print(2,"uploadRouteData():uploadHeaderInfo("+userName+"):...");
            uploadHeaderInfo(userName);
            //uploadMeterInfo();

            String fileName = new String("");
            if (this.newRouteNumber.length() == 4) {
                fileName = defaultPath+"BLOK"+stringUtils.strLength(this.newRouteNumber,4)+".DAT";
            } else {
                fileName = defaultPath+"BLOK"+this.newRouteNumber;
            }
```

```
            debugThis.print(2,"uploadRouteData():new peg.io.flatFile("+fileName+",["+this.meterInfoStructure.length+"])");
            ff = new peg.io.flatFile(fileName,this.meterInfoStructure);
            //ff.setColumnNames(this.meterInfoHeaders);
            //ff.showTable();

            if (ff.open()) {
                pb.setMin(0);
                pb.setMax(ff.getRowCount()*ff.getColumnCount());
                ff.open();
                String tmpValue = ff.getNextColumn();
                while (tmpValue != null) {
                    try {
                        debugThis.print(9,"uploadRouteData():tmpValue=" +tmpValue);
                        sR.addString(tmpValue);
                        SyncManager.writeRec(db, sR);
                    } catch (SyncException e) {
                        debugThis.print(2,"uploadRouteData():SyncException:" +e.getMessage());
                    } catch (IOException e) {
                        debugThis.print(2,"uploadRouteData():IOException:" +e.getMessage());
                    }
                    tmpValue = ff.getNextColumn();
                    pb.incValue();
                }
            }
        }
        closeSyncDB();
    } else {
        debugThis.print(2,"uploadRouteData():Could not open dataBase on Palm");
    }
    ff.close();
}

/** This method loads the Messages required for this route onto the palm. The file from which the
 * data is read is is prepared by the backoffice software.
 * @param routeNumber the routenumber to be uploaded
 */
private void uploadMessagesData(String routeNumber) {
    StringRecord sR = new StringRecord();
    peg.io.flatFile ff = null;

    String fileName = defaultPath+"MESS"+routeNumber;//+".DAT";
    debugThis.print(2,"uploadMessageData():new peg.io.flatFile("+
                      fileName+","+this.messageInfoStructure.length+")");
    ff = new peg.io.flatFile(fileName,this.messageInfoStructure);
    //ff.setColumnNames(this.messageInfoHeaders);
    //ff.showTable();

    pb.setValue(0,"Uploading Messages...");
    if (openSyncDB(this.messageDataBase)) {
        if (clearSyncDB()) {
            try {
                sR.setString(0,"dummyMessageRec");
                SyncManager.writeRec(db, sR);

                if (ff.open()) {
                    pb.setMin(0);
                    pb.setMax(ff.getRowCount()*ff.getColumnCount());
                    ff.open();
                    String tmpValue = ff.getNextColumn();
                    while (tmpValue != null) {
                        tmpValue = ff.getNextColumn();
                        pb.incValue();
                        debugThis.print(9,"uploadMessageData():tmpValue=" +tmpValue);
                        sR.addString(tmpValue);
                        SyncManager.writeRec(db, sR);
                        tmpValue = ff.getNextColumn();
                        pb.incValue();
                        debugThis.print(9,"uploadMessageData():tmpValue=" +tmpValue);
                        sR.addString(tmpValue);
                        SyncManager.writeRec(db, sR);
                        tmpValue = ff.getNextColumn();
                        pb.incValue();
                    }
                }
            } catch (SyncException e) {
                debugThis.print(2,"uploadMessageData():SyncException:" +e.getMessage());
            } catch (IOException e) {
                debugThis.print(2,"uploadMessageData():IOException:" +e.getMessage());
            }
```

```
            }
            closeSyncDB();
        } else {
            debugThis.print(2,"uploadMessageData():Could not open dataBase on Palm");
        }
        ff.close();
    }

    /** This method is responsible for opening the correct database on the palm.
     * @param dbName the name of the database
     * @return true if sucessful
     */
    private boolean openSyncDB(String dbName) {
        debugThis.print(3, "openSyncDB("+dbName+"):...");
        try {
            db = SyncManager.openDB(dbName, 0,
                    SyncManager.OPEN_READ |
                    SyncManager.OPEN_WRITE |
                    SyncManager.OPEN_EXCLUSIVE);
            return true;
        } catch (SyncException e) {
            debugThis.print(3, "openSyncDB():SyncException:"+e.getMessage());
            debugThis.print(3, "openSyncDB():SyncException:"+
                        " * Use Power Pocket Meter Reader on the palm to generate a database! *");
        }
            return false;
    }

    /** This method clears the database on the palm.
     * @return true if sucessfull
     */
    private boolean clearSyncDB() {
        debugThis.print(3, "clearSyncDB():...");
        try {
            SyncManager.purgeAllRecs(db);
            return true;
        } catch (SyncException e) {
            debugThis.print(3, "clearSyncDB():SyncException:"+e.getMessage());
            debugThis.print(3, "clearSyncDB():SyncException:"+
                    " * Use Power Pocket Meter Reader on the palm to generate a database! *");
            return false;
        }
    }

    /** Closes the database on the palm for access.
     * @return true if sucessfull
     */
    private boolean closeSyncDB() {
        debugThis.print(3, "closeSyncDB():...");
        try {
            SyncManager.closeDB(db);
            return true;
        } catch (SyncException e) {
            debugThis.print(3, "closeSyncDB():SyncException:"+e.getMessage());
            return false;
        }
    }
/*
    DBConnect Stuff:

        String driver = energyServerSetup.getSetupString(energyServerSetup.DBDriver);
        String protocol = energyServerSetup.getSetupString(energyServerSetup.DBProtocol);
        String url = energyServerSetup.getSetupString(energyServerSetup.DBurl);
        String user = energyServerSetup.getSetupString(energyServerSetup.DBuser);
        String password = energyServerSetup.getSetupString(energyServerSetup.DBpassword);


              // if (myDB.isQuerySuccess()) {
              //     javax.swing.JFrame myTempFrame = new javax.swing.JFrame();
              //     myTempFrame.getContentPane().add(new peg.swing.TableDisplay(myDB));
              //     myTempFrame.setBounds(10,10,300,450);
              //     myTempFrame.show();
              // } else {
              //     System.out.println("Error extracting data from database:" + myDB.getErrorString());
              // }

        }*/
}
```

Q.4.2 Object: `green.HHTSync.HHTSyncGUI`

```
/*
 * HHTSyncGUI.java
 *
 * Created on 04 February 2004, 12:19
 */

package green.HHTSync;

import javax.swing.*;
/** This GUI is used to configure the HHTCond Conduit Data. The data is stored in
 * the HHTConduitSetupData object.
 * @author RDoorduin
 */
public class HHTSyncGUI extends javax.swing.JFrame {
    private HHTConduitSetupData data = new HHTConduitSetupData();
    private boolean exitMethod = false; //Closes all systems if false

    /** Creates new form HHTSyncGUI */
    public HHTSyncGUI() {
        initComponents();
        lblDataPath.setText(data.getDataPath());
        btnDBCancelActionPerformed(null);
        exitMethod = false;
        this.setTitle("("+this.exitMethod+")");
    }
    /** Creates new form HHTSyncGUI, but indicates that all system process must not be killed...
     * This is handy when accessing the conduit from the HotSync | Custom... setup
     * menu.
     * @param exitMethod if true, the system does not exit.
     */
    public HHTSyncGUI(boolean exitMethod) {
        this();
        this.exitMethod = exitMethod;
        this.setTitle("("+this.exitMethod+")");
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        pnlDirectorySetup = new javax.swing.JPanel();
        btnDefaultDataPath = new javax.swing.JButton();
        lblDataPath = new javax.swing.JLabel();
        pnlDataBase = new javax.swing.JPanel();
        chkDataBaseActive = new javax.swing.JCheckBox();
        pnlDBDetails = new javax.swing.JPanel();
        lblDriver = new javax.swing.JLabel();
        txtDriver = new javax.swing.JTextField();
        lblProtocol = new javax.swing.JLabel();
        txtProtocol = new javax.swing.JTextField();
        lblURL = new javax.swing.JLabel();
        txtURL = new javax.swing.JTextField();
        lblUser = new javax.swing.JLabel();
        txtUser = new javax.swing.JTextField();
        lblPassword = new javax.swing.JLabel();
        txtPassword = new javax.swing.JPasswordField();
        btnDBCancel = new javax.swing.JButton();
        btnDBOK = new javax.swing.JButton();

        getContentPane().setLayout(new javax.swing.BoxLayout(getContentPane(), javax.swing.BoxLayout.Y_AXIS));

        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });

        pnlDirectorySetup.setLayout(new javax.swing.BoxLayout(pnlDirectorySetup, javax.swing.BoxLayout.X_AXIS));

        btnDefaultDataPath.setText("Default Data Path");
        btnDefaultDataPath.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnDefaultDataPathActionPerformed(evt);
            }
        });
```

```
        pnlDirectorySetup.add(btnDefaultDataPath);

        lblDataPath.setText("This is an initialization String");
        pnlDirectorySetup.add(lblDataPath);

        getContentPane().add(pnlDirectorySetup);

        pnlDataBase.setLayout(new javax.swing.BoxLayout(pnlDataBase, javax.swing.BoxLayout.Y_AXIS));

        pnlDataBase.setBorder(new javax.swing.border.TitledBorder("Database connectivity"));
        chkDataBaseActive.setText("Dump to DataBase");
        pnlDataBase.add(chkDataBaseActive);

        pnlDBDetails.setLayout(new java.awt.GridLayout(0, 2));

        lblDriver.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
        lblDriver.setText("Driver:");
        pnlDBDetails.add(lblDriver);

        txtDriver.setText("com.mysql.jdbc.Driver");
        pnlDBDetails.add(txtDriver);

        lblProtocol.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
        lblProtocol.setText("Protocol:");
        pnlDBDetails.add(lblProtocol);

        txtProtocol.setText("jdbc:mysql");
        pnlDBDetails.add(txtProtocol);

        lblURL.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
        lblURL.setText("URL:");
        pnlDBDetails.add(lblURL);

        txtURL.setText("//localhost:3306/palmdb?user=blah&password=blah");
        pnlDBDetails.add(txtURL);

        lblUser.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
        lblUser.setText("User:");
        pnlDBDetails.add(lblUser);

        pnlDBDetails.add(txtUser);

        lblPassword.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
        lblPassword.setText("Password:");
        pnlDBDetails.add(lblPassword);

        pnlDBDetails.add(txtPassword);

        btnDBCancel.setText("Cancel");
        btnDBCancel.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnDBCancelActionPerformed(evt);
            }
        });

        pnlDBDetails.add(btnDBCancel);

        btnDBOK.setText("OK");
        btnDBOK.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnDBOKActionPerformed(evt);
            }
        });

        pnlDBDetails.add(btnDBOK);

        pnlDataBase.add(pnlDBDetails);

        getContentPane().add(pnlDataBase);

        pack();
    }//GEN-END:initComponents

    private void btnDBCancelActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnDBCancelActionPerformed
        txtDriver.setText(data.getDBDriver());
        txtProtocol.setText(data.getDBProtocol());
        txtURL.setText(data.getDBSubname());
        txtUser.setText(data.getDBUser());
        txtPassword.setText(data.getDBPassword());
```

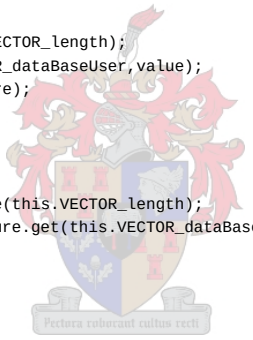
```
        chkDataBaseActive.setSelected(data.getDBActive());
    }//GEN-LAST:event_btnDBCancelActionPerformed

    private void btnDBOKActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnDBOKActionPerformed
        data.setDBDriver(txtDriver.getText());
        data.setDBProtocol(txtProtocol.getText());
        data.setDBSubname(txtURL.getText());
        data.setDBUser(txtUser.getText());
        data.setDBPassword(new String(txtPassword.getPassword()));
        data.setDBActive(chkDataBaseActive.isSelected());
    }//GEN-LAST:event_btnDBOKActionPerformed

    private void btnDefaultDataPathActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnDefaultDataPathActionPerformed
        JFileChooser chooser = new JFileChooser(lblDataPath.getText());
        chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
        int returnVal = chooser.showOpenDialog(new JFrame());
        if(returnVal == JFileChooser.APPROVE_OPTION) {
            String fileName;
            fileName = chooser.getSelectedFile().getPath();
            lblDataPath.setText(fileName);
            data.setDataPath(lblDataPath.getText());
        } else {
        }
    }//GEN-LAST:event_btnDefaultDataPathActionPerformed

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_exitForm
        if (exitMethod) {
        } else {
            System.exit(0);
        }
    }//GEN-LAST:event_exitForm

    /** Envokes the GUI form.
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        new HHTSyncGUI().show();
    }

    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btnDBCancel;
    private javax.swing.JButton btnDBOK;
    private javax.swing.JButton btnDefaultDataPath;
    private javax.swing.JCheckBox chkDataBaseActive;
    private javax.swing.JLabel lblDataPath;
    private javax.swing.JLabel lblDriver;
    private javax.swing.JLabel lblPassword;
    private javax.swing.JLabel lblProtocol;
    private javax.swing.JLabel lblURL;
    private javax.swing.JLabel lblUser;
    private javax.swing.JPanel pnlDBDetails;
    private javax.swing.JPanel pnlDataBase;
    private javax.swing.JPanel pnlDirectorySetup;
    private javax.swing.JTextField txtDriver;
    private javax.swing.JPasswordField txtPassword;
    private javax.swing.JTextField txtProtocol;
    private javax.swing.JTextField txtURL;
    private javax.swing.JTextField txtUser;
    // End of variables declaration//GEN-END:variables

}
```

Q.4.3 Object: `green.HHTSync.HHTConduitSetupData`

```
/*
 * HHTRecordManager.java
 *
 * Created on 04 February 2004, 12:29
 */

package green.HHTSync;

import java.io.*;
import java.beans.*;
import java.util.*;
```

```
/** This class is intended to store and group specific setting together, and provide
 * storage and retrieval facilities. This is currently done with XML, which
 * requires J2SDK v1.4.<BR><BR>
 *
 * History:<BR>
 * v1.0 - Initial class
 * @author RDoorduin
 * @version 1.0
 */
public class HHTConduitSetupData {

    Vector thisDataStructure = new Vector();
    private final String FILENAME = "C:\\HHTDataFile.xml";
    private final String defaultPathData = "C:\\metMeter";

    private final int VECTOR_dataPath = 0,
                      VECTOR_dataBaseDriver = 1,
                      VECTOR_dataBaseProtocol = 2,
                      VECTOR_dataBaseSubname = 3,
                      VECTOR_dataBaseUser = 4,
                      VECTOR_dataBasePassword = 5,
                      VECTOR_dataBaseActive = 6,
                      VECTOR_length = 7;
    /** Creates a new instance of HHTRecordManager and retrieves the data from the
     * storage medium.
     */
    public HHTConduitSetupData() {
        thisDataStructure = (Vector)readXMLdata(FILENAME);
        if (thisDataStructure == null) {
            thisDataStructure = new Vector();
        }
        thisDataStructure.setSize(VECTOR_length);
    }
    /** Retrieves the data Path from the hard disk the object.
     * The dataPath is the pointer to the dataFiles which will be used by the HotSync
     * class.
     * @return the dataPath.
     */
    public String getDataPath() {
        if (thisDataStructure != null) {
            try {
                this.thisDataStructure.setSize(this.VECTOR_length);
                return (String)thisDataStructure.get(0);
            } catch (ArrayIndexOutOfBoundsException e) {
                return defaultPathData;
            }
        } else {
            return defaultPathData;
        }
    }
    /** Stores the data Path.
     * The dataPath is the pointer to the dataFiles which will be used by the HotSync
     * class.
     * @param txt the datapath
     */
    public void setDataPath(String txt) {
        this.thisDataStructure.setSize(this.VECTOR_length);
        this.thisDataStructure.set(this.VECTOR_dataPath,txt);
        writeXMLdata(FILENAME,thisDataStructure);
    }

    public String getDBDriver() {
        if (thisDataStructure != null) {
                this.thisDataStructure.setSize(this.VECTOR_length);
                return (String)thisDataStructure.get(this.VECTOR_dataBaseDriver);
        } else {
            return "";
        }
    }
    public void setDBDriver(String value) {
        this.thisDataStructure.setSize(this.VECTOR_length);
        this.thisDataStructure.set(this.VECTOR_dataBaseDriver,value);
        writeXMLdata(FILENAME,thisDataStructure);
    }

    public String getDBProtocol() {
        if (thisDataStructure != null) {
                this.thisDataStructure.setSize(this.VECTOR_length);
                return (String)thisDataStructure.get(this.VECTOR_dataBaseProtocol);
```

```
    } else {
        return "";
    }
}
public void setDBProtocol(String value) {
    this.thisDataStructure.setSize(this.VECTOR_length);
    this.thisDataStructure.set(this.VECTOR_dataBaseProtocol,value);
    writeXMLdata(FILENAME,thisDataStructure);
}

public String getDBSubname() {
    if (thisDataStructure != null) {
            this.thisDataStructure.setSize(this.VECTOR_length);
            return (String)thisDataStructure.get(this.VECTOR_dataBaseSubname);
    } else {
        return "";
    }
}
public void setDBSubname(String value) {
    this.thisDataStructure.setSize(this.VECTOR_length);
    this.thisDataStructure.set(this.VECTOR_dataBaseSubname,value);
    writeXMLdata(FILENAME,thisDataStructure);
}

public String getDBUser() {
    if (thisDataStructure != null) {
            this.thisDataStructure.setSize(this.VECTOR_length);
            return (String)thisDataStructure.get(this.VECTOR_dataBaseUser);
    } else {
        return "";
    }
}
public void setDBUser(String value) {
    this.thisDataStructure.setSize(this.VECTOR_length);
    this.thisDataStructure.set(this.VECTOR_dataBaseUser,value);
    writeXMLdata(FILENAME,thisDataStructure);
}

public String getDBPassword() {
    if (thisDataStructure != null) {
            this.thisDataStructure.setSize(this.VECTOR_length);
            return (String)thisDataStructure.get(this.VECTOR_dataBasePassword);
    } else {
        return "";
    }
}
public void setDBPassword(String value) {
    this.thisDataStructure.setSize(this.VECTOR_length);
    this.thisDataStructure.set(this.VECTOR_dataBasePassword,value);
    writeXMLdata(FILENAME,thisDataStructure);
}

public void setDBActive(boolean value) {
    this.thisDataStructure.setSize(this.VECTOR_length);
    if (value) {
        this.thisDataStructure.set(this.VECTOR_dataBaseActive,"TRUE");
    } else {
        this.thisDataStructure.set(this.VECTOR_dataBaseActive,"FALSE");
    }
    writeXMLdata(FILENAME,thisDataStructure);
}

public boolean getDBActive() {
    String tmpStr = "";
    if (thisDataStructure != null) {
        this.thisDataStructure.setSize(this.VECTOR_length);
        tmpStr = (String)thisDataStructure.get(this.VECTOR_dataBaseActive);
        if (tmpStr != null) {
            if (tmpStr.equalsIgnoreCase("TRUE")) {
                return true;
            } else {
                return false;
            }
        } else {
            return false;
        }
    } else {
        return false;
    }
```

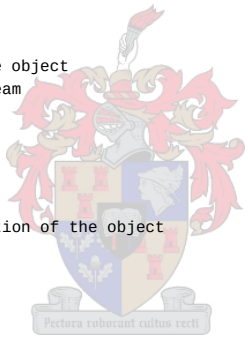


```
    }



    /** Stores the object data into XML format at the specified location.
     * @param writeFile the filename or location
     * @param dataObject the object to be stored
     */
    private void writeXMLdata(String writeFile, Object dataObject) {
        try {
            XMLEncoder xmlE = new XMLEncoder(new BufferedOutputStream(new FileOutputStream(writeFile)));
            xmlE.writeObject(dataObject);
            xmlE.close();
            System.out.println("  XML:Date successfully stored!");
        } catch (Exception e) {
            System.out.println("  XML:Error writing XMLfile: "+ e.getMessage());
        }
    }

    /** Reads the object data from a XML format file.
     * @param readFile the filename or location
     * @return the stored object
     */
    private Object readXMLdata(String readFile) {
        try {
            XMLDecoder xmlD = new XMLDecoder(new BufferedInputStream(new FileInputStream(readFile)));
            Object tmpObject = xmlD.readObject();
            xmlD.close();
            System.out.println("  XML:Successfull loaded object!");
            return tmpObject;
        } catch (Exception e) {
            System.out.println("  XML:Error loading XMLfile: "+ e.getMessage());
            return null;
        }
    }
}
```

Q.4.4 Object: `green.HHTSync.StringRecord`

```
package green.HHTSync;
import palm.conduit.*;
import java.io.*;

/* Copyright (c) 1997-2001 Palm Inc. or its subsidiaries.  All rights reserved. */

/* This class is implemented in HHTSync.jar.  The source code is modified from
 * MemoRecord provided by palmOS developement. */

/** Extends GenericRecord to implement the String Record.  This object can be
 *      used to store and translate hand-held device records used by the Strings
 *      stored in records application.
 *
 * This class was provided by PalmSource and used from an example provided with the
 * JSync toolkit.
 * @version 2.0
 *
 * Change history:
 *  v1.0 created by Palm Inc.
 *  v2.0 updated by Riaan Doorduin as indicated in code.
 */

public class StringRecord extends AbstractRecord {

    private int recID = 0; //Indicated the current record ID within the StringRecord class
    String thisString;

    /** Converts and copies StringRecord members into a DataOutputStream in preparation
     *  for writing to the hand-held device.
     *  The format of the DataOutputStream specific to the record body byte
     *  layout of the string records stored on the hand-held device for the string Pad
     *  application.
     *  @param out The DataOutputStream to fill with string record body information
     */

    /** Returns the string text.
     * @return The string contained in the data structure.
     */
```

```
    public String getString() {
        String tmpStr = thisString.substring(0,thisString.length()-1);
        return tmpStr;
    }

    /** Sets the string text.
     * @param thisString the string to be set
     */
    public void setString(String thisString) {
        this.thisString = thisString;
    }


    /** Writes the data from into the byte stream for uploading to the palm.
     * @param out the output stream
     * @throws IOException if some error occurs with the streaming.
     */
    public void writeData(DataOutputStream out) throws IOException{
        // Memo
        if (thisString != null) {
            out.write(thisString.getBytes());
        }
    }

    /** Converts a DataInputStream representing the body of the hand-held device's
     *  Memo Pad application record into MemoRecord object members.
     * @param in The DataInputStream containing record body information from
     *  the hand-held device's string application record.
     * @throws IOException if an error occurs with the streaming
     */
    public void readData(DataInputStream in) throws IOException{
        this.thisString = readCString(in);
    }

    /** Returns a string representation of the object
     * @return the string form of the datastream
     */
    public String toString() {
        return getString();
    }

    /** Returns a formatted string representation of the object
     * @return the formatted string
     */
    public String toFormattedString() {
        return ("String record: {" +
                "  string: " + getString() +
                "}" +
                super.toFormattedString());
    }

    //Additions made from here on by RDoorduin...
    //

    /** Sets the string for a specific record.
     * @param recID the record ID
     * @param strToAdd the string to places at the specific record location
     */
    public void setString(int recID, String strToAdd) {
        this.setId(recID);
        this.setString(strToAdd);
    }

    /** Adds a string to the record structure
     * @param strToAdd the string to add
     */
    public void addString(String strToAdd) {
        this.setString(strToAdd);
        this.setId(this.getId()+1);
    }
}
```



Q.5 Package: `peg.io`

Q.5.1 Object: `peg.io.byteBuffer`

```
/*
 * byteBuffer.java
 *
 * Created on 28 May 2004, 11:57
 */

package peg.io;

/**
 *
 * @author  RDoorduin
 */
public class byteBuffer {

    private byte[] bufferByte = new byte[0];

    /** Creates a new instance of byteBuffer */
    public byteBuffer() {
        bufferByte = new byte[0];
    }
    public byteBuffer(byte [] b) {
        bufferByte = b;
    }

    public void addByteArray(byte[] b,int length) {
        byte[] tmpByte = new byte[bufferByte.length+length];

        for (int i = 0; i < bufferByte.length; i++) {
            tmpByte[i] = bufferByte[i];
        }
        for (int i = 0; i < length; i++)  {
            tmpByte[i+bufferByte.length] = b[i];
        }
        bufferByte = tmpByte;
    }

    public byte[] substring(int startsWith) {
        byte[] tmpByte = new byte[bufferByte.length - startsWith];
        for (int i = 0; i < (bufferByte.length-startsWith); i++) {
            tmpByte[i] = bufferByte[i+startsWith];
        }
        return tmpByte;
    }
    public byte[] substring(int startsWith, int endsWith) {
        byte[] tmpByte = new byte[endsWith - startsWith];
        if (endsWith > bufferByte.length) {
            endsWith = bufferByte.length;
        }
        if (endsWith > startsWith) {
            for (int i = 0; i < (endsWith-startsWith); i++) {
                tmpByte[i] = bufferByte[i+startsWith];
            }
            return tmpByte;
        } else {
            return null;
        }
    }

    public int indexOf(String str) {
        return indexOf(str.getBytes());
    }
    public int indexOf(byte[] b) {
        int j = -1;
        int i = 0;
        int k = 0;
        boolean correct = false;

        if ((b != null) & bufferByte != null) {
            if (bufferByte.length > b.length) {
                while ((i < ((bufferByte.length-b.length)+1)) & (j == -1)) {
                    correct = true;
                    for (k = 0; k < b.length; k++) {
                        if (bufferByte[i+k] != b[k]) {
```

```
                        correct = false;
                    }
                }
                if (correct) {
                    j = i;
                }
                i++;
            }
        }
    }
    return j;
  }
}
```

Q.5.2 Object: `peg.io.flatFile`

```
/*
 * flatFile.java
 *
 * Created on 14 June 2004, 08:13
 */

package peg.io;
import javax.swing.table.AbstractTableModel;
import java.io.*;
/** The flatFile object is designed for easy serial extraction and addition to a
 * flatFile with fixed field withs. By changing the fields widths in the
 * datastructures the implementation can easily be changed.<BR>
 *
 * a flatfile is then a database or table where entries are added and collecting in
 * rows and columns.<BR><BR>
 *
 * History:<BR>
 * v1.0 - Create flatFile with only sequencial appending function.<BR>
 * <BR>
 * To Do:<BR>
 * <UL>
 *    <LI>Improvement of indexing.
 *    <LI>Replace the file access with RandomFileAccess to allow for simultanious write
 * and read operations and to improve the indexing performance.
 * </UL>
 * @author RDoorduin
 * @version 1.0
 */
public class flatFile extends AbstractTableModel {

    /** Indicates that the flatFile format is fixedwidth. */
    public static final int TYPE_FIXEDWIDTH = 0,
                            TYPE_CSV = 1;

    /** The maximum number of types declared. */
    public static final int
                            TYPE_MAXTYPE = TYPE_CSV;

    private String fileName = null;
    private int type = TYPE_FIXEDWIDTH;
    private int[] dataWidths = {};
    private String[] columnNames = {};

    private int columnCount = 0;
    private int rowCount = 0;

    private String lastString = null;
    private int lastColumn = 0;
    private int lastRow = 0;

    private peg.utils.debug debugThis = new peg.utils.debug("flatFile",3);
    private BufferedReader in = null;
    private BufferedWriter out = null;

    /** Creates a new instance of flatFile */
    public flatFile() {
    }
    /** Creates a new instance of flatFile with the filename to access included in the
     * constructor
     * @param fileName the filename of the flat file.
     */
```

```
public flatFile(String fileName) {
    setFileName(fileName);
}
/** Creates a new instance of flatFile with the filename and database type.
 * @param fileName the filename
 * @param type the database type
 */
public flatFile(String fileName, int type) {
    setFileName(fileName);
    setType(type);
}
/** Creates a new instance of flatFile with the filename and the dataWidhts for
 * fixed width flatFile operation.
 * @param fileName the filename
 * @param dataWidths the array with datawidths of each column
 */
public flatFile(String fileName, int[] dataWidths) {
    setFileName(fileName);
    setType(TYPE_FIXEDWIDTH);
    setDataWidths(dataWidths);
}

/** Sets the filename being used by the flatfile
 * @param fileName the filename
 */
public void setFileName(String fileName) {
    this.fileName = fileName;
}
/** Retrieves the filename currently in use.
 * @return the filename
 */
public String getFileName() {
    return this.fileName;
}

/** Sets the type of database currently being used. This is limited for now only to
 * a fixed width.
 * @param type the type of database
 */
public void setType(int type) {
    if (type < TYPE_FIXEDWIDTH) {
        this.type = TYPE_FIXEDWIDTH;
    } else if (type > TYPE_MAXTYPE) {
        this.type = TYPE_MAXTYPE;
    } else {
        this.type = type;
    }
}
/** Retrieves the current database type in use.
 * @return the database type
 */
public int getType() {
    return this.type;
}

/** Sets the datawidhts of all the different columns. A zero represents any widht.
 * This will work best if implemented for the last column. Unknown behaviour can be
 * expted when a zero is used in the middle of the datastructure.<BR>
 * From this array the total length and amount of columns is easily calculated.
 * @param dataWidths the widths of each column
 */
public void setDataWidths(int[] dataWidths) {
    this.dataWidths = dataWidths;
}
/** Returns the current implemented widths of each column.
 * @return the column widhts
 */
public int[] getDataWidths() {
    return this.dataWidths;
}

/** Returns the amount of columns implemented
 * @return the amount of columns
 */
public int getColumnCount() {
    if (columnCount == 0) {
        if ((type == TYPE_FIXEDWIDTH) & (dataWidths.length > 0)) {
            columnCount = dataWidths.length;
        } else {
```

```
            update();
        }
    }
    return columnCount;
}

/** Returns the number of rows contained in the flatfile.
 * @return number of rows
 */
public int getRowCount() {
    if (rowCount == 0) {
        update();
    }
    return rowCount;
}

/** Returns the value at a specific location.<BR>
 * The reason for this is to be complient with the AbstractTable interface.<BR>
 * This routine is not very efficient and furhter work can be done here.
 * @param rowIndex the row required
 * @param columnIndex the column required
 * @return the value at the specific location. If the database type is fixed columns then
 * the string will be the total length of the column and the user is required to
 * correct any trailing spaces.
 */
public Object getValueAt(int rowIndex, int columnIndex) {
    debugThis.print(6,"getValueAt("+rowIndex+","+columnIndex+"):...");
    return ""+getColumnString(columnIndex,getRowString(rowIndex));
}
/** From the presented row and column index the method determines the correct
 * value for the column.
 * @param columnIndex the column index
 * @param rowString the string representing the total row contents.
 * @return the column value
 */
private String getColumnString(int columnIndex, String rowString) {
    int firstChar = 0;
    if (rowString != null) {
        String tmpStr = new String("");
        if (columnIndex < dataWidths.length) {
            for (int i = 0; i < columnIndex; i++) {
                firstChar = firstChar + dataWidths[i];
            }
            try {
                if (dataWidths[columnIndex] == 0) {
                    tmpStr = rowString.substring(firstChar);
                } else {
                    tmpStr = rowString.substring(firstChar,firstChar+dataWidths[columnIndex]);
                }
            } catch (IndexOutOfBoundsException e) {
                tmpStr = " ";
            }
            return tmpStr;
        } else {
            return null;
        }
    } else {
        return null;
    }
}

/** Retrieves the content of a specified row
 * @param rowIndex the row index
 * @return the row contents
 */
private String getRowString(int rowIndex) {
    if (rowIndex < this.getRowCount()) {
        String tmpLine = null;
        debugThis.print(6,"getRowString("+rowIndex+"):...");
        if (this.open()) {;
            int i = 0;
            tmpLine = this.getNextRow();
            while (i < rowIndex) {
                tmpLine = this.getNextRow();
                i++;
            }
            debugThis.print(7,"getRowString["+i+"]:tmpLine="+tmpLine);
            this.close();
        }
```

```
            return tmpLine;
        } else {
            debugThis.print(7,"getRowString():return null");
            return null;
        }
    }
    /* Updates the row and column counts of the database to reflect any updates
     * or possible changes.
     */
    private void update() {
        columnCount = 0;
        rowCount = 0;

        debugThis.print(4, "update():...");
        if (this.open()) {
            int i = 0;
            while (this.getNextRow() != null) {
                i++;
                debugThis.print(4,"Row["+i+"]");
            }
            rowCount = i;
            if ((type == TYPE_FIXEDWIDTH) & (dataWidths.length > 0)) {
                columnCount = dataWidths.length;
            }
            this.close();
        }
        debugThis.print(4, "update():rowCount="+rowCount);
        debugThis.print(4, "update():columnCount="+columnCount);
    }

    /** Set the column names of the flatFile. This is done to enable nice table displays
     * when using the AbstractTable interface.
     * @param columnNames The column Names in an array.
     */
    public void setColumnNames(String[] columnNames) {
        this.columnNames = columnNames;

    }

    /** Return the specific columns name. Required for compatibility with the
     * AbstractTable inteface.
     * @param column the column
     * @return the column name
     */
    public String getColumnName(int column) {
        if (this.columnNames.length > column) {
            return this.columnNames[column];
        }
        return "[#"+column+"]";
    }

    /** This method opens the database file. The database can be opened for write
     * access. <B>If opened for write access, this will however overwrite any existing data.</B>
     * @param output if the datastream should be opened for writing to the file.
     * @return true if successfull
     */
    public boolean open(boolean outputEnabled) {
        if (outputEnabled) {
            in = null;
            try {
                debugThis.print(7, "open(true):BufferedReader("+fileName+")");
                out = new BufferedWriter(new FileWriter(fileName));
                this.lastColumn = 0;
                this.lastRow = 0;
                debugThis.print(7, "open(true):return true");
                return true;
            } catch (IOException e) {
                debugThis.print(3, "open(true):IOException:"+e.getMessage());
                out = null;
                debugThis.print(7, "open(true):return false");
                return false;
            }
        } else {
            out = null;
            try {
                debugThis.print(7, "open(false):BufferedReader("+fileName+")");
                in = new BufferedReader(new FileReader(fileName));
                this.lastColumn = 0;
                this.lastRow = 0;
```

```
            debugThis.print(7, "open(false):return true");
            return true;
        } catch (IOException e) {
            debugThis.print(3, "open(false):IOException:"+e.getMessage());
            in = null;
            debugThis.print(7, "open(false):return false");
            return false;
        }
    }
}
/** Opens the database in read mode.
 * @return true if successfull
 */
public boolean open() {
    return open(false);
}

/** Opens the database. The specific filename is specified in this method.
 * @param fileName the filename
 * @return true if successfull
 */
public boolean open(String fileName) {
    this.setFileName(fileName);
    return open();
}

/** Opens the database with a specific file, and can be opened for write access.
 * <B>If opened for write access, this will however overwrite any existing data.</B>
 * @param fileName the filename to open
 * @param output true, if the database should be opened for writing data to.
 * @return true if successfull.
 */
public boolean open(String fileName, boolean output) {
    this.setFileName(fileName);
    return open(output);
}

/** Opens the database for appending data.
 * @return true if successfull
 */
public boolean append() {
    in = null;
    try {
        debugThis.print(7, "append():BufferedReader("+fileName+")");
        out = new BufferedWriter(new FileWriter(fileName,true));
        this.lastColumn = 0;
        this.lastRow = 0;
        debugThis.print(7, "append():return true");
        return true;
    } catch (IOException e) {
        debugThis.print(3, "append():IOException:"+e.getMessage());
        out = null;
        debugThis.print(7, "append():return false");
        return false;
    }
}

/** Returns the contents of the next row.
 * @return the contents of the row.
 */
public String getNextRow() {
    debugThis.print(8, "getNextRow():...");
    String tmpLine = null;
    if (in != null) {
        try {
            if (in.ready()) {
                tmpLine = in.readLine();
                this.lastColumn = 0;
                this.lastRow++;
            } else {
                debugThis.print(8, "getNextRow():Null String");
            }
        } catch (IOException e) {
            debugThis.print(3, "getNextRow():IOException:"+e.getMessage());
        }
    } else {
        debugThis.print(3,"getNextRow():in == null!");
        tmpLine = null;
    }
```

```
     this.lastString = tmpLine;
     return tmpLine;
}

/** Returns a specific column contents. If the end of the column is reached, the
 * next row is automaticaly read.
 * @return the contents of the column at the specific row.
 */
public String getNextColumn() {
     debugThis.print(9, "getNextColumn():...");
     if (lastString == null) {
         this.getNextRow();
     }
     String tmpStr = this.getColumnString(this.lastColumn, this.lastString);
     if (tmpStr == null) {
         this.getNextRow();
         tmpStr = this.getColumnString(this.lastColumn, this.lastString);
     }
     this.lastColumn++;
     return tmpStr;
}

/** Close any open database streams */
public void close() {
     debugThis.print(7, "close():...");
     if (in != null) {
         try {
             in.close();
             debugThis.print(3, "close():closed!");
         } catch (IOException e) {
             debugThis.print(3, "close():in:IOException:"+e.getMessage());
         }
     }
     if (out != null) {
         try {
             out.flush();
             out.close();
             debugThis.print(3, "close():closed!");
         } catch (IOException e) {
             debugThis.print(3, "close():out:IOException:"+e.getMessage());
         }
     }
}

/** Appends a column to the current stream. If the end of a column is reached, the
 * row is advanced to the next row automatically.
 * @param value the value to append
 */
public void appendNextColumn(String value) {
     if (out != null) {
         try {
             if (this.getType() == this.TYPE_FIXEDWIDTH) {
                 if ((this.lastColumn) >= this.getColumnCount()) {
                     this.appendNextRow();
                 }
                 if (this.dataWidths[this.lastColumn] == 0) {
                     out.write(value);
                 } else {
                     out.write(peg.utils.stringUtils.strLength(value,this.dataWidths[this.lastColumn]));
                 }
                 debugThis.print(9,"appendNextColumn():lastColumn="+this.lastColumn);
             } else if (this.getType() == this.TYPE_CSV) {
                 if (this.lastColumn != 0) {
                     out.write(",");
                 }
                 out.write(value);
             }
             this.lastColumn++;
         } catch (IOException e) {
             debugThis.print(5,"appendNextColumn():IOException:" +e.getMessage());
         }
     } else {
        debugThis.print(4, "appendNextColumn(...):Stream not ready for output!");
     }
}
/** Appends the next row to the current stream. */
public void appendNextRow() {
     try {
         out.newLine();
```

```
            this.lastColumn = 0;
        } catch (IOException e) {
            debugThis.print(4,"appendNextRow():IOException:" +e.getMessage());
        }
    }
    /** Append a new row to the current stream. It also appends the next column
     * according the database type to the row.
     * @param value the value to insert into the first column.
     */
    public void appendNextRow(String value) {
         appendNextColumn(value);
         appendNextRow();
    }

    /** Allows the table to be displayd. Handy for debugging purposes. */
    public void showTable() {
         javax.swing.JFrame myTempFrame = new javax.swing.JFrame();
         myTempFrame.getContentPane().add(new javax.swing.JScrollPane(new javax.swing.JTable(this)));
         myTempFrame.setBounds(10,10,300,450);
         myTempFrame.show();
    }
}
```

Q.5.3 Object: `peg.io.gsmATinterface`

```
/*
 * gsmATinterface.java
 *
 * Created on 25 May 2004, 01:55
 */

package peg.io;

import java.io.*;
import java.util.*;

/**
 *
 * @author  RDoorduin
 */
public class gsmATinterface {
    private static final int BUFFER_SIZE = 512;

    private byteBuffer bB = new byteBuffer();
    private String stringBuffer = new String();

    private OutputStream out = null;
    private Vector resultVector = new Vector();
    private boolean resultReady = false;
    private boolean inWaitState = false;

    public Vector decodedResults = new Vector();
    public Vector decodedData = new Vector();
    public String currentSMS = "";

    public int ATState = AT_idle;
    public static final int AT_idle = 0x00,
                        AT_header_decoded = 0x01, //Header Vector Ready
                        AT_CMTI = 0x02,    //Incomming SMS
                        AT_CMGL = 0x03,    //List SMS
                        AT_CMGR = 0x04,    //Read SMS
                        AT_delete_all_SMS = 0x05,
                        AT_wait_for_next_SMS_listing = 0x06,
                        AT_delete_all_SMSes_1 = 0x07,
                        AT_delete_all_SMSes_2 = 0x08,
                        AT_delete_all_SMSes_3 = 0x09,
                        AT_delete_all_SMSes_4 = 0x0A,
                        AT_delete_all_SMSes_5 = 0x0B,
                        AT_delete_all_SMSes_6 = 0x0C,
                        AT_delete_all_SMSes_7 = 0x0D,
                        AT_delete_all_SMSes_8 = 0x0E,
                        AT_delete_all_SMSes_9 = 0x0F,
                        AT_delete_all_SMSes_10 = 0x010,
                        AT_delete_all_SMSes_11 = 0x011,
                        AT_delete_all_SMSes_12 = 0x012,
                        AT_delete_all_SMSes_13 = 0x013,
                        AT_delete_all_SMSes_14 = 0x014,
```

```
                    AT_no_state = 0x0FF;

public static final int EV_GSMnewResultAvailable = 0x00; //New GSM Result Available

/** Creates a new instance of gsmATinterface */
public gsmATinterface() {
}

public gsmATinterface(OutputStream thisOut) {
    out = thisOut;
}

private void addResult(byte[] b) {
    if (b.length != 0) {
        System.out.println("["+b.length+"]"+ new String(b));
        resultVector.add(b);
        resultReady = true;
    }
    System.out.flush();
}

private void addResult(String str) {
    if (str.length() != 0) {
        //System.out.println("["+str.length()+"]"+ str);
        resultVector.add(str);
        resultReady = true;
    }
    System.out.flush();
}

public String getNextResult() {
    if (resultVector.size() != 0) {
        try {
            String tmpStr = (String)resultVector.remove(0);
            if (resultVector.size() == 0) {
                resultReady = false;
            }
            return tmpStr;
        } catch (NoSuchElementException e) {
            resultReady = false;
            return "";
        }
    } else {
        resultReady = false;
        return "";
    }
}
public byte[] getNextByteResult() {
    if (resultVector.size() != 0) {
        try {
            byte[] b = (byte[])resultVector.remove(0);
            if (resultVector.size() == 0) {
                resultReady = false;
            }
            return b;
        } catch (NoSuchElementException e) {
            resultReady = false;
            return null;
        }
    } else {
        resultReady = false;
        return null;
    }
}

public boolean isResultReady() {
    return (resultVector.size() != 0);
}

public void clearDecodeBuffer() {
    stringBuffer = "";
}

public void decodeByte(InputStream in) {
        int c;
        int i;  //index of string
        String thisString = new String("");
        String tmpStr = "";
        byte[] tmpByte = null;
```

```
        byte[] b = new byte[BUFFER_SIZE];
        try {
            c = BUFFER_SIZE;
            //Read serial data into buffer
            while (c == BUFFER_SIZE) {
                    //System.out.print('.');
                    //System.out.flush();
                c = in.read(b);
                bB.addByteArray(b,c);
            }
            while (bB.indexOf("\r\n") == 0) {
                bB = new byteBuffer(bB.substring(2));
            }
            i = bB.indexOf("\r\n");
            while (i != -1) {
      while (bB.indexOf("\r\n") == 0) {
                 bB = new byteBuffer(bB.substring(2));
                }
                i = bB.indexOf("\r\n");
                if (i != -1) {
                    tmpByte = bB.substring(0,i);
                    addResult(tmpByte);
                    bB = new byteBuffer(bB.substring(i));
                }
                i = bB.indexOf("\r\n");
            }
        } catch (IOException e) {
            System.out.println("Error reading input stream:"+e.getMessage());
        }
}

public void decode(InputStream in) {
        int c;
        int i;
        String thisString = new String("");
        String tmpStr = "";
        byte[] b = new byte[BUFFER_SIZE];

        try {
            c = BUFFER_SIZE;
            //Read serial data into buffer
            while (c == BUFFER_SIZE) {
                    //System.out.print('.');
                    //System.out.flush();
                c = in.read(b);
                tmpStr = new String(b,0,c);
                tmpStr = tmpStr.substring(0,c);
                stringBuffer = stringBuffer + tmpStr;
            }
            while (stringBuffer.indexOf("\r\n") == 0) {
                stringBuffer = stringBuffer.substring(2);
            }
            i = stringBuffer.indexOf("\r\n");
            while (i != -1) {
      while (i == 0) {
                 stringBuffer = stringBuffer.substring(2);
                    i = stringBuffer.indexOf("\r\n");
                }
                i = stringBuffer.indexOf("\r\n");
                if (i != -1) {
                    tmpStr = stringBuffer.substring(0,i);
                    addResult(tmpStr);
                    stringBuffer = stringBuffer.substring(i);
                }
                i = stringBuffer.indexOf("\r\n");
            }
        } catch (IOException e) {
            System.out.println("Error reading input stream:"+e.getMessage());
        }
}

public void command(String str) {
    try {
        out.write(new String(str).getBytes());
        out.write(new String("\n\r").getBytes());
        System.out.println("["+str+"]");
        System.out.flush();
    } catch (IOException e) {
        System.out.println("Error with GSM AT Command Transmit: " + e.getMessage());
```

```
    }
}

public int commandWait(String str, String result, int tryOuts, long msTimeOut, String errorStr) {
    String tmpResult = "";
    long startTime = System.currentTimeMillis();
    long msT = 0;
    boolean thisResultReady = false;
    boolean thisTimeOut = false;
    boolean thisResultError = false;
    int returnResult = -1;
    int trys = 0;

    inWaitState = true;
    trys = 0;
    while ((returnResult != 0) & (trys < tryOuts)) {
        startTime = System.currentTimeMillis();
        command(str);
        while (!thisResultReady & !thisTimeOut & !thisResultError) {
            msT = System.currentTimeMillis() - startTime;
            thisTimeOut = (msT > msTimeOut);
            if (resultReady) {
                tmpResult = this.getNextResult();
                thisResultReady = tmpResult.equalsIgnoreCase(result);
                thisResultError = tmpResult.equalsIgnoreCase(errorStr);
            }
        }
        if (thisResultReady) {
            returnResult = 0;
        } else if (thisTimeOut) {
            returnResult = (int)msT;
        } else {
            returnResult = -1;
        }
        trys++;
    }

    inWaitState = false;
    return returnResult;
}

public int commandWait(String str, String result, int tryOuts, long msTimeOut) {
    return commandWait(str,result,tryOuts,msTimeOut,"ERROR");
}
public int commandWait(String str, String result, int tryOuts) {
    return commandWait(str,result,tryOuts,2000,"ERROR");
}
public int commandWait(String str, String result) {
    return commandWait(str,result,5,2000,"ERROR");
}
public int commandWait(String str, int tryOuts) {
    return commandWait(str,"OK",tryOuts,2000,"ERROR");
}
public int commandWait(String str) {
    return commandWait(str,"OK",5,2000,"ERROR");
}

public int sendSMS(String number, String message) {
    commandWait("AT+CMGS="+number,"> ");
    command(message+((char)0x01A));
    return 0;
}
public boolean isInWaitState() {
    return inWaitState;
}

public static byte[] hexStringToByte(String incomming) {
    byte[] tmpByte = new byte[incomming.length()];
    int j = 0;
    for (int i = 0; i < incomming.length();i++) {
        try {
            tmpByte[j] = (byte)Integer.parseInt(""+incomming.charAt(i)+incomming.charAt(++i),0x010);
            j++;
        } catch (NumberFormatException e){
            System.out.println("Error while parsing HexDigit: " + e.getMessage());
        }
    }
    byte[] tmpByteOut = new byte[j];
    for (int i = 0; i< j; i++) {
```

```
        tmpByteOut[i] = tmpByte[i];
    }
    return tmpByteOut;
}


public void SMSdecoder(String incomming) {
    Vector thisVector = new Vector();

    int i = incomming.indexOf(":");
    if (i != -1) {
        thisVector.add(incomming.substring(0,i));
        incomming = incomming.substring(i+2);
        i = incomming.indexOf(",");
        while (i != -1) {
            thisVector.add(incomming.substring(0,i));
            incomming = incomming.substring(i+1);
            i = incomming.indexOf(",");
        }
    }
    thisVector.add(incomming);

    decodedResults = thisVector;
}

public String getATResultCode() {
    if (decodedResults != null) {
        if (decodedResults.size() > 0) {
            return (String)decodedResults.get(0);
        } else {
            return "";
        }
    } else {
        return "";
    }
}

public int getATResultParametersCount() {
    if (decodedResults != null) {
        if (decodedResults.size() > 0) {
            return (decodedResults.size() - 1);
        } else {
            return 0;
        }
    } else {
        return 0;
    }
}

public String getATResultParameter(int index) {
    if (decodedResults != null) {
        if (decodedResults.size() > 0) {
            return (String)decodedResults.get(1+index);
        } else {
            return "";
        }
    } else {
        return "";
    }
}

public void printATResultParameters() {
    System.out.println("{"+this.getATResultCode()+"}");
    for (int i = 0; i < this.getATResultParametersCount(); i++) {
        System.out.println("[" + i + "] " + this.getATResultParameter(i));
    }
}


public void clearData() {
    decodedData = new Vector();
}
public boolean addData(String incomming) {
    try {
        java.math.BigInteger i = new java.math.BigInteger(incomming,16);
        decodedData.add(i);
        return true;
    } catch (NumberFormatException e) {
        System.out.println("Error Decoding HexNumbers: " + e.getMessage());
```

```
            decodedData.add(new Integer(0));
            return false;
        }
    }

    public int getATDataCount() {
        if (decodedData != null) {
            return decodedData.size();
        } else {
            return -1;
        }
    }

    public int getATDataAt(int i) {
        if (decodedData != null) {
            try {
                return ((java.math.BigInteger)decodedData.get(i)).intValue();
            } catch (ArrayIndexOutOfBoundsException e) {
                return 0;
            }
        } else {
            return 0;
        }
    }

    public void HexDataDecoder(String incomming) {
        clearData();
        if (incomming.length() == 76) {
            if ((incomming.charAt(25) == ':') & (incomming.charAt(50) == ':') & (incomming.charAt(75) == ':')) {
               //System.out.println(
                addData(incomming.substring(1,9));
                addData(incomming.substring(9,17));
                addData(incomming.substring(17,25));

                addData(incomming.substring(26,34));
                addData(incomming.substring(34,42));
                addData(incomming.substring(42,50));

                addData(incomming.substring(51,59));
                addData(incomming.substring(59,67));
                addData(incomming.substring(67,75));
            }
        } else {
            System.out.println("Length incorrect: [" + incomming.length() +"]");
        }
    };
    public void printHexData() {
            for (int i = 0; i < this.getATDataCount(); i++) {
                System.out.println("[" + i + "]" + this.getATDataAt(i));
            }
    }

    public void getNextState() {
        if (getATResultCode().equalsIgnoreCase("+CMGL")) {
            ATState = AT_CMGL;
        } else if (getATResultCode().equalsIgnoreCase("+CMGR")) {
            ATState = AT_CMGR;
        } else if (getATResultCode().equalsIgnoreCase("+CMTI")) {
            ATState = AT_CMTI;
            currentSMS = this.getATResultParameter(1);
        }
    }

}
```

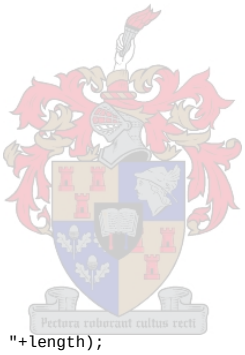
Q.5.4 Object: `peg.io.protocol`

```
/*
 * protocol.java
 *
 * Created on 01 April 2004, 04:00
 */

package peg.io;

import java.lang.*;
import java.io.*;
```

```
/**
 *
 * @author  RDoorduin
 */
public class protocol {
    private int length = 0;
    private boolean active = false;
    private int possition = 0;
    private int CRC = 0;
    private int component = 0;
    private int register = 0;
    private int data[] = new int[0]; //data
    private int i = 0; //data index
    private int latestValue = 0;
    private int totalPackets = 0;


    public final int MAX_LENGTH = 0x0F;
    public final int ACTIVE = 0x0AA;
    public final int CRC_CHAR = 0x00;
    public final int HEADER_LENGTH = 0x04;
    public final int COMP_MCU = 0x00;
    public final int COMP_AD7758 = 0x01;
    //COMP_MCU Registers and values
    public static final byte MCU_ACKNOWLEDGE = 0x000,
                             MCU_REGISTERDUMP = 0x001,
                                VALUE_MCU_MAINSCREEN = 0x001,
                                VALUE_MCU_MONITOR = 0x002,
                             MCU_CALIBRATION = 0x002,
                                VALUE_MCU_V1_A = 0x001,
                                VALUE_MCU_V2_A = 0x002,
                                VALUE_MCU_I1_A = 0x003,
                                VALUE_MCU_I2_A = 0x004,
                                VALUE_MCU_V1_B = 0x005,
                                VALUE_MCU_V2_B = 0x006,
                                VALUE_MCU_I1_B = 0x007,
                                VALUE_MCU_I2_B = 0x008,
                                VALUE_MCU_V1_C = 0x009,
                                VALUE_MCU_V2_C = 0x00A,
                                VALUE_MCU_I1_C = 0x00B,
                                VALUE_MCU_I2_C = 0x00C,
                                VALUE_MCU_GAIN_A = 0x00D,
                                VALUE_MCU_GAIN_B = 0x00E,
                                VALUE_MCU_GAIN_C = 0x00F,
                                VALUE_MCU_FREQ = 0x010,
                                VALUE_MCU_FLASH = 0x011, //force dump to FLASH mem
                                VALUE_MCU_PHASE1A = 0x012,
                                VALUE_MCU_PHASE1B = 0x013,
                                VALUE_MCU_PHASE1C = 0x014,
                                VALUE_MCU_PHASE2A = 0x015,
                                VALUE_MCU_PHASE2B = 0x016,
                                VALUE_MCU_PHASE2C = 0x017,
                             MCU_CURRENTCAL = 0x03,  //use the same values as MCU_CALLIBRATION
                             MCU_MEASURE_ENERGY = 0x04,
                                VALUE_MCU_STOP = 0x00,
                                VALUE_MCU_START = 0x01,
                             MCU_MEASURE_ENERGY_W = 0x05,
                             MCU_MEASURE_ENERGY_VA = 0x06,
                             MCU_MEASURE_ENERGY_VAR = 0x07,
                             MCU_MESSAGE = 0x08,
                                VALUE_MCU_OVERFLOW = 0x01,
                             MCU_CALLIBRATION_WDIVA = 0x09,
                             MCU_CALLIBRATION_WDIVB = 0x0A,
                             MCU_CALLIBRATION_WDIVC = 0x0B,
                             MCU_CALLIBRATION_VADIVA = 0x0C,
                             MCU_CALLIBRATION_VADIVB = 0x0D,
                             MCU_CALLIBRATION_VADIVC = 0x0E,
                             MCU_CALLIBRATION_VARDIVA = 0x0F,
                             MCU_CALLIBRATION_VARDIVB = 0x010,
                             MCU_CALLIBRATION_VARDIVC = 0x011,
                             MCU_FLASH = 0x012,
                                VALUE_MCU_FLASH_WRITE = 0x01,
                                VALUE_MCU_FLASH_READ = 0x02,
                             MCU_MEASURE_ENERGY_W_A = 0x013,
                             MCU_MEASURE_ENERGY_VA_A = 0x014,
                             MCU_MEASURE_ENERGY_VAR_A = 0x15,
                             MCU_MEASURE_ENERGY_W_B = 0x016,
                             MCU_MEASURE_ENERGY_VA_B = 0x017,
                             MCU_MEASURE_ENERGY_VAR_B = 0x18,
```

```
                        MCU_MEASURE_ENERGY_W_C = 0x019,
                        MCU_MEASURE_ENERGY_VA_C = 0x01A,
                        MCU_MEASURE_ENERGY_VAR_C = 0x01B,
                        MCU_MEASURE_PHASE_W1A = 0x01C,
                        MCU_MEASURE_PHASE_W1B = 0x01D,
                        MCU_MEASURE_PHASE_W1C = 0x01E,
                        MCU_MEASURE_PHASE_W2A = 0x01F,
                        MCU_MEASURE_PHASE_W2B = 0x020,
                        MCU_MEASURE_PHASE_W2C = 0x021;


    //COMKP_AD7758 Registers
    public static final int AD7758_AWATTHR = 0x001,
                    AD7758_BWATTHR = 0x002,
                    AD7758_CWATTHR = 0x003,
                    AD7758_AVARHR = 0x004,
                    AD7758_BVARHR = 0x005,
                    AD7758_CVARHR = 0x006,
                    AD7758_AVAHR = 0x007,
                    AD7758_BVAHR = 0x008,
                    AD7758_CVAHR = 0x009,
                    AD7758_AIRMS = 0x00A,
                    AD7758_BIRMS = 0x00B,
                    AD7758_CIRMS = 0x00C,
                    AD7758_AVRMS = 0x00D,
                    AD7758_BVRMS = 0x00E,
                    AD7758_CVRMS = 0x00F,
                    AD7758_FREQ = 0x010,
                    AD7758_TEMP = 0x011,
                    AD7758_WAVE = 0x012,
                    AD7758_OPMODE = 0x013,
                    AD7758_MMODE = 0x014,
                    AD7758_WAVMODE = 0x015,
                    AD7758_COMPMODE = 0x016,
                    AD7758_LCYCMODE = 0x017,
                    AD7758_MASK = 0x018,
                    AD7758_STATUS = 0x019,
                    AD7758_RSTATUS = 0x01A,
                    AD7758_ZXTOUT = 0x01B,
                    AD7758_LINECYC = 0x01C,
                    AD7758_SAGCYC = 0x01D,
                    AD7758_SAGLVL = 0x01E,
                    AD7758_VPINTLVL = 0x01F,
                    AD7758_IPINTLVL = 0x020,
                    AD7758_VPEAK = 0x021,
                    AD7758_IPEAK = 0x022,
                    AD7758_GAIN = 0x023,
                    AD7758_AVRMSGAIN = 0x024,
                    AD7758_BVRMSGAIN = 0x025,
                    AD7758_CVRMSGAIN = 0x026,
                    AD7758_AIGAIN = 0x027,
                    AD7758_BIGAIN = 0x028,
                    AD7758_CIGAIN = 0x029,
                    AD7758_AWG = 0x02A,
                    AD7758_BWG = 0x02B,
                    AD7758_CWG = 0x02C,
                    AD7758_AVARG = 0x02D,
                    AD7758_BVARG = 0x02E,
                    AD7758_CVARG = 0x02F,
                    AD7758_AVAG = 0x030,
                    AD7758_BVAG = 0x031,
                    AD7758_CVAG = 0x032,
                    AD7758_AVRMSOS = 0x033,
                    AD7758_BVRMSOS = 0x034,
                    AD7758_CVRMSOS = 0x035,
                    AD7758_AIRMSOS = 0x036,
                    AD7758_BIRMSOS = 0x037,
                    AD7758_CIRMSOS = 0x038,
                    AD7758_AWATTOS = 0x039,
                    AD7758_BWATTOS = 0x03A,
                    AD7758_CWATTOS = 0x03B,
                    AD7758_AVAROS = 0x03C,
                    AD7758_BVAROS = 0x03D,
                    AD7758_CVAROS = 0x03E,
                    AD7758_APHCAL = 0x03F,
                    AD7758_BPHCAL = 0x040,
                    AD7758_CPHCAL = 0x041,
                    AD7758_WDIV = 0x042,
                    AD7758_VARDIV = 0x043,
```

```
                AD7758_VADIV = 0x044,
                AD7758_APCFNUM = 0x045,
                AD7758_APCFDEN = 0x046,
                AD7758_VARCFNUM = 0x047,
                AD7758_VARCFDEN = 0x048,
                AD7758_CHKSUM = 0x07E,
                AD7758_VERSION = 0x07F;

//public final boolean debug = true;
public final boolean debug = false;


/** Creates a new instance of protocol */
public protocol() {
    active = false;
    length = 0;
    possition = 0;
    CRC = 0;
}

public void setActive(boolean active) {
    this.active = active;
    if (active) {
      setLength(0);
      setPossition(1);
      if (debug) System.out.println("Active!");
    } else {
      if (debug) {
          System.out.println("Not Active!");
          System.out.flush();
      }
    }
}
public boolean isActive() {
    return active;
}

public void setLength(int length) {
    i = 0;
    this.length = length;
    if (length > 2) {
      data = new int[length - 4];
    } else {
      data = new int[0];
    }
    nextByte(length);
    if (debug) System.out.println("Length: "+length);
    if (debug) System.out.println("["+data.length+"]");
}
public int getLength() {
    return length;
}

public void setPossition(int possition) {
    this.possition = possition;
}
public int getPossition() {
    return possition;
}

public void setComponent(int c) {
    i = 0;
    this.component = c;
    nextByte(c);
    if (debug) System.out.println("Component: "+c);
}
public int getComponent() {
    return component;
}

public void setRegister(int register) {
    i = 0;
    this.register = register;
    nextByte(register);
    if (debug) System.out.println("Register: "+register);
}
public int getRegister() {
    return register;
}
```



```
public void nextByte(int c) {
    possition++;
    c = c & 0x0FF;
    CRC = CRC + c;
}
public void addByte(int c) {
    if (i < data.length) {
        data[i] = c;
        i++;
        nextByte(c);
    }
    if (debug) System.out.println("Data["+i+"]: "+c);
}
public void checkCRC(int c) {
    c = c & 0x0FF;
    if (CRC_CHAR == c) {
        totalPackets++;
    } else {
     //   data = new int[0];
    }
}
public int getTotalPackets() {
    return totalPackets;
}
public int getValue() {
    int value = 0;
    if (data.length == 4) {
        value = (data[0]*0x01000000) + (data[1]*0x010000) + (data[2]*0x0100) + data[3];
    } else if (data.length == 2) {
        value = (data[0]*0x0100) + data[1];
    } else if (data.length == 1) {
        value = data[0];
    } else {
        latestValue = 0;
        return 0;
    }
    latestValue = value;
    return value;
}

//public String getString() {
//    return new String(data);
//}
public int getLatestValue() {
    return latestValue;
}

public String getLatestHexValue() {
    return "0x0"+Integer.toHexString(getLatestValue()).toUpperCase();
}

public void decode(InputStream in) {
        int c;
        byte b;
        boolean incomplete = true;
        try {
            c = in.read();
            while ((c != -1) & incomplete){
                if (isActive()) {
                    if (getLength() == 0) {
                        setLength(c);
                    } else {
                        if (getPossition() == 2) {
                            setComponent(c);
                        } else if (getPossition() == 3) {
                            setRegister(c);
                        } else if (getPossition() == getLength()) {
                            checkCRC(c);
                            incomplete = false;
                            setActive(false);
                            getValue();
                        } else {
                            addByte(c);
                        }
                    }
                } else {
                    if (c == ACTIVE) {
                        setActive(true);
                    }
```

```
                }
                c = in.read();
            }
        } catch (IOException e) {
            System.out.println("Error reading input stream:"+e.getMessage());
        }
}
public byte[] encode(int Component, int Register, byte[] data) {
    if (debug) System.out.print("Encoding...");
    int length = HEADER_LENGTH+data.length;
    byte[] out = new byte[MAX_LENGTH+length+1+MAX_LENGTH];
    byte tmpByte =0;
    out[MAX_LENGTH+0] = new Integer(ACTIVE).byteValue();
    out[MAX_LENGTH+1] = new Integer(length).byteValue();
    out[MAX_LENGTH+2] = new Integer(Component).byteValue();
    out[MAX_LENGTH+3] = new Integer(Register).byteValue();
    for (int i = 0; i < data.length; ++i) {
      out[MAX_LENGTH+i+HEADER_LENGTH] = data[i];
    }
    out[MAX_LENGTH+length] = new Integer(CRC_CHAR).byteValue();
    if (debug) {
        for (int i = MAX_LENGTH; i < out.length - MAX_LENGTH; ++i) {
          System.out.print("["+out[i]+"]");
        }
        System.out.println("");
    }
    return out;
}

public byte[] data(byte value1) {
    byte[] tmpByte = new byte[1];
    tmpByte[0] = value1;
    return tmpByte;
}
public byte[] data(byte value1, byte value2) {
    byte[] tmpByte = new byte[2];
    tmpByte[0] = value1;
    tmpByte[1] = value2;
    return tmpByte;
}
public byte[] data(byte value1, byte value2, byte value3) {
    byte[] tmpByte = new byte[3];
    tmpByte[0] = value1;
    tmpByte[1] = value2;
    tmpByte[2] = value3;
    return tmpByte;
}

public byte[] data(byte value1, byte value2, byte value3, byte value4) {
    byte[] tmpByte = new byte[4];
    tmpByte[0] = value1;
    tmpByte[1] = value2;
    tmpByte[2] = value3;
    tmpByte[3] = value4;
    return tmpByte;
}

public byte[] dataInt2(int value) {
    byte[] tmpByte = new byte[2];
    tmpByte[0] = (byte)(0x00FF & ((0x0FF00&value)/0x0100));
    tmpByte[1] = (byte)(0x00FF & value);
    if (debug) System.out.println(" dataInt2[0]: "+tmpByte[0]);
    if (debug) System.out.println(" dataInt2[1]: "+tmpByte[1]);
    return tmpByte;
}

public byte[] dataInt4(int value) {
    byte[] tmpByte = {0,0,0,0};
    tmpByte[0] = (byte)(0x00FF & ((0x00FF000000&value)/0x01000000));
    tmpByte[1] = (byte)(0x00FF & ((0x00FF0000&value)/0x010000));
    tmpByte[2] = (byte)(0x00FF & ((0x00FF00&value)/0x0100));
    tmpByte[3] = (byte)(0x00FF & (value));
    if (debug) System.out.println(" dataInt2[0]: "+tmpByte[0]);
    if (debug) System.out.println(" dataInt2[1]: "+tmpByte[1]);
    if (debug) System.out.println(" dataInt2[2]: "+tmpByte[2]);
    if (debug) System.out.println(" dataInt2[3]: "+tmpByte[3]);
    return tmpByte;
}
```

```
}
```

Q.5.5 Object: `peg.io.serialInterface`

```
/*
 * serialInterface.java
 *
 * Created on 30 March 2004, 11:50
 */

package peg.io;

import java.util.*;
import java.io.*;

import javax.comm.*;
/**
 *
 * @author  RDoorduin
 */
public class serialInterface {
    private SerialPort port = null;
    public InputStream in;
    public OutputStream out;
    private Enumeration ports;
    private CommPortIdentifier portID;
    private Vector portIDvec = new Vector();
    private int numPorts = 0;
    private boolean open = false;
    private String identifier = "Port";
    private peg.utils.debug db = new peg.utils.debug("serialInterface",9);

    /** Creates a new instance of serialInterface
     *  During initialization all the basic components will be generated
     */
    public serialInterface(String identifier) {
        this.identifier = identifier;
        db.print(1, "CommPortIdentifier.getPortIdentifiers()...");
        ports = CommPortIdentifier.getPortIdentifiers();
        getSerialPorts();
    }

    public void getSerialPorts() {
        if (ports != null) {
          db.print(5, "Search for Ports...");
          while (ports.hasMoreElements()) {
              portID = (CommPortIdentifier)ports.nextElement();
              if (portID.getPortType() == CommPortIdentifier.PORT_SERIAL) {
                  portIDvec.add(portID);
                  db.print(0,""+portID.getName());
                  numPorts++;
              }
          }
          db.print(5, "Done!");
        }
    }

    public boolean setCurrentPort(int index) {
        if (!this.open) {
            portID = (CommPortIdentifier)portIDvec.get(index);
            return true;
        } else {
            return false;
        }
    }

    public boolean openCurrentPort() {
        if (this.open) {
            closeCurrentPort();
        }
        open = false;
        try {
            port = (SerialPort)portID.open(identifier, 2000);
        } catch (PortInUseException e) {
            System.out.println("Port in use: "+e.getMessage());
            port = null;
```

```
        }
        if (port == null) {
            System.out.println("Error opening Port!");
            return false;
        } else {
            this.open = true;
            try {
                in = this.port.getInputStream();
            } catch (IOException e) {
                System.out.println("Cannot open input stream!");
            }
            try {
                out = this.port.getOutputStream();
            } catch (IOException e) {
                System.out.println("Cannont open output stream!");
            }
        }
        return true;
    }

    public boolean openPort(String thisIdentifier) {
        if (this.open) {
            closeCurrentPort();
        }
        open = false;
        try {
            port = (SerialPort)portID.open(thisIdentifier, 2000);
        } catch (PortInUseException e) {
            System.out.println("Port in use: "+e.getMessage());
            port = null;
        }
        if (port == null) {
            System.out.println("Error opening Port!");
            return false;
        } else {
            this.open = true;
            try {
                in = this.port.getInputStream();
            } catch (IOException e) {
                System.out.println("Cannot open input stream!");
            }
            try {
                out = this.port.getOutputStream();
            } catch (IOException e) {
                System.out.println("Cannot open output stream!");
            }
        }
        return true;
    }

    public boolean closeCurrentPort() {
        if (this.open) {

        }
        return false;
    }

    public boolean addListener(SerialPortEventListener o) {
        try {
            port.addEventListener(o);
        } catch (TooManyListenersException e) {
            System.out.println("Too many Listeners:" + e.getMessage());
            e.printStackTrace();
            return false;
        }
        port.notifyOnDataAvailable(true);
        return true;
    }

    public boolean isOpen() {
        return open;
    }

    public void sendPacket(byte[] stream) {
        try {
            this.out.write(stream);
            this.out.flush();
        } catch (IOException e) {
            System.out.println("Error communicating with device: " + e.getMessage());
```

```
            System.out.println(":" + stream);
        }
    }

    public void print(String str) {
        try {
            out.write(str.getBytes());
        } catch (IOException e) {
            System.out.println("Error printing to Serial Port: "+e.getMessage());
        }
    }
}
```

Q.5.6 Object: `peg.io.stdInput`

```
/*
 * stdInput.java
 *
 * Created on 18 September 2003, 08:55
 */

package peg.io;
import java.io.*;
/** This class assumes that the standard input device will be used, e.g. Keyboard.
 *
 *  <PRE>Example implementation: </PRE>
 *   <CODE>
 *   <PRE>
 *   String myString = String();
 *   myString = (new peg.io.stdInput()).readLine();
 *   </PRE>
 *   </CODE>
 *
 * @author Riaan Doorduin
 */
public class stdInput extends java.io.BufferedReader {

    /** Creates a new instance of input */
    public stdInput() {
      super(new java.io.InputStreamReader(System.in));
    }

    /** Method <B>readLn</B> will return the input from the standard input device, e.g.
     * keyboard in a string format.
     * @return String The input from the standard input will be returned in this String by
     * using this method
     */
    public String readLn() {
        String tmpString = new String();
        try {
            tmpString = this.readLine();
        } catch (IOException e) {
            tmpString = ("Error: " + e.getMessage());
        }
        return tmpString;
    }
}
```

Q.6 Package: `peg.sql`

Q.6.1 Object: `peg.sql.DBConnect`

```
/*
 * DBConnect.java
 *
 * Created on 16 September 2003, 10:53
 */

package peg.sql;

import javax.sql.*;
import java.sql.*;
```

```
import java.lang.*;
import javax.swing.table.AbstractTableModel;
import java.util.*;

import gov.noaa.pmel.util.*;
import gov.noaa.pmel.sgt.dm.*;
import gov.noaa.pmel.sgt.*;

import java.math.*;
import java.io.Serializable;

import peg.utils.*;
/** DBConnect is used to connect to any database, provided it has a complient JDBC2
 * driver. It returns the data in the TableModel format, which can be displayed in
 * the JTable component.
 *
 * <PRE>
 * (DataBase) --> (TableModel)
 * </PRE.
 *
 * @author Riaan Doorduin
 * @version 1.0
 */
public class DBConnect extends AbstractTableModel {
    private final int EXTRA_ROW = 0;
    //The driver supplied by database vendor
    private String driver = null;
    //The protocol used by database
    private String protocol = null;
    //The database connection to on remote or local machine
    private String subname = null;
    //The user name used to connect to the database
    private String user = null;
    //The users password to connect to the database
    private String password = null;
    //The query to extract data from the database
    private String query = null;
    //Connection to database
    private Connection con = null;
    //Statement to be sent to database
    private Statement stmt = null;
    //ResultSet of query
    private ResultSet rs = null;
    //UpdateTime to determine query duration
    private long updateTime;
    //DataVector to keep data from Each ROw
    private Vector col = new Vector();
    //ColRow will keep each row in the collumn
    private Vector row = new Vector();
    //ColumnNames Array
    private String[] columnNames;
    //ColumnClass Array
    private Class[] columnClass;
    //ColumnCount value
    private int columnCount;
    //StartTime
    private long startTime;
    //ResultSet Status
    private boolean isResultSet;
    //isConnected indicates if DBase is connected.
    private boolean isConnectedVar = false;
    //isQuerySuccess indicates if last query was executed successfully.
    private boolean isQuerySuccessVar = false;
    //errorString is used to report errors to user.
    private String errorString = new String();
    //isAlwaysUpdate indicates to the object that the database is query should be requisted each time the query is asked.
    private boolean isAlwaysUpdate = false;

    private debug d = new debug("DBConnect",0);


//-Initialisation of variables----------------------------------------------------------------

    /** Create a new instance of <B>DBConnect</B>. */
    public DBConnect() {
        this(null,null,null,null,null,null);
        d.print(1,"DBConnect()");
    }
    /** Create a new instance of <B>DBConnect</B> to connect to the database.
```

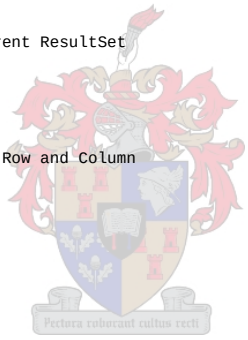
```
     * The class will by default try and connect to the database upon construction.
     * @param driver the driver supplied by the vendor, e.g. ODBC:sun.jdbc.odbc.JdbcOdbcDriver, FireBird:org.firebirdsql.jdbc.FBDriver
     * @param protocol the protocol to be used for data transfer, e.g. jdbc:odbc or dbc:firebirdsql
     * @param subname the location of the database e.g. MySmallTextDB or localhost/3050:d:/CurrentWork/RevenueProtectionSoftware/Databases/myT
     */
    public DBConnect(String driver, String protocol, String subname){
        this(driver,protocol,subname,null,null,null);
        d.print(1,"DBConnect(driver,protocol,subname)");
    }

    /** Create a new instance of <B>DBConnect</B> to connect to the database.
     * The class will by default try and connect to the database upon construction.
     * @param driver the driver supplied by the vendor, e.g. ODBC:sun.jdbc.odbc.JdbcOdbcDriver, FireBird:org.firebirdsql.jdbc.FBDriver
     * @param protocol the protocol to be used for data transfer, e.g. jdbc:odbc or dbc:firebirdsql
     * @param subname the location of the database e.g. MySmallTextDB or localhost/3050:d:/CurrentWork/RevenueProtectionSoftware/Databases/myT
     * @param query the query to be processed to extract the data
     */
    public DBConnect(String driver, String protocol, String subname, String query){
        this(driver,protocol,subname,null,null,query);
        d.print(1,"DBConnect(driver,protocol,subname,query)");
    }

    /**
     * Create a new instance of <B>DBConnect</B> to connect to the database.
     * The class will by default try and connect to the database upon construction.
     *
     * @param driver the driver supplied by the vendor, e.g. ODBC:sun.jdbc.odbc.JdbcOdbcDriver, FireBird:org.firebirdsql.jdbc.FBDriver
     * @param protocol the protocol to be used for data transfer, e.g. jdbc:odbc or dbc:firebirdsql
     * @param subname the location of the database e.g. MySmallTextDB or localhost/3050:d:/CurrentWork/RevenueProtectionSoftware/Databases/myT
     * @param user the username required by the database
     * @param password the password to acompany the username
     */
    public DBConnect(String driver, String protocol, String subname, String user, String password){
        this(driver,protocol,subname,user,password,null);
        d.print(1,"DBConnect(driver,protocol,subname,password)");
    }

    /**
     * Create a new instance of <B>DBConnect</B> to connect to the database.
     * The class will by default try and connect to the database upon construction.
     *
     * @param driver the driver supplied by the vendor, e.g. ODBC:sun.jdbc.odbc.JdbcOdbcDriver, FireBird:org.firebirdsql.jdbc.FBDriver
     * @param protocol the protocol to be used for data transfer, e.g. jdbc:odbc or dbc:firebirdsql
     * @param subname the location of the database e.g. MySmallTextDB or localhost/3050:d:/CurrentWork/RevenueProtectionSoftware/Databases/myT
     * @param user the username required by the database
     * @param password the password to acompany the username
     * @param query the query to be processed to extract the data
     */
    public DBConnect(String driver, String protocol, String subname, String user, String password, String query){
        d.print(1,"DBConnect(driver="+driver);
        setDriver(driver);
        d.print(1,"          protocol="+protocol);
        setProtocol(protocol);
        d.print(1,"          subname="+subname);
        setSubname(subname);
        d.print(1,"          user="+user);
        setUser(user);
        d.print(1,"          password="+password);
        setPassword(password);
        d.print(1,"          query="+query);
        setQuery(query);
        d.print(1,"DBConnect(...):connect()...");
        connect();
        if (query != null) {
            d.print(3,"DBConnect(...):this.ExecuteSQL()...");
            this.executeSQL();
        }
    }

    /** Set the <B>Driver</B>  to connect to the database
     * @param driver the driver supplied by the vendor, e.g. ODBC:sun.jdbc.odbc.JdbcOdbcDriver, FireBird:org.firebirdsql.jdbc.FBDriver
     */
    public void setDriver(String driver) {
        if (driver != null) {
          this.driver = driver;
        }
    }

    /**
```

```
     * Set the <B>Protocol</B>  to connect to the database
     *
     * @param protocol the protocol to be used for data transfer, e.g. jdbc:odbc or dbc:firebirdsql
     */
    public void setProtocol(String protocol) {
        if (protocol != null) {
            if (protocol.endsWith(":")) {  //in the unlikely event that a user is used to typing the ":" after the protocol as in the protocol
                this.protocol = protocol.substring(0,protocol.length()-1);
            } else {
                this.protocol = protocol;
            }
        }
    }

    /** Set the <B>Subname</B>  to connect to the database
     * @param subname the location of the database e.g. MySmallTextDB or localhost/3050:d:/CurrentWork/RevenueProtectionSoftware/Databases/myT
     */
    public void setSubname(String subname) {
        if (subname != null) {
          this.subname = subname;
        }
    }

    /**
     * Set the <B>User</B>  to connect to the database
     *
     * @param user the username required by the database
     */
    public void setUser(String user) {
        if (user != null) {
            if (user.length() != 0) {
                this.user = user;
            } else {
                this.user = null;
            }
        }
    }

    /**
     * Set the <B>Password</B>  to connect to the database
     *
     * @param password the password to acompany the username
     */
    public void setPassword(String password) {
        if (password != null) {
            if (password.length() != 0) {
                this.password = password;
            } else {
                this.password = null;
            }
        }
    }

    /** Set the <B>Query</B>  to execute in the database
     * @param query The query to be processed to extract the data
     */
    public void setQuery(String query) {
        if (query != null) {
          this.query = query;
        }
    }

//-Methods to interface to database-----------------------------------------------------------------

   //--Query-->

    /** <B>executeSQL</B> executes the query to the database. Update takes care of the connection and driver. It also closes the database conn
     * Sucees of query is returned via isQuerySuccess method.
     * @param query Query for new SQL statement to be executed.
     */
    public void executeSQL(String query) {
        setQuery(query);
        executeSQL();
    }

    /** <B>executeSQL</B> executes the stored query to the database. Update takes care of the connection and driver. It also closes the databa
     * Sucees of query is returned via isQuerySuccess method.
     */
    public void executeSQL() {
```

```
        d.print(3,"executeSQL():setQuerySuccess(false)");
        setQuerySuccess(false);
        if (query == null) {
            errorString = "Error: Query not specified.";
            d.print(5,"executeSQL():"+errorString);
        } else {
            if (!isConnected()){
                d.print(3,"executeSQL():connect()...");
                connect();  // if not connected, try again.
            }
            if (isConnected()) {
                try {
                    d.print(3,"executeSQL:stmt=con.createStatement...");
                    stmt = con.createStatement();  //ResultSet.TYPE_SCROLL_INSENSITIVE,ResultSet.CONCUR_UPDATABLE);//java.sql.ResultSet.TYPE_F
                    try {
                        StringTokenizer st = new StringTokenizer(query,";");
                        String tmpQuery = new String();
                        while (st.hasMoreTokens()) {
                           tmpQuery = st.nextToken();
                           d.print(3,"executeSQL:isResultSet("+tmpQuery+")");
                           isResultSet = stmt.execute(tmpQuery);
                        }
                        d.print(3,"executeSQL:getResultSet()");
                        rs = stmt.getResultSet();
                        d.print(3,"executeSQL:setQureySuccess(true)");
                        setQuerySuccess(true);
                    } catch (SQLException e) {
                        errorString = "Query: " + e.getMessage();
                        d.print(5,"executeSQL():"+errorString);
                    } catch (Exception e) {
                        errorString = "StatementError: " + e.getMessage();
                        d.print(5,"executeSQL():"+errorString);
                    }
                } catch (SQLException e) {
                    errorString = "Error CreateStatement: " + e.getMessage();
                    d.print(5,"executeSQL():"+errorString);
                }
            }
        }
    }

    /** Set the connection state of the last executed query.
     * @param Success state of query.
     */
    private void setQuerySuccess(boolean isQuerySuccessHere) {
        isQuerySuccessVar = isQuerySuccessHere;
    }

    /** Returns the connection state of the last executed query.
     * @return If last query was executed sucessfully.
     */
    public boolean isQuerySuccess() {
        return isQuerySuccessVar;
    }

//--Connection Establishment-->

    /** Method to initiate connection to database.
     * The status of the connection is shown via the isConnected() method.
     */
    public void connect() {
        //load the driver
        d.print(3,"connect():setConnect(false)...");
        setConnected(false);
        if ((driver == null) || (protocol == null) || (subname == null)) {
            errorString = "Connect: Driver, protocol or subname not specified.";
            d.print(5,"connect():"+errorString);
        } else {
            try{
                d.print(3,"connect():getDriver...");
                Class.forName(driver).newInstance();
                try {
                    if ((user == null) || (password == null)) {
                        con = DriverManager.getConnection((protocol+":"+subname));
                    } else {
                        con = DriverManager.getConnection((protocol+":"+subname), user, password);
                    }
                    d.print(3,"connect():setConnected(true)");
                    setConnected(true);
```

```
                } catch (SQLException e) {
                    errorString = "Connection: " + e.getMessage();
                    d.print(5, "connect():"+errorString);
                }
            } catch (Exception e) {
                errorString = "Driver: " + e.getMessage();
                d.print(5, "connect():"+ errorString);
            }
        }
    }

    /** Method to close the database connection.
     * The status of the connection is shown via the isConnected() method.
     */
    public void close() {
        //closing connection
        setConnected(false);
        if (isConnected()) {
          try {
            stmt.close();
            con.close();
          } catch (SQLException e) {
            errorString = "Closing: " + e.getMessage();
          }
        }
    }

    /** Returns the connection state of the database connection.
     * @return If database is still connected true is returned.
     */
    public boolean isConnected() {
        return isConnectedVar;
    }

    /** Returns the connection state of the database connection.
     * @return If database is still connected true is returned.
     */
    private void setConnected(boolean value) {
        isConnectedVar = value;
    }

    /** Returns the lastest error message generated by the DBConnect class.
     * @return The last error message.
     */
    public String getErrorString() {
        return errorString;
    }

//-Table Model interface-----------------------------------------------------------------------
    /* For the case of incompotent JDBC Driver this method can be used to store values of the query
     * @return True if successfull and False if not.
     */
    private boolean toTableModel() {
        int rowCount = -1;
        col = new Vector();
        row = new Vector();
        if (isResultSet) {
            try {
                columnCount = rs.getMetaData().getColumnCount();
                columnNames = new String[columnCount];
                columnClass = new Class[columnCount];
            } catch (SQLException e) {
                errorString = "Error during table creation: " + e.getMessage();
                return false;
            }
            try {
                for (int i=1; i <= columnCount; i++) {
                    columnNames[i-1] = (rs.getMetaData().getColumnName(i));
                    try {
                        columnClass[i-1] = (Class.forName(rs.getMetaData().getColumnClassName(i)));
                    } catch (ClassNotFoundException n){
                        columnClass[i-1] = ("").getClass();
                    }
                }
            } catch (SQLException e) {
                errorString = "Error while retrieving column classes: " + e.getMessage();
            }
            try {
                while (rs.next()) {
```

```
                col = new Vector();
                for (int i=1; i <= columnCount; i++){
                    col.addElement(rs.getObject(i));
                }
                row.addElement(col);
            }
        } catch (SQLException e) {
            errorString = "Error retrieving row and column data: " + e.getMessage();
            return false;
        }
    } else {
        try {
            columnCount = 0;
            rowCount = stmt.getUpdateCount();
        } catch (SQLException e) {
            errorString = "Error retreiving update Data: " + e.getMessage();
        }
    }
    return true;
}

/** Returns the amount of columns of the current ResultSet
 * @return The number of columns
 */
public int getColumnCount() {
    //return columnCount; //Using internal Row and Column
    try {
        return rs.getMetaData().getColumnCount();
    } catch (SQLException e) {
        errorString = "Error during Resultset Column Count!";
        return 0;
    }
}
/** Returns the amount of rows of the current ResultSet
 * @return The number of rows
 */
public int getRowCount(){
    int i = 0;
    //return row.size(); //Using internal Row and Column
    try {
        rs.first();
        i = 1;
        while (rs.next()) {
            i++;
        }
        return (i+EXTRA_ROW);
    } catch (SQLException e) {
        errorString = "Error while retrieving rowcount:" +e.getMessage();
        return (0+EXTRA_ROW);
    }
}
/** Returns the name of each column
 * @return Returns the name of the column specified
 * @param cl The column index number of the required column
 */
public String getColumnName(int cl) {
    //return columnNames[cl]; //Old method
    try {
        return rs.getMetaData().getColumnName(cl+1);
    } catch (SQLException e){
        errorString = "Error reading Column Names" + e.getMessage();
        return null;
    }
}

/** Return the value of the selected column and row.
 * @param cl The column in the ResultSet
 * @param rw The row in the current ResultSet.
 * @return The value/ object at the current row/column position.
 */
public Object getValueAt(int rw, int cl) {
    //return (Object)((Vector)row.get(rw)).get(cl);
    try {
        if (rw+EXTRA_ROW == (getRowCount())) {
            return "";
        }
        rs.absolute(rw+1);
        return rs.getObject(cl+1);
    } catch (SQLException e){
```



```
            errorString = "Error reading values: (Row:Col"+rw+":"+cl+") " + e.getMessage();
            return null;
        }
    }

    /** Returns the column class of a spcific column.
     * @param c The column in the ResultSet
     * @return The Class of the specific column
     */
    public Class getColumnClass(int c) {
        Class thisClass;
        try {
            try {
                thisClass = (Class.forName(rs.getMetaData().getColumnClassName(c+1)));
                if (rs.getMetaData().getColumnClassName(c+1) == "java.sql.Date") {
                  thisClass = String.class;
                }
                return thisClass;
            } catch (ClassNotFoundException n) {
                errorString = "Error while determining Column Class: "+n.getMessage();
                return String.class;
            }
        } catch (SQLException e) {
            errorString = "Error while determining Column Class: "+e.getMessage();
            return String.class;
        }
    }
    /** Returns if the editable or not.
     * @param rw Row to be checked for editability.
     * @param cl Column to be checked for editability.
     * @return True if cell is editable, currently hardcoded FALSE
     */
    public boolean isCellEditable(int rw, int cl) {
        return false;
    }
    /** Sets the value for the selected row/column
     * @param value The value of the soon to be changed cell.
     * @param rw The row of the column to be changed
     * @param cl Column of the soon to be changed value.
     */
    public void setValueAt(Object value, int rw, int cl){
        if (rw+1 == (getRowCount())) {
            try {
            rs.moveToInsertRow();
            rs.updateObject(cl+1,value);
            rs.insertRow();
            } catch (SQLException e){
                errorString = "Error inserting value: (Row:Col,"+rw+":"+cl+") " + e.getMessage();
            }
        } else {
            try {
                rs.absolute(rw+1);
                rs.updateObject(cl+1,value);
                rs.updateRow();
            } catch (SQLException e){
                errorString = "Error updateing values: (Row:Col"+rw+":"+cl+") " + e.getMessage();
            }
        }
    }
}
```

Q.7 Package: `peg.swing`

Q.7.1 Object: `peg.swing.GraphDisplay`

```
/*
 * GraphOut.java
 *
 * Created on November 12, 2003, 12:12 PM
 */
package peg.swing;

import gov.noaa.pmel.sgt.dm.*;
import gov.noaa.pmel.sgt.swing.*;
```

```
import gov.noaa.pmel.sgt.*;
/** The Class Graph display is designed to display graphs in a panel. This enables a
 * programmer to easily add it to a from or even inside other panels.
 *
 * Datasets are expected in a SimpleLine variable as in
 * gov.noaa.pmel.sgt.dm.SimpleLine.
 *
 *  <PRE>
 * (SimpleLine) --> (GUI)
 * </PRE>
 *
 * Example usage:
 * <PRE>
 * SimpleLine sL = ...; //Here the graph data is build according to the SimpleLine
 * Class
 * GraphDisplay sLPlot = new GraphDisplay(); //New Graph display envoked
 * sLPlot.setTitles("Main Title","Sub Title","Sub sub title"); //Create titles
 * sLPlot.addData(sL,"New data Set"); //Add the data to GraphDisplay
 * JFrame myTempFrame = new JFrame(); //Creates a new frame
 * myTempFrame.setContentPane(sLPlot); //Set GraphDisplay to the contentPane of the
 * frame
 * myTempFrame.show(); //Make the frame visible to the user
 * </PRE>
 * @author Riaan Doorduin
 */
public class GraphDisplay extends javax.swing.JPanel {
    private JPlotLayout jPlotLayout1 = null;

    /** Creates new panel GraphOut */
    public GraphDisplay(boolean isTime) {
        initComponents();
        jPlotLayout1 = new JPlotLayout(false,isTime,false,"SimpleLinePlotFrame",null,false);
        displayGraph();
    }

    public GraphDisplay() {
        this(true);
    }

    /** Create a new instance with the titles ready to be used.
     * @param title1 Main title of the graph
     * @param title2 Sub title to the graph
     * @param title3 Small Title to give that extra bit of meaning to the graph
     */
    public GraphDisplay(String title1, String title2, String title3) {
        initComponents();
        setTitles(title1,title2,title3);
        displayGraph();
    }


    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jToolBar1 = new javax.swing.JToolBar();
        btnRefresh = new javax.swing.JButton();
        btnReset = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();

        setLayout(new java.awt.BorderLayout());

        btnRefresh.setText("Zoom");
        btnRefresh.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnRefreshActionPerformed(evt);
            }
        });

        jToolBar1.add(btnRefresh);

        btnReset.setText("Reset");
        btnReset.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnResetActionPerformed(evt);
            }
        });
```

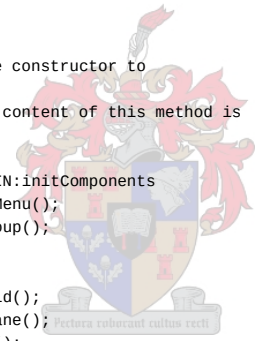
```
        jToolBar1.add(btnReset);

        add(jToolBar1, java.awt.BorderLayout.NORTH);

        jPanel1.setLayout(new java.awt.BorderLayout());

        add(jPanel1, java.awt.BorderLayout.CENTER);

    }//GEN-END:initComponents

    private void btnResetActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnResetActionPerformed
        // Add your handling code here:
        jPlotLayout1.resetZoom();
        this.updateUI();
    }//GEN-LAST:event_btnResetActionPerformed

    private void btnRefreshActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnRefreshActionPerformed
        // Add your handling code here:
        this.updateUI();
    }//GEN-LAST:event_btnRefreshActionPerformed


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JButton btnRefresh;
    private javax.swing.JButton btnReset;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JToolBar jToolBar1;
    // End of variables declaration//GEN-END:variables

    /** Set the titles of the graph panel
     * @param title1 Main title of the graph
     * @param title2 Sub title to the graph
     * @param title3 Small Title to give that extra bit of meaning to the graph
     */
    public void setTitles(String title1, String title2, String title3) {
        jPlotLayout1.setTitles(title1,title2,title3);
    }

    /** Add a new dataset to the current graph
     * @param mySL SGT SimpleLine Data set
     */
    public void addData(SimpleLine mySL) {
        addData(mySL,null,null);
    }

    /** Add a new dataset to the current graph
     * @param mySL SGT SimpleLine Data set
     * @param dataTitle Legend for the current dataset
     */
    public void addData(SimpleLine mySL, String dataTitle) {
        addData(mySL,dataTitle,null);
    }
    /** Add a new dataset to the current graph
     * @param mySL SGT SimpleLine Data set
     * @param dataTitle Legend for the current dataset
     * @param lineAttr Set the different line attributes for the dataset e.g. Solid or marked line
     */
    public void addData(SimpleLine mySL, String dataTitle, LineAttribute lineAttr) {
        if (mySL!=null) {
            if (jPanel1.getComponentCount() == 0) {
                jPanel1.add(jPlotLayout1);
            }
            SGTData sLData = mySL;
            if ((dataTitle != null) || (lineAttr != null)) { //Only addData method if both are not null
                jPlotLayout1.addData(sLData, lineAttr, dataTitle);
            } else {
                jPlotLayout1.addData(sLData);
            }
            jPlotLayout1.resetZoom();
            jPlotLayout1.setClipping(false);
            jPanel1.updateUI();
        }
    }
    /** Invokes and displayes the current graph */
    public void displayGraph() {
        // Add your handling code here:
        jPanel1.removeAll();
        if (jPlotLayout1.getComponentCount() != 0) {
            // For possible line and attribute changes.
```

```
            jPlotLayout1.getComponent();
        }
        jPanel1.updateUI();
    }
}
```

Q.7.2 Object: `peg.swing.myNode`

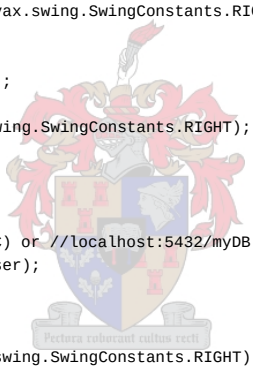
```
/*
 * FrameTest.java
 *
 * Created on October 29, 2003, 9:40 PM
 */

package peg.swing;

import peg.sql.*;
import peg.utils.*;
import javax.swing.*;
import java.awt.*;
import java.math.*;
import java.util.*;
import javax.swing.table.AbstractTableModel;

import gov.noaa.pmel.util.*;
import gov.noaa.pmel.sgt.dm.*;
import gov.noaa.pmel.sgt.swing.*;
import gov.noaa.pmel.sgt.swing.prop.*;
import gov.noaa.pmel.sgt.LineAttribute;

import edu.uah.math.distributions.*;

import java.io.Serializable;
/** This class is used to connect to the main source of the data used by myTree.
 * Currently only a connection to databases are available. It offers a storage area
 * for the children of the node, provide by the myTree class.
 *
 * Possible expansions can include flat file imports.
 * @author Riaan Doorduin
 */
public class myNode extends JFrame implements Serializable {
// Constants
    // Output Source
    /** The source used for output is determined by these constants.
     * No output will be provided by the node.
     */
    public final int SOURCE_NONE = 0;
    /** The output source will be pointed to the children. */
    public final int SOURCE_CHILDREN = 1;
    /** The output source will be pointed to the database. */
    public final int SOURCE_DATABASE = 2;
    /** The output source will be calculated from the difference between the data in the database and children. */
    public final int SOURCE_DB_CHILDREN = 3;
    /** The output source will be calculated from the difference between the data in the children and database. */
    public final int SOURCE_CHILDREN_DB = 4;
    /** The output source will be calculated from the difference between the data in the database and children. */
    public final int SOURCE_RANDOM = 5;
    // Random Engine
    public final int RANDOM_BETA = 0;
    public final int RANDOM_EVEN = 1;
    public final int RANDOM_NORMAL = 2;
    // Output Filter
    /** Points to the filter algorithm that no filter is required. */
    public final int FILTER_NONE = 0;
    /** Points to the filter algorithm that a moving average filter is required. */
    public final int FILTER_MOVAVG = 1;
    // Hystogram settings



//--FORM functionality-------------------------------------------------------------------------
    private static boolean mayCloseSystem = false;
    /** For debugging purposes */
    private debug d = new debug("myNode",0);
    private hystogram hyst = new hystogram();

    /** Creates new instance of myNode */
```

```
public myNode() {
    d.print(1,"myNode():initComponents...");
    initComponents();
    setBounds(getX(),getY(),getWidth()+10,getHeight()+10);
    setTitle(txtNodeName.getText());
}
/** Creates new instance from myNode and clones another.
 * @param newMyNode the myNode to be cloned
 */
public myNode(myNode newMyNode){
    this();
    d.print(1,"myNode(myNode):...");
    //General
    setNodeName(newMyNode.getNodeName());
    setArea(newMyNode.getArea());
    setUnit(newMyNode.getUnit());
    setDescription(newMyNode.getDescription());
    //Output
    setFunction(newMyNode.getFunction());
    setOutputSource(newMyNode.getOutputSource());
    setOutputFilter(newMyNode.getOutputFilter());
    setOutputDateFormat(newMyNode.getOutputDateFormat());

    //Input
    setInputDateFormat(newMyNode.getInputDateFormat());
    setDriver(newMyNode.getDriver());
    setProtocol(newMyNode.getProtocol());
    setUrl(newMyNode.getUrl());
    setUser(newMyNode.getUser());
    setPassword(newMyNode.getPassword());
    setQuery(newMyNode.getQuery());
    setTitle(txtNodeName.getText());
}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
private void initComponents() {//GEN-BEGIN:initComponents
    jPopupMenu1 = new javax.swing.JPopupMenu();
    bgDiffInt = new javax.swing.ButtonGroup();
    lblDriver = new javax.swing.JLabel();
    lblSQL = new javax.swing.JLabel();
    txtDriver = new javax.swing.JTextField();
    scrpQuery = new javax.swing.JScrollPane();
    txtQuery = new javax.swing.JTextArea();
    lblProtocol = new javax.swing.JLabel();
    txtProtocol = new javax.swing.JTextField();
    btnTestQuery = new javax.swing.JButton();
    lblNodeName = new javax.swing.JLabel();
    txtNodeName = new javax.swing.JTextField();
    lblErrorStr = new javax.swing.JLabel();
    lblUrl = new javax.swing.JLabel();
    txtUrl = new javax.swing.JTextField();
    lblUser = new javax.swing.JLabel();
    lblPassword = new javax.swing.JLabel();
    txtUser = new javax.swing.JTextField();
    txtPassword = new javax.swing.JPasswordField();
    txtTicToc = new javax.swing.JLabel();
    btnShowTable = new javax.swing.JButton();
    btnShowGraph = new javax.swing.JButton();
    lblArea = new javax.swing.JLabel();
    txtArea = new javax.swing.JTextField();
    lblUnit = new javax.swing.JLabel();
    txtUnit = new javax.swing.JTextField();
    lblDescription = new javax.swing.JLabel();
    txtDescription = new javax.swing.JTextField();
    lblMonthFormat = new javax.swing.JLabel();
    txtInputDateFormat = new javax.swing.JTextField();
    scrpErrorStr = new javax.swing.JScrollPane();
    txtErrorStr = new javax.swing.JTextArea();
    cmbFunction = new javax.swing.JComboBox();
    lblFunction = new javax.swing.JLabel();
    btnLineAttr = new javax.swing.JButton();
    cmbFilter = new javax.swing.JComboBox();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
```



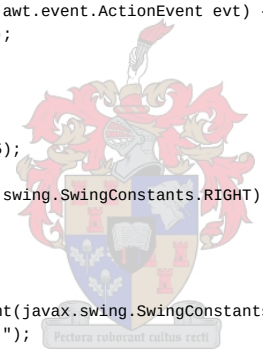
```
jLabel4 = new javax.swing.JLabel();
cmbSource = new javax.swing.JComboBox();
jLabel5 = new javax.swing.JLabel();
lblMonthFormat1 = new javax.swing.JLabel();
txtOutputDateFormat = new javax.swing.JTextField();
txtMovAvg = new javax.swing.JTextField();
jLabel6 = new javax.swing.JLabel();
btnOutputGraph = new javax.swing.JButton();
cmbPDF = new javax.swing.JComboBox();
lblMean = new javax.swing.JLabel();
txtMean = new javax.swing.JTextField();
lblDeviation = new javax.swing.JLabel();
txtDeviation = new javax.swing.JTextField();
lblScalingFactor = new javax.swing.JLabel();
txtScalingFactor = new javax.swing.JTextField();
txtOffset = new javax.swing.JTextField();
lblOffset = new javax.swing.JLabel();
jLabel7 = new javax.swing.JLabel();
jLabel8 = new javax.swing.JLabel();
jTextArea1 = new javax.swing.JTextArea();
txtStartTime = new javax.swing.JTextField();
btnStartTime = new javax.swing.JButton();
btnStopTime = new javax.swing.JButton();
txtStopTime = new javax.swing.JTextField();
lblAveragePeriod = new javax.swing.JLabel();
txtAveragePeriod = new javax.swing.JTextField();
txtHystMin = new javax.swing.JTextField();
lblHystMin = new javax.swing.JLabel();
lblHystMax = new javax.swing.JLabel();
txtHystMax = new javax.swing.JTextField();
cmbHystogram = new javax.swing.JComboBox();
jLabel9 = new javax.swing.JLabel();
btnResetHystogram = new javax.swing.JButton();
btnUpdate = new javax.swing.JCheckBox();
rdbDifferentiate = new javax.swing.JRadioButton();
rdbIntegrate = new javax.swing.JRadioButton();
rdbNone = new javax.swing.JRadioButton();

getContentPane().setLayout(null);

addWindowListener(new java.awt.event.WindowAdapter() {
    public void windowOpened(java.awt.event.WindowEvent evt) {
        formWindowOpened(evt);
    }
    public void windowClosing(java.awt.event.WindowEvent evt) {
        exitForm(evt);
    }
});

lblDriver.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblDriver.setText("Driver:");
getContentPane().add(lblDriver);
lblDriver.setBounds(80, 360, 50, 16);

lblSQL.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblSQL.setText("Query:");
getContentPane().add(lblSQL);
lblSQL.setBounds(80, 460, 50, 16);

txtDriver.setToolTipText("classpath to driver e.g. sun.jdbc.odbc.JdbcOdbcDriver");
txtDriver.setNextFocusableComponent(txtProtocol);
getContentPane().add(txtDriver);
txtDriver.setBounds(140, 360, 190, 20);

txtQuery.setToolTipText("SQL Query output columns must be structured the following way: Months(Months from 2000), Output in (kWh); e.g
txtQuery.setNextFocusableComponent(btnTestQuery);
scrpQuery.setViewportView(txtQuery);

getContentPane().add(scrpQuery);
scrpQuery.setBounds(140, 460, 320, 40);

lblProtocol.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblProtocol.setText("Protocol:");
getContentPane().add(lblProtocol);
lblProtocol.setBounds(70, 380, 60, 16);

txtProtocol.setToolTipText("e.g. jdbc:odbc");
txtProtocol.setNextFocusableComponent(txtUrl);
getContentPane().add(txtProtocol);
```

```
txtProtocol.setBounds(140, 380, 190, 20);

btnTestQuery.setText("Test Query");
btnTestQuery.setNextFocusableComponent(btnShowTable);
btnTestQuery.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnTestQueryActionPerformed(evt);
    }
});

getContentPane().add(btnTestQuery);
btnTestQuery.setBounds(350, 340, 110, 20);

lblNodeName.setFont(new java.awt.Font("Dialog", 1, 14));
lblNodeName.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblNodeName.setText("Node Name:");
getContentPane().add(lblNodeName);
lblNodeName.setBounds(30, 40, 100, 19);

txtNodeName.setFont(new java.awt.Font("Dialog", 0, 14));
txtNodeName.setText("Test Area");
txtNodeName.setNextFocusableComponent(txtArea);
txtNodeName.addFocusListener(new java.awt.event.FocusAdapter() {
    public void focusLost(java.awt.event.FocusEvent evt) {
        txtNodeNameFocusLost(evt);
    }
});

getContentPane().add(txtNodeName);
txtNodeName.setBounds(140, 40, 190, 23);

lblErrorStr.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblErrorStr.setText("Error String:");
getContentPane().add(lblErrorStr);
lblErrorStr.setBounds(50, 490, 80, 16);

lblUrl.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblUrl.setText("url:");
getContentPane().add(lblUrl);
lblUrl.setBounds(90, 400, 40, 16);

txtUrl.setToolTipText("e.g. TEST (ODBC) or //localhost:5432/myDB (PostgreSQL)");
txtUrl.setNextFocusableComponent(txtUser);
getContentPane().add(txtUrl);
txtUrl.setBounds(140, 400, 320, 20);

lblUser.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblUser.setText("User:");
getContentPane().add(lblUser);
lblUser.setBounds(90, 420, 40, 16);

lblPassword.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblPassword.setText("Password:");
getContentPane().add(lblPassword);
lblPassword.setBounds(60, 440, 70, 16);

txtUser.setNextFocusableComponent(txtPassword);
getContentPane().add(txtUser);
txtUser.setBounds(140, 420, 190, 20);

txtPassword.setNextFocusableComponent(txtQuery);
getContentPane().add(txtPassword);
txtPassword.setBounds(140, 440, 190, 20);

txtTicToc.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
txtTicToc.setText("0ms");
getContentPane().add(txtTicToc);
txtTicToc.setBounds(350, 420, 110, 16);

btnShowTable.setText("Show Table");
btnShowTable.setNextFocusableComponent(btnShowGraph);
btnShowTable.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnShowTableActionPerformed(evt);
    }
});

getContentPane().add(btnShowTable);
btnShowTable.setBounds(350, 360, 110, 20);
```



```
btnShowGraph.setText("Show Graph");
btnShowGraph.setNextFocusableComponent(txtNodeName);
btnShowGraph.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnShowGraphActionPerformed(evt);
    }
});

getContentPane().add(btnShowGraph);
btnShowGraph.setBounds(350, 380, 110, 20);

lblArea.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblArea.setText("Area:");
getContentPane().add(lblArea);
lblArea.setBounds(90, 70, 40, 16);

txtArea.setNextFocusableComponent(txtUnit);
getContentPane().add(txtArea);
txtArea.setBounds(140, 70, 190, 20);

lblUnit.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblUnit.setText("Unit:");
getContentPane().add(lblUnit);
lblUnit.setBounds(50, 90, 80, 16);

txtUnit.setText("per Unit");
txtUnit.setNextFocusableComponent(txtDescription);
getContentPane().add(txtUnit);
txtUnit.setBounds(140, 90, 190, 20);

lblDescription.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblDescription.setText("Query Description:");
getContentPane().add(lblDescription);
lblDescription.setBounds(0, 110, 130, 16);

txtDescription.setNextFocusableComponent(txtInputDateFormat);
getContentPane().add(txtDescription);
txtDescription.setBounds(140, 110, 320, 20);

lblMonthFormat.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblMonthFormat.setText("Date Format:");
getContentPane().add(lblMonthFormat);
lblMonthFormat.setBounds(20, 340, 110, 16);

txtInputDateFormat.setText("yyyy-MM-dd");
txtInputDateFormat.setNextFocusableComponent(txtDriver);
getContentPane().add(txtInputDateFormat);
txtInputDateFormat.setBounds(140, 340, 190, 20);

txtErrorStr.setEditable(false);
scrpErrorStr.setViewportView(txtErrorStr);

getContentPane().add(scrpErrorStr);
scrpErrorStr.setBounds(140, 500, 320, 40);

cmbFunction.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Add", "Subtract", "Multiply", "Divide" }));
cmbFunction.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        cmbFunctionActionPerformed(evt);
    }
});

getContentPane().add(cmbFunction);
cmbFunction.setBounds(140, 160, 140, 25);

lblFunction.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblFunction.setText("Select Function:");
getContentPane().add(lblFunction);
lblFunction.setBounds(30, 160, 100, 16);

btnLineAttr.setText("Line Attr.");
btnLineAttr.setBorder(new javax.swing.border.EtchedBorder());
btnLineAttr.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnLineAttrActionPerformed(evt);
    }
});

getContentPane().add(btnLineAttr);
```

```
btnLineAttr.setBounds(430, 160, 110, 20);

cmbFilter.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "None", "MovAvg" }));
getContentPane().add(cmbFilter);
cmbFilter.setBounds(140, 220, 80, 25);

jLabel1.setFont(new java.awt.Font("Dialog", 1, 14));
jLabel1.setForeground(new java.awt.Color(255, 0, 0));
jLabel1.setText("INPUT:");
getContentPane().add(jLabel1);
jLabel1.setBounds(80, 310, 50, 19);

jLabel2.setFont(new java.awt.Font("Dialog", 1, 14));
jLabel2.setForeground(new java.awt.Color(255, 0, 0));
jLabel2.setText("GENERAL:");
getContentPane().add(jLabel2);
jLabel2.setBounds(40, 10, 80, 19);

jLabel3.setFont(new java.awt.Font("Dialog", 1, 14));
jLabel3.setForeground(new java.awt.Color(255, 0, 0));
jLabel3.setText("OUTPUT:");
getContentPane().add(jLabel3);
jLabel3.setBounds(50, 140, 70, 19);

jLabel4.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel4.setText("Output Source:");
getContentPane().add(jLabel4);
jLabel4.setBounds(30, 190, 100, 16);

cmbSource.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "None", "Children", "Database", "DB-Children", "Children-DB", "
cmbSource.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        cmbSourceActionPerformed(evt);
    }
});

getContentPane().add(cmbSource);
cmbSource.setBounds(140, 190, 140, 25);

jLabel5.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel5.setText("Output Filter:");
getContentPane().add(jLabel5);
jLabel5.setBounds(40, 220, 90, 16);

lblMonthFormat1.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblMonthFormat1.setText("Date Format:");
getContentPane().add(lblMonthFormat1);
lblMonthFormat1.setBounds(40, 250, 90, 16);

txtOutputDateFormat.setText("yyyy-MM");
txtOutputDateFormat.setBorder(new javax.swing.border.EtchedBorder());
getContentPane().add(txtOutputDateFormat);
txtOutputDateFormat.setBounds(140, 250, 140, 20);

txtMovAvg.setHorizontalAlignment(javax.swing.JTextField.RIGHT);
txtMovAvg.setText("0");
txtMovAvg.setBorder(new javax.swing.border.EtchedBorder());
getContentPane().add(txtMovAvg);
txtMovAvg.setBounds(230, 220, 50, 20);

jLabel6.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel6.setText(":");
getContentPane().add(jLabel6);
jLabel6.setBounds(220, 220, 10, 16);

btnOutputGraph.setText("Show Graph");
btnOutputGraph.setBorder(new javax.swing.border.EtchedBorder());
btnOutputGraph.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnOutputGraphActionPerformed(evt);
    }
});

getContentPane().add(btnOutputGraph);
btnOutputGraph.setBounds(430, 180, 110, 20);

cmbPDF.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Beta", "Even", "Normal" }));
cmbPDF.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```



```
        cmbPDFActionPerformed(evt);
    }
});

getContentPane().add(cmbPDF);
cmbPDF.setBounds(480, 340, 160, 20);

lblMean.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblMean.setText("Mean(u):");
getContentPane().add(lblMean);
lblMean.setBounds(480, 360, 90, 16);

txtMean.setText("0.0");
getContentPane().add(txtMean);
txtMean.setBounds(580, 360, 60, 20);

lblDeviation.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblDeviation.setText("Deviation:");
getContentPane().add(lblDeviation);
lblDeviation.setBounds(480, 380, 90, 16);

txtDeviation.setText("0.0");
getContentPane().add(txtDeviation);
txtDeviation.setBounds(580, 380, 60, 20);

lblScalingFactor.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblScalingFactor.setText("Scaling Factor:");
getContentPane().add(lblScalingFactor);
lblScalingFactor.setBounds(480, 400, 90, 16);

txtScalingFactor.setText("0.0");
getContentPane().add(txtScalingFactor);
txtScalingFactor.setBounds(580, 400, 60, 20);

txtOffset.setText("0.0");
getContentPane().add(txtOffset);
txtOffset.setBounds(580, 420, 60, 20);

lblOffset.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblOffset.setText("Offset:");
getContentPane().add(lblOffset);
lblOffset.setBounds(480, 420, 90, 16);

jLabel7.setFont(new java.awt.Font("Dialog", 2, 12));
jLabel7.setForeground(new java.awt.Color(255, 0, 0));
jLabel7.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel7.setText("Random");
getContentPane().add(jLabel7);
jLabel7.setBounds(480, 320, 160, 16);

jLabel8.setFont(new java.awt.Font("Dialog", 2, 12));
jLabel8.setForeground(new java.awt.Color(255, 0, 0));
jLabel8.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel8.setText("Database");
getContentPane().add(jLabel8);
jLabel8.setBounds(60, 320, 400, 16);

getContentPane().add(jTextArea1);
jTextArea1.setBounds(530, 530, 0, 16);

txtStartTime.setText("2003-01");
txtStartTime.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        txtStartTimeActionPerformed(evt);
    }
});

getContentPane().add(txtStartTime);
txtStartTime.setBounds(480, 460, 160, 20);

btnStartTime.setText("Start Time:");
btnStartTime.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnStartTimeActionPerformed(evt);
    }
});

getContentPane().add(btnStartTime);
btnStartTime.setBounds(480, 440, 100, 20);
```

```
btnStopTime.setText("Stop Time:");
btnStopTime.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnStopTimeActionPerformed(evt);
    }
});

getContentPane().add(btnStopTime);
btnStopTime.setBounds(480, 480, 100, 20);

txtStopTime.setText("2003-12");
getContentPane().add(txtStopTime);
txtStopTime.setBounds(480, 500, 160, 20);

lblAveragePeriod.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblAveragePeriod.setText("Average Period:");
getContentPane().add(lblAveragePeriod);
lblAveragePeriod.setBounds(30, 270, 100, 16);

txtAveragePeriod.setText("0");
getContentPane().add(txtAveragePeriod);
txtAveragePeriod.setBounds(140, 270, 40, 20);

txtHystMin.setText("0.0");
getContentPane().add(txtHystMin);
txtHystMin.setBounds(330, 230, 90, 20);

lblHystMin.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblHystMin.setText("Min:");
getContentPane().add(lblHystMin);
lblHystMin.setBounds(300, 230, 30, 16);

lblHystMax.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
lblHystMax.setText("Max:");
getContentPane().add(lblHystMax);
lblHystMax.setBounds(300, 250, 30, 16);

txtHystMax.setText("100.0");
getContentPane().add(txtHystMax);
txtHystMax.setBounds(330, 250, 90, 20);

cmbHystogram.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "None", "Children", "Database", "DB-Children", "Children-DB"
cmbHystogram.setToolTipText("Determines how the hystogram is built up.");
getContentPane().add(cmbHystogram);
cmbHystogram.setBounds(300, 270, 120, 25);

jLabel9.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
jLabel9.setText("Histogram:");
getContentPane().add(jLabel9);
jLabel9.setBounds(230, 280, 62, 16);

btnResetHystogram.setText("Reset Histogram");
btnResetHystogram.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnResetHystogramActionPerformed(evt);
    }
});

getContentPane().add(btnResetHystogram);
btnResetHystogram.setBounds(420, 270, 129, 26);

btnUpdate.setText("Always Update");
getContentPane().add(btnUpdate);
btnUpdate.setBounds(350, 440, 109, 20);

rdbDifferentiate.setText("Differentiate");
bgDiffInt.add(rdbDifferentiate);
getContentPane().add(rdbDifferentiate);
rdbDifferentiate.setBounds(310, 150, 95, 24);

rdbIntegrate.setText("Integrate");
bgDiffInt.add(rdbIntegrate);
getContentPane().add(rdbIntegrate);
rdbIntegrate.setBounds(310, 170, 76, 24);

rdbNone.setText("None");
bgDiffInt.add(rdbNone);
getContentPane().add(rdbNone);
rdbNone.setBounds(310, 190, 104, 24);
```

```
    pack();
}//GEN-END:initComponents

private void txtNodeNameFocusLost(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_txtNodeNameFocusLost
    this.setTitle(txtNodeName.getText());
}//GEN-LAST:event_txtNodeNameFocusLost

private void cmbPDFActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_cmbPDFActionPerformed
    // Add your handling code here:
}//GEN-LAST:event_cmbPDFActionPerformed

private void btnResetHystogramActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnResetHystogramActionPerformed
   hyst = new hystogram();        // Add your handling code here:
}//GEN-LAST:event_btnResetHystogramActionPerformed

private void cmbSourceActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_cmbSourceActionPerformed
    // Add your handling code here:
}//GEN-LAST:event_cmbSourceActionPerformed

private void btnStopTimeActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnStopTimeActionPerformed
    d.print(3,"*btnStopTimeActionPerformed...");
    GeoDateDialog tmpGDD = new GeoDateDialog(getStopTime());
    tmpGDD.setHideTime(true);
    tmpGDD.showDialog(tmpGDD.getDate(),1,1);
    setStopTime(tmpGDD.getDate());
}//GEN-LAST:event_btnStopTimeActionPerformed

private void btnStartTimeActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnStartTimeActionPerformed
    d.print(3,"*btnStartTimeActionPerformed...");
    GeoDateDialog tmpGDD = new GeoDateDialog(getStartTime());
    tmpGDD.setHideTime(true);
    tmpGDD.showDialog(tmpGDD.getDate(),1,1);
    setStartTime(tmpGDD.getDate());
}//GEN-LAST:event_btnStartTimeActionPerformed

private void txtStartTimeActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_txtStartTimeActionPerformed

}//GEN-LAST:event_txtStartTimeActionPerformed

private void btnOutputGraphActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnOutputGraphActionPerformed
    d.print(3,"*btnOutputGraphActionPerformed...");
    final int MARGIN = 50;
    SimpleLine sL = toDataCollection().toSimpleLine();
    if (sL!=null) {
        GraphDisplay sLPlot = new GraphDisplay();
        sLPlot.setTitles(txtNodeName.getText(),txtArea.getText(),txtDescription.getText());
        sLPlot.addData(sL,txtDescription.getText());
        JFrame myTempFrame = new JFrame();
        myTempFrame.setContentPane(sLPlot);
        myTempFrame.setBounds(MARGIN,MARGIN,GraphicsEnvironment.getLocalGraphicsEnvironment().getMaximumWindowBounds().width-(2*MARGIN),Gr
        myTempFrame.show();
       // this.setTitle(this.toString());
    } else {
        txtErrorStr.setText("Graph: Data set incorrect for Simple Line Plot");
    }
}//GEN-LAST:event_btnOutputGraphActionPerformed

private void btnLineAttrActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnLineAttrActionPerformed
    d.print(3,"*btnLineAttrActionPerformed...");
    // Add your handling code here:
    LineAttribute lnAttr = getLineAttribute();
    LineAttributeDialog sgtLineAttributeDialog = new LineAttributeDialog();
    sgtLineAttributeDialog.setLineAttribute(lnAttr);
    sgtLineAttributeDialog.show();
    setLineAttribute(lnAttr);
}//GEN-LAST:event_btnLineAttrActionPerformed

private void cmbFunctionActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_cmbFunctionActionPerformed
}//GEN-LAST:event_cmbFunctionActionPerformed

private void btnShowGraphActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnShowGraphActionPerformed
    // Add your handling code here:
    d.print(3,"*btnShowGraphActionPerformed...");
    final int MARGIN = 50;
    SimpleLine sL = getDBDataCollection().toSimpleLine();
    if (sL!=null) {
        GraphDisplay sLPlot = new GraphDisplay();
        sLPlot.setTitles(txtNodeName.getText(),txtArea.getText(),txtDescription.getText());
        sLPlot.addData(sL,txtDescription.getText());
```

```
        JFrame myTempFrame = new JFrame();
        myTempFrame.setContentPane(sLPlot);
        myTempFrame.setBounds(MARGIN,MARGIN,GraphicsEnvironment.getLocalGraphicsEnvironment().getMaximumWindowBounds().width-(2*MARGIN),Gr
        myTempFrame.show();
    } else {
        txtErrorStr.setText("Graph: Data set incorrect for Simple Line Plot");
    }
}//GEN-LAST:event_btnShowGraphActionPerformed

private void btnShowTableActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnShowTableActionPerformed
    // Add your handling code here:
    d.print(3,"*btnShowTableActionPerformed...");
    if (myDB.isQuerySuccess()) {
      JFrame myTempFrame = new JFrame();
      myTempFrame.getContentPane().add(new TableDisplay(myDB));
      myTempFrame.setBounds(10,10,300,450);
      myTempFrame.show();
    }
}//GEN-LAST:event_btnShowTableActionPerformed

private void formWindowOpened(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_formWindowOpened
}//GEN-LAST:event_formWindowOpened

private void btnTestQueryActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event_btnTestQueryActionPerformed
    d.print(3,"btnTestQueryActionPerformed...");
    //Test the connection to the Database
    getDataBaseData();
    txtTicToc.setText(getDataBaseAccessTime());
    txtErrorStr.setText(getDataBaseErrorString());
}//GEN-LAST:event_btnTestQueryActionPerformed

/** Exit the Application */
private void exitForm(java.awt.event.WindowEvent evt) {//GEN-FIRST:event_exitForm
   d.print(3,"*exitForm...");
   if (mayCloseSystem) {
       System.exit(0);
   }
}//GEN-LAST:event_exitForm

/** Initiates the myNode class.
 * @param args the command line arguments
 */
public static void main(String args[]) {
    mayCloseSystem = true;
    new myNode().show();
}
```


```
// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.ButtonGroup bgDiffInt;
private javax.swing.JButton btnLineAttr;
private javax.swing.JButton btnOutputGraph;
private javax.swing.JButton btnResetHystogram;
private javax.swing.JButton btnShowGraph;
private javax.swing.JButton btnShowTable;
private javax.swing.JButton btnStartTime;
private javax.swing.JButton btnStopTime;
private javax.swing.JButton btnTestQuery;
private javax.swing.JCheckBox btnUpdate;
private javax.swing.JComboBox cmbFilter;
private javax.swing.JComboBox cmbFunction;
private javax.swing.JComboBox cmbHystogram;
private javax.swing.JComboBox cmbPDF;
private javax.swing.JComboBox cmbSource;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JPopupMenu jPopupMenu1;
private javax.swing.JTextArea jTextArea1;
private javax.swing.JLabel lblArea;
private javax.swing.JLabel lblAveragePeriod;
private javax.swing.JLabel lblDescription;
```

```
    private javax.swing.JLabel lblDeviation;
    private javax.swing.JLabel lblDriver;
    private javax.swing.JLabel lblErrorStr;
    private javax.swing.JLabel lblFunction;
    private javax.swing.JLabel lblHystMax;
    private javax.swing.JLabel lblHystMin;
    private javax.swing.JLabel lblMean;
    private javax.swing.JLabel lblMonthFormat;
    private javax.swing.JLabel lblMonthFormat1;
    private javax.swing.JLabel lblNodeName;
    private javax.swing.JLabel lblOffset;
    private javax.swing.JLabel lblPassword;
    private javax.swing.JLabel lblProtocol;
    private javax.swing.JLabel lblSQL;
    private javax.swing.JLabel lblScalingFactor;
    private javax.swing.JLabel lblUnit;
    private javax.swing.JLabel lblUrl;
    private javax.swing.JLabel lblUser;
    private javax.swing.JRadioButton rdbDifferentiate;
    private javax.swing.JRadioButton rdbIntegrate;
    private javax.swing.JRadioButton rdbNone;
    private javax.swing.JScrollPane scrpErrorStr;
    private javax.swing.JScrollPane scrpQuery;
    private javax.swing.JTextField txtArea;
    private javax.swing.JTextField txtAveragePeriod;
    private javax.swing.JTextField txtDescription;
    private javax.swing.JTextField txtDeviation;
    private javax.swing.JTextField txtDriver;
    private javax.swing.JTextArea txtErrorStr;
    private javax.swing.JTextField txtHystMax;
    private javax.swing.JTextField txtHystMin;
    private javax.swing.JTextField txtInputDateFormat;
    private javax.swing.JTextField txtMean;
    private javax.swing.JTextField txtMovAvg;
    private javax.swing.JTextField txtNodeName;
    private javax.swing.JTextField txtOffset;
    private javax.swing.JTextField txtOutputDateFormat;
    private javax.swing.JPasswordField txtPassword;
    private javax.swing.JTextField txtProtocol;
    private javax.swing.JTextArea txtQuery;
    private javax.swing.JTextField txtScalingFactor;
    private javax.swing.JTextField txtStartTime;
    private javax.swing.JTextField txtStopTime;
    private javax.swing.JLabel txtTicToc;
    private javax.swing.JTextField txtUnit;
    private javax.swing.JTextField txtUrl;
    private javax.swing.JTextField txtUser;
    // End of variables declaration//GEN-END:variables
    private LineAttribute lineAttr = new LineAttribute();
    private DBConnect myDB = new DBConnect();
    private SimpleLine sL = null;
    private TicToc t = new TicToc();
    private dataCollection childDC = null;
    //private LineAttribute myLineAttribute = new LineAttribute();

//-Bean Paterns-----------------------------------------------------------------------------
    /** Sets the nodename of the node
     * @param text Node name of to be given to the Node.
     */
    public void setNodeName(String text) {
        txtNodeName.setText(text);
        setTitle(txtNodeName.getText());
    }
    /** Gets the name of the node
     * @return The nodename
     */
    public String getNodeName() {
        return txtNodeName.getText();
    }

    /** Gives a name to the area.
     * @param text The area name of this node
     */
    public void setArea(String text) {
        txtArea.setText(text);
    };
    /** Retrieves the Area name of the node
     * @return the area name of the node
     */
```

```
public String getArea() {
    return txtArea.getText();
}

/** Sets the measurement unit of the node.
 * @param text The unit as defined by the user of the Node
 */
public void setUnit(String text){
    txtUnit.setText(text);
}
/** Retrieves the unit of measurement of the node.
 * @return the measurement unit
 */
public String getUnit() {
    return txtUnit.getText();
}

/** Set the description of the node
 * @param text The description
 */
public void setDescription(String text) {
    txtDescription.setText(text);
}
/** Retrieves the description of the node
 * @return The description of the node
 */
public String getDescription() {
    return txtDescription.getText();
}

/** Sets the input date format
 * The output date format is used to determine the output resolution of dates used
 * in the dataCollection class.
 * @param text The date format e.g. yyyy-MM-dd
 */
public void setOutputDateFormat(String text) {
    txtOutputDateFormat.setText(text);
}
/** Retrieves the output date format of node
 * @return The date format
 */
public String getOutputDateFormat() {
    return txtOutputDateFormat.getText();
}

/** Sets the input date format
 * The output date format is used to determine the output resolution of dates used
 * in the dataCollection class.
 * @param text The date format e.g. yyyy-MM-dd
 */
public void setInputDateFormat(String text) {
    txtInputDateFormat.setText(text);
}
/** Retrieves the input date format of node
 * @return The date format
 */
public String getInputDateFormat() {
    return txtInputDateFormat.getText();
}

/** Sets the JDBC driver for the database connectino associated with the node
 * @param text the database driver e.g. org.postgresql.com
 */
public void setDriver(String text) {
    txtDriver.setText(text);
}
/** Retrieves the database driver.
 * @return The database driver
 */
public String getDriver() {
    return txtDriver.getText();
}

/** Set the protocol to connect to the database
 * @param text the protocol e.g. jdbc:mysql
 */
public void setProtocol(String text) {
    txtProtocol.setText(text);
}
```

```
/** Retrieves the protocol used for this node's database connection
 * @return The database protocol.
 */
public String getProtocol() {
    return txtProtocol.getText();
}

/** Set the url to connect to the database
 * @param text The connection to the database e.g. //localhost:5432
 */
public void setUrl(String text) {
    txtUrl.setText(text);
}
/** Retrieves the url to the database.
 * @return The database url link to the database of this node
 */
public String getUrl() {
    return txtUrl.getText();
}

/** Set the username to connect to the database
 * @param text The username
 */
public void setUser(String text) {
    txtUser.setText(text);
}
/** Retrieves the user
 * @return The user currently being used for the dataconnection for this node
 */
public String getUser() {
    return txtUser.getText();
}

/** Sets the password used to access the database connection of this node
 * @param text the password
 */
public void setPassword(String text) {
    txtPassword.setText(text);
}
/** Returns the password stored in teh node
 * @return the password
 */
public String getPassword() {
    return new String(txtPassword.getPassword());
}

/** Sets the query to retrieve the nodes data collection
 * @param text the SQL query for the data in this node's database
 */
public void setQuery(String text) {
    txtQuery.setText(text);
}
/** Retrieves the query statement associated with this node.
 * @return the SQL statement
 */
public String getQuery() {
    return txtQuery.getText();
}

public void setUpdate(boolean tmpB) {
    btnUpdate.setSelected(tmpB);
}
public boolean getUpdate() {
    return btnUpdate.isSelected();
}
/** Sets the function associated with the node e.g. add/substract/multiplication or
 * division
 * @param i An integer associated with the constants FUNCTION_ADD...
 */
public void setFunction(int i) {
    cmbFunction.setSelectedIndex(i);
}
/** Retrieves the function used by the node
 * @return the represented function integer value assosicate with the FUNCTION_ADD...
 */
public int getFunction() {
    return cmbFunction.getSelectedIndex();
}
```

```
/** Stores the line attribute associated with the node, and used when plotting the
 * graphs.
 * @param tmpLineAttr The LineAttribute as specified by SGT
 */
public void setLineAttribute(LineAttribute tmpLineAttr) {
      lineAttr = tmpLineAttr;
}
/** Retrieves the associated Line attribute of the node
 * @return the lineAttribue properties
 */
public LineAttribute getLineAttribute() {
    return lineAttr;
}

/** Sets the source to be used for output e.g. database or children
 * @param i The output as defined by the SOURCE constants
 */
public void setOutputSource(int i) {
    cmbSource.setSelectedIndex(i);
}
/** Retrieves the output source specified for the node
 * @return The output source
 */
public int getOutputSource() {
    return cmbSource.getSelectedIndex();
}

/** Specifies the filter to be used for the output
 * @param i The output filter as specified by the FILTER constants
 */
public void setOutputFilter(int i) {
    cmbFilter.setSelectedIndex(i);
}
/** Retrieves teh associated output filter for this node
 * @return The output filter as specified by the FILTER contstants
 */
public int getOutputFilter() {
    return cmbFilter.getSelectedIndex();
}

/** Sets the amount of samles to use for the moving average filter of this node
 * @param tmpMovAvg Number of samples of the output resolution
 */
public void setMovAvgSamples(int tmpMovAvg) {
    txtMovAvg.setText(new BigInteger(""+tmpMovAvg).toString());
}
/** Retrieves the samples to be used for each moving average
 * @return The moving average filter samples
 */
public int getMovAvgSamples() {
    return new BigInteger(""+txtMovAvg.getText()).intValue();
}

//The following four methods is used to implement the integration and
// differentiation options in the node's output.
// These are used in the toDataCollection method.
/** Sets the status of the differentiation process.
 * @param tmpB Status of differentiation
 */
public void setDifferentiate(boolean tmpB) {
    rdbDifferentiate.setSelected(tmpB);
}
/** Returns the status of this object differentiation
 * @return the differentiation status
 */
public boolean isDifferentiate() {
    return rdbDifferentiate.isSelected();
}

/** Sets the status of the integration process.
 * @param tmpB Status of integration
 */
public void setIntegrate(boolean tmpB) {
    rdbIntegrate.setSelected(tmpB);
}
/** Returns the status of this object integration
 * @return the integration status
 */
public boolean isIntegrate() {
```

```
    return rdbIntegrate.isSelected();
}

/** Stores the dataCollection of the children functions.
 * The node cannot access its own childrens. This function is therfore used by the
 * tree function to store the nodes children data.
 * @param tmpDC Data collection output of all the children functions of this node.
 */
public void setChildDataCollection(dataCollection tmpDC) {
    childDC = tmpDC;
}
/** Retrievs the nodes children datacollection as stored by the tree using this
 * node.
 * @return the children datacollection
 */
public dataCollection getChildDataCollection() {
    return childDC;
}

//Random engine
// The random engine is used to aid with simulation outputs.
/* Sets the mean of
 * @param d the mean to be used
 */
public void setMean(double d) {
    txtMean.setText(""+d);
}
/** Returns the mean stored in the object.
 * @return the mean
 */
public double getMean() {
    return new BigDecimal(txtMean.getText()).doubleValue();
}

public void setDeviation(double d) {
    txtDeviation.setText(""+d);
}
public double getDeviation() {
    return new BigDecimal(txtDeviation.getText()).doubleValue();
}

public void setScalingFactor(double d) {
    txtScalingFactor.setText(""+d);
}
public double getScalingFactor() {
    return new BigDecimal(txtScalingFactor.getText()).doubleValue();
}

public void setOffset(double d) {
    txtOffset.setText(""+d);
}
public double getOffset() {
    return new BigDecimal(txtOffset.getText()).doubleValue();
}

public void setStartTime(GeoDate d) {
    txtStartTime.setText(d.toString(txtOutputDateFormat.getText()));
}
public GeoDate getStartTime() {
    try {
        return new GeoDate(txtStartTime.getText(),txtOutputDateFormat.getText());
    } catch (IllegalTimeValue e) {
        return null;
    }
}

public void setStopTime(GeoDate d) {
    txtStopTime.setText(d.toString(txtOutputDateFormat.getText()));
}
public GeoDate getStopTime() {
    try {
        return new GeoDate(txtStopTime.getText(),txtOutputDateFormat.getText());
    } catch (IllegalTimeValue e) {
        return null;
    }
}

public void setRandomEngine(int e) {
    cmbPDF.setSelectedIndex(e);
```

```
    }
    public int getRandomEngine() {
        return cmbPDF.getSelectedIndex();
    }

    //Averaging Period for to String function and output.

    public void setAveragePeriod(int period) {
        txtAveragePeriod.setText(""+period);
    }
    public int getAveragePeriod() {
        return new BigInteger(txtAveragePeriod.getText()).intValue();
    }

    // Hystogram Details
    public hystogram getHystogram() {
        return hyst;
    }

    public void setHystMin(double min) {
        txtHystMin.setText(""+min);
    }
    public double getHystMin() {
        return new BigDecimal(txtHystMin.getText()).doubleValue();
    }

    public void setHystMax(double max) {
        txtHystMax.setText(""+max);
    }
    public double getHystMax() {
        return new BigDecimal(txtHystMax.getText()).doubleValue();
    }

    /** Sets the source to be used for hystogram generation e.g. database or children
     * @param i The output as defined by the SOURCE constants
     */
    public void setHystogramSource(int i) {
        cmbHystogram.setSelectedIndex(i);
    }
    /** Retrieves the hystogram generation source specified for the node
     * @return The output source
     */
    public int getHystogramSource() {
        return cmbHystogram.getSelectedIndex();
    }

//--Outputs------------------------------------------------------------------------------------------------
    /** Converst the node to a human readible form
     * @return The string containing human readible data concerning the node
     */
    public String toString() {
        d.print(3,"toString...");
        String tmpStr = "";
        java.math.BigDecimal bdDataCollection = null;
        java.math.BigDecimal bdChildDataCollection = null;
        if ((toDataCollection() != null) &  (getChildDataCollection() != null)) {
            bdDataCollection = new java.math.BigDecimal(toDataCollection().avg()); //getAveragePeriod()
            bdChildDataCollection = new java.math.BigDecimal(getChildDataCollection().avg()); //getAveragePeriod()
            if (getOutputSource() == SOURCE_CHILDREN_DB) {
                if (bdChildDataCollection != new java.math.BigDecimal(0.0)) {
                    tmpStr = "["+bdDataCollection.multiply(new java.math.BigDecimal(100.0)).divide(bdChildDataCollection,2,java.math.BigDecima
                } else {
                    tmpStr = "[DivZero%]";
                }
            } else {
                if ((bdDataCollection.compareTo(new java.math.BigDecimal(0.0)) != 0) && (bdDataCollection != null)) {
                  tmpStr = "["+bdChildDataCollection.multiply(new java.math.BigDecimal(100.0)).divide(bdDataCollection,2,java.math.BigDecimal.
                } else {
                    tmpStr = "[DivZero%]";
                }
            }
        } else {
            tmpStr = "[null%]";
        }

        if (toDataCollection() != null) {
            tmpStr = tmpStr + "("+(new java.math.BigDecimal(toDataCollection().avg(getAveragePeriod()))).setScale(2,java.math.BigDecimal.ROUND
        } else {
            tmpStr = tmpStr + "(null:";
```

```
        }
        if (getChildDataCollection() != null) {
            tmpStr = tmpStr + (new java.math.BigDecimal(getChildDataCollection().avg(getAveragePeriod()))).setScale(2,java.math.BigDecimal.ROU
        } else {
            tmpStr = tmpStr + "null) ";
        }
        tmpStr = (tmpStr + ("["+getUnit()+"]"+getNodeName() + ":" + getArea() + ":" + getDescription()));
        d.print(3,"toString:["+tmpStr+"]");
        return tmpStr;
    }

    //DBase...
    /** Retrieves the last access time to the database
     * @return The access time to the database in milliseconds.
     */
    public String getDataBaseAccessTime() {
        return t.toString();
    }

    /** If an error occurred with accessing the database, this method provides the error
     * message.
     * @return The error string associated with the database error
     */
    public String getDataBaseErrorString() {
        return myDB.getErrorString();
    }

    /** Retrieves the latest available data from the database associated with this node.
     * @return The data contained in the database statement.
     */
    public DBConnect getDataBaseData(){

        t.tic();
        myDB = new DBConnect(txtDriver.getText(),txtProtocol.getText(),txtUrl.getText(),txtUser.getText(),txtPassword.toString());
        myDB.setDriver(txtDriver.getText());
        myDB.setProtocol(txtProtocol.getText());
        myDB.setSubname(txtUrl.getText());
        myDB.setUser(txtUser.getText());
        myDB.setPassword(new String(txtPassword.getPassword()));
        myDB.setQuery(txtQuery.getText());
        myDB.executeSQL(txtQuery.getText());
        t.toc();

        return myDB;
    }

    /** Retrieves the latest database information in the simpleLine dataclass.
     * @return The data contained in the database in SimpleLine "format"
     */
    public SimpleLine toSimpleLine() {
        d.print(3,"toSimpleLine...");
        sL = toDataCollection().toSimpleLine();
        if (sL != null) {
            sL.setXMetaData(new SGTMetaData("Time","Months",false,false));
            sL.setYMetaData(new SGTMetaData(txtDescription.getText(),txtUnit.getText(),false,false));
        }
        return sL;
    }

    /** Retrieve the data contained in the database in dataCollection format
     * @return database data associated with the node
     */
   public dataCollection getDBDataCollection() {
        dataCollection tmpDC = null;
        if ((!myDB.isQuerySuccess()) || (btnUpdate.isSelected())) {
            myDB = getDataBaseData();
        }
        if (myDB.isQuerySuccess()) {
            tmpDC = new dataCollection(myDB,getInputDateFormat());
            tmpDC.setOutputDateFormat(getOutputDateFormat());
        }
        if (tmpDC != null) {
            return tmpDC;//.group();
        } else {
            return null;
        }
    }

   /** Outputs the data as specified by the node.
```

```
*
* applied filters,
* sorted,
* grouped,
* source selection.
* @return the data as presented by the node
*/
public dataCollection toDataCollection() {
    d.print(3,"toDataCollection:"+getOutputSource());
   // dataCollection tmpMyDbDC = null;
    dataCollection tmpDC = null;
    //Source Selection output
    if (getOutputSource() == SOURCE_NONE) {
        d.print(5,"toDataCollection:SOURCE_NONE");
        tmpDC = null;
    } else if (getOutputSource() == SOURCE_DATABASE) {
        d.print(5,"toDataCollection:SOURCE_DATABASE");
        tmpDC = getDBDataCollection();
    } else if (getOutputSource() == SOURCE_CHILDREN) {
        d.print(5,"toDataCollection:SOURCE_CHILDREN");
        tmpDC = getChildDataCollection();
    } else if (getOutputSource() == SOURCE_DB_CHILDREN) {
        d.print(5,"toDataCollection:SOURCE_DB_CHILDREN");
        if (!myDB.isQuerySuccess()) {
            myDB = getDataBaseData();
        }
        if (myDB.isQuerySuccess()) {
            tmpDC = new dataCollection(myDB,getInputDateFormat());
            if (tmpDC != null) {
                tmpDC.setOutputDateFormat(getOutputDateFormat());
            }
        }
        if (tmpDC == null) {
            d.print(7,"tmpDC=null");
            tmpDC = new dataCollection().math(getChildDataCollection(),new dataCollection().MATH_SUB);
            d.print(7,""+tmpDC);
        } else {
            tmpDC = tmpDC.math(getChildDataCollection(),new dataCollection().MATH_SUB);
        }
    } else if (getOutputSource() == SOURCE_CHILDREN_DB) {
        d.print(5,"toDataCollection:SOURCE_CHILDREN_DB");
        if (!myDB.isQuerySuccess()) {
            myDB = getDataBaseData();
        }
        if (myDB.isQuerySuccess()) {
            tmpDC = new dataCollection(myDB,getInputDateFormat());
            if (tmpDC != null) {
                tmpDC.setOutputDateFormat(getOutputDateFormat());
            }
        }
        if (tmpDC == null) {
            d.print(7,"tmpDC=null");
            tmpDC = new dataCollection().math(getChildDataCollection(),new dataCollection().MATH_SUB);
            d.print(7,""+tmpDC);
        } else {
            tmpDC = tmpDC.math(getChildDataCollection(),new dataCollection().MATH_SUB_NEG);
        }
    } else if (getOutputSource() == SOURCE_RANDOM) {
        d.print(5,"toDataCollection:SOURCE_RANDOM");
        try {
            double mu = getMean();
            d.print(6,"toDataCollection:mu="+mu);
            double dev = getDeviation();
            d.print(6,"toDataCollection:dev="+dev);
            double variance = dev*dev;
            d.print(6,"toDataCollection:var="+variance);
            GeoDate tmpDateNow = new GeoDate(txtStartTime.getText(), txtOutputDateFormat.getText());
            GeoDate tmpEndDate = new GeoDate(txtStopTime.getText(), txtOutputDateFormat.getText());
            BetaDistribution bD = null;
            ContinuousUniformDistribution cD = null;
            NormalDistribution nD = null;
            if (getRandomEngine() == RANDOM_BETA) {
                double alpha = mu*((getScalingFactor()*mu)-(mu*mu)-variance)/(getScalingFactor()*variance);
                double beta = (getScalingFactor()-mu)*((getScalingFactor()*mu)-(mu*mu)-variance)/(getScalingFactor()*variance);
                d.print(6,"toDataCollection:alpha="+alpha);
                d.print(6,"toDataCollection:beta="+beta);
                bD = new BetaDistribution(alpha,beta);
            } else if (getRandomEngine() == RANDOM_EVEN) {
                double left = mu-dev;
                double right = mu+dev;
```

```
                    cD = new ContinuousUniformDistribution(left,right);
                } else if (getRandomEngine() == RANDOM_NORMAL) {
                    nD = new NormalDistribution(mu, dev);
                }
                double tmpValue = 0.0;
                if ((tmpDateNow != null) & (tmpEndDate != null)) {
                    tmpDC = new dataCollection();
                    tmpDC.setOutputDateFormat(getOutputDateFormat());
                    tmpDC.setInputDateFormat(getOutputDateFormat());
                    do {
                        if (getRandomEngine() == RANDOM_BETA) {
                          tmpValue = bD.simulate()*getScalingFactor();
                        } else if (getRandomEngine() == RANDOM_EVEN){
                            tmpValue = cD.simulate()*getScalingFactor();
                        } else if (getRandomEngine() == RANDOM_NORMAL){
                            tmpValue = nD.simulate()*getScalingFactor();
                        }
                        tmpDC.add(new dataPoint(new GeoDate(tmpDateNow),(tmpValue)+getOffset()));
                        tmpDateNow.increment(1.0, tmpDateNow.MONTHS);
                    } while (tmpDateNow.compareTo(tmpEndDate) <= 0);
                } else {
                    tmpDC = null;
                }
            } catch (IllegalTimeValue e) {
                tmpDC = null;
            }
        } else {
            tmpDC = null;
        }
        //Integrate and differentiated output
        if (tmpDC != null) {
            if (rdbDifferentiate.isSelected()) {
                d.print(5,"toDataCollection:differentiate...");
                tmpDC = tmpDC.differentiate();
            } else if (rdbIntegrate.isSelected()) {
                d.print(5,"toDataCollection:integrate...");
                tmpDC = tmpDC.integrate();
            }
        }

        //Filter output
        if (tmpDC != null) {
            if (getOutputFilter() == FILTER_NONE) {
                tmpDC = tmpDC;
            } else if (getOutputFilter() == FILTER_MOVAVG) {
                tmpDC = tmpDC.corr(new dataCollection().movAvg(getMovAvgSamples()));
            }
            if (tmpDC != null) {
              tmpDC = tmpDC.group(getOutputDateFormat());
            }
        }
        //Histogram
        if (tmpDC == null) {
            d.print(5,"return=Null");
        } else {
            if (this.getHystogramSource() != SOURCE_NONE) {
              hyst.setMax(this.getHystMax());
              hyst.setMin(this.getHystMin());
              hyst.add(tmpDC.avg(getAveragePeriod()));
            }
        }

        if (tmpDC != null) {
            return tmpDC.group();
        } else {
            return tmpDC;
        }
    }

/*--Implements the Abstract Table Model-------------------------------------------------------*/

    public AbstractTableModel toTableModel() {
        AbstractTableModel thisTable = new AbstractTableModel() {
            public String getColumnName(int col) {
                if (col == 0) {
                    return new String("Date/Time");
                } else {
                    return new String("Data");
                }
```

```
            }
            public int getRowCount() {
                return toDataCollection().getElementCount();
            }
            public int getColumnCount() {
                return 2;
            }
            public Object getValueAt(int row, int col) {
                if (col == 0) {
                    return toDataCollection().get(row).getDate();
                } else {
                    return (""+toDataCollection().get(row).getValue());
                }
            }
            public boolean isCellEditable(int row, int col) {
                return true;
            }
            public void setValueAt(Object value, int row, int col) {
            }
        };
        return thisTable;
    }
}
```

Q.7.3 Object: `peg.swing.progressBar`

```
/*
 * progressBar.java
 *
 * Created on 21 June 2004, 09:37
 */

package peg.swing;

/** The progressBar class allows easy generation of a progress bar. This
 * progressbar will not display the frames decorations.
 * @author RDoorduin
 */
public class progressBar extends javax.swing.JFrame{
    private javax.swing.JProgressBar pb;
    private String title = "";
    /** Creates a new instance of progressBar
     * @param title the title of the progressbar
     * @param min the minimum value
     * @param max tha maximum value
     */
    public progressBar(String title, int min, int max) {
        this.title = title;
        int maxHeight = java.awt.GraphicsEnvironment.getLocalGraphicsEnvironment().getMaximumWindowBounds().height;
        int maxWidth = java.awt.GraphicsEnvironment.getLocalGraphicsEnvironment().getMaximumWindowBounds().width;
        int sizeHeight = 24;
        int sizeWidth = maxWidth/2;
        int y = (maxHeight-sizeHeight)/2;
        int x = (maxWidth-sizeWidth)/2;
        this.setTitle("0%"+title);
        //javax.swing.JFrame thisFrame = new javax.swing.JFrame("palmHHT Synchronization");
        this.setUndecorated(true);
        pb = new javax.swing.JProgressBar();
        this.getContentPane().add(pb);
        pb.setStringPainted(true);
        pb.setString("");
        this.setMin(min);
        this.setMax(max);
        this.setBounds(x,y,sizeWidth,sizeHeight);
        this.show();
    }

    /** Creates a new instance of progressBar
     * @param title the title of the progressbar
     */
    public progressBar(String title) {
        this(title,0,100);
    }

    /** Sets the minimum value fo the progressbar.
     * @param min the minimum value
     */
```

```
    public void setMin(int min) {
        pb.setMinimum(min);
    }

    /** Sets the minimum value fo the progressbar.
     * @param max the maximum value
     */
    public void setMax(int max) {
        pb.setMaximum(max);
    }

    /** Increments the progress bar with one unit. */
    public void incValue() {
        pb.setValue(pb.getValue()+1);
        this.setTitle(""+(pb.getPercentComplete()*100)+"% - "+title);
        super.repaint();
    }
    /** Sets the specified value which is represented by the progress bar.
     * @param value the value
     */
    public void setValue(int value) {
        pb.setValue(value);
        this.setTitle(""+(pb.getPercentComplete()*100)+"% - "+title);
        super.repaint();
    }
    /** Sets the specified value which is represented by the progress bar and allows the
     * user to specify a string to be displayed over the progress. The string indicates
     * which process is busy executing, or the phase of excution.
     * @param value the value
     * @param text the string of text to be displayed while the progress bar is running
     */
    public void setValue(int value, String text) {
        this.setValue(value);
        pb.setString(text);
    }
}
```

Q.7.4 Object: `peg.swing.TableDisplay`

```
package peg.swing;

/*
 * TableDisplay.java
 *
 * Created on November 12, 2003, 11:40 AM
 */

/** This class forms the interface with the DBConnect class and the JForm class. The
 * DBConnect class extends the AbstractTableModel class, which can interface with
 * the JTable class.
 *
 *  <PRE>
 *  (TableModel) --> (GUI)
 *  </PRE>
 *
 * An implemenation example of its use is shown below.
 *
 * <PRE>
 * if (myDB.isQuerySuccess()) {
 *  JFrame myTempFrame = new JFrame();
 *  myTempFrame.getContentPane().add(new TableDisplay(myDB));
 *  myTempFrame.setBounds(10,10,300,450);
 *  myTempFrame.show();
 * }
 * </PRE>
 *
 * By using this class in your software you can present the DBConnect class
 * uniformly throughout the application.
 * @author Riaan Doorduin
 */
public class TableDisplay extends javax.swing.JPanel {

    /** Creates new form TableDisplay */
    public TableDisplay() {
        initComponents();
    }
```

```
    /** Creates new form TableDisplay directly with the inteded DBConnect class.
     * @param myDB The current contents of the this DBConnect class.
     */
    public TableDisplay(peg.sql.DBConnect myDB) {
        this(); //initiate the class
        setDB(myDB); //sets the database
        displayTable(); //set up all the GUI components and displays the contents
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    private void initComponents() {//GEN-BEGIN:initComponents
        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();

        setLayout(new java.awt.BorderLayout());

        jTable1.setModel(new javax.swing.table.DefaultTableModel(
            new Object [][] {
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null},
                {null, null, null, null}
            },
            new String [] {
                "Title 1", "Title 2", "Title 3", "Title 4"
            }
        ));
        jScrollPane1.setViewportView(jTable1);

        add(jScrollPane1, java.awt.BorderLayout.CENTER);

    }//GEN-END:initComponents


    // Variables declaration - do not modify//GEN-BEGIN:variables
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTable jTable1;
    // End of variables declaration//GEN-END:variables
    private peg.sql.DBConnect myDB;

    /** Sets the current DBConnect class to use.
     * @param myDB The DBConnect to class to be used.
     */
    public void setDB(peg.sql.DBConnect myDB) {
        this.myDB = myDB; //sets the database class
    }

    /** Initiates the display of the contents of the current database connection. */
    public void displayTable(){
        String errorString = "";
        if (myDB != null) {
            // if no successfull query the posibility exist that no previous query was
            //  initiated. Therefor a check and a reQuery is done here.
            if (!myDB.isQuerySuccess()) {
              myDB.executeSQL();
              if (myDB.getErrorString() != null)
                errorString = ("Error:" + myDB.getErrorString());
            }
            if (myDB.isQuerySuccess()) {
                errorString = ("GotData!");
                jScrollPane1.setViewportView(new javax.swing.JTable(myDB));
            } else {
                jScrollPane1.setViewportView(null);
            }
        } else {
            errorString = ("Database not yet intialized");
        }
    }
}
```

Q.8 Package: `peg.utils`

Q.8.1 Object: `peg.utils.dataCollection`

```
/*
 * sgt.java
 *
 * Created on November 13, 2003, 9:03 AM
 */

package peg.utils;

import java.util.*;
import java.math.*;

import gov.noaa.pmel.util.*;
import gov.noaa.pmel.sgt.dm.*;
import gov.noaa.pmel.sgt.*;

import peg.sql.*;
import peg.utils.*;
/** The dataCollection class is used to form an interface between various data
 * types. The main function of this class is to perform manipulations on the data,
 * e.g. addition, sorting, grouping, sampling manipulations and filter
 * applications.
 *
 * <PRE>
 * (SimpleLine) ---------->+                  +--> (SimpleLine)
 * (DBConnect) ----------->+--> dataPoint[] -->+--> (GeoDate[])
 * (Geodate[], double[]) ->+     |            +--> (double[])
 *                               |
 *                               +--Sort
 *                               +--Group
 *                               +--Add, Sub, Mult, Div
 *                               +--MovAvg
 *                               +--Differentiate
 *                               +--Integrate
 *                               +--Downsample
 *                               +--Fill Zeros
 *                               |
 *                               +--(Geodate, double)[]
 *
 * </PRE>
 * @author Riaan Doorduin
 */
public class dataCollection {

    private Vector myData = new Vector();
    private int elementCount = 0;
    //DateFormat is used to format the dates stored in the database.
    private String myInputDateFormat = new String("yyyy-MM-dd");
    private String myOutputDateFormat = new String("yyyy-MM-dd");
    private boolean sortedState = false;
    private debug d = new debug("dataCollection",0);

    //The math manipulation functions:
    /** Indicates an addition function to the math method. */
    public final int MATH_ADD = 0;
    /** Indicates a subtraction function to the math method.on function to the math method. */
    public final int MATH_SUB = 1;
    /** Indicates a multiplication function to the math method. */
    public final int MATH_MUL = 2;
    /** Indicates a devision function to the math method. */
    public final int MATH_DIV = 3;
    /** Indicates a subtraction function to the math method.on function to the math method. The result is multiplied by -1 after subtraction*/
    public final int MATH_SUB_NEG = 4;

/*--Constructors-------------------------------------------------------------------------*/
    /** Creates a new instance of dataCollection */
    public dataCollection() {
        d.print(1,"dataCollection():clear()...");
        clear();
    }
    /** Creates a new instance of dataCollection
     * @param sL SimpleLine used as input
     */
    public dataCollection(SimpleLine sL) {
```

```
        d.print(1,"dataCollection(SimpleLine):...");
        if (sL!=null) {
            if (sL.isXTime() && !sL.isYTime()) {
                setDataCollection(sL.getTimeArray(), sL.getYArray());
            }
        }
    }
    /** Creates a new instance of dataCollection
     * @param myDB Use the input from myDB directly.
     */
    public dataCollection(DBConnect myDB) {
        d.print(1,"dataCollection(DBConnect):setDataCollection(myDB)...");
        setDataCollection(myDB);
    }
    /** Creates a new instance of dataCollection
     * @param myDB Use the DBConnect for direct input into this class.
     * @param myInputDateFormat The input date/time format used for this data.
     */
    public dataCollection(DBConnect myDB, String myInputDateFormat) {
        d.print(1,"dataCollection(DBConnect,myInputDateFormat():...");
        setInputDateFormat(myInputDateFormat);
        setDataCollection(myDB);
    }
    /** Creates a new instance of dataCollection
     * @param date The GeoDae array used in the dataset.
     * @param value The corresponding value range assosiated with each dat in the same data
     * collecollection.
     */
    public dataCollection(GeoDate[] date, double[] value){
        d.print(1,"dataCollection(date[].vale[]):...");
        setDataCollection(date, value);
    }

/*---Beans-------------------------------------------------------------------------------*/
    private void setElementCount(int i) {
        elementCount = i;
    }
    /** Returns the number of ellements in the dataCollection vector.
     * @return the number of elements in the collection
     */
    public int getElementCount() {
        return elementCount;
    }
    private void incElementCount() {
        elementCount++;
    }

    /** Sets the input date format used for input of data when using the DBConnect class
     * to import data. The default assumed format is "yyyy-MM-dd".
     * @param myDateFormat A string representing the date format.
     */
    public void setInputDateFormat(String myDateFormat) {
        if (getElementCount() == 0) {
            this.myInputDateFormat = new String(myDateFormat);
        }
    }
    /** Returns the input format used for the current data collection.
     * @return A string showing the used input date format for the current loaded data collection.
     */
    public String getInputDateFormat() {
        return myInputDateFormat;
    }

    /** Sets the resolution of the required output date format.
     * @param myDateFormat The date format required for determining output resolution
     */
    public void setOutputDateFormat(String myDateFormat) {
        this.myOutputDateFormat = new String(myDateFormat);
    }
    /** Returns the current output date format used.
     * @return The current output date format
     */
    public String getOutputDateFormat() {
        return myOutputDateFormat;
    }

    /** Clear the current dateCollection */
    public void clear() {
        myData.clear();
```

```
        setElementCount(0);
    }
    /** Adds a point the the data collection.
     * This point will be added at the end of the collection and the collection's
     * sorted state will be unsorted.
     * @param tmpDP The datapoint for input.
     */
    public void add(dataPoint tmpDP) {
        myData.add(tmpDP);
        setSorted(false);
        incElementCount();
    }
    /** Returns a data point in the collection.
     * @param i The index of the point to be returned.
     * @return The datapoint requested
     */
    public dataPoint get(int i) {
        return (dataPoint)myData.get(i);
    }

    /** Sets the sorted state of the collection */
    private void setSorted(boolean s) {
        sortedState = s;
    }
    /** Returns the sorted state of the collection.
     * @return True if the collection is sorted.
     */
    public boolean isSorted() {
        return sortedState;
    }

/*--DataCollection Beans--------------------------------------------------------------------------*/
    /** Loads a new set of data in the collection.
     * !!This will remove all the previously loaded data.!!
     * @param myDB The data from a database from the DBConnect class.
     */
    public void setDataCollection(DBConnect myDB) {
        d.print(3,"setDataCollection(DBConnect):clear()...");
        clear();
        addDataCollection(myDB);
    }

    /** Adds a new set of data to the collection.
     * @param myDB The data from a database from the DBConnect class.
     */
    public void addDataCollection(DBConnect myDB) {
        d.print(3,"addDataCollection(myDB):...");
        SimpleLine data;
        GeoDate[] tArray = new GeoDate[myDB.getRowCount()];
        double[] xArray = new double[myDB.getRowCount()];

        try {
            if ((myDB.getColumnCount() == 2) && (myDB.getRowCount() != 0)) {
                //Time Data
                for (int rw=0; rw < (myDB.getRowCount());rw++){
                    tArray[rw] = new GeoDate(myDB.getValueAt(rw, 0).toString(),getInputDateFormat());
                    d.print(9,"addDataCollection(myDB):tArray[rw]="+tArray[rw]);
                }

                //X Data
                for (int rw=0; rw < myDB.getRowCount();rw++){
                    try {
                        xArray[rw] = new BigDecimal(myDB.getValueAt(rw, 1).toString()).doubleValue();
                    } catch (NumberFormatException n) {
                        d.print(5,"dataCollection(DBConnect):Invallid number format:(Row:"+rw+"):"+n.getMessage());
                        xArray[rw] = 0.0;
                    }
                }
                addDataCollection(tArray, xArray);
            } else {
                System.out.println("Format not valid for SGT input.");
            }
        } catch (IllegalTimeValue tv) {
            System.out.println("Date Format not valid for SGT input:" + tv.getMessage());
        }
    }

    /** Sets the data in the collection from a GeoData en valua array. Both arrays must
     * have the same lenghs to be successfull.
```

```
     * !! Previous data will be lost using this method !!
     * @param date The GeoDate array for the collection.
     * @param value The values assosiated with the dates.
     */
    public void setDataCollection(GeoDate[] date, double[] value){
        clear();
        addDataCollection(date,value);
    }

    /** Adds data to the collection from a GeoData en valua array. Both arrays must
     * have the same lenghs to be successfull.
     * @param date The GeoDate array for the collection.
     * @param value The values assosiated with the dates.
     */
    public void addDataCollection(GeoDate[] date, double[] value){
        int i = 0;
        if (date.length == value.length) {
            for (i = 0; i < date.length; i++) {
                add(new dataPoint(date[i],value[i]));
            }
            sort();
        } else {
            clear();
        }
    }

    /** Returns the current data in the collection in Vector format
     * @return The data in the data collection in vector format
     */
    public Vector getDataCollection() {
        return myData;
    }
/*--Conversion methods-----------------------------------------------------------------------*/
    /** Outputs only the date part of the collection
     * @return Only the GeoDate part of the collection in array format.
     */
    public GeoDate[] toGeoDateArray() {
        GeoDate tmpGeoDate[] = new GeoDate[getElementCount()];
        int count = 0;
        for (Iterator i = myData.iterator(); i.hasNext(); ){
            tmpGeoDate[count++] = ((dataPoint)i.next()).getDate();
        }
        return tmpGeoDate;
    }

    /** Returns the current data contained in the collection.
     * @return The current data in a double array
     */
    public double[] toValueArray() {
        double tmpValue[] = new double[getElementCount()];
        int count = 0;
        for (Iterator i = myData.iterator(); i.hasNext(); ){
            tmpValue[count++] = ((dataPoint)i.next()).getValue();
        }
        return tmpValue;
    }

    /** Returns the current dataCollection in a SimpleLine format to be used with the
     * SGT Graph class.
     * @return Simple Line data of the current dataset.
     */
    public SimpleLine toSimpleLine() {
        if (getElementCount() != 0) {
            SimpleLine data = new SimpleLine(toGeoDateArray(),toValueArray(),"");
            data.setXMetaData(new SGTMetaData("","",false,false));
            data.setYMetaData(new SGTMetaData("","",false,false));
            return data;
        } else {
            return null;
        }
    }

    /** Converts the current data into a String representation
     * @return The string representation of the current content of the collection.
     */
    public String toString() {
        return "Total:"+totalSum()+";Avg:"+avg();
    }
```

```
    /** Displays the current contents of the class.
     * The output is sent the standard output of the system.
     */
    public void print() {
        for (Iterator i = myData.iterator(); i.hasNext(); ){
            System.out.println(i+":["+(dataPoint)i.next()+"]");
        }
    }

/*--Sort and manipulation algotrithms-----------------------------------------------------------*/
    /** Sorts the current data according to the date. */
    public void sort() {
        if (!isSorted()) {
            Collections.sort(myData);
            setSorted(true);
        }
    }

    /** Search for a specific date in the collection and returns the index
     * @return The index where the current data is stored.
     * -1 if no data is found.
     * @param key The date of the index required
     */
    public int search(GeoDate key) {
      sort();
      return Collections.binarySearch(myData,new dataPoint(key));
    }

    /** Adds all the values contained in the data class.
     * @return The total sum of all the values
     */
    public double totalSum() {
        double tmpSum = 0.0;
        for (Iterator i = myData.iterator(); i.hasNext(); ){
            tmpSum = tmpSum+((dataPoint)i.next()).getValue();
        }
        return tmpSum;
    }
    /** Calculates the weighted average of the contained values per entry.
     * @return the Average value of the data Collection.
     */
    public double avg() {
        if (getElementCount() != 0) {
            return (totalSum()/getElementCount());
        } else {
            return 0.0;
        }
    }
    public double avg(int period) {
        if (period == 0) {
            return avg();
        } else {
            sort();
            double tmpSum = 0.0;
            if ((getElementCount() != 0) & (getElementCount() >= period)) {
                for (int i = getElementCount()-period; i < getElementCount(); i++) {
                   tmpSum = tmpSum + get(i).getValue();
                }
                return (tmpSum/period);
            } else {
                return 0.0;
            }
        }
    }
    /** Adds all the values of all the datapoints in this collection which shares the
     * same date.
     * The output will be in a sorted state after this method was
     * performed.
     * @return A new grouped datacollection
     */
    public dataCollection group() {
        dataCollection myDC = new dataCollection();
        boolean first = true;
        if ((getElementCount() > 1) && (getOutputDateFormat() != null)) {
            sort();
            dataPoint myDataPoint;
            Iterator i = myData.iterator();
            GeoDate currentDate = new GeoDate();
            myDC.setInputDateFormat(getOutputDateFormat());
```

```
        double currentValue = 00;
        while (i.hasNext()) {
            myDataPoint = (dataPoint)(i.next());
            if (first) {
                first = false;
                currentValue = myDataPoint.getValue();
                currentDate = myDataPoint.getDate(getOutputDateFormat());
            } else {
                if (currentDate.equals(myDataPoint.getDate(getOutputDateFormat()))) {
                    currentValue = currentValue + myDataPoint.getValue();
                } else {
                    myDC.add(new dataPoint(currentDate,currentValue));
                    currentValue = myDataPoint.getValue();
                    currentDate = myDataPoint.getDate(getOutputDateFormat());
                }
            }
        }
        myDC.add(new dataPoint(currentDate,currentValue));
        myDC.setOutputDateFormat(getOutputDateFormat());
        myDC.sort();
        return myDC;
    } else {
      return this;
    }
}
/** Adds all the values of all the datapoints in this collection which shares the
 * same date.
 * The output will be in a sorted state after this method was
 * performed.
 * @param dateFormat The resolution of the dateFormat required for the output of this method.
 * @return A new grouped datacollection
 */
public dataCollection group(String dateFormat) {
    setOutputDateFormat(dateFormat);
    return group();
}

/** Does mathamatical operations on two datasets, according to the value in <B>operation </B>.
 * The datasets must be grouped and sorted prior to input.
 * Only values with mathching datas will be considered when operations are executed.
 * @param addDC The dataCollection to be used in conjuction with the current object to perform
 * the mathematical operation.
 * @param operation The operation to be used.
 * MATH_ADD - addition
 * MATH_SUB - subtraction
 * MATH_MUL - multiplication
 * MATH_DIV - division
 * @return A dataCollection containing the result of the operation.
 */
public dataCollection math(dataCollection addDC, int operation) {
    return math(addDC, operation, false);
}
/** Does mathamatical operations on two datasets, according to the value in <B>operation </B>.
 * The datasets must be grouped and sorted prior to input.
 * If the newInstace is false, only values with mathching datas will be considered when operations are executed.
 * @param addDC The dataCollection to be used in conjuction with the current object to perform
 * the mathematical operation.
 * @param operation The operation to be used.
 * MATH_ADD - addition
 * MATH_SUB - subtraction
 * MATH_MUL - multiplication
 * MATH_DIV - division
 * @return A dataCollection containing the result of the operation.
 * @param newInstance Indicates if a new instance should be generated.
 */
public dataCollection math(dataCollection addDC, int operation, boolean newInstance) {
    Iterator i;
    d.print(3, "math("+addDC+","+operation+","+newInstance+"):");
    dataCollection myDC = new dataCollection();
    if (addDC != null) {
        if (((getElementCount() > 0) && (addDC.getElementCount() > 0)) || newInstance){
            d.print(5, "math:sort...:addDC="+addDC);
            sort();
            d.print(5, "math:addDC.group...:addDC="+addDC);
            addDC = addDC.group(getOutputDateFormat());
            d.print(5, "math:addDC.group:addDC="+addDC);

            myDC.setInputDateFormat(getOutputDateFormat());
            dataPoint myDataPoint;
```

```
            if (!newInstance) {
                i = myData.iterator();
            } else {
                i = addDC.getDataCollection().iterator();
            }
            GeoDate currentDate = new GeoDate();
            d.print(5, "math:_prior_while_addDC="+addDC);
            double currentValue = 00;
            while (i.hasNext()) {
                myDataPoint = (dataPoint)(i.next());
                currentValue = myDataPoint.getValue();
                currentDate = myDataPoint.getDate(getOutputDateFormat());
                if (!newInstance) {
                    d.print(5, "math:_in_while_addDC="+addDC);
                    int j = addDC.search(currentDate);
                    d.print(5, "math:j="+j);
                    if (j >= 0) {
                       if (operation == MATH_ADD) {
                         currentValue = currentValue + addDC.get(j).getValue();
                       }
                       if (operation == MATH_SUB) {
                         currentValue = currentValue - addDC.get(j).getValue();
                       }
                       if (operation == MATH_SUB_NEG) {
                         currentValue = addDC.get(j).getValue() - currentValue;
                       }
                       if (operation == MATH_MUL) {
                         currentValue = currentValue * addDC.get(j).getValue();
                       }
                       if ((operation == MATH_DIV) && (addDC.get(j).getValue() != 0.0)) {
                         currentValue = currentValue / addDC.get(j).getValue();
                       }

                       myDC.add(new dataPoint(currentDate,currentValue));
                    }
                } else {
                       if (operation == MATH_ADD) {
                         currentValue = 0.0 + currentValue;
                       }
                       if (operation == MATH_SUB) {
                         currentValue = 0.0 - currentValue;
                       }
                       if (operation == MATH_SUB_NEG) {
                         currentValue = currentValue - 0.0;
                       }
                       if (operation == MATH_MUL) {
                         currentValue = 1.0 * currentValue;
                       }
                       if ((operation == MATH_DIV) && (currentValue != 0.0)) {
                         currentValue = 1.0 / currentValue;
                       }
                       myDC.add(new dataPoint(currentDate,currentValue));
                }
            }
            myDC.setOutputDateFormat(getOutputDateFormat());
            myDC.sort();
            return myDC;
        } else {
          return null;
        }
    } else {
        return null;
    }
}

/** Generates a array to be used with the correlation for implementing a mov average
 * function.
 * @param size The amount of samples to incorporate, which translates into the output size if
 * the function.
 * @return The moving function filter function.
 */
public double[] movAvg(int size) {
    double[] myOut = new double[size];
    for (int i = 0; i < size; i++) {
        myOut[i] = 1.0/(size);
    }
    return myOut;
}
```

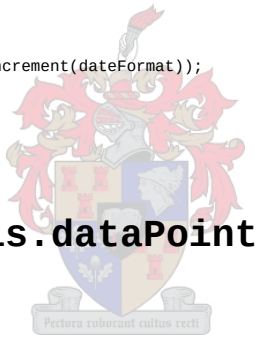
```
/** Differentiates the data points. The first data point is lost.
 *  This method assumes the operation should be done on its own data.
 */
public dataCollection differentiate() {
    return differentiate(this);
}

/** Differentiates the data points. The first data point is lost.
 * @param the dataCollection to be differentiated
 */
public dataCollection differentiate(dataCollection diffDC) {
    Iterator i;
    dataCollection myDC = new dataCollection();
    if (diffDC != null) {
        if (diffDC.getElementCount() > 1){
            diffDC.sort();
            //diffDC = diffDC.group(getOutputDateFormat());
            myDC.setInputDateFormat(getOutputDateFormat());
            dataPoint myDataPoint;
            i = diffDC.getDataCollection().iterator();

            GeoDate currentDate = new GeoDate();
            double currentValue = 0.0;
            double prevValue = 0.0;
            myDataPoint = (dataPoint)(i.next());
            prevValue = myDataPoint.getValue();
            while (i.hasNext()) {
                myDataPoint = (dataPoint)(i.next());
                currentValue = myDataPoint.getValue();
                currentDate = myDataPoint.getDate(getOutputDateFormat());

                myDC.add(new dataPoint(currentDate,(0.0+currentValue-prevValue)));
                prevValue = currentValue;
            }

            myDC.setOutputDateFormat(getOutputDateFormat());
            myDC.sort();
            return myDC;
        } else {
          return null;
        }
    } else {
        return null;
    }
}

/** Differentiates the data points. The first data point is lost.
 *  This method assumes the operation should be done on its own data.
 */
public dataCollection integrate() {
    return integrate(this);
}

/** Differentiates the data points. The first data point is lost.
 * @param the dataCollection to be differentiated
 */
public dataCollection integrate(dataCollection intDC) {
    Iterator i;
    dataCollection myDC = new dataCollection();
    if (intDC != null) {
        if (intDC.getElementCount() > 0){
            sort();
            //intDC = intDC.group(getOutputDateFormat());
            myDC.setInputDateFormat(getOutputDateFormat());
            dataPoint myDataPoint;
            i = intDC.getDataCollection().iterator();

            GeoDate currentDate = new GeoDate();
            double currentValue = 0.0;
            double sumValue = 0.0;
            while (i.hasNext()) {
                myDataPoint = (dataPoint)(i.next());
                currentValue = myDataPoint.getValue();
                currentDate = myDataPoint.getDate(getOutputDateFormat());
                sumValue = 0.0+sumValue + currentValue;
                myDC.add(new dataPoint(currentDate,sumValue));
            }

            myDC.setOutputDateFormat(getOutputDateFormat());
```

```
            myDC.sort();
            return myDC;
        } else {
          return null;
        }
    } else {
        return null;
    }
}

/* Correlation function */
/** Autocorrelation function
 * @return Output of the autocorrelation function into a new dataCollection set
 */
public dataCollection corr() {
    return corr(this); // detemines autocorrelation
}
/** Correlation of another dataset onto the current dataCollection
 * @param corrData The array to correlate with the current collection
 * @return The output of the correlation function into a new dataCollection set.
 */
public dataCollection corr(double[] corrData) {
    return corr(null, corrData);
}
/** Correlation of another dataset onto the current dataCollection
 * @return The output of the correlation function into a new dataCollection set.
 * @param corrData The data to correlate to the current dataCollection
 */
public dataCollection corr(dataCollection corrData) {
    corrData.sort();
    return corr(null,  corrData.toValueArray());
}
/** Correlationan external dataset onto another dataCollection
 * @return The output of the correlation function into a new dataCollection set.
 * @param corrData The data to correlate the external data to.
 * @param externalData The externalData
 */
public dataCollection corr(dataCollection externalData, dataCollection corrData) {
    corrData.sort();
    return corr(externalData, corrData.toValueArray());
}
/** Correlation of an external dataset onto another dataCollection
 * @return The output of the correlation function into a new dataCollection set.
 * @param externalData The data to be used for correlation
 * @param corrData The array to be used for correlation (filter)
 */
public dataCollection corr(dataCollection externalData, double[] corrData) {
    //Variable declaration
    dataCollection myDC = new dataCollection();
    dataPoint myDataPoint;
    GeoDate[] inputDate;
    double[] inputValue;
    GeoDate currentDate = new GeoDate();
    double currentValue = 00;

    if (corrData.length < 1) return null; //no output can be determined with no corrData
    //Setup Data
    if (externalData == null) {
        sort();
        inputDate = toGeoDateArray();
        inputValue = toValueArray();
        myDC.setInputDateFormat(getOutputDateFormat());
    } else {
        externalData.sort();
        myDC.setInputDateFormat(externalData.getInputDateFormat());
        inputDate = externalData.toGeoDateArray();
        inputValue = externalData.toValueArray();
    }
    if (corrData == null) corrData = inputValue; //calculates autocorrelation of dataset.
    int l_max = inputValue.length;
    double[] valueRegister = new double[l_max]; //to store temp values

    //Correlation routine - code copied from Digital Signal Processing - Proakis and Manolakis p132
    for (int i = 0; i < l_max; i++) {

        int nl = (corrData.length+i-1);

        if (nl >= (inputValue.length-1)) {
            nl = inputValue.length-1;
```

```
        }
        currentValue = 0.0;
        for (int j = i; j <= nl; j++) {
            currentValue = currentValue + inputValue[j]*corrData[j-i];
        }
        if (i <= (inputValue.length - corrData.length)) {
          myDC.add(new dataPoint(inputDate[i + corrData.length -1],currentValue));
        }
    }
    //End of Correlation routine
    myDC.setOutputDateFormat(getOutputDateFormat());
    myDC.group();
    return myDC;
}

/** Fill all the gaps between the first and last date of the data set.
 * The set must be in sorted state.
 * It will fill the gaps up to current set resolution with the InputDateFormat
 * setting.
 * @return The output with all gaps filled with zeros.
 */
public dataCollection fillGapsZero() {
    return fillGapsZero(this);
}
/** Fill all the gaps between the first and last date of the data set.
 * The set must be in sorted state.
 * It will fill the gaps up to current set resolution with the InputDateFormat
 * setting.
 * @return The output with all gaps filled with zeros.
 * @param outputDateFormat specifies the resolution of the output.
 */
public dataCollection fillGapsZero(String outputDateFormat) {
    this.setOutputDateFormat(outputDateFormat);
    return fillGapsZero(this);
}
/** Fill all the gaps between the first and last date of the data set.
 * The set must be in sorted state.
 * It will fill the gaps up to current set resolution with the InputDateFormat
 * setting.
 * @return The output with all gaps filled with zeros.
 * @param inputDC Fill the gaps of this dataCollection
 */
public dataCollection fillGapsZero(dataCollection inputDC) {
    if (inputDC != null) {
        inputDC.group();

        int increment = getDateFormatIncrement(inputDC.getOutputDateFormat());

        GeoDate g = inputDC.get(0).getDate(getOutputDateFormat());
        int i = 0;
        dataCollection outputDC = new dataCollection();
        outputDC.setInputDateFormat(inputDC.getInputDateFormat());
        outputDC.setOutputDateFormat(inputDC.getOutputDateFormat());
        while (g.compareTo(inputDC.get(getElementCount()-1).getDate()) == -1) {
            if (inputDC.get(i).getDate().equals(g)) {
                outputDC.add(new dataPoint(new dataPoint(g).getDate(inputDC.getOutputDateFormat()),inputDC.get(i).getValue()));
                i++;
            } else {
                outputDC.add(new dataPoint(new dataPoint(g).getDate(inputDC.getOutputDateFormat()),0.0));
            }
            if (increment == GeoDate.MONTHS) {
                g.decrement(10,GeoDate.DAYS); // to make sure that month doesn't flip over due to extra day.
            }
            g = new dataPoint(g.increment(1.0,increment)).getDate(inputDC.getOutputDateFormat());

        }
        return outputDC;
    } else {
        return null;
    }
}

/** Change the resolution of the dataCollection. The missing data points are
 * averaged out between the different available datapoints.
 * @return The downsampled dataCollection
 */
public dataCollection downSample() {
    return downSample(this);
}
```

```
/** Change the resolution of the dataCollection. The missing data points are
 * averaged out between the different available datapoints.
 * @return The downsampled dataCollection
 * @param outputDateFormat specifies the resolution of the output.
 */
public dataCollection downSample(String outputDateFormat) {
    this.setOutputDateFormat(outputDateFormat);
    return downSample(this);
}
/** Change the resolution of the dataCollection. The missing data points are
 * averaged out between the different available datapoints. The first point however is not changed.
 * @return The downsampled dataCollection
 * @param inputDC Fill the gaps of this dataCollection
 */
public dataCollection downSample(dataCollection inputDC) {
    if ((inputDC != null) && (inputDC.getElementCount() > 0)) {
        inputDC.group();

        int increment = getDateFormatIncrement(inputDC.getOutputDateFormat());

        GeoDate deltaTime = new GeoDate();
        double timeDivider = 1.0;
        GeoDate currentDate = inputDC.get(0).getDate(inputDC.getOutputDateFormat());
        double currentValue = 0.0;

        dataCollection outputDC = new dataCollection();
        outputDC.setInputDateFormat(inputDC.getInputDateFormat());
        outputDC.setOutputDateFormat(inputDC.getOutputDateFormat());
        for (int i = 0; i < inputDC.getElementCount(); i++) {
            currentDate = inputDC.get(i).getDate(getOutputDateFormat());
            currentValue = inputDC.get(i).getValue();
            if (i==0) {
                outputDC.add(new dataPoint(inputDC.get(i).getDate(inputDC.getOutputDateFormat()),currentValue));
            } else {
                deltaTime = inputDC.get(i).getDate().subtract(inputDC.get(i-1).getDate());
                if (increment == GeoDate.YEARS) {
                    timeDivider = deltaTime.getGMTYear()-1970;
                }
                if (increment == GeoDate.MONTHS) {
                    timeDivider = ((deltaTime.getGMTYear()-1970)*12)+(deltaTime.getGMTMonth()-1);
                }
                if (increment == GeoDate.DAYS) {
                    timeDivider = ((deltaTime.getGMTYear()-1970)*12*365)+deltaTime.getYearday()-1;
                }
                if (increment == GeoDate.HOURS) {
                    timeDivider = ((((deltaTime.getGMTYear()-1970)*12*365)+(deltaTime.getYearday()-1))*24) + deltaTime.getGMTHours();
                }
                if (increment == GeoDate.MINUTES) {
                    timeDivider = ((((((deltaTime.getGMTYear()-1970)*12*365)+(deltaTime.getYearday()-1))*24) + deltaTime.getGMTHours())*60
                }
                if (increment == GeoDate.SECONDS) {
                    timeDivider = (((((((deltaTime.getGMTYear()-1970)*12*365)+(deltaTime.getYearday()-1))*24) + deltaTime.getGMTHours())*6
                }

                if (timeDivider != 0.0) {
                    currentValue = currentValue/timeDivider;
                    deltaTime = inputDC.get(i-1).getDate().increment(1,increment);
                    if (increment == GeoDate.MONTHS) {
                        deltaTime.decrement(10,GeoDate.DAYS); // to make sure that month doesn't flip over due to extra day.
                    }
     for (GeoDate g = deltaTime; g.compareTo(inputDC.get(i).getDate()) == -1; g.increment(1,increment)) {
                        outputDC.add(new dataPoint(new dataPoint(g).getDate(inputDC.getOutputDateFormat()),currentValue));
                    }
                } else {
                 outputDC.add(new dataPoint(currentDate,currentValue));
                }
            }
        }
        return outputDC;
    } else {
        return null;
    }
}

/** This method returns the increment resolution size.
 * @param tmpDateFormat The dateFormat
 * @return The increment size
 */
public int getDateFormatIncrement(String tmpDateFormat) {
```

```
        int increment = GeoDate.YEARS;
        if (tmpDateFormat.toUpperCase().indexOf("MM") >= 0) {
            increment = GeoDate.MONTHS;
        }
        if (tmpDateFormat.toUpperCase().indexOf("DD") >= 0) {
            increment = GeoDate.DAYS;
        }
        if ((tmpDateFormat.toUpperCase().indexOf("HH") >= 0) || (tmpDateFormat.toUpperCase().indexOf("HOUR") >= 0)) {
            increment = GeoDate.HOURS;
        }
        if ((tmpDateFormat.toUpperCase().indexOf(":MM") >= 0) || (tmpDateFormat.toUpperCase().indexOf("MINUTE") >= 0)) {
            increment = GeoDate.MINUTES;
        }
        if ((tmpDateFormat.toUpperCase().indexOf(":SS") >= 0) || (tmpDateFormat.toUpperCase().indexOf("SECOND") >= 0)) {
            increment = GeoDate.SECONDS;
        }
        return increment;
    }

    public GeoDate incrementByFormat(GeoDate tmpDate, String dateFormat) {
        try {
            tmpDate = new GeoDate(tmpDate.toString(dateFormat),dateFormat);
        } catch (IllegalTimeValue e) {
            tmpDate = tmpDate;
        }
        tmpDate.increment(1.0, getDateFormatIncrement(dateFormat));
        return tmpDate;
    }
    public GeoDate decrementByFormat(GeoDate tmpDate, String dateFormat) {
        try {
            tmpDate = new GeoDate(tmpDate.toString(dateFormat),dateFormat);
        } catch (IllegalTimeValue e) {
            tmpDate = tmpDate;
        }
        tmpDate.decrement(1.0, getDateFormatIncrement(dateFormat));
        return tmpDate;
    }
}
```



Q.8.2 Object: `peg.utils.dataPoint`

```
/*
 * data.java
 *
 * Created on November 17, 2003, 7:58 PM
 */

package peg.utils;

import gov.noaa.pmel.util.*;
import gov.noaa.pmel.sgt.dm.*;
import gov.noaa.pmel.sgt.*;

import peg.sql.*;
import java.io.Serializable;
/** dataPoint represents a value in any point in time. For some applications it is
 * neccassary to manipulate data data part, by keep the time part untouched.
 * @author Riaan Doorduin
 */
public class dataPoint implements Serializable, Comparable {

    private GeoDate myDate;
    private double myValue;
    /** This string contains the latest error message from this class.
     * Currently this is the only error feedback available and future work can be done
     * here.
     */
    public String errorStr = "";

    /** Create a new dataPoint */
    public dataPoint() {
    }
    /** Create a new dataPoint in Time.
     * @param myDate The date/time of the point
     */
    public dataPoint(GeoDate myDate) {
        setDate(myDate);
```

```
}
/** Create a new dataPoint with a value.
 * @param myValue The value associated with the point
 */
public dataPoint(double myValue) {
    setValue(myValue);
}
/** Create a new dataPoint with both value and time parameters known.
 * @param myDate The date/time for this point
 * @param myValue The value for this point
 */
public dataPoint(GeoDate myDate, double myValue) {
    setDate(myDate);
    setValue(myValue);
}

/** Set the value associated with the current point
 * @param myValue The value for the point
 */
public void setValue(double myValue) {
    this.myValue = myValue;
}
/** Returns the value associated with the point
 * @return the value of the point
 */
public double getValue() {
    return myValue;
}

/** Set the date/time for the current point
 * @param myDate The date/time for the point
 */
public void setDate(GeoDate myDate) {
    this.myDate = myDate;
}
/** Return the date/time of the point.
 * @return the date time of the data point.
 */
public GeoDate getDate() {
    return myDate;
}

/** Returns only the date/time components in the dataFormat string
 * @param myDateFormat The date format data fields which should be included in the date for the point
 * @return Only the data of the fields specified in the myDateFormat field
 */
public GeoDate getDate(String myDateFormat) {
    int YYYY = 0;
    int MM = 12;
    int DD;
    int H = 23;
    int M = 59;
    int S = 59;

    YYYY = myDate.getGMTYear(); // The year format will always be included in the date.
        // A typical top down approach.
        //  Months will be include if the year is included, therefor
        //  YYYY -> MM -> DD -> HH -> MM -> SEC
    if (myDateFormat.toUpperCase().indexOf("MM") >= 0) {
        MM = myDate.getGMTMonth();
        if (myDateFormat.toUpperCase().indexOf("DD") >= 0) {
            DD = myDate.getGMTDay();
            if (myDateFormat.toUpperCase().indexOf("HH:MM:SS") >= 0) {
                H = myDate.getGMTHours();
                M = myDate.getGMTMinutes();
                S = new java.math.BigDecimal(myDate.getGMTSeconds()).intValue();
            }
            if (myDateFormat.toUpperCase().indexOf("HH:MM") >= 0) {
                H = myDate.getGMTHours();
                M = myDate.getGMTMinutes();
            }
            if (myDateFormat.toUpperCase().indexOf("HOUR") >= 0) {
                H = myDate.getGMTHours();
                if (myDateFormat.toUpperCase().indexOf("MIN") >= 0) {
                    M = myDate.getGMTMinutes();
                    if (myDateFormat.toUpperCase().indexOf("SEC") >= 0) {
                        S = new java.math.BigDecimal(myDate.getGMTSeconds()).intValue();
                    }
                }
```

```
                }
            } else {
                DD = myDate.getDaysInMonth();
            }
        } else {
            DD = myDate.getDaysInMonth();
        }
        try {
            return new GeoDate(MM,DD,YYYY,H,M,S,0);
        } catch (IllegalTimeValue i) {
            errorStr = ("dataPoint:Illegal Date/Time value encountered: "+i.getMessage());
            return null;
        }
    }

    /** Compares the object to another object using the natural order of the date/time component
     * of the data point.
     * @param o The object to be compared with
     * @return The result of the compare
     */
    public int compareTo(Object o) {
        return compareTo((dataPoint)o);
    }

    /** Compares the object to another object using the natural order of the date/time component
     * of the data point.
     * @param tmpDP The point to be compared with
     * @return The result of the compare
     */
    public int compareTo(dataPoint tmpDP) {
        return myDate.compareTo(tmpDP.getDate());  // The date part will be compared.
    }

    /** Returns the value in a string of the data point
     * @return The value of the current data point in a human readable string.
     */
    public String toString() {
        return (myDate.toString() + ":" + myValue);
    }
}
```

Q.8.3 Object: `peg.utils.debug`

```
/*
 * debug.java
 *
 * Created on 14 February 2004, 10:59
 */

package peg.utils;

/** This class is based on the Debug class as showed in several text books.
 *
 * It helps with debugging in the sense that the debug layer and depth can be set
 * in the class and can be disabled when the debugging of a class is completed.
 * @author RDoorduin
 */
public class debug {
    private String name;
    private final int maxDebugLevel = 10;
    private int level = 5;
    /** Creates a new instance of debug */
    public debug() {
        this("NoName");
    }

    /** Creates a new Named instance of the debug class.
     * @param name The name of the debug instance.
     */
    public debug(String name) {
        this(name,5);
    }

    /** Creates a new Named instance of the debug class and indicates the level of
     * debugging in this instance.
     * @param name The name of the debugging instance
```

```
     * @param level The level of debugging in this instance
     */
    public debug(String name, int level) {
        setName(name);
        setLevel(level);
    }

    /** Sets the name of this debugging instance
     * @param name The new name of the instance
     */
    public void setName(String name) {
        this.name = name;
    }

    /** Sets the level of this debugging instance. This level determines the lowest
     * level dived into for debugging.
     * @param level The debugging level of this instance allowed for printing.
     */
    public void setLevel(int level) {
        this.level = level;
    }

    /** Prints the debugging message and ensures that flushing of all text to the
     * standard output occurs.
     * @param level Indicate the level of this message.
     * @param text The message associated with the debugging event.
     */
    public void print(int level,String text) {
        if ((this.level >= level) & (maxDebugLevel >= level)) {
            if (level == 1) {
                System.out.print("+");
            } else if (level == 2) {
                System.out.print(" +");
            } else if (level == 3) {
                System.out.print("  +");
            } else if (level == 4) {
                System.out.print("   +");
            } else if (level == 5) {
                System.out.print("    +");
            } else if (level == 6) {
                System.out.print("     +");
            } else if (level == 7) {
                System.out.print("      +");
            } else if (level == 8) {
                System.out.print("       +");
            } else if (level == 9) {
                System.out.print("        +");
            } else if (level == 10) {
                System.out.print("         +");
            } else {
                System.out.print("          ");
            }
            System.out.println(name+":"+text);
            System.out.flush();
        }
    }
}
```


Q.8.4 Object: `peg.utils.hystogram`

```
/*
 * hystogram.java
 *
 * Created on 19 February 2004, 10:29
 */

package peg.utils;

import java.math.*;

import gov.noaa.pmel.sgt.dm.*;

import peg.utils.*;
/**
 *
 * @author  RDoorduin
 */
```

```
public class hystogram {
    private final int samples = 200;
    private double min = 0.0;
    private double max = 100.0;
    private double[] hyst = new double[samples+1];
    private double[] values = new double[samples+1];
    private debug d = new debug("hystogram",0);
    private double sum = 0.0;

    /** Creates a new instance of hystogram */
    public hystogram() {
        d.print(1,"hystogram()");
        for (int i=0; i <= samples; i++){
            hyst[i] = 0.0;  //amount of values (y)
        }
        generateValues();  // (x)
    }

    public void setMin(double min) {
        if (this.min != min) {
          this.min = min;
          generateValues();
        }
    }
    public double getMin() {
        return min;
    }

    public void setMax(double max) {
        if (this.max != max) {
          this.max = max;
          generateValues();
        }
    }
    public double getMax() {
        return max;
    }

    public void add(double value) {
        d.print(3,"add("+value+")");
        if (value < min) {
            value = min;
        }
        if (value > max) {
            value = max;
        }

        double tmpD = ((value - min)/(max-min)*samples);
        int i = new BigDecimal(tmpD).intValue();
        hyst[i] = hyst[i] + 1.0;
   d.print(7, "hyst["+i+"]="+hyst[i]);
    }

    public double[] getHyst() {
        double[] tmpHyst = new double[samples+1];
        double sum = 0.0;
        for (int i=0; i <= samples; i++) {
            sum = sum + hyst[i];
        }
        if (sum > 0.0) {
            for (int i=0; i <= samples; i++) {
                tmpHyst[i] = hyst[i]/sum;
            }
        }
        this.sum = sum;
        return tmpHyst;
    }

    public double getSum() {
        return sum;
    }

    private void generateValues() {
        for (int i=0; i <= samples; i++){
            values[i] = i*(max-min)/samples + min;
        }
    }

    public SimpleLine toSimpleLine() {
```

```
        d.print(3,"toSimpleLine...");
        SimpleLine data = new SimpleLine(values,getHyst(),"");
        data.setXMetaData(new SGTMetaData("","",false,false));
        data.setYMetaData(new SGTMetaData("","",false,false));
        return data;
    }
    public void print() {
        System.out.println("Hystogram:");
        for (int i=0; i <= samples; i++) {
            System.out.println(""+values[i]+","+hyst[i]);
        }
    }
}
```

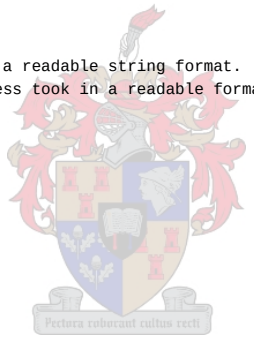
Q.8.5 Object: `peg.utils.stringUtils`

```
/*
 * stringUtils.java
 *
 * Created on 14 June 2004, 06:50
 */

package peg.utils;

/** This class is used to extend the functionality of strings.
 * @author RDoorduin
 */
public class stringUtils {

    /** Creates a new instance of stringUtils */
    public stringUtils() {
    }

    /** This method returns the incomming String with a fixed length. If the String is
     * shorter than the specified length spaces will be appended to the string to
     * ensure the correct length.
     * @param inStr The incomming String required to be a certain length
     * @param length The lenght of the new String
     * @return A String with a fixed length as defined by the length parameter.
     */
    public static String strLength(String inStr, int length) {
        String tmpStr = "";
        if (inStr.length() < length) {
            for (int i = 0; i < (length - inStr.length()); i++) {
                tmpStr = tmpStr + " ";
            }
            tmpStr = inStr + tmpStr;
        } else if (inStr.length() > length) {
            tmpStr = inStr.substring(0,length);
        } else {
            tmpStr = inStr;
        }
        return tmpStr;
    }
}
```

Q.8.6 Object: `peg.utils.TicToc`

```
/*
 * tictoc.java
 *
 * Created on 20 September 2003, 08:30
 */

package peg.utils;

/** Class to easily determine the time a process took to complete.
 * <PRE>usage:</PRE>
 * <PRE>
 * <CODE>
 *  Tictoc = new Tictoc();
 *  t.tic();
 *  //Process
 *  t.toc();
```

```
 *  System.out.println(t);
 * </CODE>
 * </PRE>
 * @author Riaan Doorduin
 */
public class TicToc {

    private long startTime = 0;

    private long totalTime = 0;

    /** Creates a new instance of tictoc */
    public TicToc() {
    }

    /** Initializes the timer and stores the current time. */
    public void tic() {
        startTime = System.currentTimeMillis();
    }

    /** Calculates the time since tic() was initiated. */
    public void toc() {
        long stopTime;
        stopTime = System.currentTimeMillis();
        totalTime = stopTime-startTime;
    }

    /** Returns the time calculated by toc().
     * @return long the time calulated by toc.
     */
    public long getTime() {
        return totalTime;
    }

    /** Converts the time the process took to a readable string format.
     * @return String The total time the process took in a readable format
     */
    public String toString() {
        return (""+totalTime+"ms");
    }

}
```



Appendix R

C Code

The code presented in this appendix is used to program the Check Meter as discussed in Chapter 6. The code was compiled with MSPGCC[69].

R.1 Global and Hardware setup software

R.1.1 main.c

This piece of code is what keeps everything together, ensuring all the correct drivers are running and the eternal loop is initiated with the correct program.

```
//******************************************************************************
//  main.c
//
// This is the main procedure code for the CheckMeter for the use on
//  a Texas Instruments MSP 430 F 149 micro processor.
//
// This code invokes all the different hardware and act as the kernel for the system
//    The MSP430 can handle only one thread at a sigle instance. The kernel emulates
//    a multi threaded environment with the use of different state even machines on the
//    levels in the system.
// The main loop these levels van be seen.
//    First all the perphirals incorporated in the MCU is handeld.
//    Secondly the external devices are and communications is done.
//    The top layer can be seen as the logic and application layer.
//
// History:
//    v1.0 - 2004-03-22 - First creation of code
//    v2.0 - 2004-07-01 - Revamp of all the tradional code into a integrated state
//                        event machine.
//
//  R. Doorduin
//  PEG - University of Stellenbosch
//  March 2004
//******************************************************************************

//System
#include <io.h>            //General Purpose IO for MSP430
#include <signal.h>        //Signals for interupt purposes
#include "memory.h"        //Memory Map and general declerations
#include "clock.h"         //Clock configuration
#include "utils.h"         //Utils for general purpose
//Internal Hardware/Periphery
#include "timer.h"         //Timer Interface
#include "LEDs.h"          //Interface for LED's extension
#include "dipSwitch.h"     //Interface for Dip switch extension
#include "spi.h"           //Data interface for SPI bus to AD7758
#include "uart.h"          //UART interface part
#include "flash.h"         //Flash header file for storing and retrieving info in the flash memory
```

```
//External Periphery
#include "ad7758.h"        //Energy Measurement Chip
#include "rs232.h"         //Interface to communicate via RS232 for callibration purposes
#include "atCommand.h"     //Interface to GSM via the AT command set

//Applications
#include "initMeter.h"     //Sets up all the default values and starts the various systems
#include "energy.h"        //Driver responsible for servicing the energy measurement registers
#include "onLine.h"        //Operation in an online RS232 environment
#include "gsmMode.h"       //Opreation in GSM mode.

//Protocols
//#include "gsm.h"            //GSM modem module

//StateEventSysem
//#include "checkMeter.h"     //checkMeter state machine
//#include "stateEvent.h"     //StateEvent Machine
#include "testStateEvent.h"   //Tests or dummy state event machine

//
//  MAIN
//
// The main procedure groups all the functions of the checkmeter together in
//    the relevent layers and correct sequence
// The first part sets up the MCU and ensures that the clock is setup
//    correctly. This is done to ensure correct operation of the MCU.
// Secondly the hardware is setup to ensure that the software can interact
//    with the hardware. This is done in the MCU and only uses the MCU's
//    registers.
// The protocols section adresses all the various external components.
//
// The state event machine services each of the drivers for the various layers
//    and based on the users setting (external jumpers) the correct operation
//    for either GSM or RS232 mode is sected.
//
// MODES:
//    The RS232 mode is used for callibration and debugging purposes
//    The GSM mode is when the unit is out in the field, to give feedback
//    to the server.
//
//------------------------------------------------------------------------------
int main(void) {
   // MCU and software setup
   WDTCTL = WDTPW + WDTHOLD;        // Stop Watch Dog Timer (WDT)
   CLOCK_startXTAL();               // Setup correct operation for timer

   //MCU internal periphery
   LEDs_init();                     // LED setup
   DIPs_init();                     // DIP's Input setup
   TIMER_init();                    // Initialize System Timer

   //State event machine
   if (DIPs_isDIP7() == 0) {
      mode = MODE_GSM;
   } else {
      mode = MODE_RS232;
   }
   // mode = MODE_test;

   for (;;) {                 // State Event Machine
      //Main Loop Heartbeat
      LEDs_toggleLED6();
         // insert watchdog implementation here...

      //MCU internal periphery
      LEDs_driver();
      DIPs_driver();
      SPI_driver();
      UART_driver();
      FLASH_driver();
      //TIMER_driver();

      //External Hardware Communications
      AD7758_driver();

      //Application Drivers
      switch (mode) {   //mode is determined by the DIP switch/jumper setting
         case MODE_test:
            test_StateEventMachine();
```

```
            break;
         case MODE_RS232:
            RS232_driver();
            OnLine_driver();
            break;
         case MODE_GSM:
            Energy_Driver();
            AT_driver();
            GSM_driver();
            break;
      }
   }
}
```

R.2 MCU and System

R.2.1 memory.h

All the global variables used throughout the system is declared here.

```
/*********************************************************************
*  MEMORY
*
*  This header file is contains all the global variables in the
*  system.
*
*  R. Doorduin
*  PEG - Universityo of Stellenbosch
*  March 2004
*********************************************************************/

//Acumulator info - Storage where 1 byte value = 1 Wh/varh/VAh
signed long energy_W_A = 0x0;
signed long energy_W_B = 0x0;
signed long energy_W_C = 0x0;
signed long energy_VA_A = 0x0;
signed long energy_VA_B = 0x0;
signed long energy_VA_C = 0x0;
signed long energy_VAR_A = 0x0;
signed long energy_VAR_B = 0x0;
signed long energy_VAR_C = 0x0;

//Per phase registers for storage of uncorrected data
signed long energy_W_a = 0x0;
signed long energy_W_b = 0x0;
signed long energy_W_c = 0x0;
signed long energy_VA_a = 0x0;
signed long energy_VA_b = 0x0;
signed long energy_VA_c = 0x0;
signed long energy_VAR_a = 0x0;
signed long energy_VAR_b = 0x0;
signed long energy_VAR_c = 0x0;


//Calibration details
unsigned char AD7758_overFlow = 0x0;

signed long MCU_WDIVA = 0x01;
signed long MCU_WDIVB = 0x01;
signed long MCU_WDIVC = 0x01;
signed long MCU_VADIVA = 0x01;
signed long MCU_VADIVB = 0x01;
signed long MCU_VADIVC = 0x01;
signed long MCU_VARDIVA = 0x01;
signed long MCU_VARDIVB = 0x01;
signed long MCU_VARDIVC = 0x01;

//Mode definitions
#define MODE_test                       0x000
#define MODE_RS232                      0x001
#define MODE_GSM                        0x002
int mode = MODE_test;           //Opereation mode of state event machine
```

R.2.2 clock.h

Ensures that the clock is sets up correctly.

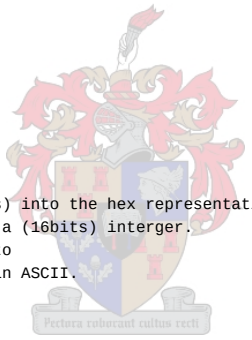
```
/*****************************************************************
*  CLOCK
*
*  This header file is intended for easy interfacing with the clock
*  system. The external crystal frequency is 7.327MHz. This is
*  configured on XIN, XOUT/TCLK pins on the MSP430F149 MCU.
*
*  R. Doorduin
*  PEG - Universityo of Stellenbosch
*  March 2004
*****************************************************************/

//  CLOCK_outputMCLK
//
// Enables the user to put the clock MCLK out on pin 5.4.
// This is done for testing purposes.
//---------------------------------------------------------------
void CLOCK_outputMCLK(void) {
  P5SEL |= 0x10;
  P5DIR |= 0x10;                        // Set P5.4 to output direction
}

//  CLOCK_outputSMCLK
//
// Enables the user to put the clock SMCLK out on pin 5.5.
// This is done for testing purposes.
//---------------------------------------------------------------
void CLOCK_outputSMCLK(void) {
  P5SEL |= 0x20;
  P5DIR |= 0x20;                        // Set P5.5 to output direction
}

//  CLOCK_startXTAL
//
// This enables the use of the first external clock input.
// This procedure is based on the TI datasheets.
//---------------------------------------------------------------
void CLOCK_startXTAL(void) {

  unsigned int j;
  BCSCTL1 |= XTS;                       // ACLK = LFXT1 = HF XTAL

  do {
    IFG1 &= ~OFIFG;                     // Clear OSCFault flag
    for (j = 0xFF; j > 0; j--);         // Time for flag to set
  } while ((IFG1 & OFIFG) == OFIFG);

  BCSCTL2 |= SELM1+SELM0;               // MCLK = LFXT1 (safe)

  //Outputs the clocks on the external pins.
  CLOCK_outputMCLK();
  CLOCK_outputSMCLK();
}
```

R.2.3 utils.h

Handy procedures that can be used troughout the system is implemented here.

```
//*******************************************************************************
//
// Utils
//
// General Purpose utility type functions
//
// History:
// v1.0 - 2004-07-24 Creation of Driver
//
//  R. Doorduin
//  PEG - University of Stellenbosch
//  March 2004
```

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

//  signedIntToLong
//
// When converting a signed integer to a long, the leading FF's are ignored.
// This handy procedure adds the leading F's for 2's compliment to the long.
//
//-----------------------------------------------------------------------------
signed long signedIntToLong(signed int i) {
   signed long tmpLong= 0x0;
   tmpLong = i;
   if (i < 0) {
      tmpLong = tmpLong | 0x0FFFF0000;
   }
   return tmpLong;
}

//  strCmp
//
// Standard procedure to compare two strings. Returns a positive value
//    representing the extra length if the first variable is longer than the
//    second of the compared or a negative value
//    representing the additional length of the second variable
//
// e.g   PIET = PIET    return 0
//       PIETER = PIET  return 2
//       PIET = PIETER  return -2
//
//-----------------------------------------------------------------------------
int strCmp(char *s,char *t) {
   for( ; *s == *t; s++, t++) {
      if (*s == '\0') {
         return 0;
      }
   }
   return *s - *t;
}

// charToHex
//
// This function converts the character (8bits) into the hex representation of
//    two Hexadecimal text equiwivalents into a (16bits) interger.
// e.g. a charachter 0x0AB will be converted to
//          0x04748 which translates to "AB" in ASCII.
//
// Returns:
//    0 if character is read
//    -1 if charcater is not read.
//
//-----------------------------------------------------------------------------
unsigned int charToHex(unsigned char c) {
   int i = 0x0;
   i = c;
   i = i >> 4;
   i = i & 0x0F;
   if (i < 0x0A) {
      i = i + 0x030;
   } else if (i < 0x010) {
      i = i + 0x037;
   } else {
      i = 0;
   }
   i = i << 8;
   c = c & 0x0F;
   if (c < 0x0A) {
      c = c + 0x030;
   } else if (c < 0x010) {
      c = c + 0x037;
   } else {
      c = 0;
   }
   i = i + c;
   return i;
}
```



R.3 MCU periphery

R.3.1 timer.h

```
/*********************************************************************
*  TIMER
*
*  This header file is intended for easy interfacing with the timing
*  system
*
*  History
*  v1.0  - 2004-07-02
*  R. Doorduin
*  PEG - University of Stellenbosch
*********************************************************************/

// Timers declerations and values
//
// Various timers are created and their corresponding variables and values are
//    tracked here.
// (Values in milli seconds(ms))
//
//-----------------------------------------------------------------------------
//WDTsLEDs
#define delayTestSE              250
#define  tmrTestSE                     0x00
#define delayWDTGSM              500
#define  tmrWDTGSM                     0x01
#define delayWDTOnLine           125
#define  tmrWDTOnLine                  0x02
//AD7758
#define delayAD7758_Clear        1
#define  tmrAD7758_Clear               0x03
//UART
#define delayRS232on_off         25
#define  tmrRS232on_off                0x04
//AT
#define delayAT_Error_delay      3000
#define  tmrAT_Error_delay             0x05
#define delayAT_SMS_message      1000
#define  tmrAT_SMS_message             0x06
#define delayAT_SMS_messageWait  10000
#define  tmrAT_SMS_messageWait         0x07
//GSM
#define delayGSM_On_Off_Pulse    1250
#define  tmrGSM_On_Off_Pulse           0x08
#define delayGSM_WritePIN_delay  4000
#define  tmrGSM_WritePIN_delay         0x09

//Energy Measurement
#define delayEnergyMeasurement   100
#define  tmrEnergyMeasurement          0x0A

/*
#define delayGSM_PIN_Active      5000
#define  tmrGSM_PIN_Active             0x05
#define delayUART_Switch         50
#define  tmrUART_Switch                0x014
#define delayGSM_TimeOut         2000
#define  tmrGSM_TimeOut                0x015
#define delayGSM_SMS_TimeOut     10000
#define  tmrGSM_SMS_TimeOut            0x016
#define delayGSM_short_delay     750
#define  tmrGSM_short_delay            0x018*/

#define TIMER_NumberOfTimers 0x0B
#define TIMER_InactiveValue 0x0FFFFFFF
unsigned long TIMER_timers[TIMER_NumberOfTimers];

// TIMER_startTimer
//
// Starts the timer specified
//
// Expects:
//    timer, the timer number which should start
//
//-----------------------------------------------------------------------------
```

```
void TIMER_startTimer(int timer) {
      TIMER_timers[timer] = 0x0;
}

// TIMER_stopTimer
//
// Stop the timer specified
//
// Expects:
//    timer, the timer number which should be stopped
//
//---------------------------------------------------------------------------
void TIMER_stopTimer(int timer) {
      TIMER_timers[timer] = TIMER_InactiveValue;
}

// TIMER_isExpired
//
// Checks if the timer is completed or expired.
//
// Expects:
//    timer, the timer number which should be stopped
//    expireTime, the time the timer is supposed to be expiring
//
//---------------------------------------------------------------------------
int TIMER_isExpired(int timer, unsigned long expireTime){
   if (TIMER_timers[timer] > expireTime) {
      TIMER_stopTimer(timer);
      return 0;
   } else {
      return -1;
   }
}

// TIMER_clearAll
//
// Clears all the timers and set their state to TIMER_InactiveValue
//
//---------------------------------------------------------------------------
void TIMER_clearAll(void) {
   int i = 0x0;
   for (i; i < TIMER_NumberOfTimers; i++) {
      TIMER_timers[i] = TIMER_InactiveValue;
   }
}

//
//  TIMER_init
//
// The timer is used to time internal events in the system.
// This procedure sets up all the interupts.
//
//---------------------------------------------------------------------------
void TIMER_init(void) {
   TACTL |= TASSEL1+TACLR;    //Timer source = SMCLK, clear TAR
   TACCR0 = 0x00FFF;          //Timer Period
   CCTL0 = CCIE;
   CCR0 = 737;                //For 1ms interupts.
   TACTL |= BIT4;             //Start timer in UP modes

   TIMER_clearAll();          //Clears all the timer values

   eint();                    //enable interupts
}


// TIMER_driver
//
// Service all the timers (currently no actions here...)
//
//---------------------------------------------------------------------
void TIMER_driver(void) {
   //Remember to switch on driver if anything is added here in main.c
}

//  TIMER_incrementTimers
//
// This procedure increment all the various timers.
//
```

```
//-------------------------------------------------------------------------
void TIMER_incrementTimers(void) {
   int i = 0x0;
   for (i; i < TIMER_NumberOfTimers; i++) {
      //if (TIMER_timers[i] != TIMER_InactiveValue) {
         TIMER_timers[i]++;
      //}
   }
}

// Timer_A
//
// The interupt routine for handling timer events
//
//-------------------------------------------------------------------------
interrupt (TIMERA0_VECTOR) Timer_A(void) {
   TIMER_incrementTimers();
}
```

R.3.2 leds.h

```
/*********************************************************************
*  LED's
*
*  This header file is intended to easy interfacing with the LED's
*  The LEDs are connected on the following pins of the MSP430F149
*     MCU:
*  LED0  = P1.0
*  LED1  = P1.1
*  LED2  = P1.2
*  LED3  = P1.3
*  LED4  = P1.5
*  LED5  = P1.6
*  LED6  = P1.7
*
*  R. Doorduin
*  PEG - Universityo of Stellenbosch
*  March 2004
*********************************************************************/

#define LED0 BIT0 //GSM Module active
#define LED1 BIT1
#define LED2 BIT2
#define LED3 BIT3
#define LED4 BIT5
#define LED5 BIT6
#define LED6 BIT7
#define LEDs P1OUT

unsigned char LEDstate = 0x00;

//  LEDs_init
//
// Initialize the LEDS attached to the device
//
//-------------------------------------------------------------------------
void LEDs_init(void) {
  P1DIR |= LED0+LED1+LED2+LED3+LED4+LED5+LED6;     // Set P1.0,1,2,3,5,6,7 to output direction
  LEDstate = (LED0+LED1+LED2+LED3+LED4+LED5+LED6); // Set all outputs off.
}

//  LEDs_driver
//
// Updates the LEDs according to last setting in the LEDstate
//
//-------------------------------------------------------------------------
void LEDs_driver(void) {
  LEDs = LEDstate;
}

// LEDs_resetLEDx
//
// Sets the LEDs to the off state
//
//-------------------------------------------------------------------------
void LEDs_resetLED0(void) {
  LEDstate |= LED0;
```

```
}
void LEDs_resetLED1(void) {
  LEDstate |= LED1;
}
void LEDs_resetLED2(void) {
  LEDstate |= LED2;
}
void LEDs_resetLED3(void) {
  LEDstate |= LED3;
}
void LEDs_resetLED4(void) {
  LEDstate |= LED4;
}
void LEDs_resetLED5(void) {
  LEDstate |= LED5;
}
void LEDs_resetLED6(void) {
  LEDstate |= LED6;
}

// LEDs_setLEDx
//
// Sets the LEDs to the on state
//
//-----------------------------------------------------------------------------
void LEDs_setLED0(void) {
  LEDstate &= ~LED0;
}
void LEDs_setLED1(void) {
  LEDstate &= ~LED1;
}
void LEDs_setLED2(void) {
  LEDstate &= ~LED2;
}
void LEDs_setLED3(void) {
  LEDstate &= ~LED3;
}
void LEDs_setLED4(void) {
  LEDstate &= ~LED4;
}
void LEDs_setLED5(void) {
  LEDstate &= ~LED5;
}
void LEDs_setLED6(void) {
  LEDstate &= ~LED6;
}

// LEDs_toggleLEDx
//
// Toggles the LEDs either on or off depending on their current state
//
//-----------------------------------------------------------------------------
void LEDs_toggleLED0(void) {
  LEDstate ^= LED0;
}
void LEDs_toggleLED1(void) {
  LEDstate ^= LED1;
}
void LEDs_toggleLED2(void) {
  LEDstate ^= LED2;
}
void LEDs_toggleLED3(void) {
  LEDstate ^= LED3;
}
void LEDs_toggleLED4(void) {
  LEDstate ^= LED4;
}
void LEDs_toggleLED5(void) {
  LEDstate ^= LED5;
}
void LEDs_toggleLED6(void) {
  LEDstate ^= LED6;
}
```

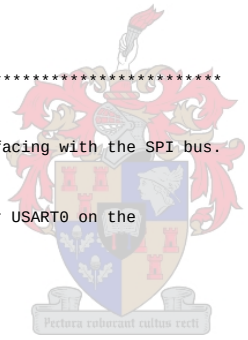
R.3.3 dipswitch.h

```
/*********************************************************************
*  DIP SWITCH
*
*  This header file is intended to easy interfacing with the DIP
*  switches. The dipswitches/jumpers are connected on the following
*  pins of the MSP430F149 MCU:
*  DIP0  = BIT0
*  DIP1  = BIT1
*  DIP2  = BIT2
*  DIP3  = BIT3
*  DIP4  = BIT4
*  DIP5  = BIT5
*  DIP6  = BIT6
*  DIP7  = BIT7
*
*  History
*  v1.0 - 2004-07-02 - Enabling of DIPswitch and debounce handling
*
*  R. Doorduin
*  PEG - Universityo of Stellenbosch
*********************************************************************/

#define DIP0 BIT0
#define DIP1 BIT1
#define DIP2 BIT2
#define DIP3 BIT3
#define DIP4 BIT4
#define DIP5 BIT5
#define DIP6 BIT6   //GSM Mode or RS232 Mode
#define DIP7 BIT7 //Measurement Enabled or Disabled
#define DIPs P4IN

unsigned char DIPstate = 0x00;

//  DIPs_init
//
// Initialize and setup input from the jumper or DIP switches
//
//-------------------------------------------------------------------------------
void DIPs_init() {
  P4DIR = 0x00;                          // Set P4 for input
  DIPstate = DIPs;
}

//  DIPs_driver
//
// Since this is only an input method, this driver only generates events for
//    the state event machine.
// To do:
//    Debouncing
//    Handling of more than one change concurrently
//-------------------------------------------------------------------------------
void DIPs_driver(void) {
   if (DIPs != DIPstate) {
/*    switch (DIPs ^ DIPstate) {
         case DIP0:
            if ((DIPs & DIP0) != 0x00) {
               event = evDIP0on;
            } else {
               event = evDIP0off;
            }
         break;
         case DIP1:
            if ((DIPs & DIP1) != 0x00) {
               event = evDIP1on;
            } else {
               event = evDIP1off;
            }
         break;
         case DIP2:
            if ((DIPs & DIP2) != 0x00) {
               event = evDIP2on;
            } else {
               event = evDIP2off;
            }
         break;
         case DIP3:
```

```
            if ((DIPs & DIP3) != 0x00) {
               event = evDIP3on;
            } else {
               event = evDIP3off;
            }
         break;
         case DIP4:
            if ((DIPs & DIP4) != 0x00) {
               event = evDIP4on;
            } else {
               event = evDIP4off;
            }
         break;
         case DIP5:
            if ((DIPs & DIP5) != 0x00) {
               event = evDIP5on;
            } else {
               event = evDIP5off;
            }
         break;
         case DIP6:
            if ((DIPs & DIP6) != 0x00) {
               event = evDIP6on;
            } else {
               event = evDIP6off;
            }
         break;
         case DIP7:
            if ((DIPs & DIP7) != 0x00) {
               event = evDIP7on;
            } else {
               event = evDIP7off;
            }
         break;
      }*/
   }
   DIPstate = DIPs;
}

//  DIPs_isDIPx
//
// These functions allows the user to test for the various state of a DIPswitch.
// The functions returns a zero if the switch is left open.
//
// Returns:
//    0 Open
//    -1 Closed
//
//-------------------------------------------------------------------------------
int DIPs_isDIP0(void) {
   if ((DIPstate & DIP0) != 0) {
      return 0;
   } else {
      return -1;
   }
}

int DIPs_isDIP1(void) {
   if ((DIPstate & DIP1) != 0) {
      return 0;
   } else {
      return -1;
   }
}

int DIPs_isDIP2(void) {
   if ((DIPstate & DIP2) != 0) {
      return 0;
   } else {
      return -1;
   }
}
int DIPs_isDIP3(void) {
   if ((DIPstate & DIP3) != 0) {
      return 0;
   } else {
      return -1;
   }
}
```

```
int DIPs_isDIP4(void) {
   if ((DIPstate & DIP4) != 0) {
      return 0;
   } else {
      return -1;
   }
}
int DIPs_isDIP5(void) {
   if ((DIPstate & DIP5) != 0) {
      return 0;
   } else {
      return -1;
   }
}
int DIPs_isDIP6(void) {
   if ((DIPstate & DIP6) != 0) {
      return 0;
   } else {
      return -1;
   }
}
int DIPs_isDIP7(void) {
   if ((DIPstate & DIP7) != 0) {
      return 0;
   } else {
      return -1;
   }
}
```

R.3.4 spi.h




```
/*********************************************************************
*  SPI
*
*  This header file is intended to easy interfacing with the SPI bus.
*  system
*
*  On the system the SPI bus is configured for USART0 on the
*     MSP430F149 MCU:
*  SPI_OUT  = P3.1
*  SPI_IN   = P3.2
*  SPI_CLK  = P3.3
*
*  History:
*  v1.0 - 2004-04-01 - Creation of simple interface
*  v2.0 - 2004-07-05 - Event driven driver development
*
*  R. Doorduin
*  PEG - Universityo of Stellenbosch
*********************************************************************/

#include <io.h>
// State Event Declerations
//-------------------------------------------------------------------
//States
#define SPI_stInit               0x00
#define SPI_stIdle               0x01
#define SPI_stWaitForWriteClear  0x02
#define SPI_stInitiateRead       0x03
#define SPI_stWaitForReadClear   0x04

//Variables
int SPI_state = SPI_stInit;           //The current state within a the SPI driver

#define SPI_InitFlag          BIT0
#define SPI_NewCharToWrite    BIT1
#define SPI_NewCharToRead     BIT2
#define SPI_NewReadCharReady  BIT3
int SPI_flags = 0x00;                 //Flags to allow the driver to keep track of user input

unsigned char SPI_lastReadChar = 0x0;
unsigned char SPI_charToWrite = 0x0;


//  SPI_init
//
```

```
// Intialize the SPI interface on UART0
//
//-------------------------------------------------------------------------
void SPI_init(void) {
  P3SEL |= 0x0E;                //P3.1,2,3 SPI option select

  U0CTL |= SWRST;               //Software Restart
  U0CTL |= MM+SYNC+CHAR;        //Enable 8bit SPI master
  U0TCTL |= STC+SSEL0+SSEL1;    //Transmit Control register
  U0BR0  = 0x0F;                //Set SPI Baud Rate
  U0BR1  = 0x00;                //MCLK/2 baud rate
  U0MCTL = 0x00;                //No modulation for SPI
  ME1 |= USPIE0;                //Enable UART for SPI mode
  U0CTL &= ~SWRST;              //Software Restart

  SPI_flags |= SPI_InitFlag;     //Set Initialization done
}

// SPI_write
//
// Procedure to let the driver know that a new charachter is ready to write on the
// SPI bus.
//
// Returns:
//    0 if successful que placement
//    -1 if a character is already placed for a write itteration
//
//-------------------------------------------------------------------------
int SPI_write(unsigned char c) {
   if ((SPI_flags & (SPI_NewCharToRead+SPI_NewCharToWrite)) == 0x00) {
      SPI_charToWrite = c;
      SPI_flags |= SPI_NewCharToWrite; //To let driver know that a new char is ready
      return 0;
   } else {
      return -1;
   }
}

//  SPI_isWriting
//
// Indicates to users that the charachter is in the process of writing
//
//-------------------------------------------------------------------------
int SPI_isWriting() {
   if (SPI_state == SPI_stWaitForWriteClear) {
      return 0;
   } else {
      return -1;
   }
}

// SPI_read
//
// Procedure to let the driver know that an charchter is required to be read from
// the SPI bus.
//
// Returns:
//    0 if the driver is not currently busy with a read operation
//    -1 if a character is already requested for a read operation
//
//-------------------------------------------------------------------------
int SPI_read(void) {
   if ((SPI_flags & (SPI_NewCharToRead+SPI_NewCharToWrite)) == 0x00) {
      SPI_flags |= SPI_NewCharToRead; //To let driver know that a new char is ready
      return 0;
   } else {
      return -1;
   }
}

// SPI_isCharReady
//
// By calling this procedure the user can establish if a character is read correcty
//
// Returns:
//    0 if character is read
//    -1 if charcater is not read.
//
//-------------------------------------------------------------------------
```

```
int SPI_isCharReady(void) {
   if ((SPI_flags & SPI_NewReadCharReady) != 0x00) {
      return 0;
   } else {
      return -1;
   }
}

// SPI_getChar
//
// Returns the last read character
//
//-----------------------------------------------------------------------------
unsigned char SPI_getChar(void) {
   SPI_flags &= ~SPI_NewReadCharReady;
   return SPI_lastReadChar;
}

//  SPI_isNotIdle
//
// Indicates to users that the charachter is not in the idle state
//
//-----------------------------------------------------------------------------
int SPI_isNotIdle() {
   if (SPI_state != SPI_stIdle) {
      return 0;
   } else {
      return -1;
   }
}


// SPI_driver
//
// The driver is a state event machine and is the only software component
//    communicating on hardware level with the SPI bus.
//
//-----------------------------------------------------------------------------
void SPI_driver(void) {
   // Monitor events
   // Execute state
   switch (SPI_state) {

      case SPI_stInit:
         SPI_init();
         SPI_state = SPI_stIdle;
         break;
      case SPI_stIdle:
         if ((SPI_flags & SPI_InitFlag) == 0x0) {                //Ensure that driver is initialized
            SPI_state = SPI_stInit;
         } else if ((SPI_flags & SPI_NewCharToWrite) != 0x0) {    //Check if write action is required
            U0TXBUF = SPI_charToWrite;
            SPI_state = SPI_stWaitForWriteClear;
         } else if ((SPI_flags & SPI_NewCharToRead) != 0x0) {     //Check if a read event did occur
            U0TXBUF = 0x00;
            SPI_state = SPI_stInitiateRead;
         }
         break;
      case SPI_stWaitForWriteClear:                //Write state... Waits for a cleared buffer.
         if ((IFG1 & UTXIFG0) != 0x0) {
            IFG1 &= ~UTXIFG0;
            SPI_flags &= ~SPI_NewCharToWrite;
            SPI_state = SPI_stIdle;
         }
         break;
      case SPI_stInitiateRead:                     //Read initiate state
         if ((IFG1 & UTXIFG0) != 0x0) {
            IFG1 &= ~UTXIFG0;
            IFG1 &= ~URXIFG0;
            SPI_state = SPI_stWaitForReadClear;
         }
         break;
      case SPI_stWaitForReadClear:                 //Wait for complete read operation
         if ((IFG1 & URXIFG0) != 0x00) {
            IFG1 &= ~URXIFG0;
            SPI_lastReadChar = U0RXBUF;
            SPI_flags |= SPI_NewReadCharReady;
            SPI_flags &= ~SPI_NewCharToRead;
            SPI_state = SPI_stIdle;
```

```
        }
        break;
      default:
        SPI_state = SPI_stInit;
        break;
   }
}
```

R.3.5 uart.h

```
/************************************************************************
*  UART
*
*  This header file is intended to easy interfacing with the UART
*  system for serial communications.
*  The UART is configured for the MSP430F149 MCU with the following
*     pins used:
*  TX = P3.6
*  RX = P3.7
*
*  The MAX3222C is connected to the UART for communicating on RS232.
*  The MAX is connected via the following pins:
*  /ENABLE  = P2.1
*  /PWRDWN  = P2.0
*  The MAX is controlled with the UART_setRS232on() and
*     UART_setRS232off() functions.
*
*  The following functions are used for interfacing with the UART driver
*     WRITE ops:
*        int UART_writeChar(unsigned char c)
*        int UART_writeInt(unsigned int c)
*        int UART_writeLong(unsigned long c)
*        int UART_writeString(unsigned char *string)
*        unsigned int UART_charToHex(unsigned char c)
*     READ ops:
*        int UART_isCharReady(void)
*        unsigned char UART_getChar(void)
*     RS232 ops:
*        void UART_setRS232on(void)
*        void UART_setRS232off(void)
*        int UART_isRS232on(void)
*        int UART_isRS232off(void)
*     Driver:
*        voide UART_driver(void)
*
*  History:
*  v 1.0 - 2004-04 Initial configuration and primitive functionality
*  v 2.0 - 2004-07 Change to state machine/driver implementation
*
*  R. Doorduin
*  PEG - University of Stellenbosch
************************************************************************/

/*    This refference is included to help the programmer...
 * \a - alert bell
 * \b - backspace
 * \f - formfeed
 * \n - new line
 * \r - carriage return
 * \t - horizontal tab
 * \v - vertical tab
 * \\ - backslash
 * \? - question mark
 * \' - single quote
 * \" - double quote
 * \ooo - ocatal number
 * \xhh - hex number
 *
 */

#include <io.h>
#include <signal.h>

// State Event Declerations
//------------------------------------------------------------------
//States
```

```
#define UART_stInit             0x000
#define UART_stIdle             0x001
#define UART_stWaitRS232on      0x002
#define UART_stWaitRS232off     0x003
#define UART_stWriteChar        0x004
#define UART_stWriteInt         0x005
#define UART_stWriteLong1       0x006
#define UART_stWriteLong        0x007
#define UART_stWriteString      0x008
#define UART_stWriteLongHex0    0x009
#define UART_stWriteLongHex1    0x00A
#define UART_stWriteLongHex2    0x00B
#define UART_stWriteLongHex3    0x00C
#define UART_stWriteLongHex4    0x00D
#define UART_stWriteLongHex5    0x00E
#define UART_stWriteLongHex6    0x00F
#define UART_stWriteLongHex7    0x010


//Variables
int UART_state = UART_stInit;          //The current state within a the UART driver

//Flags
#define UART_InitFlag           BIT0
#define UART_NewCharToWrite     BIT1
#define UART_NewIntToWrite      BIT2
#define UART_NewLongToWrite     BIT3
#define UART_NewStringToWrite   BIT4
#define UART_NewLongHexToWrite  BIT5
#define UART_NewReadCharReady   BIT6
#define UART_isRS232onFlag      BIT7
#define UART_isRS232offFlag     BIT8
#define UART_RS232on            BIT9
#define UART_RS232off           BITA
int UART_flags = 0x00;                 //Flags to allow the driver to keep track of user input

unsigned char UART_lastReadChar = 0x0;
unsigned char UART_char1ToWrite = 0x0;
unsigned char UART_char2ToWrite = 0x0;
unsigned char UART_char3ToWrite = 0x0;
unsigned char UART_char4ToWrite = 0x0;

char *UART_stringToWrite = "";

//  UART_init
//
// Intialize the UART interface on UART1.
//
//-------------------------------------------------------------------------------
void UART_init(void) {
   //------Generic setup for
   P3SEL |= 0x0C0;              //P3.6,7 UART option select  1100 0000b
   P3DIR |= 0x040;              //Enable output for P3.6;

   U1CTL = SWRST;               //Software Restart
   U1CTL |= CHAR;               //Enable 8bit charachter mode
   U1TCTL |= SSEL0;             //Transmit Control register / UCLK = MCLK
   U1RCTL = 0x00;               //Clear Receive control regitster
   U1BR0 = 0x00;                // CLK / BAUD, e.g. 7372000/9600 = 0x300
   U1BR1 = 0x03;
   U1MCTL = 0x00;               //No modulation
   ME2 |= UTXE1+URXE1;          //Enable USART1 TXD/RXD
   IE2 |= URXIE1;
   U1CTL &= ~SWRST;             //Software Restart Clear

   //-----Implementation specific----
   P2DIR = 0x007;               // Control lines for selection between RS2323 and GSM
   P2OUT = 0x002;               // Default

   UART_flags |= UART_InitFlag;
}

// UART_writeChar
//
// Procedure to let the driver know that a new charachter(8bits) is ready to write on the
// UART bus.
//*   The following functions are used for interfacing with the UART driver

// Returns:    case 0x04:     //Write special charchters out
```

```
//    0 if successful que placement
//    -1 if a character is already placed for a write itteration
//
//-----------------------------------------------------------------------------
int UART_writeChar(unsigned char c) {
   if ((UART_flags & (UART_NewCharToWrite + UART_NewIntToWrite + UART_NewLongToWrite + UART_NewLongHexToWrite + UART_NewStringToWrite)) == 0x0
      UART_char1ToWrite = c;
      UART_flags |= UART_NewCharToWrite; //To let driver know that a new char is ready
      return 0;
   } else {
      return -1;
   }
}

// UART_writeInt
//
// Procedure to let the driver know that a new integer(16bits) is ready to write on the
// UART bus.
//
// Returns:
//    0 if successful que placement
//    -1 if a character is already placed for a write itteration
//
//-----------------------------------------------------------------------------
int UART_writeInt(unsigned int c) {
   if ((UART_flags & (UART_NewCharToWrite + UART_NewIntToWrite + UART_NewLongToWrite + UART_NewLongHexToWrite + UART_NewStringToWrite)) == 0x0
      UART_char2ToWrite = c >> 8;
      UART_char1ToWrite = c;
      UART_flags |= UART_NewIntToWrite; //To let driver know that a new char is ready
      return 0;
   } else {
      return -1;
   }
}

// UART_writeLong
//
// Procedure to let the driver know that a new Long(32bits) is ready to write on the
// UART bus.
//
// Returns:*   The following functions are used for interfacing with the UART driver

//    0 if successful que placement
//    -1 if a character is already placed for a write itteration
//
//-----------------------------------------------------------------------------
int UART_writeLong(unsigned long c) {
   if ((UART_flags & (UART_NewCharToWrite + UART_NewIntToWrite + UART_NewLongToWrite + UART_NewLongHexToWrite + UART_NewStringToWrite)) == 0x0
      UART_char4ToWrite = c >> 24;
      UART_char3ToWrite = c >> 16;
      UART_char2ToWrite = c >> 8;
      UART_char1ToWrite = c;
      UART_flags |= UART_NewLongToWrite; //To let driver know that a new char is ready
      return 0;
   } else {
      return -1;
   }
}

// UART_writeLongHex
//
// Procedure to let the driver know that a new Long(32bits) is ready to be writen to the
// UART bus in HEX format.
//
// Returns:
//
//    0 if successful que placement
//    -1 if a character is already placed for a write itteration
//
//-----------------------------------------------------------------------------
int UART_writeLongHex(unsigned long c) {
   if ((UART_flags & (UART_NewCharToWrite + UART_NewIntToWrite + UART_NewLongToWrite + UART_NewLongHexToWrite + UART_NewStringToWrite)) == 0x0
      UART_char4ToWrite = c >> 24;
      UART_char3ToWrite = c >> 16;
      UART_char2ToWrite = c >> 8;
      UART_char1ToWrite = c;
      UART_flags |= UART_NewLongHexToWrite; //To let driver know that a new char is ready
      return 0;
   } else {
```

```
      return -1;
   }
}
// UART_writeString
//
// Procedure to let the driver know that a new Long(32bits) is ready to write on the
// UART bus.
//
// Returns:
//    0 if successful que placement
//    -1 if a character is already placed for a write itteration
//
//-----------------------------------------------------------------------------
int UART_writeString(unsigned char *string) {
   if ((UART_flags & (UART_NewCharToWrite + UART_NewIntToWrite + UART_NewLongToWrite + UART_NewLongHexToWrite + UART_NewStringToWrite)) == 0x0
      UART_stringToWrite = string;
      UART_flags |= UART_NewStringToWrite; //To let driver know that a new char is ready
      return 0;
   } else {
      return -1;
   }
}

// UART_isCharReady
//
// By calling this procedure the user can establish if a character is read correcty
//
// Returns:
//    0 if character is read
//    -1 if charcater is not read.
//
//-----------------------------------------------------------------------------
int UART_isCharReady(void) {
   if ((UART_flags & UART_NewReadCharReady) != 0x00) {
      return 0;
   } else {
      return -1;
   }
}

// UART_getChar
//
// Returns the last read character
//
//-----------------------------------------------------------------------------
unsigned char UART_getChar(void) {
   UART_flags &= ~UART_NewReadCharReady;
   return UART_lastReadChar;
}

//  UART_setRS232on
//
// Forces the RS232 Max chip to be on
//
//-----------------------------------------------------------------------------
void UART_setRS232on(void) {
   UART_flags |= UART_RS232on;
}

//  UART_setRS232off
//
// Forces the RS232 to shutdown
//
//-----------------------------------------------------------------------------
void UART_setRS232off(void) {
   UART_flags |= UART_RS232off;
}

//  UART_isRS232on
//
// Checks if the MAX3222C is switched on.
//
//-----------------------------------------------------------------------------
int UART_isRS232on(void) {
   if ((UART_flags & UART_isRS232onFlag) != 0x0) {
      return 0;
   } else {
      return -1;
   }
```

```
}

//  UART_isRS232off
//
// Checks if the MAX3222C is switched off.
//
//---------------------------------------------------------------------------
int UART_isRS232off(void) {
   if ((UART_flags & UART_isRS232offFlag) != 0x0) {
      return 0;
   } else {
      return -1;
   }
}

// UART_driver
//
// The driver is a state event machine and is the only software component
//    communicating on hardware level with the UART serial interface.
//
//---------------------------------------------------------------------------
void UART_driver(void) {
   char c = 0x0;     //For use with the Hex conversion

   if ((IFG2 & URXIFG1) != 0x0) {                       //Checks if UART recieve buffer is full
      UART_lastReadChar = RXBUF1;
      UART_flags |= UART_NewReadCharReady;
   }

   switch (UART_state) {
      case UART_stInit:
         UART_init();
         UART_state = UART_stIdle;
         break;
      case UART_stIdle:
         if ((UART_flags & UART_InitFlag) == 0x0) {
            UART_state = UART_stInit;
         } else if ((UART_flags & UART_NewCharToWrite) != 0x0) {     //Write Char initiated by user
            UART_state = UART_stWriteChar;
         } else if ((UART_flags & UART_NewIntToWrite) != 0x0) {      //Write Int initiated by user
            UART_state = UART_stWriteInt;
         } else if ((UART_flags & UART_NewLongToWrite) != 0x0) {     //Write Long initiated by user
            UART_state = UART_stWriteLong;
         } else if ((UART_flags & UART_NewLongHexToWrite) != 0x0) {  //Write Long Hex initiated by user
            UART_state = UART_stWriteLongHex0;
         } else if ((UART_flags & UART_NewStringToWrite) != 0x0) {   //Write String initiated by user
            UART_state = UART_stWriteString;
         } else if ((UART_flags & UART_RS232on) != 0x0) {            //RS232 requested to be switched on
            P2OUT |= BIT0;                                      //Power down High - On
            P2OUT &= ~BIT1;                                     //Enable Low - Enabled
            TIMER_startTimer(tmrRS232on_off);
            UART_state = UART_stWaitRS232on;
         } else if ((UART_flags & UART_RS232off) != 0x0) {           //RS232 requested to be shut down
//          P2OUT |= BIT0;                                      //Kept on to allow GSM feedback. This line should be disabled for production
            P2OUT &= ~BIT0;                                     //Power down High - On
            P2OUT |= BIT1;                                      //Enable Low - Enabled
            TIMER_startTimer(tmrRS232on_off);
            UART_state = UART_stWaitRS232off;
         }
         break;
      case UART_stWriteChar:                                    //Write Char
         if ((IFG2 & UTXIFG1) != 0x0) {                     //Check if buffer is clear
            TXBUF1 = UART_char1ToWrite;
            UART_flags &= ~UART_NewCharToWrite;
            UART_flags &= ~UART_NewIntToWrite;
            UART_flags &= ~UART_NewLongToWrite;
            UART_state = UART_stIdle;
         }
         break;
      case UART_stWriteInt:                                     //Write Int
         if ((IFG2 & UTXIFG1) != 0x0) {                     //Check if buffer is clear
            TXBUF1 = UART_char2ToWrite;
            UART_state = UART_stWriteChar;
         }
         break;
      case UART_stWriteLong1:                                   //Write second Long byte
         if ((IFG2 & UTXIFG1) != 0x0) {                     //Check if buffer is clear
            TXBUF1 = UART_char3ToWrite;
            UART_state = UART_stWriteInt;
```

```
   }
   break;
case UART_stWriteLong:                                  //Write Long
   if ((IFG2 & UTXIFG1) != 0x0) {                     //Check if buffer is clear
      TXBUF1 = UART_char4ToWrite;
      UART_state = UART_stWriteLong1;
   }
   break;
case UART_stWriteString:                                //Write String
   if ((IFG2 & UTXIFG1) != 0x0) {                     //Check if buffer is clear
      if ((UART_char1ToWrite = *UART_stringToWrite) != '\0') {
         UART_stringToWrite++;
         TXBUF1 = UART_char1ToWrite;
      } else {
         UART_flags &= ~UART_NewStringToWrite;
         UART_state = UART_stIdle;
      }
   }
   break;
case UART_stWriteLongHex0:                                  //Write Long
   if ((IFG2 & UTXIFG1) != 0x0) {                     //Check if buffer is clear
      TXBUF1 = (0x0FF & (charToHex(UART_char4ToWrite) >> 8));
      UART_state = UART_stWriteLongHex1;
   }
   break;
case UART_stWriteLongHex1:                                  //Write Long
   if ((IFG2 & UTXIFG1) != 0x0) {                     //Check if buffer is clear
      TXBUF1 = (0x0FF & charToHex(UART_char4ToWrite));
      UART_state = UART_stWriteLongHex2;
   }
   break;
case UART_stWriteLongHex2:                                  //Write Long
   if ((IFG2 & UTXIFG1) != 0x0) {                     //Check if buffer is clear
      TXBUF1 = (0x0FF & (charToHex(UART_char3ToWrite) >> 8));
      UART_state = UART_stWriteLongHex3;
   }
   break;
case UART_stWriteLongHex3:                                  //Write Long
   if ((IFG2 & UTXIFG1) != 0x0) {                     //Check if buffer is clear
      TXBUF1 = (0x0FF & charToHex(UART_char3ToWrite));
      UART_state = UART_stWriteLongHex4;
   }
   break;
case UART_stWriteLongHex4:                                  //Write Long
   if ((IFG2 & UTXIFG1) != 0x0) {                     //Check if buffer is clear
      TXBUF1 = (0x0FF & (charToHex(UART_char2ToWrite) >> 8));
      UART_state = UART_stWriteLongHex5;
   }
   break;
case UART_stWriteLongHex5:                                  //Write Long
   if ((IFG2 & UTXIFG1) != 0x0) {                     //Check if buffer is clear
      TXBUF1 = (0x0FF & charToHex(UART_char2ToWrite));
      UART_state = UART_stWriteLongHex6;
   }
   break;
case UART_stWriteLongHex6:                                  //Write Long
   if ((IFG2 & UTXIFG1) != 0x0) {                     //Check if buffer is clear
      TXBUF1 = (0x0FF & (charToHex(UART_char1ToWrite) >> 8));
      UART_state = UART_stWriteLongHex7;
   }
   break;
case UART_stWriteLongHex7:                                  //Write Long
   if ((IFG2 & UTXIFG1) != 0x0) {                     //Check if buffer is clear
      TXBUF1 = (0x0FF & charToHex(UART_char1ToWrite));
      UART_flags &= ~UART_NewLongHexToWrite;
      UART_state = UART_stIdle;
   }
   break;
case UART_stWaitRS232on:                                    //Wait for RS232 to be switched on
   if (TIMER_isExpired(tmrRS232on_off,delayRS232on_off) == 0) {
      UART_flags &= ~UART_RS232on;
      UART_flags |= UART_isRS232onFlag;
      UART_flags &= ~UART_isRS232offFlag;
      UART_state = UART_stIdle;
   }
   break;
case UART_stWaitRS232off:                                   //Wait for RS232 to be switched off
   if (TIMER_isExpired(tmrRS232on_off,delayRS232on_off) == 0) {
      UART_flags &= ~UART_RS232off;
```

```
            UART_flags &= ~UART_isRS232onFlag;
            UART_flags |= UART_isRS232offFlag;
            UART_state = UART_stIdle;
          }
          break;
    }
}
```

R.3.6 flash.h

```
/*********************************************************************
*  Flash
*
*  This header file is intended for easy interfacing with the flash
*  module of the MSP430
*
*  The code in this module is based on code supplied by TI.com
*
*  History
*  v1.0  - 2004-05-06 Based on TI's
*  v2.0  - 2004-07-24 Upgrade to State Event Driven
*  R. Doorduin
*  PEG - Universityo of Stellenbosch
*********************************************************************/
//Flash Memory Map
// SegmentB 0x01000-0x107F
// SegmentA 0x01080-0x10FF

#define FLASHMEM_energy_W_A     0x01000
#define FLASHMEM_energy_W_B     0x01004
#define FLASHMEM_energy_W_C     0x01008
#define FLASHMEM_energy_VA_A    0x0100C
#define FLASHMEM_energy_VA_B    0x01010
#define FLASHMEM_energy_VA_C    0x01014
#define FLASHMEM_energy_VAR_A   0x01018
#define FLASHMEM_energy_VAR_B   0x0101C
#define FLASHMEM_energy_VAR_C   0x01020

#define FLASHMEM_MCU_WDIVA      0x01024
#define FLASHMEM_MCU_WDIVB      0x01028
#define FLASHMEM_MCU_WDIVC      0x0102C
#define FLASHMEM_MCU_VADIVA     0x01030
#define FLASHMEM_MCU_VADIVB     0x01034
#define FLASHMEM_MCU_VADIVC     0x01038
#define FLASHMEM_MCU_VARDIVA    0x0103C
#define FLASHMEM_MCU_VARDIVB    0x01040
#define FLASHMEM_MCU_VARDIVC    0x01044

#define FLASHMEM_AVRMSOS        0x01048
#define FLASHMEM_BVRMSOS        0x0104A
#define FLASHMEM_CVRMSOS        0x0104C
#define FLASHMEM_AIRMSOS        0x0104E
#define FLASHMEM_BIRMSOS        0x01050
#define FLASHMEM_CIRMSOS        0x01052

#define FLASHMEM_APHCAL        0x01054
#define FLASHMEM_BPHCAL        0x01055
#define FLASHMEM_CPHCAL        0x01056

// State Event Declerations
//-------------------------------------------------------------------
//States
#define Flash_stInit                0x000
#define Flash_stIdle                0x001
#define Flash_ClearSegment_Init     0x002
#define Flash_ClearSegment_Setup    0x003
#define Flash_ClearSegment_Clear    0x004
#define Flash_WriteChar_Init        0x005
#define Flash_WriteInt_Init         0x006
#define Flash_WriteLong_Init        0x007
#define Flash_Write_End             0x008
//Variables
int Flash_state = Flash_stInit;

//Flags
#define Flash_flagClearSeg       BIT0
#define Flash_flagWriteChar      BIT1
```

```
#define Flash_flagWriteInt       BIT2
#define Flash_flagWriteLong      BIT3
int Flash_flags = 0x00;               //Flags to allow the driver to keep track of user input

char *Flash_charPtr;        //Flash Pointer
int *Flash_intPtr;        //Flash Pointer
long *Flash_longPtr;        //Flash Pointer
char Flash_tmpChar = 0x0;
int  Flash_tmpInt = 0x0;
long Flash_tmpLong = 0x0;


// FLASH_ClearSegA
//
// Clears the values in Segment A. This involves the setting of each byte to
//    0x0FF. In ths function the adress of the segment is set, and the flag
//    indicating that the state machine should perform the clear action
//    is set.
//
// Returns:
//    0 if successful
//    -1 if waiting for current data in the cue.
//
//-------------------------------------------------------------------------------
int FLASH_ClearSegA() {
   if (Flash_flags == 0) {
      Flash_charPtr = (char *) 0x01080; //Initialize flash pointer
      Flash_flags |= Flash_flagClearSeg;
      return 0;
   } else {
      return -1;
   }
}

// FLASH_ClearSegB
//
// Clears the values in Segment B. This involves the setting of each byte to
//    0x0FF. In ths function the adress of the segment is set, and the flag
//    indicating that the state machine should perform the clear action
//    is set.
//
// Returns:
//    0 if successful
//    -1 if waiting for current data in the cue.
//
//-------------------------------------------------------------------------------
int FLASH_ClearSegB() {
   if (Flash_flags == 0) {
      Flash_charPtr = (char *) 0x01000; //Initialize flash pointer
      Flash_flags |= Flash_flagClearSeg;
      return 0;
   } else {
      return -1;
   }
}

// FLASH_readChar
//
// This function reads a character at the specified adress. No wait states are
//    required and therefor this procedure can access the flash directly.
//
// Returns:
//    0 if unsuccessful
//    the data stored at the adress
//
//-------------------------------------------------------------------------------
char FLASH_readChar(unsigned int adress) {
   if (Flash_state == Flash_stIdle) {
       char *Flash_ptr;
       char tmpVal = 0x0;
       Flash_ptr = (char *) adress;

       FCTL3 = FWKEY;          //Clear LOCK bit
       tmpVal = *Flash_ptr;
       FCTL3 = FWKEY + LOCK;   //Set LOCK Bit
       return tmpVal;
   } else {
      return 0;
   }
```

```
}

// FLASH_readInt
//
// This function reads a integer at the specified adress. No wait states are
//    required and therefor this procedure can access the flash directly.
//
// Returns:
//    0 if unsuccessful
//    the data stored at the adress
//
//---------------------------------------------------------------------------
int FLASH_readInt(unsigned int adress) {
   if (Flash_state == Flash_stIdle) {
      int *Flash_ptr;
      int tmpVal = 0x0;
      Flash_ptr = (int *) adress;

      FCTL3 = FWKEY;          //Clear LOCK bit
      tmpVal = *Flash_ptr;
      FCTL3 = FWKEY + LOCK;   //Set LOCK Bit
      return tmpVal;
   } else {
      return 0;
   }
}

// FLASH_readLong
//
// This function reads a long at the specified adress. No wait states are
//    required and therefor this procedure can access the flash directly.
//
// Returns:
//    0 if unsuccessful
//    the data stored at the adress
//
//---------------------------------------------------------------------------
long FLASH_readLong(unsigned int adress) {
   if (Flash_state == Flash_stIdle) {
      long *Flash_ptr;
      long tmpVal = 0x0;

      Flash_ptr = (long *) adress;
      FCTL3 = FWKEY;          //Clear LOCK bit
      tmpVal = *Flash_ptr;
      FCTL3 = FWKEY + LOCK;   //Set LOCK Bit
      return tmpVal;
   } else {
      return 0;
   }
}

// FLASH_writeChar
//
// Writes the specified character to an adress.
// The function sets the flags and adress for correct implementation via the
//    state machine.
//
// Returns:
//    0 if successful
//    -1 if waiting for current data in the cue.
//
//---------------------------------------------------------------------------
int FLASH_writeChar(unsigned int adress, char value) {
   if (Flash_flags == 0) {
      Flash_charPtr = (char *) adress;
      Flash_tmpChar = value;
      Flash_flags |= Flash_flagWriteChar;
      return 0;
   } else {
      return -1;
   }
}

// FLASH_writeInt
//
// Writes the specified integer to an adress.
// The function sets the flags and adress for correct implementation via the
//    state machine.
```

```
//
// Returns:
//    0 if successful
//    -1 if waiting for current data in the cue.
//
//-------------------------------------------------------------------------
int FLASH_writeInt(unsigned int adress, int value) {
   if (Flash_flags == 0) {
      Flash_intPtr = (int *) adress;
      Flash_tmpInt = value;
      Flash_flags |= Flash_flagWriteInt;
      return 0;
   } else {
      return -1;
   }
}

// FLASH_writeLong
//
// Writes the specified long to an adress.
// The function sets the flags and adress for correct implementation via the
//    state machine.
//
// Returns:
//    0 if successful
//    -1 if waiting for current data in the cue.
//
//-------------------------------------------------------------------------
int FLASH_writeLong(unsigned int adress, long value) {
   if (Flash_flags == 0) {
      Flash_longPtr = (long *) adress;
      Flash_tmpLong = value;
      Flash_flags |= Flash_flagWriteLong;
      return 0;
   } else {
      return -1;
   }
}

// FLASH_driver
//
// The driver ensures that the correct actions are implemented according to
//    the set flags.
//
// Returns:
//    0 if successful
//    -1 if waiting for current data in the cue.
//
//-------------------------------------------------------------------------
void FLASH_driver(void) {
   switch (Flash_state) {
      case Flash_stInit:
         FCTL2 = FWKEY+FSSEL1+FN0; //Setup SMCLK/2 for Flash Timing generator
         Flash_state = Flash_stIdle;
         break;
      case Flash_stIdle:
         if ((Flash_flags & Flash_flagClearSeg) != 0) {
            Flash_state = Flash_ClearSegment_Init;
         } else if ((Flash_flags & Flash_flagWriteChar) != 0) {
            Flash_state = Flash_WriteChar_Init;
         } else if ((Flash_flags & Flash_flagWriteInt) != 0) {
            Flash_state = Flash_WriteInt_Init;
         } else if ((Flash_flags & Flash_flagWriteLong) != 0) {
            Flash_state = Flash_WriteLong_Init;
         }
         break;
      case Flash_ClearSegment_Init:
         dint();
         if (BUSY & FCTL3) {
         } else {
            FCTL1 = FWKEY+ERASE;    //Set Erase Bit
            FCTL3 = FWKEY;          //Clear Lock Bit
            *Flash_charPtr = 0;         //Initiate Dummy Write
            Flash_state = Flash_ClearSegment_Clear;
         }
         break;
      case Flash_ClearSegment_Clear:
         if (BUSY & FCTL3) {
         } else {
```

```
            FCTL1 = FWKEY;          //Clear Erase Bit
            FCTL3 = FWKEY + LOCK;   //Set Lock Bit
            Flash_flags = 0x0;      //Reset all flags
            eint();
            Flash_state = Flash_stIdle;
         }
         break;
      case Flash_WriteChar_Init:
         dint();
         if (BUSY & FCTL3) {
         } else {
            FCTL3 = FWKEY;          //Clear LOCK bit
            FCTL1 = FWKEY+WRT;      //Enable WRT bit
            *Flash_charPtr = Flash_tmpChar;
            Flash_state = Flash_Write_End;
         }
         break;
      case Flash_WriteInt_Init:
         dint();
         if (BUSY & FCTL3) {
         } else {
            FCTL3 = FWKEY;          //Clear LOCK bit
            FCTL1 = FWKEY+WRT;      //Enable WRT bit
            *Flash_intPtr = Flash_tmpInt;
            Flash_state = Flash_Write_End;
         }
         break;
      case Flash_WriteLong_Init:
         dint();
         if (BUSY & FCTL3) {
         } else {
            FCTL3 = FWKEY;          //Clear LOCK bit
            FCTL1 = FWKEY+WRT;      //Enable WRT bit
            *Flash_longPtr = Flash_tmpLong;
            Flash_state = Flash_Write_End;
         }
         break;
      case Flash_Write_End:
         if (BUSY & FCTL3) {
         } else {
            FCTL1 = FWKEY;          //Clear WRT Bit
            FCTL3 = FWKEY + LOCK;   //Set LOCK Bit
            Flash_flags = 0x0;
            eint();
            Flash_state = Flash_stIdle;
         }
         break;
   }
}
```

R.4 Communications

R.4.1 ad7758.h

Responsible for correct communications to the ADE7758.

```
/*****************************************************************
*  AD7758
*
*  This header file is intended to easy interfacing with the AD7758
*     energy measurement system, providing routines and all the
*     registers already defined.
*  Although the device is connected on the MSP430F149 using the SPI
*     bus as configured in the SPI.h driver the folling additional
*     pins are used:
*  /CS   = P3.0
*  IRQ   = P2.7   (Currently not implemented)
*
*  The following functions are inteded for user interaction
*     Driver
*     Write
*        int AD7758_write(char c)
*        int AD7758_writeChar(unsigned char reg, unsigned char data)
```

```
*        int AD7758_writeInt(unsigned char reg, unsigned int data)
*        int AD7758_writeLong(unsigned char reg, unsigned long data)
*     Read
*        int AD7758_readChar(unsigned char reg)
*        int AD7758_readInt(unsigned char reg)
*        int AD7758_readLong(unsigned char reg)
*        int AD7758_isDataReady(void)
*        unsigned char AD7758_getChar(void)
*        unsigned int AD7758_getInt(void)
*        unsigned long AD7758_getLong(void)
*
*  History:
*  v1.0 - 2004-04   - Initial Experiments getting the AD7758 working
*  v2.0 - 2004-07-5 - Changing the interface into a state event driven
                      interface.
*  R. Doorduin
*  PEG - University of Stellenbosch
*****************************************************************/

#include <io.h>

#define AD7758_CS 01 //Assign Pin 3.0 for CS for AD7758
#define AD7758_INT 80 //Assign Pin 2.7 for IRQ for AD7758

#define AWATTHR 0x001
#define BWATTHR 0x002
#define CWATTHR 0x003
#define AVARHR 0x004
#define BVARHR 0x005
#define CVARHR 0x006
#define AVAHR 0x007
#define BVAHR 0x008
#define CVAHR 0x009
#define AIRMS 0x00A
#define BIRMS 0x00B
#define CIRMS 0x00C
#define AVRMS 0x00D
#define BVRMS 0x00E
#define CVRMS 0x00F
#define FREQ 0x010
#define TEMP 0x011
#define WAVE 0x012
#define OPMODE 0x013
#define MMODE 0x014
#define WAVMODE 0x015
#define COMPMODE 0x016
#define LCYCMODE 0x017
#define MASK 0x018
#define STATUS 0x019
#define RSTATUS 0x01A
#define ZXTOUT 0x01B
#define LINECYC 0x01C
#define SAGCYC 0x01D
#define SAGLVL 0x01E
#define VPINTLVL 0x01F
#define IPINTLVL 0x020
#define VPEAK 0x021
#define IPEAK 0x022
#define GAIN 0x023
#define AVRMSGAIN 0x024
#define BVRMSGAIN 0x025
#define CVRMSGAIN 0x026
#define AIGAIN 0x027
#define BIGAIN 0x028
#define CIGAIN 0x029
#define AWG 0x02A
#define BWG 0x02B
#define CWG 0x02C
#define AVARG 0x02D
#define BVARG 0x02E
#define CVARG 0x02F
#define AVAG 0x030
#define BVAG 0x031
#define CVAG 0x032
#define AVRMSOS 0x033
#define BVRMSOS 0x034
#define CVRMSOS 0x035
#define AIRMSOS 0x036
#define BIRMSOS 0x037
```

```
#define CIRMSOS 0x038
#define AWATTOS 0x039
#define BWATTOS 0x03A
#define CWATTOS 0x03B
#define AVAROS 0x03C
#define BVAROS 0x03D
#define CVAROS 0x03E
#define APHCAL 0x03F
#define BPHCAL 0x040
#define CPHCAL 0x041
#define WDIV 0x042
#define VARDIV 0x043
#define VADIV 0x044
#define APCFNUM 0x045
#define APCFDEN 0x046
#define VARCFNUM 0x047
#define VARCFDEN 0x048
#define CHKSUM 0x07E
#define VERSION 0x07F

// State Event Declerations
//-----------------------------------------------------------------
//States
#define AD7758_stInit            0x00
#define AD7758_stIdle            0x01
#define AD7758_stWrite1          0x02
#define AD7758_stWrite1Wait      0x03
#define AD7758_stWrite2          0x04
#define AD7758_stWrite2Wait      0x05
#define AD7758_stWrite3          0x06
#define AD7758_stWrite3Wait      0x07
#define AD7758_stWrite4          0x08
#define AD7758_stWrite4Wait      0x09
#define AD7758_stWriteClear      0x0C
#define AD7758_stWriteClearWait  0x0D
#define AD7758_stWriteClearTimer 0x0E

#define AD7758_stRead1           0x010
#define AD7758_stRead1Wait       0x011
#define AD7758_stReadChar1       0x012
#define AD7758_stReadWait1       0x013
#define AD7758_stRead2           0x014
#define AD7758_stRead2Wait       0x015
#define AD7758_stReadChar2       0x016
#define AD7758_stReadWait2       0x017
#define AD7758_stRead3           0x018
#define AD7758_stRead3Wait       0x019
#define AD7758_stReadChar3       0x01A
#define AD7758_stReadWait3       0x01B


//Variables
int AD7758_state = AD7758_stInit;           //The current state within a the SPI driver

#define AD7758_InitFlag          BIT0
#define AD7758_NewData1ToWrite   BIT1
#define AD7758_NewData2ToWrite   BIT2
#define AD7758_NewData3ToWrite   BIT3
#define AD7758_NewData4ToWrite   BIT4
#define AD7758_NewData1ToRead    BIT5
#define AD7758_NewData2ToRead    BIT6
#define AD7758_NewData3ToRead    BIT7
int AD7758_flags = 0x00;                  //Flags to allow the driver to keep track of user input

unsigned char AD7758_Data1ToWrite = 0x0;  //Characters buffer for writing to AD7758
unsigned char AD7758_Data2ToWrite = 0x0;  //Character buffer for writing to AD7758
unsigned char AD7758_Data3ToWrite = 0x0;  //Character buffer for writing to AD7758
unsigned char AD7758_Data4ToWrite = 0x0;  //Character buffer for writing to AD7758

unsigned char AD7758_ReadRegister = 0x0;
unsigned char AD7758_Data1ToRead = 0x0;   //Characters buffer for writing to AD7758
unsigned char AD7758_Data2ToRead = 0x0;   //Character buffer for writing to AD7758
unsigned char AD7758_Data3ToRead = 0x0;   //Character buffer for writing to AD7758


// AD7758_write
//
// Writes a character to the AD7758. Since the firts byte is only a register,
//    this write bit will be cleared for this operation
```

```
//
// Returns:
//    0 if successful
//    -1 if waiting for current data in the cue.
//
//-------------------------------------------------------------------------
int AD7758_write(char c) {
   if (AD7758_flags == AD7758_InitFlag) {
      AD7758_flags |= AD7758_NewData1ToWrite;
      c &= ~0x80;              //Disables write operation for data
                              //Since no data of one byte can be written to AD7758
      AD7758_Data1ToWrite = c;  //Put character in buffer
      return 0;
   } else {
      return -1;
   }
}

// AD7758_writeChar
//
// Writes a character to the specific AD7758 register.
//
// Expects:
//    reg - the AD7758 register
//    data - a char/byte of data for the specific register
// Returns:
//    0 if successful
//    -1 if waiting for current data in the cue.
//
//-------------------------------------------------------------------------
int AD7758_writeChar(unsigned char reg, unsigned char data) {
   if (AD7758_flags == AD7758_InitFlag) {
      AD7758_flags |= AD7758_NewData2ToWrite;
      reg |= 0x80;                  //Enable write operation for data
      AD7758_Data2ToWrite = reg;    //Put character in buffer
      AD7758_Data1ToWrite = data;   //Put character in buffer
      return 0;
   } else {
      return -1;
   }
}

// AD7758_writeInt
//
// Writes a integer (16bits) to the specific AD7758 register.
//
// Expects:
//    reg - the AD7758 register
//    data - a integer of data for the specific register
// Returns:
//    0 if successful
//    -1 if waiting for current data in the cue.
//
//-------------------------------------------------------------------------
int AD7758_writeInt(unsigned char reg, unsigned int data) {
   if (AD7758_flags == AD7758_InitFlag) {
      AD7758_flags |= AD7758_NewData3ToWrite;
      reg |= 0x80;                  //Enable write operation for data
      AD7758_Data3ToWrite = reg;    //Put character in buffer
      AD7758_Data2ToWrite = data >> 8; //Put character in buffer
      AD7758_Data1ToWrite = data;   //Put character in buffer
      return 0;
   } else {
      return -1;
   }
}

// AD7758_writeLong
//
// Writes a 24bit value to the specific AD7758 register.
// The AD7758 only expects 24bit values, therefor 32bits are not supported.
//
// Expects:
//    reg - the AD7758 register
//    data - a long(24bits) of data for the specific register
// Returns:
//    0 if successful
//    -1 if waiting for current data in the cue.
//
```

```
//---------------------------------------------------------------------------
int AD7758_writeLong(unsigned char reg, unsigned long data) {
   if (AD7758_flags == AD7758_InitFlag) {
      AD7758_flags |= AD7758_NewData4ToWrite;
      reg |= 0x80;                        //Enable write operation for data
      AD7758_Data4ToWrite = reg;          //Put character in buffer
      AD7758_Data3ToWrite = data >> 16;   //Put character in buffer
      AD7758_Data2ToWrite = data >> 8;    //Put character in buffer
      AD7758_Data1ToWrite = data;         //Put character in buffer
      return 0;
   } else {
      return -1;
   }
}

// AD7758_readChar
//
// Configures the AD7758 for reading a register of type character.
//
// Expects:
//    reg - the AD7758 register
// Returns:
//    0 if successful
//    -1 if waiting for current data in the cue.
//
// The result of the register can be retrieved by AD7758_getChar()
//   if the AD7758_isDataReady() flag is set
//---------------------------------------------------------------------------
int AD7758_readChar(unsigned char reg) {
   if (AD7758_flags == AD7758_InitFlag) {
      AD7758_flags |= AD7758_NewData1ToRead;
      reg &= ~0x80;                    //Enables read operation for data
      AD7758_ReadRegister = reg;       //Put character in buffer
      return 0;
   } else {
      return -1;
   }
}

// AD7758_readInt
//
// Configures the AD7758 for reading a register of type integer(16bits).
//
// Expects:
//    reg - the AD7758 register
// Returns:
//    0 if successful
//    -1 if waiting for current data in the cue.
//
// The result of the register can be retrieved by AD7758_getInt()
//   if the AD7758_isDataReady() flag is set
//---------------------------------------------------------------------------
int AD7758_readInt(unsigned char reg) {
   if (AD7758_flags == AD7758_InitFlag) {
      AD7758_flags |= AD7758_NewData2ToRead;
      reg &= ~0x80;                    //Enables read operation for data
      AD7758_ReadRegister = reg;       //Put character in buffer
      return 0;
   } else {
      return -1;
   }
}

// AD7758_readLong
//
// Configures the AD7758 for reading a register of type long(24bits).
//    The AD7758 only have 24bits.
//
// Expects:
//    reg - the AD7758 register
// Returns:
//    0 if successful
//    -1 if waiting for current data in the cue.
//
// The result of the register can be retrieved by AD7758_getLong()
//   if the AD7758_isDataReady() flag is set
//---------------------------------------------------------------------------
int AD7758_readLong(unsigned char reg) {
   if (AD7758_flags == AD7758_InitFlag) {
```

```
      AD7758_flags |= AD7758_NewData3ToRead;
      reg &= ~0x80;                    //Enables read operation for data
      AD7758_ReadRegister = reg;       //Put character in buffer
      return 0;
   } else {
      return -1;
   }
}

// AD7758_isDataReady
//
// If the data in the registers are valid, e.g. no other operations are
//    currently using or changing the memory.
//
// Returns:
//    0 if data is ready
//    -1 if data is not ready.
//
//---------------------------------------------------------------------------
int AD7758_isDataReady(void) {
   if (AD7758_flags == AD7758_InitFlag) {
      return 0;
   } else {
      return -1;
   }
}

// AD7758_getChar
//
// Returns the charachter representation (8bits) of the last read register value.
//
// Returns:
//    8bit register value (unsigned char)
//
//---------------------------------------------------------------------------
unsigned char AD7758_getChar(void) {
   return AD7758_Data1ToRead;
}

// AD7758_getInt
//
// Returns the integer representation (16bits) of the last read register value.
//
// Returns:
//    16bit register value (unsigned int)
//
//---------------------------------------------------------------------------
unsigned int AD7758_getInt(void) {
   unsigned int c = 0x0;
   c = AD7758_Data2ToRead & (0x0FF);
   c = c << 8;
   c = c + (AD7758_Data1ToRead  & (0x0FF));
   return c;
}

// AD7758_getChar
//
// Returns the long representation (24bits) of the last read register value.
//
// Returns:
//    32bit register value (unsigned long), MSB = 0x0000
//
//---------------------------------------------------------------------------
unsigned long AD7758_getLong(void) {
   unsigned long c = 0x0;
   c = AD7758_Data3ToRead & (0x0FF);
   c = c << 8;
   c = c + (AD7758_Data2ToRead & (0x0FF));
   c = c << 8;
   c = c + (AD7758_Data1ToRead & (0x0FF));
   return c;
}

// AD7758_driver
//
// The driver is a state event machine intende to manage the AD7758 operation
//    and is the only software component communicating with the SPI driver.
//
//---------------------------------------------------------------------------
```

```
void AD7758_driver(void) {
   switch (AD7758_state) {
   //INIT
      case AD7758_stInit:
         P3DIR |= AD7758_CS;  //Enable ChipSelect
         P3OUT |= AD7758_CS;  //AD7758 not selected
         AD7758_flags = 0x00;
         AD7758_write(0x00);  //Clears output buffer
         AD7758_flags |= AD7758_InitFlag;
         AD7758_state = AD7758_stIdle;
         break;
   //IDLE
      case AD7758_stIdle:
         if ((AD7758_flags & AD7758_InitFlag) == 0x0) {                //Ensure that driver is initialized
            AD7758_state = AD7758_stInit;
         } else if ((AD7758_flags & AD7758_NewData1ToWrite) != 0x0) {
            if ((P3OUT & AD7758_CS) != 0x0) {
               AD7758_state = AD7758_stWrite1;
            }
         } else if ((AD7758_flags & AD7758_NewData2ToWrite) != 0x0) {
            if ((P3OUT & AD7758_CS) != 0x0) {
               AD7758_state = AD7758_stWrite2;
            }
         } else if ((AD7758_flags & AD7758_NewData3ToWrite) != 0x0) {
            if ((P3OUT & AD7758_CS) != 0x0) {
               AD7758_state = AD7758_stWrite3;
            }
         } else if ((AD7758_flags & AD7758_NewData4ToWrite) != 0x0) {
            if ((P3OUT & AD7758_CS) != 0x0) {
               AD7758_state = AD7758_stWrite4;
            }
         } else if ((AD7758_flags & AD7758_NewData1ToRead) != 0x0) {
            if ((P3OUT & AD7758_CS) != 0x0) {
               AD7758_state = AD7758_stRead1;
            }
         } else if ((AD7758_flags & AD7758_NewData2ToRead) != 0x0) {
            if ((P3OUT & AD7758_CS) != 0x0) {
               AD7758_state = AD7758_stRead2;
            }
         } else if ((AD7758_flags & AD7758_NewData3ToRead) != 0x0) {
            if ((P3OUT & AD7758_CS) != 0x0) {
               AD7758_state = AD7758_stRead3;
            }
         }
         break;

   //READ
      case AD7758_stRead3:
         if (SPI_write(AD7758_ReadRegister) == 0) {
            P3OUT &= ~AD7758_CS;    //Select AD7758... This can be done after the SPI_write, since enough
                                    //time will elapse prior to starting the signal on the SPI bus.
            AD7758_state = AD7758_stRead3Wait;
         }
         break;
      case AD7758_stRead3Wait:
         if (SPI_isWriting() != 0) {
            AD7758_state = AD7758_stReadChar3;
         }
         break;
      case AD7758_stReadChar3:
         if (SPI_read() == 0) {
            AD7758_state = AD7758_stReadWait3;
         }
         break;
      case AD7758_stReadWait3:
         if (SPI_isCharReady() == 0) {
            AD7758_Data3ToRead = SPI_getChar();
            AD7758_state = AD7758_stReadChar2;
         }
         break;
      case AD7758_stRead2:
         if (SPI_write(AD7758_ReadRegister) == 0) {
            P3OUT &= ~AD7758_CS;    //Select AD7758... This can be done after the SPI_write, since enough
                                    //time will elapse prior to starting the signal on the SPI bus.
            AD7758_state = AD7758_stRead2Wait;
         }
         break;
      case AD7758_stRead2Wait:
         if (SPI_isWriting() != 0) {
```

```
            AD7758_state = AD7758_stReadChar2;
         }
         break;
      case AD7758_stReadChar2:
         if (SPI_read() == 0) {
            AD7758_state = AD7758_stReadWait2;
         }
         break;
      case AD7758_stReadWait2:
         if (SPI_isCharReady() == 0) {
            AD7758_Data2ToRead = SPI_getChar();
            AD7758_state = AD7758_stReadChar1;
         }
         break;
      case AD7758_stRead1:
         if (SPI_write(AD7758_ReadRegister) == 0) {
            P3OUT &= ~AD7758_CS;    //Select AD7758... This can be done after the SPI_write, since enough
                                    //time will elapse prior to starting the signal on the SPI bus.
            AD7758_state = AD7758_stRead1Wait;
         }
         break;
      case AD7758_stRead1Wait:
         if (SPI_isWriting() != 0) {
            AD7758_state = AD7758_stReadChar1;
         }
         break;
      case AD7758_stReadChar1:
         if (SPI_read() == 0) {
            AD7758_state = AD7758_stReadWait1;
         }
         break;
      case AD7758_stReadWait1:
         if (SPI_isCharReady() == 0) {
            AD7758_Data1ToRead = SPI_getChar();
            AD7758_state = AD7758_stWriteClear;
         }
         break;

   //WRITE
      case AD7758_stWrite4:
         if (SPI_write(AD7758_Data4ToWrite) == 0) {
               P3OUT &= ~AD7758_CS;    //Select AD7758... This can be done after the SPI_write, since enough
                                       //time will elapse prior to starting the signal on the SPI bus.
               AD7758_state = AD7758_stWrite4Wait;
         }
         break;
      case AD7758_stWrite4Wait:
         if (SPI_isWriting() != 0) {
            AD7758_state = AD7758_stWrite3;
         }
         break;
      case AD7758_stWrite3:
         if (SPI_write(AD7758_Data3ToWrite) == 0) {
               P3OUT &= ~AD7758_CS;    //Select AD7758... This can be done after the SPI_write, since enough
                                                   AD7758_flags &= ~AD7758_NewData1ToRead;
//time will elapse prior to starting the signal on the SPI bus.
               AD7758_state = AD7758_stWrite3Wait;
         }
         break;
      case AD7758_stWrite3Wait:
         if (SPI_isWriting() != 0) {
            AD7758_state = AD7758_stWrite2;
         }
         break;
      case AD7758_stWrite2:
         if (SPI_write(AD7758_Data2ToWrite) == 0) {
               P3OUT &= ~AD7758_CS;    //Select AD7758... This can be done after the SPI_write, since enough
                                       //time will elapse prior to starting the signal on the SPI bus.
               AD7758_state = AD7758_stWrite2Wait;
         }
         break;
      case AD7758_stWrite2Wait:
         if (SPI_isWriting() != 0) {
            AD7758_state = AD7758_stWrite1;
         }
         break;
      case AD7758_stWrite1:
         if (SPI_write(AD7758_Data1ToWrite) == 0) {
               P3OUT &= ~AD7758_CS;    //Select AD7758... This can be done after the SPI_write, since enough
```

```
                                    //time will elapse prior to starting the signal on the SPI bus.
               AD7758_state = AD7758_stWrite1Wait;
            }
            break;
         case AD7758_stWrite1Wait:
            if (SPI_isWriting() != 0) {
               AD7758_state = AD7758_stWriteClear;
            }
            break;
         case AD7758_stWriteClear:
            if (SPI_write(0x0) == 0) {
               AD7758_state = AD7758_stWriteClearWait;
            }
            break;
         case AD7758_stWriteClearWait:
            if (SPI_isWriting() != 0) {
               P3OUT |= AD7758_CS;
               TIMER_startTimer(tmrAD7758_Clear);
               AD7758_state = AD7758_stWriteClearTimer;
            }
            break;
         case AD7758_stWriteClearTimer:
            if (TIMER_isExpired(tmrAD7758_Clear,delayAD7758_Clear) == 0) {
               TIMER_stopTimer(tmrAD7758_Clear);
               AD7758_flags &= ~AD7758_NewData1ToWrite;
               AD7758_flags &= ~AD7758_NewData2ToWrite;
               AD7758_flags &= ~AD7758_NewData3ToWrite;
               AD7758_flags &= ~AD7758_NewData4ToWrite;
               AD7758_flags &= ~AD7758_NewData1ToRead;
               AD7758_flags &= ~AD7758_NewData2ToRead;
               AD7758_flags &= ~AD7758_NewData3ToRead;
               AD7758_state = AD7758_stIdle;
            }
            break;
         default:
            AD7758_state = AD7758_stInit;
            break;
      }
}
```

R.4.2 rs232.h

Allows for communication to the callibration system.

```
/*********************************************************************
*  RS232
*
*  This header file is intended to easy interfacing with the RS232
*  system for serial communications.
*  This driver interfaces with the UART driver.
*
*  The following functions are used for interfacing with the RS232 driver
*     Driver
*        void RS232_driver(void)
*     Send Functions
*        int RS232_sendChar(unsigned char device, unsigned char deviceRegister, unsigned char data)
*        int RS232_sendInt(unsigned char device, unsigned char deviceRegister, unsigned int data)
*        int RS232_sendLong(unsigned char device, unsigned char deviceRegister, unsigned long data)
*     Read Functions
*        int RS232_isRS232Ready(void)
*        int RS232_isRS232ReadyReset(void)
*        unsigned char RS232_getDevice(void)
*        unsigned char RS232_getDeviceRegister(void)
*        unsigned char RS232_getDataLength(void)
*        char RS232_getChar(void)
*        int RS232_getInt(void)
*        long RS232_getLong(void)
*
*  History:
*  v 1.0 - 2004-04 Initial configuration and primitive functionality
*  v 2.0 - 2004-07 Change to state machine/driver implementation
*
*  R. Doorduin
*  PEG - University of Stellenbosch
*********************************************************************/
```

```
// State Event Declerations
//-----------------------------------------------------------------
//Receive States
#define RS232_stReceiveInit                 0x00
#define RS232_stReceiveIdle                 0x01
#define RS232_stReceiveWaitLength           0x02
#define RS232_stReceiveWaitDevice           0x03
#define RS232_stReceiveWaitDeviceRegister   0x04
#define RS232_stReceiveWaitData             0x05
#define RS232_stReceiveWaitStop             0x06

//Send States
#define RS232_stSendInit              0x00
#define RS232_stSendIdle              0x01
#define RS232_stSendStartChar         0x02 //Start
#define RS232_stSendStartInt          0x03
#define RS232_stSendStartLong         0x04
#define RS232_stSendLengthChar        0x05 //Length
#define RS232_stSendLengthInt         0x06
#define RS232_stSendLengthLong        0x07
#define RS232_stSendDeviceChar        0x08 //Device
#define RS232_stSendDeviceInt         0x09
#define RS232_stSendDeviceLong        0x0A
#define RS232_stSendDeviceRegisterChar 0x0B //DeviceRegister
#define RS232_stSendDeviceRegisterInt  0x0C
#define RS232_stSendDeviceRegisterLong 0x0D
#define RS232_stSendDataChar          0x0E //Data
#define RS232_stSendDataInt           0x0F
#define RS232_stSendDataLong          0x10
#define RS232_stSendStop              0x11 //Stop
#define RS232_stSendZero              0x12

//Variables
int RS232_receiveState = RS232_stReceiveInit;
int RS232_sendState = RS232_stSendInit;

//Flags
#define RS232_flagInit          BIT0
#define RS232_flagSendChar      BIT1
#define RS232_flagSendInt       BIT2
#define RS232_flagSendLong      BIT3
#define RS232_flagPacketReady   BIT4
int RS232_flags = 0x00;               //Flags to allow the driver to keep track of user input

//Protocol Declerations
//-----------------------------------------------------------------
//Definitions
#define PROTOCOL_START_CHAR         0xAA
#define PROTOCOL_LAST_CHAR          0x00
#define PROTOCOL_MAX_DATA_LENGTH    0x0F
#define PROTOCOL_HEADER_LENGTH      0x04   //include the Length and CRC byte at the end

#define PROTOCOL_COMPONENT_MCU                    0x00
   #define PROTOCOL_REGISTER_ACKNOWLEDGE             0x00
   #define PROTOCOL_REGISTER_MCU_REGISTERDUMP        0x01
      #define PROTOCOL_VALUE_MCU_MAINSCREEN             0x01
      #define PROTOCOL_VALUE_MCU_MONITOR                0x02
   #define PROTOCOL_REGISTER_MCU_CALIBRATION         0x02
      #define PROTOCOL_VALUE_MCU_V1_A                   0x01
      #define PROTOCOL_VALUE_MCU_V2_A                   0x02
      #define PROTOCOL_VALUE_MCU_I1_A                   0x03
      #define PROTOCOL_VALUE_MCU_I2_A                   0x04
      #define PROTOCOL_VALUE_MCU_V1_B                   0x05
      #define PROTOCOL_VALUE_MCU_V2_B                   0x06
      #define PROTOCOL_VALUE_MCU_I1_B                   0x07
      #define PROTOCOL_VALUE_MCU_I2_B                   0x08
      #define PROTOCOL_VALUE_MCU_V1_C                   0x09
      #define PROTOCOL_VALUE_MCU_V2_C                   0x0A
      #define PROTOCOL_VALUE_MCU_I1_C                   0x0B
      #define PROTOCOL_VALUE_MCU_I2_C                   0x0C
      #define PROTOCOL_VALUE_MCU_GAIN_A                 0x0D
      #define PROTOCOL_VALUE_MCU_GAIN_B                 0x0E
      #define PROTOCOL_VALUE_MCU_GAIN_C                 0x0F
      #define PROTOCOL_VALUE_MCU_FREQ                   0x10
      #define PROTOCOL_VALUE_MCU_FLASH                  0x11
      #define PROTOCOL_VALUE_MCU_PHASE1A                0x12
      #define PROTOCOL_VALUE_MCU_PHASE1B                0x13
      #define PROTOCOL_VALUE_MCU_PHASE1C                0x14
      #define PROTOCOL_VALUE_MCU_PHASE2A                0x15
```

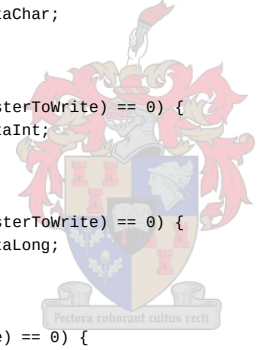
```
      #define PROTOCOL_VALUE_MCU_PHASE2B                0x16
      #define PROTOCOL_VALUE_MCU_PHASE2C                0x17
   #define PROTOCOL_REGISTER_MCU_CURRENTCAL          0x03
   #define PROTOCOL_REGISTER_MCU_MEASURE_ENERGY      0x04
      #define PROTOCOL_VALUE_MCU_STOP                   0x00
      #define PROTOCOL_VALUE_MCU_START                  0x01
   #define PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_W    0x05
   #define PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VA   0x06
   #define PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VAR  0x07
   #define PROTOCOL_REGISTER_MCU_MESSAGE             0x08
      #define PROTOCOL_VALUE_MCU_OVERFLOW               0x01
   #define PROTOCOL_REGISTER_MCU_CALIBRATION_WDIVA   0x09
   #define PROTOCOL_REGISTER_MCU_CALIBRATION_WDIVB   0x0A
   #define PROTOCOL_REGISTER_MCU_CALIBRATION_WDIVC   0x0B
   #define PROTOCOL_REGISTER_MCU_CALIBRATION_VADIVA  0x0C
   #define PROTOCOL_REGISTER_MCU_CALIBRATION_VADIVB  0x0D
   #define PROTOCOL_REGISTER_MCU_CALIBRATION_VADIVC  0x0E
   #define PROTOCOL_REGISTER_MCU_CALIBRATION_VARDIVA 0x0F
   #define PROTOCOL_REGISTER_MCU_CALIBRATION_VARDIVB 0x010
   #define PROTOCOL_REGISTER_MCU_CALIBRATION_VARDIVC 0x011
   #define PROTOCOL_REGISTER_MCU_FLASH               0x012
      #define PROTOCOL_VALUE_MCU_FLASH_WRITE            0x01
      #define PROTOCOL_VALUE_MCU_FLASH_READ             0x02
   #define PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_W_A  0x013
   #define PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VA_A 0x014
   #define PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VAR_A 0x015
   #define PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_W_B  0x016
   #define PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VA_B 0x017
   #define PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VAR_B 0x018
   #define PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_W_C  0x019
   #define PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VA_C 0x01A
   #define PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VAR_C 0x01B
   #define PROTOCOL_REGISTER_MCU_MEASURE_PHASE_W1A   0x01C
   #define PROTOCOL_REGISTER_MCU_MEASURE_PHASE_W1B   0x01D
   #define PROTOCOL_REGISTER_MCU_MEASURE_PHASE_W1C   0x01E
   #define PROTOCOL_REGISTER_MCU_MEASURE_PHASE_W2A   0x01F
   #define PROTOCOL_REGISTER_MCU_MEASURE_PHASE_W2B   0x020
   #define PROTOCOL_REGISTER_MCU_MEASURE_PHASE_W2C   0x021

#define PROTOCOL_COMPONENT_AD7758                0x01

//Packet Data
//    To Write
unsigned char RS232_deviceToWrite = 0x0;                //The Device to which the data is adressed
unsigned char RS232_deviceRegisterToWrite = 0x0;        //The register within the device
unsigned char RS232_charToWrite = 0x0;                  //Data...
unsigned int  RS232_intToWrite = 0x0;
unsigned long RS232_longToWrite = 0x0;
//    To Read
unsigned char RS232_deviceRead = 0x0;                           //The Device to which the data is adressed
unsigned char RS232_deviceRegisterRead = 0x0;                   //The register within the device
unsigned char RS232_dataRead[PROTOCOL_MAX_DATA_LENGTH+0x01];    //The data in the packet (1 for length @[0])
unsigned char RS232_lengthToRead = 0x0;                         //The length according to the packet
unsigned char RS232_readLength = 0x0;                           //The current position within the packet
unsigned char RS232_readDataPos = 0x0;                          //The current position of the read data


//  RS232_sendChar
//
// Sends a character via RS232 protocol
//
//------------------------------------------------------------------------------
int RS232_sendChar(unsigned char device, unsigned char deviceRegister, unsigned char data) {
   if ((RS232_flags & (RS232_flagSendChar + RS232_flagSendInt + RS232_flagSendLong)) == 0x00) {
      RS232_deviceToWrite = device;
      RS232_deviceRegisterToWrite = deviceRegister;
      RS232_charToWrite = data;
      RS232_flags |= RS232_flagSendChar; //To let driver know that a new char is ready
      return 0;
   } else {
      return -1;
   }
}

//  RS232_sendInt
//
// Sends a character via RS232 protocol
//
//------------------------------------------------------------------------------
```

```
int RS232_sendInt(unsigned char device, unsigned char deviceRegister, unsigned int data) {
   if ((RS232_flags & (RS232_flagSendChar + RS232_flagSendInt + RS232_flagSendLong)) == 0x00) {
      RS232_deviceToWrite = device;
      RS232_deviceRegisterToWrite = deviceRegister;
      RS232_intToWrite = data;
      RS232_flags |= RS232_flagSendInt; //To let driver know that a new char is ready
      return 0;
   } else {
      return -1;
   }
}

//  RS232_sendLong
//
// Sends a character via RS232 protocol
//
//------------------------------------------------------------------------------
int RS232_sendLong(unsigned char device, unsigned char deviceRegister, unsigned long data) {
   if ((RS232_flags & (RS232_flagSendChar + RS232_flagSendInt + RS232_flagSendLong)) == 0x00) {
      RS232_deviceToWrite = device;
      RS232_deviceRegisterToWrite = deviceRegister;
      RS232_longToWrite = data;
      RS232_flags |= RS232_flagSendLong; //To let driver know that a new char is ready
      return 0;
   } else {
      return -1;
   }
}

//  RS232_isRS232Ready
//
// Checks to see if a packet is ready to be read
//
//------------------------------------------------------------------------------
int RS232_isRS232Ready(void) {
   if ((RS232_flags & RS232_flagPacketReady) != 0x0) {
      return 0;
   } else {
      return -1;
   }
}

//  RS232_isRS232ReadyReset
//
// Checks to see if a packet is ready to be read
// After calling this function the flag is reset, to indicate that the
//    packet is already serviced.
//
//------------------------------------------------------------------------------
int RS232_isRS232ReadyReset(void) {
   if ((RS232_flags & RS232_flagPacketReady) != 0x0) {
      RS232_flags &= ~RS232_flagPacketReady;
      return 0;
   } else {
      return -1;
   }
}

//  RS232_getDevice
//
// Returns the device adressed in the received packet
//
//------------------------------------------------------------------------------
unsigned char RS232_getDevice(void) {
   return RS232_deviceRead;
}

//  RS232_getDeviceRegister
//
// Returns the register adressed in the device contained in the received packet
//
//------------------------------------------------------------------------------
unsigned char RS232_getDeviceRegister(void) {
   return RS232_deviceRegisterRead;
}

//  RS232_getDataLength
//
// Returns the length of the data
```

```
//
//---------------------------------------------------------------------------
unsigned char RS232_getDataLength(void) {
   return RS232_dataRead[0];
}

//  RS232_getChar
//
// Returns the char equivilent of the data packet
//
//---------------------------------------------------------------------------
char RS232_getChar(void) {
   return RS232_dataRead[1];
}

//  RS232_getInt
//
// Returns the char equivilent of the data packet
//
//---------------------------------------------------------------------------
int RS232_getInt(void) {
   int c = 0x0;
   c = (RS232_dataRead[1]&0x0FF);
   c = c << 8;
   c = c + (RS232_dataRead[2]&0x0FF);
   return c;
}

//  RS232_getLong
//
// Returns the char equivilent of the data packet
//
//---------------------------------------------------------------------------
long RS232_getLong(void) {
   long c = 0x0;
   c = (RS232_dataRead[1]&0x0FF);
   c = c << 8;
   c = c + (RS232_dataRead[2]&0x0FF);
   c = c << 8;
   c = c + (RS232_dataRead[3]&0x0FF);
   c = c << 8;
   c = c + (RS232_dataRead[4]&0x0FF);
   return c;
}

// RS232_receiveStateDriver
//
// Handles the reception and decoding of data.
// Includes time outs if no data is recieved for some time.
//
//---------------------------------------------------------------------------
void RS232_receiveStateDriver(void) {
   unsigned char c = 0x0;
   //Watchdog
   //  ...can be inserted here...

   //State Events
   switch (RS232_receiveState) {
      case RS232_stReceiveInit:
         RS232_flags &= ~RS232_flagPacketReady;
         RS232_receiveState = RS232_stReceiveIdle;
         break;
      case RS232_stReceiveIdle:
         if (UART_isCharReady() == 0) {
            if (UART_getChar() == PROTOCOL_START_CHAR) {
               RS232_flags &= ~RS232_flagPacketReady;
               RS232_receiveState = RS232_stReceiveWaitLength;
            }
         }
         break;
      case RS232_stReceiveWaitLength:
         if (UART_isCharReady() == 0) {
            c = UART_getChar();
            if ((c <= 0x04) || (c > (PROTOCOL_HEADER_LENGTH + PROTOCOL_MAX_DATA_LENGTH))) {
               RS232_receiveState = RS232_stReceiveIdle;    //Implies that data falls outside range
            } else {
               //Correct data lengths
               //Clear old data
               RS232_lengthToRead = c;
```

```
               RS232_readLength = 0x01;
               RS232_readDataPos = 0x0;
               RS232_receiveState = RS232_stReceiveWaitDevice;
            }
         }
         break;
      case RS232_stReceiveWaitDevice:
         if (UART_isCharReady() == 0) {
            RS232_deviceRead = UART_getChar();
            RS232_readLength = 0x02;
            RS232_receiveState = RS232_stReceiveWaitDeviceRegister;
         }
         break;
      case RS232_stReceiveWaitDeviceRegister:
         if (UART_isCharReady() == 0) {
            RS232_deviceRegisterRead = UART_getChar();
            RS232_readLength = 0x03;
            RS232_receiveState = RS232_stReceiveWaitData;
         }
         break;
      case RS232_stReceiveWaitData:
         if (UART_isCharReady() == 0) {
            RS232_readLength++;
            if ((RS232_readLength < RS232_lengthToRead) & (RS232_readDataPos < PROTOCOL_MAX_DATA_LENGTH)) {
               RS232_dataRead[++RS232_readDataPos] = UART_getChar();
               if (RS232_readLength == (RS232_lengthToRead - 0x01)) {
                  RS232_dataRead[0] = RS232_readDataPos;
                  RS232_receiveState = RS232_stReceiveWaitStop;
               }
            } else {
               RS232_receiveState = RS232_stReceiveIdle;
            }
         }
         break;
      case RS232_stReceiveWaitStop:
         if (UART_isCharReady() == 0) {
            if (UART_getChar() == PROTOCOL_LAST_CHAR) {
               RS232_flags |= RS232_flagPacketReady;
            }
            RS232_receiveState = RS232_stReceiveIdle;
         }
         break;
   }
}

// RS232_sendStateDriver
//
// Handles the packaging and encoding of data.
//
//-----------------------------------------------------------------------------
void RS232_sendStateDriver(void) {
   switch (RS232_sendState) {
      case RS232_stSendInit:
         RS232_sendState = RS232_stSendIdle;
         break;
      case RS232_stSendIdle:
         if ((RS232_flags & RS232_flagSendChar) != 0) {
            RS232_sendState = RS232_stSendStartChar;
         } else if ((RS232_flags & RS232_flagSendInt) != 0) {
            RS232_sendState = RS232_stSendStartInt;
         } else if ((RS232_flags & RS232_flagSendLong) != 0) {
            RS232_sendState = RS232_stSendStartLong;
         }
         break;
      case RS232_stSendStartChar:
         if (UART_writeChar(PROTOCOL_START_CHAR) == 0) {
            RS232_sendState = RS232_stSendLengthChar;
         }
         break;
      case RS232_stSendStartInt:
         if (UART_writeChar(PROTOCOL_START_CHAR) == 0) {
            RS232_sendState = RS232_stSendLengthInt;
         }
         break;
      case RS232_stSendStartLong:
         if (UART_writeChar(PROTOCOL_START_CHAR) == 0) {
            RS232_sendState = RS232_stSendLengthLong;
         }
         break;
```

```
      case RS232_stSendLengthChar:
         if (UART_writeChar(PROTOCOL_HEADER_LENGTH+0x001) == 0) {
            RS232_sendState = RS232_stSendDeviceChar;
         }
         break;
      case RS232_stSendLengthInt:
         if (UART_writeChar(PROTOCOL_HEADER_LENGTH+0x002) == 0) {
            RS232_sendState = RS232_stSendDeviceInt;
         }
         break;
      case RS232_stSendLengthLong:
         if (UART_writeChar(PROTOCOL_HEADER_LENGTH+0x004) == 0) {
            RS232_sendState = RS232_stSendDeviceLong;
         }
         break;
      case RS232_stSendDeviceChar:
         if (UART_writeChar(RS232_deviceToWrite) == 0) {
            RS232_sendState = RS232_stSendDeviceRegisterChar;
         }
         break;
      case RS232_stSendDeviceInt:
         if (UART_writeChar(RS232_deviceToWrite) == 0) {
            RS232_sendState = RS232_stSendDeviceRegisterInt;
         }
         break;
      case RS232_stSendDeviceLong:
         if (UART_writeChar(RS232_deviceToWrite) == 0) {
            RS232_sendState = RS232_stSendDeviceRegisterLong;
         }
         break;
      case RS232_stSendDeviceRegisterChar:
         if (UART_writeChar(RS232_deviceRegisterToWrite) == 0) {
            RS232_sendState = RS232_stSendDataChar;
         }
         break;
      case RS232_stSendDeviceRegisterInt:
         if (UART_writeChar(RS232_deviceRegisterToWrite) == 0) {
            RS232_sendState = RS232_stSendDataInt;
         }
         break;
      case RS232_stSendDeviceRegisterLong:
         if (UART_writeChar(RS232_deviceRegisterToWrite) == 0) {
            RS232_sendState = RS232_stSendDataLong;
         }
         break;
      case RS232_stSendDataChar:
         if (UART_writeChar(RS232_charToWrite) == 0) {
            RS232_sendState = RS232_stSendStop;
         }
         break;
      case RS232_stSendDataInt:
         if (UART_writeInt(RS232_intToWrite) == 0) {
            RS232_sendState = RS232_stSendStop;
         }
         break;
      case RS232_stSendDataLong:
         if (UART_writeLong(RS232_longToWrite) == 0) {
            RS232_sendState = RS232_stSendStop;
         }
         break;
      case RS232_stSendStop:
         if (UART_writeChar(PROTOCOL_LAST_CHAR) == 0) {
            RS232_sendState = RS232_stSendZero;
         }
         break;
      case RS232_stSendZero:
         if (UART_writeChar(0x00) == 0) {
            RS232_sendState = RS232_stSendIdle;
            RS232_flags &= ~RS232_flagSendChar;
            RS232_flags &= ~RS232_flagSendInt;
            RS232_flags &= ~RS232_flagSendLong;
         }
         break;
   }
}

// RS232_driver
//
// The driver is a state event machine and is the only software component
```



```
//    communicating on hardware level with the UART driver interface.
// The driver assumes that the RS232 interfaces is set and reset on a application
//    level.
//
//------------------------------------------------------------------------------
void RS232_driver(void) {
   if (UART_isRS232on() == 0) {
      RS232_receiveStateDriver();
      RS232_sendStateDriver();
   }
}
```

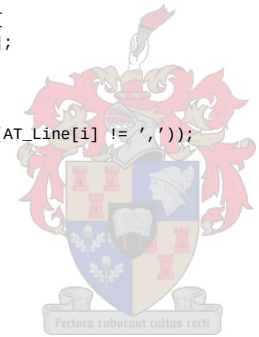
R.4.3 atCommand.h

Ensures communications to the GSM modem via AT command set.

```
/*********************************************************************
*  AT COMMAND
*
*  This header file is intended to easy interfacing using the AT commands
*  for serial communications to the GSM modem.
*  This driver interfaces with the UART driver.
*
*  The following functions are used for interfacing with the AT driver
*     Driver
*        void AT_driver(void)
*     Send Functions
*        int AT_sendChar(unsigned char device, unsigned char deviceRegister, unsigned char data)
*        int AT_sendInt(unsigned char device, unsigned char deviceRegister, unsigned int data)
*        int AT_sendLong(unsigned char device, unsigned char deviceRegister, unsigned long data)
*     Read Functions
*        int AT_isATReady(void)
*        int AT_isATReadyReset(void)
*        unsigned char AT_getDevice(void)
*        unsigned char AT_getDeviceRegister(void)
*        unsigned char AT_getDataLength(void)
*        char AT_getChar(void)
*        int AT_getInt(void)
*        long AT_getLong(void)
*
*  History:
*  v 1.0 - 2004-07 Based on the AT commmand spec.
*
*  R. Doorduin
*  PEG - University of Stellenbosch
*********************************************************************/

// State Event Declerations
//-----------------------------------------------------------------
//Receive States
#define AT_stReceiveInit                0x00
#define AT_stReceiveIdle                0x01

//Send States
#define AT_stSendInit                   0x000
#define AT_stSendIdle                   0x001
#define AT_stSend_StringStart           0x002
#define AT_stSend_StringStored          0x003
#define AT_stSend_StringEnd             0x004
#define AT_stSend_WaitOK                0x005
#define AT_stSend_WaitErrorDelay        0x006
#define AT_stSend_SMS_EnergyStart       0x007
#define AT_stSend_SMS_EnergyNumber      0x008
#define AT_stSend_SMS_EnergyPreReading  0x009
#define AT_stSend_SMS_WaitMessagePointer 0x01A
#define AT_stSend_SMS_WaitMessageDelay  0x01B
#define AT_stSend_SMS_EnergyWA          0x00A
#define AT_stSend_SMS_EnergyWB          0x00B
#define AT_stSend_SMS_EnergyWC          0x00C
#define AT_stSend_SMS_EnergyWColon      0x00D
#define AT_stSend_SMS_EnergyVARA        0x00E
#define AT_stSend_SMS_EnergyVARB        0x00F
#define AT_stSend_SMS_EnergyVARColon    0x010
#define AT_stSend_SMS_EnergyVARC        0x011
#define AT_stSend_SMS_EnergyVAA         0x012
#define AT_stSend_SMS_EnergyVAB         0x013
```

```
#define AT_stSend_SMS_EnergyVAC           0x014
#define AT_stSend_SMS_EnergyVAColon       0x015
#define AT_stSend_SMS_End                 0x016
#define AT_stSend_SMS_LF                  0x017
#define AT_stSend_SMS_Wait                0x01C
#define AT_stSend_SMS_WaitOK              0x018
#define AT_stSend_SMS_WaitErrorDelay      0x019

//Variables
int AT_receiveState = AT_stReceiveInit;
int AT_sendState = AT_stSendInit;

//Flags
#define AT_flagInit              BIT0
#define AT_NewStringToWrite      BIT1
#define AT_NewStringToWriteOK    BIT2
#define AT_flagSendEnergyReport  BIT3
#define AT_LineReady             BIT4
int AT_flags = 0x00;               //Flags to allow the driver to keep track of user input

//General variables and defines
#define AT_GSM_onOff BIT2
char *AT_stringToWrite = "";

#define AT_Line_MAX_Length 80
#define AT_Field_MAX_Length 20
#define AT_MAX_Fields 5

char AT_Line[AT_Line_MAX_Length];
char AT_ResultLine[AT_Line_MAX_Length];
char AT_Cmd[5];
char AT_Field1[AT_Field_MAX_Length];
char AT_Field2[AT_Field_MAX_Length];
char AT_Field3[AT_Field_MAX_Length];
char AT_Field4[AT_Field_MAX_Length];
char AT_Field5[AT_Field_MAX_Length];
unsigned char AT_Fields = 0x0;

//  AT_parseLine
//
// The parser, sorts the returned results from the GSM modem into the respective
//    fields.
// The following fields are used:
//    AT_Fields   Number of fields paresed and stored
//    AT_Field1   The various fields used to store the data in
//       :
//    AT_Field5
//
// Returns:
//    0 if successful
//    -1 if unsuccessful
//
//-------------------------------------------------------------------------------
int AT_parseLine() {
   int i,j = 0x0;
   int fields = 0;
   int index = 0x0;
   if (AT_Line[0] == '+') {
      if ((AT_Line[4] == ':') & (AT_Line[5] == ' ')) {
         AT_Cmd[0] = AT_Line[1];
         AT_Cmd[1] = AT_Line[2];
         AT_Cmd[2] = AT_Line[3];
         AT_Cmd[3] = '\0';
         AT_Cmd[4] = '\0';
         index = 5;
      } else if ((AT_Line[5] == ':') & (AT_Line[6] == ' ')) {
         AT_Cmd[0] = AT_Line[1];
         AT_Cmd[1] = AT_Line[2];
         AT_Cmd[2] = AT_Line[3];
         AT_Cmd[3] = AT_Line[4];
         AT_Cmd[4] = '\0';
         index = 6;
      } else if ((AT_Line[10] == ':') & (AT_Line[11] == ' ')) {
         AT_Cmd[0] = AT_Line[1];
         AT_Cmd[1] = AT_Line[2];
         AT_Cmd[2] = AT_Line[3];
         AT_Cmd[3] = AT_Line[4];
         AT_Cmd[4] = '\0';
         index = 11;
```

```
      }

      i = index;
      do {  //Field 1
         if (j < AT_Field_MAX_Length) {
            AT_Field1[j++] = AT_Line[i];
            AT_Field1[j] = '\0';
         }
         i++;
      } while ((AT_Line[i] != '\0') & (AT_Line[i] != ','));
      AT_Fields = 1;
      if (AT_Line[i] == ',') { //Field 2
         j = 0;
         i++;
         AT_Field2[0] = '\0';
         if (AT_Line[i] != ',') {
            do {
               if (j < AT_Field_MAX_Length) {
                  AT_Field2[j++] = AT_Line[i];
                  AT_Field2[j] = '\0';
               }
               i++;
            } while ((AT_Line[i] != '\0') & (AT_Line[i] != ','));
            AT_Fields = 2;
         }
      }
      if (AT_Line[i] == ',') { //Field 3
         j = 0;
         i++;
         AT_Field3[0] = '\0';
         if (AT_Line[i] != ',') {
            do {
               if (j < AT_Field_MAX_Length) {
                  AT_Field3[j++] = AT_Line[i];
                  AT_Field3[j] = '\0';
               }
               i++;
            } while ((AT_Line[i] != '\0') & (AT_Line[i] != ','));
            AT_Fields = 3;
         }
      }
      if (AT_Line[i] == ',') { //Field 4
         j = 0;
         i++;
         AT_Field4[0] = '\0';
         if (AT_Line[i] != ',') {
            do {
               if (j < AT_Field_MAX_Length) {
                  AT_Field4[j++] = AT_Line[i];
                  AT_Field4[j] = '\0';
               }
               i++;
            } while ((AT_Line[i] != '\0') & (AT_Line[i] != ','));
            AT_Fields = 4;
         }
      }
      if (AT_Line[i] == ',') { //Field 5
         j = 0;
         i++;
         AT_Field5[0] = '\0';
         if (AT_Line[i] != ',') {
            do {
               if (j < AT_Field_MAX_Length) {
                  AT_Field5[j++] = AT_Line[i];
                  AT_Field5[j] = '\0';
               }
               i++;
            } while ((AT_Line[i] != '\0') & (AT_Line[i] != ','));
            AT_Fields = 5;
         }
      }

      AT_Fields = 5;
      if (index != 0x0) {
         return 0;
      }
   }
   return -1;
}
```



```
//  AT_isLine
//
// This function compares the latest line recieved from the GSM modem.
// The AT_LineReady flag indicates that a new line is ready and compares
//    AT_Line.
//
// Returns:
//    0 if string is equal to the last line from the GSM modem.
//    -1 if not equal
//
//---------------------------------------------------------------------------
int AT_isLine(char *string) {
   if ((AT_flags & AT_LineReady) != 0) {
      if (strCmp(AT_Line,string) == 0x00) {
         return 0;
      }
   }
   return -1;
}

//  AT_isLineReady
//
// Checks if a new line is ready for parsing or comparison
//
// Returns:
//    0 if a new line is ready
//    -1 if no new line is available
//
//---------------------------------------------------------------------------
int AT_isLineReady(void) {
   if ((AT_flags & AT_LineReady) != 0) {
      return 0;
   } else {
      return -1;
   }
}

//  AT_lineReset
//
// Reset and clear the current line and status
//
//---------------------------------------------------------------------------
void AT_lineReset(void) {
   int i = 0x0;
   while (i <= AT_Line_MAX_Length) {
      AT_Line[i++] = '\0';
   }
   AT_flags &= ~AT_LineReady;
}

//  AT_addChar
//
// Adds a character to the current line
//
//---------------------------------------------------------------------------
void AT_addChar(char newChar) {
   int i = 0x0;
   while ((i < AT_Line_MAX_Length) & (AT_Line[i] != '\0')) {
      i++;
   }
   if (i < AT_Line_MAX_Length) {
      AT_Line[i] = newChar;
      AT_Line[++i] = '\0';
   }
}

//  AT_sendCommand
//
// Sends a command to the GSM modem, the leading and trailing
//    <CR> should not be included in the string
//
// Returns:
//    0 if successful
//    -1 if unsuccessful
//
//---------------------------------------------------------------------------
int AT_sendCommand(unsigned char *string) {
   if ((AT_flags & (AT_NewStringToWrite + AT_NewStringToWriteOK + AT_flagSendEnergyReport))== 0x00) {
      AT_stringToWrite = string;
```

```
      AT_flags |= AT_NewStringToWrite;
      return 0;
   } else {
      return -1;
   }
}

//  AT_sendCommandOK
//
// Sends a command to the GSM modem, the leading and trailing
//    <CR> should not be included in the string.
// This function waits until the OK string is received from the GSM module
//    and includes some error handling.
//
// Returns:
//    0 if successful
//    -1 if unsuccessful
//
//---------------------------------------------------------------------------
int AT_sendCommandOK(unsigned char *string) {
   if ((AT_flags & (AT_NewStringToWrite + AT_NewStringToWriteOK + AT_flagSendEnergyReport)) == 0x00) {
      AT_stringToWrite = string;
      AT_flags |= AT_NewStringToWriteOK;
      return 0;
   } else {
      return -1;
   }
}

//  AT_SendEnergyReport
//
// This function signals the AT driver that a SMS must be sent containing all
//    the required energy values.
// The driver will output the values in Hex format. Each register is represented
//    by 8 hex characters, where A, B, C indicates phase and W, VA, VAR indicates
//    the energy unit as shown in the example below.
// SMS Contents:
//    WAAAAAAAWBBBBBBBWCCCCCCC:VARAAAAAVARBBBBBVARCCCCC:VAAAAAAAVABBBBBBVACCCCCC:
//
// Returns:
//    0 if successful
//    -1 if unsuccessful
//
//---------------------------------------------------------------------------
int AT_SendEnergyReport(unsigned char *string) {
   if ((AT_flags & (AT_NewStringToWrite + AT_NewStringToWriteOK + AT_flagSendEnergyReport)) == 0x00) {
      AT_stringToWrite = string;
      AT_flags |= AT_flagSendEnergyReport;
      return 0;
   } else {
      return -1;
   }
}

//  AT_isSendEnergyReportSent
//
// Confirms if the energy report is successfully sent.
//
// Returns:
//    0 if successful
//    -1 if unsuccessful
//
//---------------------------------------------------------------------------
int AT_isSendEnergyReportSent(void) {
   if ((AT_flags & (AT_flagSendEnergyReport)) == 0x00) {
      return 0;
   } else {
      return -1;
   }
}

//  AT_setGSMonSignal
//
// Pulls the On/Off pin high on the GSM module to engage startup procedure
//
//---------------------------------------------------------------------------
void AT_setGSMonSignal(void) {
   P2OUT |= AT_GSM_onOff;
}
```

```
//  AT_resetGSMonSignal
//
// Pulls the On/Off pin low on the GSM module to end the engage startup procedure
//    pulse.
//
//---------------------------------------------------------------------------
void AT_resetGSMonSignal(void) {
   P2OUT &= ~AT_GSM_onOff;
}

//  AT_isGSMonSignal
//
// Confirms if the On/Off signal is on.
//
// Returns:
//    0  if the signal is on
//    -1 if the signal is off.
//
//---------------------------------------------------------------------------
int AT_isGSMonSignal(void) {
   if (P2OUT & AT_GSM_onOff == 0) {
      return -1;
   } else {
      return 0;
   }
}

// AT_receiveStateDriver
//
// Handles the reception and decoding of data.
// Includes time outs if no data is recieved for some time.
//
//---------------------------------------------------------------------------
void AT_receiveStateDriver(void) {
   unsigned char c = 0x0;
   //State Events
   switch (AT_receiveState) {
      case AT_stReceiveInit:
            //Place holder for any initialization required
         AT_receiveState = AT_stReceiveIdle;
         break;
      case AT_stReceiveIdle:
         if (UART_isCharReady() == 0) {
            if ((UART_getChar() == '\n') | (UART_getChar() == '\r')) {
               if (AT_Line[0] != '\0') {
                  AT_flags |= AT_LineReady;
               }
            } else {
               AT_addChar(UART_getChar());
            }
         }
         break;
   }
}
// AT_sendStateDriver
//
// Handles the packaging and encoding of data.
//
//---------------------------------------------------------------------------
void AT_sendStateDriver(void) {
   switch (AT_sendState) {
      case AT_stSendInit:
         if (UART_writeChar(0x01B) == 0) {
            AT_sendState = AT_stSendIdle;
         }
         break;
      case AT_stSendIdle:
         if ((AT_flags & AT_NewStringToWrite) != 0) {
            AT_sendState = AT_stSend_StringStart;
         } else if ((AT_flags & AT_NewStringToWriteOK) != 0) {
            AT_sendState = AT_stSend_StringStart;
         }  else if ((AT_flags & AT_flagSendEnergyReport) != 0) {
            AT_sendState = AT_stSend_SMS_EnergyStart;
         }
         break;

      //SEND COMMAND
      case AT_stSend_StringStart:
         if (UART_writeString("\n\r") == 0) {
```

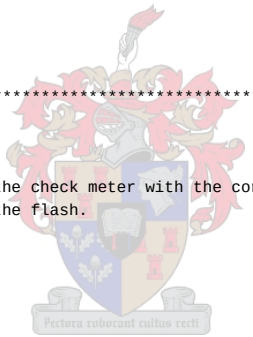
```
      AT_sendState = AT_stSend_StringStored;
   }
   break;
case AT_stSend_StringStored:
   if (UART_writeString(AT_stringToWrite) == 0) {
      AT_lineReset();      //Clear buffer while command is written to UART
      AT_sendState = AT_stSend_StringEnd;
   }
   break;
case AT_stSend_StringEnd:
   if (UART_writeString("\n\r") == 0) {
      if ((AT_flags & AT_NewStringToWrite) != 0) {
         AT_flags &= ~AT_NewStringToWrite;
         AT_sendState = AT_stSendIdle;
      } else {
         AT_sendState = AT_stSend_WaitOK;
      }
   }
   break;
case AT_stSend_WaitOK:
   if ((AT_flags & AT_LineReady) != 0x0) {
      if (AT_isLine("OK") == 0) {
         AT_flags &= ~AT_NewStringToWriteOK;
         AT_sendState = AT_stSendIdle;
      } else if (AT_isLine("ERROR") == 0) {
         TIMER_startTimer(tmrAT_Error_delay);
         AT_sendState = AT_stSend_WaitErrorDelay;
      }
   }
   break;
case AT_stSend_WaitErrorDelay:
   if (TIMER_isExpired(tmrAT_Error_delay,delayAT_Error_delay) == 0) {
      TIMER_stopTimer(tmrAT_Error_delay);
      AT_sendState = AT_stSend_StringStart;
   }
   break;
//SEND ENERGY REPORT
   //header
case AT_stSend_SMS_EnergyStart:
   if (UART_writeString("\n\rAT+CMGS=\"") == 0) {
      AT_sendState = AT_stSend_SMS_EnergyNumber;
   }
   break;
case AT_stSend_SMS_EnergyNumber:
   if (UART_writeString(AT_stringToWrite) == 0) {     //AT_stringToWrite contains the number to be sent
      AT_sendState = AT_stSend_SMS_EnergyPreReading;
   }
   break;
case AT_stSend_SMS_EnergyPreReading:
   if (UART_writeString("\"\n\r") == 0) {
      TIMER_startTimer(tmrAT_SMS_message);
      AT_sendState = AT_stSend_SMS_WaitMessageDelay;
   }
   break;
   //Meassage
case AT_stSend_SMS_WaitMessageDelay:
   if (TIMER_isExpired(tmrAT_SMS_message,delayAT_SMS_message) == 0) {
      TIMER_stopTimer(tmrAT_SMS_message);
      AT_lineReset();      //Clear buffer while command is written to UART
      AT_sendState = AT_stSend_SMS_EnergyWA;
   }
   break;
case AT_stSend_SMS_WaitMessagePointer:
   if (strCmp(AT_Line,"> ") == 0) {  //No new line characters are sent after this, therefor this implementation
      TIMER_startTimer(tmrAT_SMS_message);
      AT_sendState = AT_stSend_SMS_EnergyWA;
   }
   break;
case AT_stSend_SMS_EnergyWA:
   if (UART_writeLongHex(energy_W_A) == 0) {
      AT_sendState = AT_stSend_SMS_EnergyWB;
   }
   break;
case AT_stSend_SMS_EnergyWB:
   if (UART_writeLongHex(energy_W_B) == 0) {
      AT_sendState = AT_stSend_SMS_EnergyWC;
   }
   break;
case AT_stSend_SMS_EnergyWC:
```

```
   if (UART_writeLongHex(energy_W_C) == 0) {
      AT_sendState = AT_stSend_SMS_EnergyWColon;
   }
   break;
case AT_stSend_SMS_EnergyWColon:
   if (UART_writeChar(':') == 0) {
      AT_sendState = AT_stSend_SMS_EnergyVARA;
   }
   break;
case AT_stSend_SMS_EnergyVARA:
   if (UART_writeLongHex(energy_VAR_A) == 0) {
      AT_sendState = AT_stSend_SMS_EnergyVARB;
   }
   break;
case AT_stSend_SMS_EnergyVARB:
   if (UART_writeLongHex(energy_VAR_B) == 0) {
      AT_sendState = AT_stSend_SMS_EnergyVARC;
   }
   break;
case AT_stSend_SMS_EnergyVARC:
   if (UART_writeLongHex(energy_VAR_C) == 0) {
      AT_sendState = AT_stSend_SMS_EnergyVARColon;
   }
   break;
case AT_stSend_SMS_EnergyVARColon:
   if (UART_writeChar(':') == 0) {
      AT_sendState = AT_stSend_SMS_EnergyVAA;
   }
   break;
case AT_stSend_SMS_EnergyVAA:
   if (UART_writeLongHex(energy_VA_A) == 0) {
      AT_sendState = AT_stSend_SMS_EnergyVAB;
   }
   break;
case AT_stSend_SMS_EnergyVAB:
   if (UART_writeLongHex(energy_VA_B) == 0) {
      AT_sendState = AT_stSend_SMS_EnergyVAC;
   }
   break;
case AT_stSend_SMS_EnergyVAC:
   if (UART_writeLongHex(energy_VA_C) == 0) {
      AT_sendState = AT_stSend_SMS_EnergyVAColon;
   }
   break;
case AT_stSend_SMS_EnergyVAColon:
   if (UART_writeChar(':') == 0) {
      AT_sendState = AT_stSend_SMS_End;
   }
   break;
   //End
case AT_stSend_SMS_End:
   if (UART_writeChar(0x01A) == 0) {
      AT_lineReset();      //Clear buffer while command is written to UART
      AT_sendState = AT_stSend_SMS_WaitOK;
   }
   break;
case AT_stSend_SMS_LF:
   if (UART_writeString("\n\r") == 0) {
      TIMER_startTimer(tmrAT_SMS_messageWait);
      AT_sendState = AT_stSend_SMS_Wait;
   }
   break;
case AT_stSend_SMS_Wait:
   if (TIMER_isExpired(tmrAT_SMS_messageWait,delayAT_SMS_messageWait) == 0) {
      TIMER_stopTimer(tmrAT_SMS_messageWait);
      AT_sendState = AT_stSend_SMS_WaitOK;
   }
   break;
case AT_stSend_SMS_WaitOK:
   if ((AT_flags & AT_LineReady) != 0x0) {
      if (AT_isLine("OK") == 0) {
         AT_flags &= ~AT_flagSendEnergyReport;
         AT_sendState = AT_stSendIdle;
      } else if (AT_isLine("ERROR") == 0) {
         TIMER_startTimer(tmrAT_Error_delay);
         AT_sendState = AT_stSend_SMS_WaitErrorDelay;
      }
      AT_lineReset();      //Clear buffer while command is written to UART
   }
```

```
         break;
      case AT_stSend_SMS_WaitErrorDelay:
         if (TIMER_isExpired(tmrAT_Error_delay,delayAT_Error_delay) == 0) {
            TIMER_stopTimer(tmrAT_Error_delay);
            AT_sendState = AT_stSend_SMS_EnergyStart;
         }
         break;

   }
}

// AT_driver
//
// The driver is a state event machine and is the only software component
//    communicating on hardware level with the UART driver interface.
// The driver assumes that the AT interfaces is set and reset on a application
//    level.
//
//-----------------------------------------------------------------------------
void AT_driver(void) {
   if (UART_isRS232off() == 0) {
      AT_receiveStateDriver();
      AT_sendStateDriver();
   }
}
```

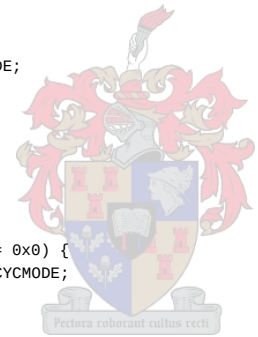
R.5 Applications

R.5.1 initMeter.h




```
//*****************************************************************************
//
// INITMETER
//
// This driver is responsible for setting up the check meter with the correct
//    registers and values, mostly read from the flash.
//
//
// History:
// v1.0 - 2004-07-20 Creation of Driver
//
//  R. Doorduin
//  PEG - University of Stellenbosch
//  March 2004
//*****************************************************************************
#define init_stInit                       0x000
#define init_stHeartBeatLED               0x002
#define init_stAD7758_InitOPMODE          0x003
#define init_stAD7758_InitLCYCMODE        0x004
#define init_stAD7758_InitCOMPMODE        0x014
#define init_stAD7758_InitGAIN            0x005
#define init_stAD7758_InitLINECYC         0x006
#define init_stAD7758_InitWDIV            0x007
#define init_stAD7758_InitVADIV           0x008
#define init_stAD7758_InitVARDIV          0x009
#define init_stFlash_ReadMem              0x00A
#define init_stFlash_ReadAD7758_AVRMSOS   0x00B
#define init_stFlash_ReadAD7758_BVRMSOS   0x00C
#define init_stFlash_ReadAD7758_CVRMSOS   0x00D
#define init_stFlash_ReadAD7758_AIRMSOS   0x00E
#define init_stFlash_ReadAD7758_BIRMSOS   0x00F
#define init_stFlash_ReadAD7758_CIRMSOS   0x010
#define init_stFlash_ReadAD7758_APHCAL    0x011
#define init_stFlash_ReadAD7758_BPHCAL    0x012
#define init_stFlash_ReadAD7758_CPHCAL    0x013
#define init_stClearAccumulators          0x0FE
#define init_stIdle                       0x0FF

//Variables
int init_state = init_stInit;

// INIT_init
//
// REsets the state machine to initialize the check meter.
```

```
//
//-----------------------------------------------------------------------
int INIT_init(void) {
   if ((init_state == init_stIdle) | (init_state == init_stInit)){
      init_state = init_stInit;
      return 0;
   } else {
      return -1;
   }
}

// INIT_isReady
//
// Tests if the initialization process is completed.
//-----------------------------------------------------------------------
int INIT_isReady(void) {
   if (init_state == init_stIdle) {
      return 0;
   } else {
      return -1;
   }
}

// INIT_driver
//
// The driver responsible for correct setup of register in the
//    check meter.
//
//-----------------------------------------------------------------------
void INIT_driver(void) {
   switch (init_state) {
      //Timer
      case init_stInit:
         TIMER_startTimer(tmrTestSE);
         init_state = init_stAD7758_InitOPMODE;
         break;
      case init_stHeartBeatLED:
            init_state = init_stInit;
         break;

      //AD7758
      case init_stAD7758_InitOPMODE:
         if (AD7758_writeChar(OPMODE,0x00) == 0x0) {
            init_state = init_stAD7758_InitLCYCMODE;
         }
         break;
      case init_stAD7758_InitLCYCMODE:
         if (AD7758_writeChar(LCYCMODE,0x078) == 0x0) {
            init_state = init_stAD7758_InitCOMPMODE;
         }
         break;
      case init_stAD7758_InitCOMPMODE:
         if (AD7758_writeChar(COMPMODE,0x09C) == 0x0) {
            init_state = init_stAD7758_InitGAIN;
         }
         break;
      case init_stAD7758_InitGAIN:
         if (AD7758_writeChar(GAIN,0x00) == 0x0) {
            init_state = init_stAD7758_InitLINECYC;
         }
         break;
      case init_stAD7758_InitLINECYC:
         if (AD7758_writeInt(LINECYC,0x01FF) == 0x0) {
            init_state = init_stAD7758_InitWDIV;
         }
         break;
      case init_stAD7758_InitWDIV:
         if (AD7758_writeChar(WDIV,0x07F) == 0x0) {
            init_state = init_stAD7758_InitVADIV;
         }
         break;
      case init_stAD7758_InitVADIV:
         if (AD7758_writeChar(VADIV,0x07F) == 0x0) {
            init_state = init_stAD7758_InitVARDIV;
         }
         break;
      case init_stAD7758_InitVARDIV:
         if (AD7758_writeChar(VARDIV,0x07F) == 0x0) {
            init_state = init_stFlash_ReadMem;
```



```
   }
   break;

//FLASH
case init_stFlash_ReadMem:
   MCU_WDIVA = FLASH_readLong(FLASHMEM_MCU_WDIVA);
   MCU_WDIVB = FLASH_readLong(FLASHMEM_MCU_WDIVB);
   MCU_WDIVC = FLASH_readLong(FLASHMEM_MCU_WDIVC);
   MCU_VARDIVA = FLASH_readLong(FLASHMEM_MCU_VARDIVA);
   MCU_VARDIVB = FLASH_readLong(FLASHMEM_MCU_VARDIVB);
   MCU_VARDIVC = FLASH_readLong(FLASHMEM_MCU_VARDIVC);
   MCU_VADIVA = FLASH_readLong(FLASHMEM_MCU_VADIVA);
   MCU_VADIVB = FLASH_readLong(FLASHMEM_MCU_VADIVB);
   MCU_VADIVC = FLASH_readLong(FLASHMEM_MCU_VADIVC);
   init_state = init_stFlash_ReadAD7758_AVRMSOS;
   break;
case init_stFlash_ReadAD7758_AVRMSOS:
   if (AD7758_writeInt(AVRMSOS,(0x00FFF & FLASH_readInt(FLASHMEM_AVRMSOS))) == 0x0) {
      init_state = init_stFlash_ReadAD7758_BVRMSOS;
   }
   break;
case init_stFlash_ReadAD7758_BVRMSOS:
   if (AD7758_writeInt(BVRMSOS,(0x00FFF & FLASH_readInt(FLASHMEM_BVRMSOS))) == 0x0) {
      init_state = init_stFlash_ReadAD7758_CVRMSOS;
   }
   break;
case init_stFlash_ReadAD7758_CVRMSOS:
   if (AD7758_writeInt(CVRMSOS,(0x00FFF & FLASH_readInt(FLASHMEM_CVRMSOS))) == 0x0) {
      init_state = init_stFlash_ReadAD7758_AIRMSOS;
   }
   break;
case init_stFlash_ReadAD7758_AIRMSOS:
   if (AD7758_writeInt(AIRMSOS,(0x00FFF & FLASH_readInt(FLASHMEM_AIRMSOS))) == 0x0) {
      init_state = init_stFlash_ReadAD7758_BIRMSOS;
   }
   break;
case init_stFlash_ReadAD7758_BIRMSOS:
   if (AD7758_writeInt(BIRMSOS,(0x00FFF & FLASH_readInt(FLASHMEM_BIRMSOS))) == 0x0) {
      init_state = init_stFlash_ReadAD7758_CIRMSOS;
   }
   break;
case init_stFlash_ReadAD7758_CIRMSOS:
   if (AD7758_writeInt(CIRMSOS,(0x00FFF & FLASH_readInt(FLASHMEM_CIRMSOS))) == 0x0) {
      init_state = init_stFlash_ReadAD7758_APHCAL;
   }
   break;
case init_stFlash_ReadAD7758_APHCAL:
   if (AD7758_writeChar(APHCAL,(0x007F & FLASH_readChar(FLASHMEM_APHCAL))) == 0x0) {
      init_state = init_stFlash_ReadAD7758_BPHCAL;
   }
   break;
case init_stFlash_ReadAD7758_BPHCAL:
   if (AD7758_writeChar(BPHCAL,(0x007F & FLASH_readChar(FLASHMEM_BPHCAL))) == 0x0) {
      init_state = init_stFlash_ReadAD7758_CPHCAL;
   }
   break;
case init_stFlash_ReadAD7758_CPHCAL:
   if (AD7758_writeChar(CPHCAL,(0x007F & FLASH_readChar(FLASHMEM_CPHCAL))) == 0x0) {
      init_state = init_stClearAccumulators;
   }
   break;
case init_stClearAccumulators:
   energy_W_a = 0x0; //Clear registers   -- Register inserted here, to ensure no uncallibrated data.
   energy_W_b = 0x0;
   energy_W_c = 0x0;
   energy_VAR_a = 0x0;
   energy_VAR_b = 0x0;
   energy_VAR_c = 0x0;
   energy_VA_a = 0x0;
   energy_VA_b = 0x0;
   energy_VA_c = 0x0;
   energy_W_A = 0x0;
   energy_W_B = 0x0;
   energy_W_C = 0x0;
   energy_VAR_A = 0x0;
   energy_VAR_B = 0x0;
   energy_VAR_C = 0x0;
   energy_VA_A = 0x0;
   energy_VA_B = 0x0;
```

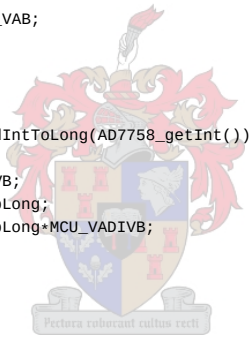
```
         energy_VA_C = 0x0;
         init_state = init_stIdle;
         break;
      case init_stIdle:

         break;
   }
}
```

R.5.2 energy.h

```
//*****************************************************************************
//
// Energy
//
// This header runs on an application level and is responsible for the measurement
//    and scaling of the energy measurements.
// All register must be set up prior to starting this state machine.
//
//
// History:
// v1.0 - 2004-07-24 Creation of Driver
//
//  R. Doorduin
//  PEG - University of Stellenbosch
//  March 2004
//*****************************************************************************

// State Event Declerations
//-----------------------------------------------------------------
//States
#define Energy_stInit            0x0000
#define Energy_stMeas_WA         0x0100
#define Energy_stMeasReady_WA    0x0101
#define Energy_stMeas_WB         0x0102
#define Energy_stMeasReady_WB    0x0103
#define Energy_stMeas_WC         0x0104
#define Energy_stMeasReady_WC    0x0105
#define Energy_stMeas_VARA       0x0106
#define Energy_stMeasReady_VARA  0x0107
#define Energy_stMeas_VARB       0x0108
#define Energy_stMeasReady_VARB  0x0109
#define Energy_stMeas_VARC       0x010A
#define Energy_stMeasReady_VARC  0x010B
#define Energy_stMeas_VAA        0x010C
#define Energy_stMeasReady_VAA   0x010D
#define Energy_stMeas_VAB        0x010E
#define Energy_stMeasReady_VAB   0x010F
#define Energy_stMeas_VAC        0x0110
#define Energy_stMeasReady_VAC   0x0111
#define Energy_stTimer           0x0112
//Variables
int Energy_state = Energy_stInit;

int Energy_isEnergyIdle(void) {
   if (Energy_state == Energy_stTimer) {
      return 0;
   } else {
      return -1;
   }
}

// Energy_driver
//
// The driver responsible for various tasks.
// The measurement are stored in the variable at memory.h
//-----------------------------------------------------------------------------------------------------------
void Energy_Driver(void) {
   signed long tmpLong = 0x0;
   LEDs_toggleLED2();
   switch (Energy_state) {
      case Energy_stInit:
            Energy_state = Energy_stMeas_WA;
         break;
      case Energy_stMeas_WA:
         if (AD7758_readInt(AWATTHR) == 0x0) {  //Measurement
            Energy_state = Energy_stMeasReady_WA;
```

```
      }
      break;
   case Energy_stMeasReady_WA:
      if (AD7758_isDataReady() == 0x0) {
         energy_W_a = energy_W_a + signedIntToLong(AD7758_getInt());
         if ((energy_W_a >= MCU_WDIVA) | (energy_W_a <= -MCU_WDIVA)) {  //Scaling
            tmpLong = energy_W_a/MCU_WDIVA;
            energy_W_A = energy_W_A + tmpLong;
            energy_W_a = energy_W_a - tmpLong*MCU_WDIVA;
         }
         Energy_state = Energy_stMeas_WB;
      }
      break;
   case Energy_stMeas_WB:
      if (AD7758_readInt(BWATTHR) == 0x0) {
         Energy_state = Energy_stMeasReady_WB;
      }
      break;
   case Energy_stMeasReady_WB:
      if (AD7758_isDataReady() == 0x0) {
         energy_W_b = energy_W_b + signedIntToLong(AD7758_getInt());
         if ((energy_W_b >= MCU_WDIVB) | (energy_W_b <= -MCU_WDIVB)) {
            tmpLong = energy_W_b/MCU_WDIVB;
            energy_W_B = energy_W_B + tmpLong;
            energy_W_b = energy_W_b - tmpLong*MCU_WDIVB;
         }
         Energy_state = Energy_stMeas_WC;
      }
      break;
   case Energy_stMeas_WC:
      if (AD7758_readInt(CWATTHR) == 0x0) {
         Energy_state = Energy_stMeasReady_WC;
      }
      break;
   case Energy_stMeasReady_WC:
      if (AD7758_isDataReady() == 0x0) {
         energy_W_c = energy_W_c + signedIntToLong(AD7758_getInt());
         if ((energy_W_c >= MCU_WDIVC) | (energy_W_c <= -MCU_WDIVC)) {
            tmpLong = energy_W_c/MCU_WDIVC;
            energy_W_C = energy_W_C + tmpLong;
            energy_W_c = energy_W_c - tmpLong*MCU_WDIVC;
         }
         Energy_state = Energy_stMeas_VARA;
      }
      break;
   case Energy_stMeas_VARA:
      if (AD7758_readInt(AVARHR) == 0x0) {
         Energy_state = Energy_stMeasReady_VARA;
      }
      break;
   case Energy_stMeasReady_VARA:
      if (AD7758_isDataReady() == 0x0) {
         energy_VAR_a = energy_VAR_a + signedIntToLong(AD7758_getInt());
         if ((energy_VAR_a >= MCU_VARDIVA) | (energy_VAR_a <= -MCU_VARDIVA)) {
            tmpLong = energy_VAR_a/MCU_VARDIVA;
            energy_VAR_A = energy_VAR_A + tmpLong;
            energy_VAR_a = energy_VAR_a - tmpLong*MCU_VARDIVA;
         }
         Energy_state = Energy_stMeas_VARB;
      }
      break;
   case Energy_stMeas_VARB:
      if (AD7758_readInt(BVARHR) == 0x0) {
         Energy_state = Energy_stMeasReady_VARB;
      }
      break;
   case Energy_stMeasReady_VARB:
      if (AD7758_isDataReady() == 0x0) {
         energy_VAR_b = energy_VAR_b + signedIntToLong(AD7758_getInt());
         if ((energy_VAR_b >= MCU_VARDIVB) | (energy_VAR_b <= -MCU_VARDIVB)) {
            tmpLong = energy_VAR_b/MCU_VARDIVB;
            energy_VAR_B = energy_VAR_B + tmpLong;
            energy_VAR_b = energy_VAR_b - tmpLong*MCU_VARDIVB;
         }
         Energy_state = Energy_stMeas_VARC;
      }
      break;
   case Energy_stMeas_VARC:
      if (AD7758_readInt(CVARHR) == 0x0) {
```

```
            Energy_state = Energy_stMeasReady_VARC;
         }
         break;
      case Energy_stMeasReady_VARC:
         if (AD7758_isDataReady() == 0x0) {
            energy_VAR_c = energy_VAR_c + signedIntToLong(AD7758_getInt());
            if ((energy_VAR_c >= MCU_VARDIVC) | (energy_VAR_c <= -MCU_VARDIVC)) {
               tmpLong = energy_VAR_c/MCU_VARDIVC;
               energy_VAR_C = energy_VAR_C + tmpLong;
               energy_VAR_c = energy_VAR_c - tmpLong*MCU_VARDIVC;
            }
            Energy_state = Energy_stMeas_VAA;
         }
         break;
      case Energy_stMeas_VAA:
         if (AD7758_readInt(AVAHR) == 0x0) {
            Energy_state = Energy_stMeasReady_VAA;
         }
         break;
      case Energy_stMeasReady_VAA:
         if (AD7758_isDataReady() == 0x0) {
            energy_VA_a = energy_VA_a + signedIntToLong(AD7758_getInt());
            if (energy_VA_a >= MCU_VADIVA) {
               tmpLong = energy_VA_a/MCU_VADIVA;
               energy_VA_A = energy_VA_A + tmpLong;
               energy_VA_a = energy_VA_a - tmpLong*MCU_VADIVA;
            }
            Energy_state = Energy_stMeas_VAB;
         }
         break;
      case Energy_stMeas_VAB:
         if (AD7758_readInt(BVAHR) == 0x0) {
            Energy_state = Energy_stMeasReady_VAB;
         }
         break;
      case Energy_stMeasReady_VAB:
         if (AD7758_isDataReady() == 0x0) {
            energy_VA_b = energy_VA_b + signedIntToLong(AD7758_getInt());
            if (energy_VA_b >= MCU_VADIVB) {
               tmpLong = energy_VA_b/MCU_VADIVB;
               energy_VA_B = energy_VA_B + tmpLong;
               energy_VA_b = energy_VA_b - tmpLong*MCU_VADIVB;
            }
            Energy_state = Energy_stMeas_VAC;
         }
         break;
      case Energy_stMeas_VAC:
         if (AD7758_readInt(CVAHR) == 0x0) {
            Energy_state = Energy_stMeasReady_VAC;
         }
         break;
      case Energy_stMeasReady_VAC:
         if (AD7758_isDataReady() == 0x0) {
            energy_VA_c = energy_VA_c + signedIntToLong(AD7758_getInt());
            if (energy_VA_c >= MCU_VADIVC) {
               tmpLong = energy_VA_c/MCU_VADIVC;
               energy_VA_C = energy_VA_C + tmpLong;
               energy_VA_c = energy_VA_c - tmpLong*MCU_VADIVC;
            }
            TIMER_startTimer(tmrEnergyMeasurement);
            Energy_state = Energy_stTimer;
         }
         break;
      case Energy_stTimer:
         if (TIMER_isExpired(tmrEnergyMeasurement,delayEnergyMeasurement) == 0) {
            TIMER_stopTimer(tmrEnergyMeasurement);
            LEDs_toggleLED1(); //Energy Measurement Cycle
            Energy_state = Energy_stMeas_WA;
         }
         break;
      default:
         Energy_state = Energy_stMeas_WA;
         break;
   }
}
```

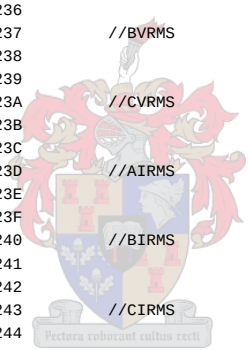


R.5.3 onLine.h

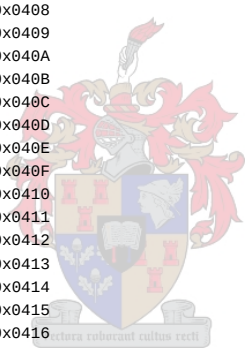
```
//*****************************************************************************
//
// OnLine
//
// This header runs on an application level and is solely responsible for the
//    callibration and coordinating all operations when in online mode.
//
//
// History:
// v1.0 - 2004-07-20 Creation of Driver
//
//  R. Doorduin
//  PEG - University of Stellenbosch
//  March 2004
//*****************************************************************************


// State Event Declerations
//-----------------------------------------------------------------
//Driver
#define OnLine_stInit                           0x00
#define OnLine_stIdle                           0x01
#define OnLine_stInitStart                      0x02
#define OnLine_stInitSystem                     0x03

//AD7758
#define OnLine_stWriteAD7758Char       0x017
#define OnLine_stReadAD7758Char        0x018
#define OnLine_stCheckDataReadyChar    0x019
#define OnLine_stSendUartChar          0x01A
#define OnLine_stWriteAD7758Int        0x01B
#define OnLine_stReadAD7758Int         0x01C
#define OnLine_stCheckDataReadyInt     0x01D
#define OnLine_stSendUartInt           0x01E
#define OnLine_stWriteAD7758Long       0x01F
#define OnLine_stReadAD7758Long        0x020
#define OnLine_stCheckDataReadyLong    0x021
#define OnLine_stSendUartLong          0x022

//FLASH
#define OnLine_stFlash_WriteMem             0x040
#define OnLine_stFlash_Write_WDIVA          0x041
#define OnLine_stFlash_Write_WDIVB          0x042
#define OnLine_stFlash_Write_WDIVC          0x043
#define OnLine_stFlash_Write_VARDIVA        0x044
#define OnLine_stFlash_Write_VARDIVB        0x045
#define OnLine_stFlash_Write_VARDIVC        0x046
#define OnLine_stFlash_Write_VADIVA         0x047
#define OnLine_stFlash_Write_VADIVB         0x048
#define OnLine_stFlash_Write_VADIVC         0x049
#define OnLine_stFlash_AD7758Read_AVRMSOS   0x04A
#define OnLine_stFlash_Write_AVRMSOS        0x04B
#define OnLine_stFlash_AD7758Read_BVRMSOS   0x04C
#define OnLine_stFlash_Write_BVRMSOS        0x04D
#define OnLine_stFlash_AD7758Read_CVRMSOS   0x04E
#define OnLine_stFlash_Write_CVRMSOS        0x04F
#define OnLine_stFlash_AD7758Read_AIRMSOS   0x050
#define OnLine_stFlash_Write_AIRMSOS        0x051
#define OnLine_stFlash_AD7758Read_BIRMSOS   0x052
#define OnLine_stFlash_Write_BIRMSOS        0x053
#define OnLine_stFlash_AD7758Read_CIRMSOS   0x054
#define OnLine_stFlash_Write_CIRMSOS        0x055
#define OnLine_stFlash_AD7758Read_APHCAL    0x056
#define OnLine_stFlash_Write_APHCAL         0x057
#define OnLine_stFlash_AD7758Read_BPHCAL    0x058
#define OnLine_stFlash_Write_BPHCAL         0x059
#define OnLine_stFlash_AD7758Read_CPHCAL    0x05A
#define OnLine_stFlash_Write_CPHCAL         0x05B

//MCU
#define OnLine_stMainScreenData             0x0130   //Main Screen Register Dump
#define OnLine_stMainScreenInitRead_RSTATUS 0x0131      //RSTATUS
#define OnLine_stMainScreenRead_RSTATUS     0x0132
#define OnLine_stMainScreenSend_RSTATUS     0x0133
#define OnLine_stMainScreenInitRead_OPMODE  0x0134      //OPMODE
#define OnLine_stMainScreenRead_OPMODE      0x0135
#define OnLine_stMainScreenSend_OPMODE      0x0136
```

```
#define OnLine_stMainScreenInitRead_MMODE    0x0137      //MMODE
#define OnLine_stMainScreenRead_MMODE        0x0138
#define OnLine_stMainScreenSend_MMODE        0x0139
#define OnLine_stMainScreenInitRead_WAVMODE  0x013A      //WAVMODE
#define OnLine_stMainScreenRead_WAVMODE      0x013B
#define OnLine_stMainScreenSend_WAVMODE      0x013C
#define OnLine_stMainScreenInitRead_COMPMODE 0x013D      //COMPMODE
#define OnLine_stMainScreenRead_COMPMODE     0x013E
#define OnLine_stMainScreenSend_COMPMODE     0x013F
#define OnLine_stMainScreenInitRead_LCYCMODE 0x0140      //LCYCMODE
#define OnLine_stMainScreenRead_LCYCMODE     0x0141
#define OnLine_stMainScreenSend_LCYCMODE     0x0142
#define OnLine_stMainScreenInitRead_MASK     0x0143      //MASK
#define OnLine_stMainScreenRead_MASK         0x0144
#define OnLine_stMainScreenSend_MASK         0x0145
#define OnLine_stMainScreenInitRead_LINECYC  0x0146      //LINECYC
#define OnLine_stMainScreenRead_LINECYC      0x0147
#define OnLine_stMainScreenSend_LINECYC      0x0148
#define OnLine_stMainScreenInitRead_AVRMSOS  0x0149      //AVRMSOS
#define OnLine_stMainScreenRead_AVRMSOS      0x014A
#define OnLine_stMainScreenSend_AVRMSOS      0x014B
#define OnLine_stMainScreenInitRead_BVRMSOS  0x014C      //BVRMSOS
#define OnLine_stMainScreenRead_BVRMSOS      0x014D
#define OnLine_stMainScreenSend_BVRMSOS      0x014E
#define OnLine_stMainScreenInitRead_CVRMSOS  0x014F      //CVRMSOS
#define OnLine_stMainScreenRead_CVRMSOS      0x0150
#define OnLine_stMainScreenSend_CVRMSOS      0x0151
#define OnLine_stMainScreenInitRead_AIRMSOS  0x0152      //AIRMSOS
#define OnLine_stMainScreenRead_AIRMSOS      0x0153
#define OnLine_stMainScreenSend_AIRMSOS      0x0154
#define OnLine_stMainScreenInitRead_BIRMSOS  0x0155      //BIRMSOS
#define OnLine_stMainScreenRead_BIRMSOS      0x0156
#define OnLine_stMainScreenSend_BIRMSOS      0x0157
#define OnLine_stMainScreenInitRead_CIRMSOS  0x0158      //CIRMSOS
#define OnLine_stMainScreenRead_CIRMSOS      0x0159
#define OnLine_stMainScreenSend_CIRMSOS      0x015A
#define OnLine_stMainScreenInitRead_APCFNUM  0x015B      //APCFNUM
#define OnLine_stMainScreenRead_APCFNUM      0x015C
#define OnLine_stMainScreenSend_APCFNUM      0x015D
#define OnLine_stMainScreenInitRead_APCFDEN  0x015E      //APCFDEN
#define OnLine_stMainScreenRead_APCFDEN      0x015F
#define OnLine_stMainScreenSend_APCFDEN      0x0160
#define OnLine_stMainScreenInitRead_VARCFNUM 0x0161      //VARCFDEN
#define OnLine_stMainScreenRead_VARCFNUM     0x0162
#define OnLine_stMainScreenSend_VARCFNUM     0x0163
#define OnLine_stMainScreenInitRead_VARCFDEN 0x0164      //VARCFDEN
#define OnLine_stMainScreenRead_VARCFDEN     0x0165
#define OnLine_stMainScreenSend_VARCFDEN     0x0166
#define OnLine_stMainScreenInitRead_AWG      0x0167      //AWG
#define OnLine_stMainScreenRead_AWG          0x0168
#define OnLine_stMainScreenSend_AWG          0x016A
#define OnLine_stMainScreenInitRead_BWG      0x016B      //BWG
#define OnLine_stMainScreenRead_BWG          0x016C
#define OnLine_stMainScreenSend_BWG          0x016D
#define OnLine_stMainScreenInitRead_CWG      0x016E      //CWG
#define OnLine_stMainScreenRead_CWG          0x016F
#define OnLine_stMainScreenSend_CWG          0x0170
#define OnLine_stMainScreenInitRead_AVAG     0x0171      //AVAG
#define OnLine_stMainScreenRead_AVAG         0x0172
#define OnLine_stMainScreenSend_AVAG         0x0173
#define OnLine_stMainScreenInitRead_BVAG     0x0174      //BVAG
#define OnLine_stMainScreenRead_BVAG         0x0175
#define OnLine_stMainScreenSend_BVAG         0x0176
#define OnLine_stMainScreenInitRead_CVAG     0x0177      //CVAG
#define OnLine_stMainScreenRead_CVAG         0x0178
#define OnLine_stMainScreenSend_CVAG         0x0179
#define OnLine_stMainScreenInitRead_AVARG    0x017A      //AVARG
#define OnLine_stMainScreenRead_AVARG        0x017B
#define OnLine_stMainScreenSend_AVARG        0x017C
#define OnLine_stMainScreenInitRead_BVARG    0x017D      //BVARG
#define OnLine_stMainScreenRead_BVARG        0x017E
#define OnLine_stMainScreenSend_BVARG        0x017F
#define OnLine_stMainScreenInitRead_CVARG    0x0180      //CVARG
#define OnLine_stMainScreenRead_CVARG        0x0181
#define OnLine_stMainScreenSend_CVARG        0x0182
#define OnLine_stMainScreenInitRead_WDIV     0x0183      //WDIV
#define OnLine_stMainScreenRead_WDIV         0x0184
#define OnLine_stMainScreenSend_WDIV         0x0185
#define OnLine_stMainScreenInitRead_VADIV    0x0186      //VADIV
```

```
#define OnLine_stMainScreenRead_VADIV        0x0187
#define OnLine_stMainScreenSend_VADIV        0x0188
#define OnLine_stMainScreenInitRead_VARDIV   0x0189      //VARDIV
#define OnLine_stMainScreenRead_VARDIV       0x018A
#define OnLine_stMainScreenSend_VARDIV       0x018B
#define OnLine_stMainScreenSend_MCU_WDIVA    0x018C      //MCU_WDIVA
#define OnLine_stMainScreenSend_MCU_WDIVB    0x018D      //MCU_WDIVB
#define OnLine_stMainScreenSend_MCU_WDIVC    0x018E      //MCU_WDIVC
#define OnLine_stMainScreenSend_MCU_VARDIVA  0x018F      //MCU_VARDIVA
#define OnLine_stMainScreenSend_MCU_VARDIVB  0x0190      //MCU_VARDIVB
#define OnLine_stMainScreenSend_MCU_VARDIVC  0x0191      //MCU_VARDIVC
#define OnLine_stMainScreenSend_MCU_VADIVA   0x0192      //MCU_VADIVA
#define OnLine_stMainScreenSend_MCU_VADIVB   0x0193      //MCU_VADIVB
#define OnLine_stMainScreenSend_MCU_VADIVC   0x0194      //MCU_VADIVC
#define OnLine_stMainScreenInitRead_APHCAL   0x0195      //APHCAL
#define OnLine_stMainScreenRead_APHCAL       0x0196
#define OnLine_stMainScreenSend_APHCAL       0x0197
#define OnLine_stMainScreenInitRead_BPHCAL   0x0198      //BPHCAL
#define OnLine_stMainScreenRead_BPHCAL       0x0199
#define OnLine_stMainScreenSend_BPHCAL       0x019A
#define OnLine_stMainScreenInitRead_CPHCAL   0x019B      //CPHCAL
#define OnLine_stMainScreenRead_CPHCAL       0x019C
#define OnLine_stMainScreenSend_CPHCAL       0x019D

#define OnLine_stMainScreenDataEnd           0x01FF

#define OnLine_stMonitor                 0x0231      //Monitor
#define OnLine_stMonitorInit_LCYCMODE    0x0232         //Init
#define OnLine_stMonitorInit_MASK        0x0233
#define OnLine_stMonitorInitRead_AVRMS   0x0234         //AVRMS
#define OnLine_stMonitorRead_AVRMS       0x0235
#define OnLine_stMonitorSend_AVRMS       0x0236
#define OnLine_stMonitorInitRead_BVRMS   0x0237         //BVRMS
#define OnLine_stMonitorRead_BVRMS       0x0238
#define OnLine_stMonitorSend_BVRMS       0x0239
#define OnLine_stMonitorInitRead_CVRMS   0x023A         //CVRMS
#define OnLine_stMonitorRead_CVRMS       0x023B
#define OnLine_stMonitorSend_CVRMS       0x023C
#define OnLine_stMonitorInitRead_AIRMS   0x023D         //AIRMS
#define OnLine_stMonitorRead_AIRMS       0x023E
#define OnLine_stMonitorSend_AIRMS       0x023F
#define OnLine_stMonitorInitRead_BIRMS   0x0240         //BIRMS
#define OnLine_stMonitorRead_BIRMS       0x0241
#define OnLine_stMonitorSend_BIRMS       0x0242
#define OnLine_stMonitorInitRead_CIRMS   0x0243         //CIRMS
#define OnLine_stMonitorRead_CIRMS       0x0244
#define OnLine_stMonitorSend_CIRMS       0x0245
#define OnLine_stMonitorInitRead_AWATTHR 0x0246         //AWATTHR
#define OnLine_stMonitorRead_AWATTHR     0x0247
#define OnLine_stMonitorSend_AWATTHR     0x0248
#define OnLine_stMonitorInitRead_BWATTHR 0x0249         //BWATTHR
#define OnLine_stMonitorRead_BWATTHR     0x024A
#define OnLine_stMonitorSend_BWATTHR     0x024B
#define OnLine_stMonitorInitRead_CWATTHR 0x024C         //CWATTHR
#define OnLine_stMonitorRead_CWATTHR     0x024D
#define OnLine_stMonitorSend_CWATTHR     0x024E
#define OnLine_stMonitorInitRead_AVARHR  0x024F         //AVARHR
#define OnLine_stMonitorRead_AVARHR      0x0250
#define OnLine_stMonitorSend_AVARHR      0x0251
#define OnLine_stMonitorInitRead_BVARHR  0x0252         //BVARHR
#define OnLine_stMonitorRead_BVARHR      0x0253
#define OnLine_stMonitorSend_BVARHR      0x0254
#define OnLine_stMonitorInitRead_CVARHR  0x0255         //CVARHR
#define OnLine_stMonitorRead_CVARHR      0x0256
#define OnLine_stMonitorSend_CVARHR      0x0257
#define OnLine_stMonitorInitRead_AVAHR   0x0258         //AVAHR
#define OnLine_stMonitorRead_AVAHR       0x0259
#define OnLine_stMonitorSend_AVAHR       0x025A
#define OnLine_stMonitorInitRead_BVAHR   0x025B         //BVAHR
#define OnLine_stMonitorRead_BVAHR       0x025C
#define OnLine_stMonitorSend_BVAHR       0x025D
#define OnLine_stMonitorInitRead_CVAHR   0x025E         //CVAHR
#define OnLine_stMonitorRead_CVAHR       0x025F
#define OnLine_stMonitorSend_CVAHR       0x0260
#define OnLine_stMonitorInitRead_TEMP    0x0278         //TEMP
#define OnLine_stMonitorRead_TEMP        0x0279
#define OnLine_stMonitorSend_TEMP        0x027A
#define OnLine_stMonitorInitRead_FREQ    0x027B         //FREQ
#define OnLine_stMonitorRead_FREQ        0x027C
```



```
#define OnLine_stMonitorSend_FREQ         0x027D
#define OnLine_stMonitorSendAck           0x027E        //End of Cycle
#define OnLine_stMonitorSendMonitor       0x027F

#define OnLine_stOffCal_InitLCYCMODE              0x0300      //OffsetCallibration
#define OnLine_stOffCal_InitMASK                  0x0301
#define OnLine_stOffCal_InitXRMSOS                0x0302
#define OnLine_stOffCal_InitRead_LCYCMODE         0x0303
#define OnLine_stOffCal_Read_LCYCMODE             0x0304
#define OnLine_stOffCal_Send_LCYCMODE             0x0305
#define OnLine_stOffCal_InitRead_MASK             0x0306
#define OnLine_stOffCal_Read_MASK                 0x0307
#define OnLine_stOffCal_Send_MASK                 0x0308
#define OnLine_stOffCal_Send_Ack                  0x0309
#define OnLine_stOffCal_ResetInt                  0x030A
#define OnLine_stOffCal_ResetIntWait              0x030B
#define OnLine_stOffCal_ReadRegister              0x030C
#define OnLine_stOffCal_AccumulateRegister        0x030D
#define OnLine_stOffCal_SendRegisterToRS232       0x03FC
#define OnLine_stOffCal_SendCurrentCalToRS232     0x03FD
#define OnLine_stOffCal_SendAccumulatorCalToRS232 0x03FE
#define OnLine_stOffCal_SendFinal_Ack             0x03FF

#define OnLine_stGainCal_InitXWG            0x0400      //Energy Gain Callibration
#define OnLine_stGainCal_InitXVARG          0x0401
#define OnLine_stGainCal_InitXVAG           0x0402
#define OnLine_stGainCal_InitLCYCMODE       0x0403
#define OnLine_stGainCal_InitRead_LCYCMODE  0x0404
#define OnLine_stGainCal_Read_LCYCMODE      0x0405
#define OnLine_stGainCal_Send_LCYCMODE      0x0406
#define OnLine_stGainCal_InitRead_LINECYC   0x0407
#define OnLine_stGainCal_Read_LINECYC       0x0408
#define OnLine_stGainCal_Send_LINECYC       0x0409
#define OnLine_stGainCal_InitMASK           0x040A
#define OnLine_stGainCal_InitRead_MASK      0x040B
#define OnLine_stGainCal_Read_MASK          0x040C
#define OnLine_stGainCal_Send_MASK          0x040D
#define OnLine_stGainCal_InitRead_FREQ      0x040E
#define OnLine_stGainCal_Read_FREQ          0x040F
#define OnLine_stGainCal_Send_FREQ          0x0410
#define OnLine_stGainCal_FirstResetInt      0x0411
#define OnLine_stGainCal_ResetInt           0x0412
#define OnLine_stGainCal_ResetIntWait       0x0413
#define OnLine_stGainCal_InitRead_xWATTHR   0x0414
#define OnLine_stGainCal_Read_xWATTHR       0x0415
#define OnLine_stGainCal_Send_xWATTHR       0x0416
#define OnLine_stGainCal_InitRead_xVARHR    0x0417
#define OnLine_stGainCal_Read_xVARHR        0x0418
#define OnLine_stGainCal_Send_xVARHR        0x0419
#define OnLine_stGainCal_InitRead_xVAHR     0x041A
#define OnLine_stGainCal_Read_xVAHR         0x041B
#define OnLine_stGainCal_Send_xVAHR         0x041C
#define OnLine_stGainCal_InitRead_WDIV      0x041D
#define OnLine_stGainCal_Read_WDIV          0x041E
#define OnLine_stGainCal_Send_WDIV          0x041F
#define OnLine_stGainCal_InitRead_VARDIV    0x0420
#define OnLine_stGainCal_Read_VARDIV        0x0421
#define OnLine_stGainCal_Send_VARDIV        0x0422
#define OnLine_stGainCal_InitRead_VADIV     0x0423
#define OnLine_stGainCal_Read_VADIV         0x0424
#define OnLine_stGainCal_Send_VADIV         0x0425
#define OnLine_stGainCal_Send_CurrentCal    0x0426
#define OnLine_stGainCal_SendFinal_Ack      0x04FF

#define OnLine_stEnergyM_InitMASK           0x0500      //Energy Measurement
#define OnLine_stEnergyM_InitLCYCMODE       0x0501
#define OnLine_stEnergyM_InitLINECYC        0x0502
#define OnLine_stEnergyM_InitCOMPMODE       0x0503
#define OnLine_stEnergyM_ReadAWATTHR        0x0504
#define OnLine_stEnergyM_ReadBWATTHR        0x0505
#define OnLine_stEnergyM_ReadCWATTHR        0x0506
#define OnLine_stEnergyM_ReadAVARHR         0x0507
#define OnLine_stEnergyM_ReadBVARHR         0x0508
#define OnLine_stEnergyM_ReadCVARHR         0x0509
#define OnLine_stEnergyM_ReadAVAHR          0x050A
#define OnLine_stEnergyM_ReadBVAHR          0x050B
#define OnLine_stEnergyM_ReadCVAHR          0x050C
#define OnLine_stEnergyM_ReadRSTATUS        0x050D
```



```
#define OnLine_stEnergySend_W               0x0601   //EnergyMeasurement Send
#define OnLine_stEnergySend_VAR             0x0602
#define OnLine_stEnergySend_VA              0x0603
#define OnLine_stEnergySend_W_A             0x0604   //EnergyMeasurement Send
#define OnLine_stEnergySend_VAR_A           0x0605
#define OnLine_stEnergySend_VA_A            0x0606
#define OnLine_stEnergySend_W_B             0x0607   //EnergyMeasurement Send
#define OnLine_stEnergySend_VAR_B           0x0608
#define OnLine_stEnergySend_VA_B            0x0609
#define OnLine_stEnergySend_W_C             0x060A   //EnergyMeasurement Send
#define OnLine_stEnergySend_VAR_C           0x060B
#define OnLine_stEnergySend_VA_C            0x060C

#define OnLine_stFreqCal_Init               0x0701   //Frequency Callibration
#define OnLine_stFreqCal_ReadData           0x0702
#define OnLine_stFreqCal_SendAD7758Data     0x0703
#define OnLine_stFreqCal_SendCalAck         0x0704

#define OnLine_stPhaseCal_InitXWG           0x0800     //Phase Callibration
#define OnLine_stPhaseCal_InitLCYCMODE      0x0803
#define OnLine_stPhaseCal_InitRead_LCYCMODE 0x0804
#define OnLine_stPhaseCal_Read_LCYCMODE     0x0805
#define OnLine_stPhaseCal_Send_LCYCMODE     0x0806
#define OnLine_stPhaseCal_InitRead_LINECYC  0x0807
#define OnLine_stPhaseCal_Read_LINECYC      0x0808
#define OnLine_stPhaseCal_Send_LINECYC      0x0809
#define OnLine_stPhaseCal_InitMASK          0x080A
#define OnLine_stPhaseCal_InitRead_MASK     0x080B
#define OnLine_stPhaseCal_Read_MASK         0x080C
#define OnLine_stPhaseCal_Send_MASK         0x080D
#define OnLine_stPhaseCal_InitRead_FREQ     0x080E
#define OnLine_stPhaseCal_Read_FREQ         0x080F
#define OnLine_stPhaseCal_Send_FREQ         0x0810
#define OnLine_stPhaseCal_FirstResetInt     0x0811
#define OnLine_stPhaseCal_ResetInt          0x0812
#define OnLine_stPhaseCal_ResetIntWait      0x0813
#define OnLine_stPhaseCal_InitRead_xWATTHR  0x0814
#define OnLine_stPhaseCal_Read_xWATTHR      0x0815
#define OnLine_stPhaseCal_Send_xWATTHR      0x0816


//Variables
int OnLine_state = OnLine_stInit;

//Flags
#define OnLine_flagInit                BIT0
#define OnLine_flagGainCalOverflow     BIT1
#define OnLine_flagEnergyMeasurement   BIT2
int OnLine_flags                    = 0x00;     //Flags to allow the driver to keep track of user input

//OnLineVariables
//Protocol
unsigned char OnLine_device         = 0x0;
unsigned char OnLine_deviceRegister = 0x0;
unsigned char OnLine_dataChar       = 0x00;
unsigned int OnLine_dataInt         = 0x00;
unsigned long OnLine_dataLong       = 0x00;

//Calibration variables
char OnLine_Cal_xWG              = 0x0;
char OnLine_Cal_xVARG            = 0x0;
char OnLine_Cal_xVAG             = 0x0;
char OnLine_Cal_xXMRSOS          = 0x0;
char OnLine_Cal_xXRMS            = 0x0;
char OnLine_Cal_xWATTHR          = 0x0;
char OnLine_Cal_xVARHR           = 0x0;
char OnLine_Cal_xVAHR            = 0x0;

char OnLine_Cal_lcycmode         = 0x0;
long OnLine_Cal_mask             = 0x0;
int  OnLine_Cal_n                = 0x0;
char OnLine_Cal_currentCal       = 0x0;
long OnLine_Cal_accumulator      = 0x0;
#define OnLine_Cal_AD7758_N      0x040


// OnLine_driver
//
// The driver responsible for coordinating various tasks.
```

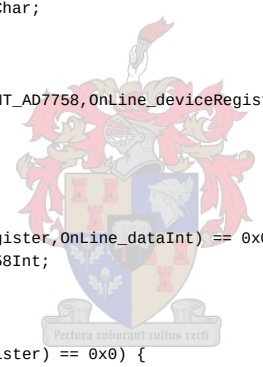
```
// Tasks are recieved via the RS232 port.
//
//-------------------------------------------------------------------------------------------------------------
void OnLine_driver() {
   //Energy measurement should be continuesly on if the enrgyMeasurement Callibration is on.
   if ((OnLine_flags & OnLine_flagEnergyMeasurement) != 0) {
      Energy_Driver();
   }

   switch (OnLine_state) {
      //Initialization
      //-------------------------------------------------------
      //Timer and UART
      case OnLine_stInit:
         UART_setRS232on();
         if (UART_isRS232on() == 0) {
            //Start General Timers
            OnLine_state = OnLine_stInitStart;
         }
         break;
      case OnLine_stInitStart:
         if (INIT_init() == 0) {
            OnLine_state = OnLine_stInitSystem;
         }
         break;
      case OnLine_stInitSystem:
         if (INIT_isReady() != 0) {
            INIT_driver();
         } else {
            TIMER_startTimer(tmrWDTOnLine);
            OnLine_state = OnLine_stIdle;
         }
         break;


      //IDLE
      // This state is responsible for correct state assignment
      // according to the decoded RS232 protocol.
      //-------------------------------------------------------
      case OnLine_stIdle:
         if (RS232_isRS232ReadyReset() == 0) {
            OnLine_device = RS232_getDevice();
            OnLine_deviceRegister = RS232_getDeviceRegister();
            if (RS232_getDevice() == PROTOCOL_COMPONENT_AD7758) {          //AD7758
               if (RS232_getDataLength() == 1) {
                  OnLine_dataChar = RS232_getChar();
                  OnLine_state = OnLine_stWriteAD7758Char;
               } else if (RS232_getDataLength() == 2) {
                  OnLine_dataInt = RS232_getInt();
                  OnLine_state = OnLine_stWriteAD7758Int;
               } else if (RS232_getDataLength() == 4) {
                  OnLine_dataLong = RS232_getLong();
                  OnLine_state = OnLine_stWriteAD7758Long;
               }
            } else if (RS232_getDevice() == PROTOCOL_COMPONENT_MCU) {      //MCU
               if (RS232_getDataLength() == 1) {
                  OnLine_dataChar = RS232_getChar();
                  switch (OnLine_deviceRegister) {
                     case PROTOCOL_REGISTER_MCU_REGISTERDUMP:
                        switch (OnLine_dataChar) {
                           case PROTOCOL_VALUE_MCU_MAINSCREEN:
                              OnLine_state = OnLine_stMainScreenData;
                              break;
                           case PROTOCOL_VALUE_MCU_MONITOR:
                              OnLine_state = OnLine_stMonitor;
                              break;
                        }
                        break;
                     case PROTOCOL_REGISTER_MCU_CALIBRATION:
                        switch (OnLine_dataChar) {
                           case PROTOCOL_VALUE_MCU_V1_A:
                              OnLine_Cal_lcycmode     = 0x0008;
                              OnLine_Cal_mask         = 0x0200;
                              OnLine_Cal_xXMRSOS      = AVRMSOS;
                              OnLine_Cal_xXRMS        = AVRMS;
                              OnLine_Cal_currentCal   = OnLine_dataChar;
                              OnLine_state = OnLine_stOffCal_InitLCYCMODE;
                              break;
                           case PROTOCOL_VALUE_MCU_V2_A:
```

```
   OnLine_Cal_lcycmode     = 0x0008;
   OnLine_Cal_mask         = 0x0200;
   OnLine_Cal_xXMRSOS      = AVRMSOS;
   OnLine_Cal_xXRMS        = AVRMS;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stOffCal_InitLCYCMODE;
   break;
case PROTOCOL_VALUE_MCU_V1_B:
   OnLine_Cal_lcycmode     = 0x0010;
   OnLine_Cal_mask         = 0x0400;
   OnLine_Cal_xXMRSOS      = BVRMSOS;
   OnLine_Cal_xXRMS        = BVRMS;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stOffCal_InitLCYCMODE;
   break;
case PROTOCOL_VALUE_MCU_V2_B:
   OnLine_Cal_lcycmode     = 0x0010;
   OnLine_Cal_mask         = 0x0400;
   OnLine_Cal_xXMRSOS      = BVRMSOS;
   OnLine_Cal_xXRMS        = BVRMS;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stOffCal_InitLCYCMODE;
   break;
case PROTOCOL_VALUE_MCU_V1_C:
   OnLine_Cal_lcycmode     = 0x00020;
   OnLine_Cal_mask         = 0x0800;
   OnLine_Cal_xXMRSOS      = CVRMSOS;
   OnLine_Cal_xXRMS        = CVRMS;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stOffCal_InitLCYCMODE;
   break;
case PROTOCOL_VALUE_MCU_V2_C:
   OnLine_Cal_lcycmode     = 0x0020;
   OnLine_Cal_mask         = 0x0800;
   OnLine_Cal_xXMRSOS      = CVRMSOS;
   OnLine_Cal_xXRMS        = CVRMS;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stOffCal_InitLCYCMODE;
   break;

case PROTOCOL_VALUE_MCU_I1_A:
   OnLine_Cal_lcycmode     = 0x0008;
   OnLine_Cal_mask         = 0x0200;
   OnLine_Cal_xXMRSOS      = AIRMSOS;
   OnLine_Cal_xXRMS        = AIRMS;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stOffCal_InitLCYCMODE;
   break;
case PROTOCOL_VALUE_MCU_I2_A:
   OnLine_Cal_lcycmode     = 0x0008;
   OnLine_Cal_mask         = 0x0200;
   OnLine_Cal_xXMRSOS      = AIRMSOS;
   OnLine_Cal_xXRMS        = AIRMS;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stOffCal_InitLCYCMODE;
   break;
case PROTOCOL_VALUE_MCU_I1_B:
   OnLine_Cal_lcycmode     = 0x0010;
   OnLine_Cal_mask         = 0x0400;
   OnLine_Cal_xXMRSOS      = BIRMSOS;
   OnLine_Cal_xXRMS        = BIRMS;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stOffCal_InitLCYCMODE;
   break;
case PROTOCOL_VALUE_MCU_I2_B:
   OnLine_Cal_lcycmode     = 0x0010;
   OnLine_Cal_mask         = 0x0400;
   OnLine_Cal_xXMRSOS      = BIRMSOS;
   OnLine_Cal_xXRMS        = BIRMS;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stOffCal_InitLCYCMODE;
   break;
case PROTOCOL_VALUE_MCU_I1_C:
   OnLine_Cal_lcycmode     = 0x00020;
   OnLine_Cal_mask         = 0x0800;
   OnLine_Cal_xXMRSOS      = CIRMSOS;
   OnLine_Cal_xXRMS        = CIRMS;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stOffCal_InitLCYCMODE;
```

```
   break;
case PROTOCOL_VALUE_MCU_I2_C:
   OnLine_Cal_lcycmode     = 0x0020;
   OnLine_Cal_mask         = 0x0800;
   OnLine_Cal_xXMRSOS      = CIRMSOS;
   OnLine_Cal_xXRMS        = CIRMS;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stOffCal_InitLCYCMODE;
   break;
case PROTOCOL_VALUE_MCU_GAIN_A:
   OnLine_Cal_xWG          = AWG;
   OnLine_Cal_xVARG        = AVARG;
   OnLine_Cal_xVAG         = AVAG;
   OnLine_Cal_lcycmode     = 0x0F;
   OnLine_Cal_xWATTHR      = AWATTHR;
   OnLine_Cal_xVARHR       = AVARHR;
   OnLine_Cal_xVAHR        = AVAHR;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stGainCal_InitXWG;
   break;
case PROTOCOL_VALUE_MCU_GAIN_B:
   OnLine_Cal_xWG          = BWG;
   OnLine_Cal_xVARG        = BVARG;
   OnLine_Cal_xVAG         = BVAG;
   OnLine_Cal_lcycmode     = 0x017;
   OnLine_Cal_xWATTHR      = BWATTHR;
   OnLine_Cal_xVARHR       = BVARHR;
   OnLine_Cal_xVAHR        = BVAHR;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stGainCal_InitXWG;
   break;
case PROTOCOL_VALUE_MCU_GAIN_C:
   OnLine_Cal_xWG          = CWG;
   OnLine_Cal_xVARG        = CVARG;
   OnLine_Cal_xVAG         = CVAG;
   OnLine_Cal_lcycmode     = 0x027;
   OnLine_Cal_xWATTHR      = CWATTHR;
   OnLine_Cal_xVARHR       = CVARHR;
   OnLine_Cal_xVAHR        = CVAHR;
   OnLine_Cal_currentCal   = OnLine_dataChar;
   OnLine_state = OnLine_stGainCal_InitXWG;
   break;

case PROTOCOL_VALUE_MCU_FREQ:
   OnLine_state = OnLine_stFreqCal_Init;
   break;

case PROTOCOL_VALUE_MCU_PHASE1A:
   OnLine_Cal_xWG          = AWG;
   OnLine_Cal_lcycmode     = 0x0F;
   OnLine_Cal_xWATTHR      = AWATTHR;
   OnLine_Cal_currentCal   = PROTOCOL_REGISTER_MCU_MEASURE_PHASE_W1A;
   OnLine_state = OnLine_stPhaseCal_InitXWG;
   break;
case PROTOCOL_VALUE_MCU_PHASE1B:
   OnLine_Cal_xWG          = BWG;
   OnLine_Cal_lcycmode     = 0x017;
   OnLine_Cal_xWATTHR      = BWATTHR;
   OnLine_Cal_currentCal   = PROTOCOL_REGISTER_MCU_MEASURE_PHASE_W1B;
   OnLine_state = OnLine_stPhaseCal_InitXWG;
   break;
case PROTOCOL_VALUE_MCU_PHASE1C:
   OnLine_Cal_xWG          = CWG;
   OnLine_Cal_lcycmode     = 0x027;
   OnLine_Cal_xWATTHR      = CWATTHR;
   OnLine_Cal_currentCal   = PROTOCOL_REGISTER_MCU_MEASURE_PHASE_W1C;
   OnLine_state = OnLine_stPhaseCal_InitXWG;
   break;
case PROTOCOL_VALUE_MCU_PHASE2A:
   OnLine_Cal_xWG          = AWG;
   OnLine_Cal_lcycmode     = 0x0F;
   OnLine_Cal_xWATTHR      = AWATTHR;
   OnLine_Cal_currentCal   = PROTOCOL_REGISTER_MCU_MEASURE_PHASE_W2A;
   OnLine_state = OnLine_stPhaseCal_InitXWG;
   break;
case PROTOCOL_VALUE_MCU_PHASE2B:
   OnLine_Cal_xWG          = BWG;
   OnLine_Cal_lcycmode     = 0x017;
   OnLine_Cal_xWATTHR      = BWATTHR;
```

```
                OnLine_Cal_currentCal   = PROTOCOL_REGISTER_MCU_MEASURE_PHASE_W2B;
                OnLine_state = OnLine_stPhaseCal_InitXWG;
                break;
             case PROTOCOL_VALUE_MCU_PHASE2C:
                OnLine_Cal_xWG          = CWG;
                OnLine_Cal_lcycmode     = 0x027;
                OnLine_Cal_xWATTHR      = CWATTHR;
                OnLine_Cal_currentCal   = PROTOCOL_REGISTER_MCU_MEASURE_PHASE_W2C;
                OnLine_state = OnLine_stPhaseCal_InitXWG;
                break;
          }
          break;
       case PROTOCOL_REGISTER_MCU_MEASURE_ENERGY:
          switch (OnLine_dataChar) {
             case PROTOCOL_VALUE_MCU_STOP:
                OnLine_flags &= ~OnLine_flagEnergyMeasurement;
                break;
             case PROTOCOL_VALUE_MCU_START:
                energy_W_a = 0x0; //Clear registers
                energy_W_b = 0x0;
                energy_W_c = 0x0;
                energy_VAR_a = 0x0;
                energy_VAR_b = 0x0;
                energy_VAR_c = 0x0;
                energy_VA_a = 0x0;
                energy_VA_b = 0x0;
                energy_VA_c = 0x0;
                energy_W_A = 0x0;
                energy_W_B = 0x0;
                energy_W_C = 0x0;
                energy_VAR_A = 0x0;
                energy_VAR_B = 0x0;
                energy_VAR_C = 0x0;
                energy_VA_A = 0x0;
                energy_VA_B = 0x0;
                energy_VA_C = 0x0;
                OnLine_state = OnLine_stEnergyM_InitMASK;
                break;
          }
          break;
       case PROTOCOL_REGISTER_MCU_FLASH:
          switch (OnLine_dataChar) {
             case PROTOCOL_VALUE_MCU_FLASH_WRITE:
                OnLine_state = OnLine_stFlash_WriteMem;
                break;
             case PROTOCOL_VALUE_MCU_FLASH_READ:
                OnLine_state = OnLine_stInitStart;
                break;
          }
          break;
    }
 } else if (RS232_getDataLength() == 4) {
    switch (OnLine_deviceRegister) {
       case PROTOCOL_REGISTER_MCU_CALIBRATION_WDIVA:
          MCU_WDIVA = RS232_getLong();
          break;
       case PROTOCOL_REGISTER_MCU_CALIBRATION_WDIVB:
          MCU_WDIVB = RS232_getLong();
          break;
       case PROTOCOL_REGISTER_MCU_CALIBRATION_WDIVC:
          MCU_WDIVC = RS232_getLong();
          break;
       case PROTOCOL_REGISTER_MCU_CALIBRATION_VARDIVA:
          MCU_VARDIVA = RS232_getLong();
          break;
       case PROTOCOL_REGISTER_MCU_CALIBRATION_VARDIVB:
          MCU_VARDIVB = RS232_getLong();
          break;
       case PROTOCOL_REGISTER_MCU_CALIBRATION_VARDIVC:
          MCU_VARDIVC = RS232_getLong();
          break;
       case PROTOCOL_REGISTER_MCU_CALIBRATION_VADIVA:
          MCU_VADIVA = RS232_getLong();
          OnLine_state = OnLine_stMainScreenData;
          break;
       case PROTOCOL_REGISTER_MCU_CALIBRATION_VADIVB:
          MCU_VADIVB = RS232_getLong();
          OnLine_state = OnLine_stMainScreenData;
          break;
```

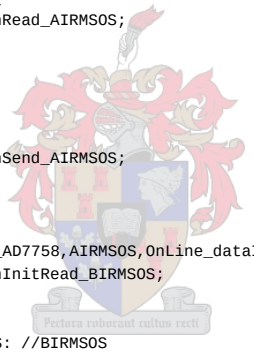
```
                case PROTOCOL_REGISTER_MCU_CALIBRATION_VADIVC:
                   MCU_VADIVC = RS232_getLong();
                   OnLine_state = OnLine_stMainScreenData;
                   break;
             }
          }
       }
    } else if (TIMER_isExpired(tmrWDTOnLine,delayWDTOnLine) == 0) {
       TIMER_startTimer(tmrWDTOnLine);
       LEDs_toggleLED0();
    } else if ((OnLine_flags & OnLine_flagEnergyMeasurement) != 0) {
       OnLine_state = OnLine_stEnergySend_W;
    }
    break;

//AD7758 Direct interface
//-------------------------------------------------------
//Char
case OnLine_stWriteAD7758Char:
    if (AD7758_writeChar(OnLine_deviceRegister,OnLine_dataChar) == 0x0) {
       OnLine_state = OnLine_stReadAD7758Char;
    }
    break;
case OnLine_stReadAD7758Char:
    if (AD7758_readChar(OnLine_deviceRegister) == 0x0) {
       OnLine_state = OnLine_stCheckDataReadyChar;
    }
    break;
case OnLine_stCheckDataReadyChar:
    if (AD7758_isDataReady() == 0x0) {
       OnLine_dataChar = AD7758_getChar();
       OnLine_state = OnLine_stSendUartChar;
    }
    break;
case OnLine_stSendUartChar:
    if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,OnLine_deviceRegister,OnLine_dataChar) == 0x0) {
       OnLine_state = OnLine_stIdle;
    }
    break;
//Int
case OnLine_stWriteAD7758Int:
    if (AD7758_writeInt(OnLine_deviceRegister,OnLine_dataInt) == 0x0) {
       OnLine_state = OnLine_stReadAD7758Int;
    }
    break;
case OnLine_stReadAD7758Int:
    if (AD7758_readInt(OnLine_deviceRegister) == 0x0) {
       OnLine_state = OnLine_stCheckDataReadyInt;
    }
    break;
case OnLine_stCheckDataReadyInt:
    if (AD7758_isDataReady() == 0x0) {
       OnLine_dataInt = AD7758_getInt();
       OnLine_state = OnLine_stSendUartInt;
    }
    break;
case OnLine_stSendUartInt:
    if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,OnLine_deviceRegister,OnLine_dataInt) == 0x0) {
       OnLine_state = OnLine_stIdle;
    }
    break;
//Long
case OnLine_stWriteAD7758Long:
    if (AD7758_writeLong(OnLine_deviceRegister,OnLine_dataLong) == 0x0) {
       OnLine_state = OnLine_stReadAD7758Long;
    }
    break;
case OnLine_stReadAD7758Long:
    if (AD7758_readLong(OnLine_deviceRegister) == 0x0) {
       OnLine_state = OnLine_stCheckDataReadyLong;
    }
    break;
case OnLine_stCheckDataReadyLong:
    if (AD7758_isDataReady() == 0x0) {
       OnLine_dataLong = AD7758_getLong();
       OnLine_state = OnLine_stSendUartLong;
    }
    break;
case OnLine_stSendUartLong:
```



```
   if (RS232_sendLong(PROTOCOL_COMPONENT_AD7758,OnLine_deviceRegister,OnLine_dataLong) == 0x0) {
      OnLine_state = OnLine_stIdle;
   }
   break;


//MCU interface
//------------------------------------------------------
//MainScreenDump : Init
case OnLine_stMainScreenData:
   OnLine_state = OnLine_stMainScreenInitRead_RSTATUS;
   break;
case OnLine_stMainScreenInitRead_RSTATUS: //RSTATUS
   if (AD7758_readChar(RSTATUS) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_RSTATUS;
   }
   break;
case OnLine_stMainScreenRead_RSTATUS:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stMainScreenSend_RSTATUS;
   }
   break;
case OnLine_stMainScreenSend_RSTATUS:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,RSTATUS,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_OPMODE;
   }
   break;
case OnLine_stMainScreenInitRead_OPMODE:  //OPMODE
   if (AD7758_readChar(OPMODE) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_OPMODE;
   }
   break;
case OnLine_stMainScreenRead_OPMODE:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stMainScreenSend_OPMODE;
   }
   break;
case OnLine_stMainScreenSend_OPMODE:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,OPMODE,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_MMODE;
   }
   break;
case OnLine_stMainScreenInitRead_MMODE:   //MMODE
   if (AD7758_readChar(MMODE) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_MMODE;
   }
   break;
case OnLine_stMainScreenRead_MMODE:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stMainScreenSend_MMODE;
   }
   break;
case OnLine_stMainScreenSend_MMODE:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,MMODE,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_WAVMODE;
   }
   break;
case OnLine_stMainScreenInitRead_WAVMODE: //WAVMODE
   if (AD7758_readChar(WAVMODE) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_WAVMODE;
   }
   break;
case OnLine_stMainScreenRead_WAVMODE:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stMainScreenSend_WAVMODE;
   }
   break;
case OnLine_stMainScreenSend_WAVMODE:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,WAVMODE,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_COMPMODE;
   }
   break;
case OnLine_stMainScreenInitRead_COMPMODE:   //COMPMODE
   if (AD7758_readChar(COMPMODE) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_COMPMODE;
```

```
   }
   break;
case OnLine_stMainScreenRead_COMPMODE:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stMainScreenSend_COMPMODE;
   }
   break;
case OnLine_stMainScreenSend_COMPMODE:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,COMPMODE,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_LCYCMODE;
   }
   break;
case OnLine_stMainScreenInitRead_LCYCMODE:   //LCYCMODE
   if (AD7758_readChar(LCYCMODE) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_LCYCMODE;
   }
   break;
case OnLine_stMainScreenRead_LCYCMODE:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stMainScreenSend_LCYCMODE;
   }
   break;
case OnLine_stMainScreenSend_LCYCMODE:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,LCYCMODE,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_MASK;
   }
   break;
case OnLine_stMainScreenInitRead_MASK: //MASK
   if (AD7758_readLong(MASK) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_MASK;
   }
   break;
case OnLine_stMainScreenRead_MASK:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataLong = AD7758_getLong();
      OnLine_state = OnLine_stMainScreenSend_MASK;
   }
   break;
case OnLine_stMainScreenSend_MASK:
   if (RS232_sendLong(PROTOCOL_COMPONENT_AD7758,MASK,OnLine_dataLong) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_LINECYC;
   }
   break;
case OnLine_stMainScreenInitRead_LINECYC: //LINECYC
   if (AD7758_readInt(LINECYC) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_LINECYC;
   }
   break;
case OnLine_stMainScreenRead_LINECYC:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_LINECYC;
   }
   break;
case OnLine_stMainScreenSend_LINECYC:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,LINECYC,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_AVRMSOS;
   }
   break;
case OnLine_stMainScreenInitRead_AVRMSOS: //AVRMSOS
   if (AD7758_readInt(AVRMSOS) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_AVRMSOS;
   }
   break;
case OnLine_stMainScreenRead_AVRMSOS:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_AVRMSOS;
   }
   break;
case OnLine_stMainScreenSend_AVRMSOS:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,AVRMSOS,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_BVRMSOS;
   }
   break;
case OnLine_stMainScreenInitRead_BVRMSOS: //BVRMSOS
   if (AD7758_readInt(BVRMSOS) == 0x0) {
```

```
      OnLine_state = OnLine_stMainScreenRead_BVRMSOS;
   }
   break;
case OnLine_stMainScreenRead_BVRMSOS:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_BVRMSOS;
   }
   break;
case OnLine_stMainScreenSend_BVRMSOS:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,BVRMSOS,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_CVRMSOS;
   }
   break;
case OnLine_stMainScreenInitRead_CVRMSOS: //CVRMSOS
   if (AD7758_readInt(CVRMSOS) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_CVRMSOS;
   }
   break;
case OnLine_stMainScreenRead_CVRMSOS:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_CVRMSOS;
   }
   break;
case OnLine_stMainScreenSend_CVRMSOS:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,CVRMSOS,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_AIRMSOS;
   }
   break;
case OnLine_stMainScreenInitRead_AIRMSOS: //AIRMSOS
   if (AD7758_readInt(AIRMSOS) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_AIRMSOS;
   }
   break;
case OnLine_stMainScreenRead_AIRMSOS:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_AIRMSOS;
   }
   break;
case OnLine_stMainScreenSend_AIRMSOS:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,AIRMSOS,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_BIRMSOS;
   }
   break;
case OnLine_stMainScreenInitRead_BIRMSOS: //BIRMSOS
   if (AD7758_readInt(BIRMSOS) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_BIRMSOS;
   }
   break;
case OnLine_stMainScreenRead_BIRMSOS:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_BIRMSOS;
   }
   break;
case OnLine_stMainScreenSend_BIRMSOS:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,BIRMSOS,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_CIRMSOS;
   }
   break;
case OnLine_stMainScreenInitRead_CIRMSOS: //CIRMSOS
   if (AD7758_readInt(CIRMSOS) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_CIRMSOS;
   }
   break;
case OnLine_stMainScreenRead_CIRMSOS:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_CIRMSOS;
   }
   break;
case OnLine_stMainScreenSend_CIRMSOS:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,CIRMSOS,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_APCFNUM;
   }
   break;
case OnLine_stMainScreenInitRead_APCFNUM: //APCFNUM
```



```
   if (AD7758_readInt(APCFNUM) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_APCFNUM;
   }
   break;
case OnLine_stMainScreenRead_APCFNUM:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_APCFNUM;
   }
   break;
case OnLine_stMainScreenSend_APCFNUM:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,APCFNUM,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_APCFDEN;
   }
   break;
case OnLine_stMainScreenInitRead_APCFDEN: //APCFDEN
   if (AD7758_readInt(APCFDEN) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_APCFDEN;
   }
   break;
case OnLine_stMainScreenRead_APCFDEN:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_APCFDEN;
   }
   break;
case OnLine_stMainScreenSend_APCFDEN:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,APCFDEN,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_VARCFNUM;
   }
   break;
case OnLine_stMainScreenInitRead_VARCFNUM:   //VARCFNUM
   if (AD7758_readInt(VARCFNUM) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_VARCFNUM;
   }
   break;
case OnLine_stMainScreenRead_VARCFNUM:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_VARCFNUM;
   }
   break;
case OnLine_stMainScreenSend_VARCFNUM:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,VARCFNUM,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_VARCFDEN;
   }
   break;
case OnLine_stMainScreenInitRead_VARCFDEN:   //VARCFDEN
   if (AD7758_readInt(VARCFDEN) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_VARCFDEN;
   }
   break;
case OnLine_stMainScreenRead_VARCFDEN:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_VARCFDEN;
   }
   break;
case OnLine_stMainScreenSend_VARCFDEN:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,VARCFDEN,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_AWG;
   }
   break;
case OnLine_stMainScreenInitRead_AWG:  //AWG
   if (AD7758_readInt(AWG) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_AWG;
   }
   break;
case OnLine_stMainScreenRead_AWG:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_AWG;
   }
   break;
case OnLine_stMainScreenSend_AWG:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,AWG,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_BWG;
   }
   break;
```

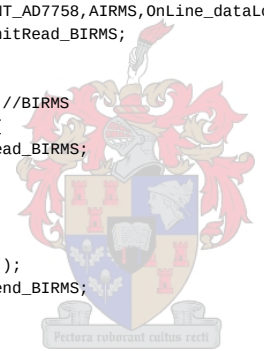
```
case OnLine_stMainScreenInitRead_BWG:  //BWG
   if (AD7758_readInt(BWG) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_BWG;
   }
   break;
case OnLine_stMainScreenRead_BWG:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_BWG;
   }
   break;
case OnLine_stMainScreenSend_BWG:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,BWG,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_CWG;
   }
   break;
case OnLine_stMainScreenInitRead_CWG:  //CWG
   if (AD7758_readInt(CWG) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_CWG;
   }
   break;
case OnLine_stMainScreenRead_CWG:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_CWG;
   }
   break;
case OnLine_stMainScreenSend_CWG:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,CWG,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_AVAG;
   }
   break;
case OnLine_stMainScreenInitRead_AVAG: //AVAG
   if (AD7758_readInt(AVAG) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_AVAG;
   }
   break;
case OnLine_stMainScreenRead_AVAG:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_AVAG;
   }
   break;
case OnLine_stMainScreenSend_AVAG:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,AVAG,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_BVAG;
   }
   break;
case OnLine_stMainScreenInitRead_BVAG: //BVAG
   if (AD7758_readInt(BVAG) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_BVAG;
   }
   break;
case OnLine_stMainScreenRead_BVAG:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_BVAG;
   }
   break;
case OnLine_stMainScreenSend_BVAG:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,BVAG,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_CVAG;
   }
   break;
case OnLine_stMainScreenInitRead_CVAG: //CVAG
   if (AD7758_readInt(CVAG) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_CVAG;
   }
   break;
case OnLine_stMainScreenRead_CVAG:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_CVAG;
   }
   break;
case OnLine_stMainScreenSend_CVAG:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,CVAG,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_AVARG;
   }
```

```
   break;
case OnLine_stMainScreenInitRead_AVARG:   //AVARG
   if (AD7758_readInt(AVARG) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_AVARG;
   }
   break;
case OnLine_stMainScreenRead_AVARG:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_AVARG;
   }
   break;
case OnLine_stMainScreenSend_AVARG:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,AVARG,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_BVARG;
   }
   break;
case OnLine_stMainScreenInitRead_BVARG:   //BVARG
   if (AD7758_readInt(BVARG) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_BVARG;
   }
   break;
case OnLine_stMainScreenRead_BVARG:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_BVARG;
   }
   break;
case OnLine_stMainScreenSend_BVARG:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,BVARG,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_CVARG;
   }
   break;
case OnLine_stMainScreenInitRead_CVARG:   //CVARG
   if (AD7758_readInt(CVARG) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_CVARG;
   }
   break;
case OnLine_stMainScreenRead_CVARG:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMainScreenSend_CVARG;
   }
   break;
case OnLine_stMainScreenSend_CVARG:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,CVARG,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_WDIV;
   }
   break;
case OnLine_stMainScreenInitRead_WDIV: //WDIV
   if (AD7758_readChar(WDIV) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_WDIV;
   }
   break;
case OnLine_stMainScreenRead_WDIV:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stMainScreenSend_WDIV;
   }
   break;
case OnLine_stMainScreenSend_WDIV:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,WDIV,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_VADIV;
   }
   break;
case OnLine_stMainScreenInitRead_VADIV:   //VADIV
   if (AD7758_readChar(VADIV) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_VADIV;
   }
   break;
case OnLine_stMainScreenRead_VADIV:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stMainScreenSend_VADIV;
   }
   break;
case OnLine_stMainScreenSend_VADIV:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,VADIV,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_VARDIV;
```

```
   }
   break;
case OnLine_stMainScreenInitRead_VARDIV:  //VARDIV
   if (AD7758_readChar(VARDIV) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_VARDIV;
   }
   break;
case OnLine_stMainScreenRead_VARDIV:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stMainScreenSend_VARDIV;
   }
   break;
case OnLine_stMainScreenSend_VARDIV:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,VARDIV,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stMainScreenSend_MCU_WDIVA;
   }
   break;
case OnLine_stMainScreenSend_MCU_WDIVA:   //MCU_WDIVA
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_CALIBRATION_WDIVA,MCU_WDIVA) == 0x0) {
      OnLine_state = OnLine_stMainScreenSend_MCU_WDIVB;
   }
   break;
case OnLine_stMainScreenSend_MCU_WDIVB:   //MCU_WDIVB
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_CALIBRATION_WDIVB,MCU_WDIVB) == 0x0) {
      OnLine_state = OnLine_stMainScreenSend_MCU_WDIVC;
   }
   break;
case OnLine_stMainScreenSend_MCU_WDIVC:   //MCU_WDIVC
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_CALIBRATION_WDIVC,MCU_WDIVC) == 0x0) {
      OnLine_state = OnLine_stMainScreenSend_MCU_VARDIVA;
   }
   break;
case OnLine_stMainScreenSend_MCU_VARDIVA: //MCU_VARDIVA
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_CALIBRATION_VARDIVA,MCU_VARDIVA) == 0x0) {
      OnLine_state = OnLine_stMainScreenSend_MCU_VARDIVB;
   }
   break;
case OnLine_stMainScreenSend_MCU_VARDIVB: //MCU_VARDIVB
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_CALIBRATION_VARDIVB,MCU_VARDIVB) == 0x0) {
      OnLine_state = OnLine_stMainScreenSend_MCU_VARDIVC;
   }
   break;
case OnLine_stMainScreenSend_MCU_VARDIVC: //MCU_VARDIVC
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_CALIBRATION_VARDIVC,MCU_VARDIVC) == 0x0) {
      OnLine_state = OnLine_stMainScreenSend_MCU_VADIVA;
   }
   break;
case OnLine_stMainScreenSend_MCU_VADIVA:  //MCU_VADIVA
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_CALIBRATION_VADIVA,MCU_VADIVA) == 0x0) {
      OnLine_state = OnLine_stMainScreenSend_MCU_VADIVB;
   }
   break;
case OnLine_stMainScreenSend_MCU_VADIVB:  //MCU_VADIVB
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_CALIBRATION_VADIVB,MCU_VADIVB) == 0x0) {
      OnLine_state = OnLine_stMainScreenSend_MCU_VADIVC;
   }
   break;
case OnLine_stMainScreenSend_MCU_VADIVC:  //MCU_VADIVC
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_CALIBRATION_VADIVC,MCU_VADIVC) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_APHCAL;
   }
   break;
case OnLine_stMainScreenInitRead_APHCAL:  //APHCAL
   if (AD7758_readChar(APHCAL) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_APHCAL;
   }
   break;
case OnLine_stMainScreenRead_APHCAL:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stMainScreenSend_APHCAL;
   }
   break;
case OnLine_stMainScreenSend_APHCAL:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,APHCAL,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_BPHCAL;
   }
   break;
```

```
case OnLine_stMainScreenInitRead_BPHCAL:  //BPHCAL
   if (AD7758_readChar(BPHCAL) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_BPHCAL;
   }
   break;
case OnLine_stMainScreenRead_BPHCAL:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stMainScreenSend_BPHCAL;
   }
   break;
case OnLine_stMainScreenSend_BPHCAL:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,BPHCAL,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stMainScreenInitRead_CPHCAL;
   }
   break;
case OnLine_stMainScreenInitRead_CPHCAL:  //CPHCAL
   if (AD7758_readChar(CPHCAL) == 0x0) {
      OnLine_state = OnLine_stMainScreenRead_CPHCAL;
   }
   break;
case OnLine_stMainScreenRead_CPHCAL:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stMainScreenSend_CPHCAL;
   }
   break;
case OnLine_stMainScreenSend_CPHCAL:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,CPHCAL,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stMainScreenDataEnd;
   }
   break;
case OnLine_stMainScreenDataEnd:       //End - Handover to Monitor sequence
   OnLine_state = OnLine_stIdle;//OnLine_stMonitorInitRead_AVRMS;
   break;

//Monitor : Init
case OnLine_stMonitor:
   OnLine_state = OnLine_stMonitorInit_LCYCMODE;
   break;
case OnLine_stMonitorInit_LCYCMODE: //Set Line cycle mode for detection on phase A
   if (AD7758_writeChar(LCYCMODE,0x038) == 0x0) {
      OnLine_state = OnLine_stMonitorInit_MASK;
   }
   break;
case OnLine_stMonitorInit_MASK:     //Set Interupt mask for zero crossing on phase A
   if (AD7758_writeLong(MASK,0x00) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_AVRMS;
   }
   break;
case OnLine_stMonitorInitRead_AVRMS:   //AVRMS
   if (AD7758_readLong(AVRMS) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_AVRMS;
   }
   break;
case OnLine_stMonitorRead_AVRMS:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataLong = AD7758_getLong();
      OnLine_state = OnLine_stMonitorSend_AVRMS;
   }
   break;
case OnLine_stMonitorSend_AVRMS:
   if (RS232_sendLong(PROTOCOL_COMPONENT_AD7758,AVRMS,OnLine_dataLong) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_BVRMS;
   }
   break;
case OnLine_stMonitorInitRead_BVRMS:   //BVRMS
   if (AD7758_readLong(BVRMS) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_BVRMS;
   }
   break;
case OnLine_stMonitorRead_BVRMS:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataLong = AD7758_getLong();
      OnLine_state = OnLine_stMonitorSend_BVRMS;
   }
   break;
case OnLine_stMonitorSend_BVRMS:
   if (RS232_sendLong(PROTOCOL_COMPONENT_AD7758,BVRMS,OnLine_dataLong) == 0x0) {
```

```
      OnLine_state = OnLine_stMonitorInitRead_CVRMS;
   }
   break;
case OnLine_stMonitorInitRead_CVRMS:   //CVRMS
   if (AD7758_readLong(CVRMS) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_CVRMS;
   }
   break;
case OnLine_stMonitorRead_CVRMS:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataLong = AD7758_getLong();
      OnLine_state = OnLine_stMonitorSend_CVRMS;
   }
   break;
case OnLine_stMonitorSend_CVRMS:
   if (RS232_sendLong(PROTOCOL_COMPONENT_AD7758,CVRMS,OnLine_dataLong) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_AIRMS;
   }
   break;
case OnLine_stMonitorInitRead_AIRMS:   //AIRMS
   if (AD7758_readLong(AIRMS) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_AIRMS;
   }
   break;
case OnLine_stMonitorRead_AIRMS:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataLong = AD7758_getLong();
      OnLine_state = OnLine_stMonitorSend_AIRMS;
   }
   break;
case OnLine_stMonitorSend_AIRMS:
   if (RS232_sendLong(PROTOCOL_COMPONENT_AD7758,AIRMS,OnLine_dataLong) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_BIRMS;
   }
   break;
case OnLine_stMonitorInitRead_BIRMS:   //BIRMS
   if (AD7758_readLong(BIRMS) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_BIRMS;
   }
   break;
case OnLine_stMonitorRead_BIRMS:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataLong = AD7758_getLong();
      OnLine_state = OnLine_stMonitorSend_BIRMS;
   }
   break;
case OnLine_stMonitorSend_BIRMS:
   if (RS232_sendLong(PROTOCOL_COMPONENT_AD7758,BIRMS,OnLine_dataLong) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_CIRMS;
   }
   break;
case OnLine_stMonitorInitRead_CIRMS:   //CIRMS
   if (AD7758_readLong(CIRMS) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_CIRMS;
   }
   break;
case OnLine_stMonitorRead_CIRMS:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataLong = AD7758_getLong();
      OnLine_state = OnLine_stMonitorSend_CIRMS;
   }
   break;
case OnLine_stMonitorSend_CIRMS:
   if (RS232_sendLong(PROTOCOL_COMPONENT_AD7758,CIRMS,OnLine_dataLong) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_AWATTHR;
   }
   break;
case OnLine_stMonitorInitRead_AWATTHR: //AWATTHR
   if (AD7758_readInt(AWATTHR) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_AWATTHR;
   }
   break;
case OnLine_stMonitorRead_AWATTHR:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMonitorSend_AWATTHR;
   }
   break;
case OnLine_stMonitorSend_AWATTHR:
```



```
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,AWATTHR,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_BWATTHR;
   }
   break;
case OnLine_stMonitorInitRead_BWATTHR: //BWATTHR
   if (AD7758_readInt(BWATTHR) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_BWATTHR;
   }
   break;
case OnLine_stMonitorRead_BWATTHR:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMonitorSend_BWATTHR;
   }
   break;
case OnLine_stMonitorSend_BWATTHR:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,BWATTHR,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_CWATTHR;
   }
   break;
case OnLine_stMonitorInitRead_CWATTHR: //CWATTHR
   if (AD7758_readInt(CWATTHR) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_CWATTHR;
   }
   break;
case OnLine_stMonitorRead_CWATTHR:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMonitorSend_CWATTHR;
   }
   break;
case OnLine_stMonitorSend_CWATTHR:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,CWATTHR,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_AVARHR;
   }
   break;
case OnLine_stMonitorInitRead_AVARHR:  //AVARHR
   if (AD7758_readInt(AVARHR) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_AVARHR;
   }
   break;
case OnLine_stMonitorRead_AVARHR:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMonitorSend_AVARHR;
   }
   break;
case OnLine_stMonitorSend_AVARHR:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,AVARHR,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_BVARHR;
   }
   break;
case OnLine_stMonitorInitRead_BVARHR:  //BVARHR
   if (AD7758_readInt(BVARHR) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_BVARHR;
   }
   break;
case OnLine_stMonitorRead_BVARHR:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMonitorSend_BVARHR;
   }
   break;
case OnLine_stMonitorSend_BVARHR:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,BVARHR,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_CVARHR;
   }
   break;
case OnLine_stMonitorInitRead_CVARHR:  //CVARHR
   if (AD7758_readInt(CVARHR) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_CVARHR;
   }
   break;
case OnLine_stMonitorRead_CVARHR:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMonitorSend_CVARHR;
   }
   break;
```

```
case OnLine_stMonitorSend_CVARHR:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,CVARHR,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_AVAHR;
   }
   break;
case OnLine_stMonitorInitRead_AVAHR:   //AVAHR
   if (AD7758_readInt(AVAHR) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_AVAHR;
   }
   break;
case OnLine_stMonitorRead_AVAHR:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMonitorSend_AVAHR;
   }
   break;
case OnLine_stMonitorSend_AVAHR:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,AVAHR,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_BVAHR;
   }
   break;
case OnLine_stMonitorInitRead_BVAHR:   //BVARHR
   if (AD7758_readInt(BVAHR) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_BVAHR;
   }
   break;
case OnLine_stMonitorRead_BVAHR:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMonitorSend_BVAHR;
   }
   break;
case OnLine_stMonitorSend_BVAHR:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,BVAHR,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_CVAHR;
   }
   break;
case OnLine_stMonitorInitRead_CVAHR:   //CVARHR
   if (AD7758_readInt(CVAHR) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_CVAHR;
   }
   break;
case OnLine_stMonitorRead_CVAHR:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMonitorSend_CVAHR;
   }
   break;
case OnLine_stMonitorSend_CVAHR:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,CVAHR,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_TEMP;
   }
   break;
case OnLine_stMonitorInitRead_TEMP: //TEMP
   if (AD7758_readChar(TEMP) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_TEMP;
   }
   break;
case OnLine_stMonitorRead_TEMP:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stMonitorSend_TEMP;
   }
   break;
case OnLine_stMonitorSend_TEMP:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,FREQ,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stMonitorInitRead_FREQ;
   }
   break;
case OnLine_stMonitorInitRead_FREQ: //FREQ
   if (AD7758_readInt(FREQ) == 0x0) {
      OnLine_state = OnLine_stMonitorRead_FREQ;
   }
   break;
case OnLine_stMonitorRead_FREQ:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stMonitorSend_FREQ;
   }
```

```
      break;
   case OnLine_stMonitorSend_FREQ:
      if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,FREQ,OnLine_dataInt) == 0x0) {
         OnLine_state = OnLine_stMonitorSendAck;
      }
      break;
   case OnLine_stMonitorSendAck: //Request another monitor packet from client to close the loop
      if (RS232_sendChar(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_ACKNOWLEDGE,0x01) == 0x0) {
         OnLine_state = OnLine_stMonitorSendMonitor;
      }
      break;
   case OnLine_stMonitorSendMonitor:   //Request another monitor packet from client to close the loop
      if (RS232_sendChar(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_REGISTERDUMP,PROTOCOL_VALUE_MCU_MONITOR) == 0x0) {
         OnLine_state = OnLine_stIdle;
      }
      break;
   //Callibration
   case OnLine_stOffCal_InitLCYCMODE:
      //Prior to calling this state, the following variables must be set up correctly:
      //    OnLine_Cal_lcycmode  - required value of AD7768 register
      //    OnLine_Cal_mask      - required value of AD7768 register
      //    OnLine_Cal_xXMRSOS   - pointer to registers in AD7758
      //    OnLine_Cal_xXRMS     - pointer to registers in AD7758
      //    OnLine_Cal_currentCal- remembers the current callibration procedure
      // This callibration procedure is done according to the AD7758 datasheet.
      if (AD7758_writeChar(LCYCMODE,OnLine_Cal_lcycmode) == 0x0) {
         OnLine_state = OnLine_stOffCal_InitMASK;
      }
      break;
   case OnLine_stOffCal_InitMASK:
      if (AD7758_writeLong(MASK,OnLine_Cal_mask) == 0x0) {
         OnLine_state = OnLine_stOffCal_InitXRMSOS;
      }
      break;
   case OnLine_stOffCal_InitXRMSOS:
      if (AD7758_writeLong(OnLine_Cal_xXMRSOS,0x0) == 0x0) {   //set offset to zero
         OnLine_state = OnLine_stOffCal_InitRead_LCYCMODE;
      }
      break;
   case OnLine_stOffCal_InitRead_LCYCMODE:   //LCYCMODE
      if (AD7758_readChar(LCYCMODE) == 0x0) {
         OnLine_state = OnLine_stOffCal_Read_LCYCMODE;
      }
      break;
   case OnLine_stOffCal_Read_LCYCMODE:
      if (AD7758_isDataReady() == 0x0) {
         OnLine_dataChar = AD7758_getChar();
         OnLine_state = OnLine_stOffCal_Send_LCYCMODE;
      }
      break;
   case OnLine_stOffCal_Send_LCYCMODE:
      if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,LCYCMODE,OnLine_dataChar) == 0x0) {
         OnLine_state = OnLine_stOffCal_InitRead_MASK;
      }
      break;
   case OnLine_stOffCal_InitRead_MASK: //MASK
      if (AD7758_readLong(MASK) == 0x0) {
         OnLine_state = OnLine_stOffCal_Read_MASK;
      }
      break;
   case OnLine_stOffCal_Read_MASK:
      if (AD7758_isDataReady() == 0x0) {
         OnLine_dataLong = AD7758_getLong();
         OnLine_state = OnLine_stOffCal_Send_MASK;
      }
      break;
   case OnLine_stOffCal_Send_MASK:
      if (RS232_sendLong(PROTOCOL_COMPONENT_AD7758,MASK,OnLine_dataLong) == 0x0) {
         OnLine_state = OnLine_stOffCal_Send_Ack;
      }
      break;
   case OnLine_stOffCal_Send_Ack:   //Send Acknowledge
      if (RS232_sendChar(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_ACKNOWLEDGE,0x01) == 0x0) {
         OnLine_Cal_n = 0x0;
         OnLine_Cal_accumulator = 0x0;
         OnLine_state = OnLine_stOffCal_ResetInt;
      }
      break;
   case OnLine_stOffCal_ResetInt:
```

```
   if (AD7758_readLong(RSTATUS) == 0x0) {
      OnLine_state = OnLine_stOffCal_ResetIntWait;
   }
   break;
case OnLine_stOffCal_ResetIntWait:
   if (AD7758_isDataReady() == 0x0) {
      if ((AD7758_getLong() & OnLine_Cal_mask) != 0) {
         OnLine_state = OnLine_stOffCal_ReadRegister;
      } else {
         OnLine_state = OnLine_stOffCal_ResetInt;
      }
   }
   break;
case OnLine_stOffCal_ReadRegister:
   if (AD7758_readLong(OnLine_Cal_xXRMS) == 0x0) {
      OnLine_state = OnLine_stOffCal_AccumulateRegister;
   }
   break;
case OnLine_stOffCal_AccumulateRegister:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_Cal_n++;
      OnLine_Cal_accumulator = AD7758_getLong() + OnLine_Cal_accumulator;
      if (OnLine_Cal_n > OnLine_Cal_AD7758_N) {
         OnLine_state = OnLine_stOffCal_SendRegisterToRS232;
      } else {
         OnLine_state = OnLine_stOffCal_ResetInt;
      }
   }
   break;
case OnLine_stOffCal_SendRegisterToRS232:
   OnLine_Cal_accumulator = OnLine_Cal_accumulator/OnLine_Cal_n;
   if (RS232_sendLong(PROTOCOL_COMPONENT_AD7758,OnLine_Cal_xXRMS,OnLine_Cal_accumulator) == 0x0) {
      OnLine_state = OnLine_stOffCal_SendCurrentCalToRS232;
   }
   break;
case OnLine_stOffCal_SendCurrentCalToRS232:
   if (RS232_sendChar(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_CURRENTCAL,OnLine_Cal_currentCal) == 0x0) {
      OnLine_state = OnLine_stOffCal_SendAccumulatorCalToRS232;
   }
   break;
case OnLine_stOffCal_SendAccumulatorCalToRS232:
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_CALIBRATION,OnLine_Cal_accumulator) == 0x0) {
      OnLine_state = OnLine_stOffCal_SendFinal_Ack;
   }
   break;
case OnLine_stOffCal_SendFinal_Ack: //Send Acknowledge
   if (RS232_sendChar(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_ACKNOWLEDGE,0x01) == 0x0) {
      OnLine_state = OnLine_stIdle;
   }
   break;

case OnLine_stGainCal_InitXWG:
   //Prior to calling this state, the following variables must be set up correctly:
   //    OnLine_Cal_lcycmode  - required value of AD7768 register
   //    OnLine_Cal_xWG       - used registers...
   //    OnLine_Cal_xVARG
   //    OnLine_Cal_xVAG
   //    OnLine_Cal_xWATTHR
   //    OnLine_Cal_xVARHR
   //    OnLine_Cal_xVAHR
   //    OnLine_Cal_currentCal- remembers the current callibration procedure
   // This callibration procedure is done according to the AD7758 datasheet.
   if (AD7758_writeInt(OnLine_Cal_xWG,0x00) == 0x0) {
      OnLine_state = OnLine_stGainCal_InitXVARG;
   }
   break;
case OnLine_stGainCal_InitXVARG:
   if (AD7758_writeInt(OnLine_Cal_xVARG,0x00) == 0x0) {
      OnLine_state = OnLine_stGainCal_InitXVAG;
   }
   break;
case OnLine_stGainCal_InitXVAG:
   if (AD7758_writeInt(OnLine_Cal_xVAG,0x00) == 0x0) {
      OnLine_state = OnLine_stGainCal_InitLCYCMODE;
   }
   break;
case OnLine_stGainCal_InitLCYCMODE:    //Set LCYCMODE
   if (AD7758_writeChar(LCYCMODE,OnLine_Cal_lcycmode) == 0x0) {
      OnLine_state = OnLine_stGainCal_InitRead_LCYCMODE;
```

```
   }
   break;
case OnLine_stGainCal_InitRead_LCYCMODE:  //LCYCMODE
   if (AD7758_readChar(LCYCMODE) == 0x0) {
      OnLine_state = OnLine_stGainCal_Read_LCYCMODE;
   }
   break;
case OnLine_stGainCal_Read_LCYCMODE:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stGainCal_Send_LCYCMODE;
   }
   break;
case OnLine_stGainCal_Send_LCYCMODE:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,LCYCMODE,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stGainCal_InitRead_LINECYC;
   }
   break;
case OnLine_stGainCal_InitRead_LINECYC:   //LINECYC
   if (AD7758_readInt(LINECYC) == 0x0) {
      OnLine_state = OnLine_stGainCal_Read_LINECYC;
   }
   break;
case OnLine_stGainCal_Read_LINECYC:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stGainCal_Send_LINECYC;
   }
   break;
case OnLine_stGainCal_Send_LINECYC:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,LINECYC,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stGainCal_InitMASK;
   }
   break;
case OnLine_stGainCal_InitMASK:        //Set MASK
   if (AD7758_writeLong(MASK,0x01000) == 0x0) {
      OnLine_state = OnLine_stGainCal_InitRead_MASK;
   }
   break;
case OnLine_stGainCal_InitRead_MASK:   //MASK
   if (AD7758_readLong(MASK) == 0x0) {
      OnLine_state = OnLine_stGainCal_Read_MASK;
   }
   break;
case OnLine_stGainCal_Read_MASK:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataLong = AD7758_getLong();
      OnLine_state = OnLine_stGainCal_Send_MASK;
   }
   break;
case OnLine_stGainCal_Send_MASK:
   if (RS232_sendLong(PROTOCOL_COMPONENT_AD7758,MASK,OnLine_dataLong) == 0x0) {
      OnLine_state = OnLine_stGainCal_InitRead_FREQ;
   }
   break;
case OnLine_stGainCal_InitRead_FREQ:   //FREQ
   if (AD7758_readInt(FREQ) == 0x0) {
      OnLine_state = OnLine_stGainCal_Read_FREQ;
   }
   break;
case OnLine_stGainCal_Read_FREQ:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stGainCal_Send_FREQ;
   }
   break;
case OnLine_stGainCal_Send_FREQ:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,FREQ,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stGainCal_FirstResetInt;
   }
   break;
case OnLine_stGainCal_FirstResetInt:      //Reset Sequence
   if (AD7758_readLong(RSTATUS) == 0x0) {
      OnLine_state = OnLine_stGainCal_ResetInt;
   }
   break;
case OnLine_stGainCal_ResetInt:
   if (AD7758_readLong(RSTATUS) == 0x0) {
      OnLine_state = OnLine_stGainCal_ResetIntWait;
```

```
   }
   break;
case OnLine_stGainCal_ResetIntWait:
   if (AD7758_isDataReady() == 0x0) {
      if ((AD7758_getLong() & 0x01000) == 0) {
         OnLine_state = OnLine_stGainCal_ResetInt;
      } else {
         OnLine_state = OnLine_stGainCal_InitRead_xWATTHR;
         if ((AD7758_getLong() & 0x040) != 0) {
            OnLine_flags |= OnLine_flagGainCalOverflow;
         } else {
            OnLine_flags &= ~OnLine_flagGainCalOverflow;
         }
      }
   }
   break;
case OnLine_stGainCal_InitRead_xWATTHR:   //xWATTHR
   if (AD7758_readInt(OnLine_Cal_xWATTHR) == 0x0) {
      OnLine_state = OnLine_stGainCal_Read_xWATTHR;
   }
   break;
case OnLine_stGainCal_Read_xWATTHR:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stGainCal_Send_xWATTHR;
   }
   break;
case OnLine_stGainCal_Send_xWATTHR:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,OnLine_Cal_xWATTHR,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stGainCal_InitRead_xVARHR;
   }
   break;
case OnLine_stGainCal_InitRead_xVARHR: //xVARHR
   if (AD7758_readInt(OnLine_Cal_xVARHR) == 0x0) {
      OnLine_state = OnLine_stGainCal_Read_xVARHR;
   }
   break;
case OnLine_stGainCal_Read_xVARHR:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stGainCal_Send_xVARHR;
   }
   break;
case OnLine_stGainCal_Send_xVARHR:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,OnLine_Cal_xVARHR,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stGainCal_InitRead_xVAHR;
   }
   break;
case OnLine_stGainCal_InitRead_xVAHR:  //xVAHR
   if (AD7758_readInt(OnLine_Cal_xWATTHR) == 0x0) {
      OnLine_state = OnLine_stGainCal_Read_xVAHR;
   }
   break;
case OnLine_stGainCal_Read_xVAHR:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stGainCal_Send_xVAHR;
   }
   break;
case OnLine_stGainCal_Send_xVAHR:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,OnLine_Cal_xVAHR,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stGainCal_InitRead_WDIV;
   }
   break;
case OnLine_stGainCal_InitRead_WDIV:   //WDIV
   if (AD7758_readChar(WDIV) == 0x0) {
      OnLine_state = OnLine_stGainCal_Read_WDIV;
   }
   break;
case OnLine_stGainCal_Read_WDIV:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stGainCal_Send_WDIV;
   }
   break;
case OnLine_stGainCal_Send_WDIV:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,WDIV,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stGainCal_InitRead_VARDIV;
   }
```

```
   break;
case OnLine_stGainCal_InitRead_VARDIV: //VARDIV
   if (AD7758_readChar(WDIV) == 0x0) {
      OnLine_state = OnLine_stGainCal_Read_VARDIV;
   }
   break;
case OnLine_stGainCal_Read_VARDIV:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stGainCal_Send_VARDIV;
   }
   break;
case OnLine_stGainCal_Send_VARDIV:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,VARDIV,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stGainCal_InitRead_VADIV;
   }
   break;
case OnLine_stGainCal_InitRead_VADIV:  //VADIV
   if (AD7758_readChar(VADIV) == 0x0) {
      OnLine_state = OnLine_stGainCal_Read_VADIV;
   }
   break;
case OnLine_stGainCal_Read_VADIV:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataChar = AD7758_getChar();
      OnLine_state = OnLine_stGainCal_Send_VADIV;
   }
   break;
case OnLine_stGainCal_Send_VADIV:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,WDIV,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stGainCal_Send_CurrentCal;
   }
   break;
case OnLine_stGainCal_Send_CurrentCal:
   if ((OnLine_flags & OnLine_flagGainCalOverflow) == 0) {
      if (RS232_sendChar(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_CURRENTCAL,OnLine_Cal_currentCal) == 0x0) {
         OnLine_state = OnLine_stGainCal_SendFinal_Ack;
      }
   } else {
      if (RS232_sendChar(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_MESSAGE,PROTOCOL_VALUE_MCU_OVERFLOW) == 0x0) {
         OnLine_state = OnLine_stGainCal_SendFinal_Ack;
      }
   }
   break;
case OnLine_stGainCal_SendFinal_Ack:   //Send Acknowledge
   if (RS232_sendChar(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_ACKNOWLEDGE,0x01) == 0x0) {
      OnLine_state = OnLine_stIdle;
   }
   break;

//EnergyMeasurements
   //Sets all the register for correct energy measurement opperation
case OnLine_stEnergyM_InitMASK:
   if (AD7758_writeLong(MASK,0x07) == 0x0) {          //Enable interupts when the line accumulator is halfway
      OnLine_state = OnLine_stEnergyM_InitLCYCMODE;
   }
   break;
case OnLine_stEnergyM_InitLCYCMODE:
   if (AD7758_writeChar(LCYCMODE,0x078) == 0x0) {     //Enable correct setup for  line accumulation
      OnLine_state = OnLine_stEnergyM_InitLINECYC;
   }
   break;
case OnLine_stEnergyM_InitLINECYC:
   if (AD7758_writeInt(LINECYC,0x01FF) == 0x0) {      //Halfcycles for line accumulation read
      OnLine_state = OnLine_stEnergyM_InitCOMPMODE;
   }
   break;
case OnLine_stEnergyM_InitCOMPMODE:
   if (AD7758_writeChar(COMPMODE,0x09C) == 0x0) {     //Enables no load detection on each phase as well as
                                                      //correct computation of the energy registers values
      OnLine_state = OnLine_stEnergyM_ReadAWATTHR;
   }
   break;
case OnLine_stEnergyM_ReadAWATTHR:                    //Read all register to zero readings
   if (AD7758_readInt(AWATTHR) == 0x0) {
      OnLine_state = OnLine_stEnergyM_ReadBWATTHR;
   }
   break;
case OnLine_stEnergyM_ReadBWATTHR:
```

```
   if (AD7758_readInt(AWATTHR) == 0x0) {
      OnLine_state = OnLine_stEnergyM_ReadCWATTHR;
   }
   break;
case OnLine_stEnergyM_ReadCWATTHR:
   if (AD7758_readInt(AWATTHR) == 0x0) {
      OnLine_state = OnLine_stEnergyM_ReadAVARHR;
   }
   break;
case OnLine_stEnergyM_ReadAVARHR:
   if (AD7758_readInt(AWATTHR) == 0x0) {
      OnLine_state = OnLine_stEnergyM_ReadBVARHR;
   }
   break;
case OnLine_stEnergyM_ReadBVARHR:
   if (AD7758_readInt(AWATTHR) == 0x0) {
      OnLine_state = OnLine_stEnergyM_ReadCVARHR;
   }
   break;
case OnLine_stEnergyM_ReadCVARHR:
   if (AD7758_readInt(AWATTHR) == 0x0) {
      OnLine_state = OnLine_stEnergyM_ReadAVAHR;
   }
   break;
case OnLine_stEnergyM_ReadAVAHR:
   if (AD7758_readInt(AWATTHR) == 0x0) {
      OnLine_state = OnLine_stEnergyM_ReadBVAHR;
   }
   break;
case OnLine_stEnergyM_ReadBVAHR:
   if (AD7758_readInt(AWATTHR) == 0x0) {
      OnLine_state = OnLine_stEnergyM_ReadCVAHR;
   }
   break;
case OnLine_stEnergyM_ReadCVAHR:
   if (AD7758_readInt(AWATTHR) == 0x0) {
      OnLine_state = OnLine_stEnergyM_ReadRSTATUS;
   }
   break;
case OnLine_stEnergyM_ReadRSTATUS:
   if (AD7758_readLong(RSTATUS) == 0x0) {
      OnLine_flags |= OnLine_flagEnergyMeasurement;
      OnLine_state = OnLine_stMainScreenData;
   }
   break;

//Send Energy Measurements to PC
case OnLine_stEnergySend_W:   //Send Acknowledge
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_W,energy_W_A+energy_W_B+energy_W_C) == 0x0) {
      OnLine_state = OnLine_stEnergySend_VAR;
   }
   break;
case OnLine_stEnergySend_VAR: //Send Acknowledge
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VAR,energy_VAR_A+energy_VAR_B+energy_VAR_C) == 0x0) {
      OnLine_state = OnLine_stEnergySend_VA;
   }
   break;
case OnLine_stEnergySend_VA:  //Send Acknowledge
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VA,energy_VA_A+energy_VA_B+energy_VA_C) == 0x0) {
      OnLine_state = OnLine_stEnergySend_W_A;
   }
   break;
case OnLine_stEnergySend_W_A: //Send Acknowledge
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_W_A,energy_W_A) == 0x0) {
      OnLine_state = OnLine_stEnergySend_VAR_A;
   }
   break;
case OnLine_stEnergySend_VAR_A:  //Send Acknowledge
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VAR_A,energy_VAR_A) == 0x0) {
      OnLine_state = OnLine_stEnergySend_VA_A;
   }
   break;
case OnLine_stEnergySend_VA_A:   //Send Acknowledge
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VA_A,energy_VA_A) == 0x0) {
      OnLine_state = OnLine_stEnergySend_W_B;
   }
   break;
case OnLine_stEnergySend_W_B: //Send Acknowledge
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_W_B,energy_W_B) == 0x0) {
```

```
      OnLine_state = OnLine_stEnergySend_VAR_B;
   }
   break;
case OnLine_stEnergySend_VAR_B:  //Send Acknowledge
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VAR_B,energy_VAR_B) == 0x0) {
      OnLine_state = OnLine_stEnergySend_VA_B;
   }
   break;
case OnLine_stEnergySend_VA_B:   //Send Acknowledge
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VA_B,energy_VA_B) == 0x0) {
      OnLine_state = OnLine_stEnergySend_W_C;
   }
   break;
case OnLine_stEnergySend_W_C: //Send Acknowledge
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_W_C,energy_W_C) == 0x0) {
      OnLine_state = OnLine_stEnergySend_VAR_C;
   }
   break;
case OnLine_stEnergySend_VAR_C:  //Send Acknowledge
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VAR_C,energy_VAR_C) == 0x0) {
      OnLine_state = OnLine_stEnergySend_VA_C;
   }
   break;
case OnLine_stEnergySend_VA_C:   //Send Acknowledge
   if (RS232_sendLong(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_MEASURE_ENERGY_VA_C,energy_VA_C) == 0x0) {
      OnLine_state = OnLine_stIdle;
   }
   break;

//Flash Write Procedure:
case OnLine_stFlash_WriteMem:
   if (FLASH_ClearSegB() == 0) {
      OnLine_state = OnLine_stFlash_Write_WDIVA;
   }
   break;
case OnLine_stFlash_Write_WDIVA:
   if (FLASH_writeLong(FLASHMEM_MCU_WDIVA,MCU_WDIVA) == 0) {
      OnLine_state = OnLine_stFlash_Write_WDIVB;
   }
   break;
case OnLine_stFlash_Write_WDIVB:
   if (FLASH_writeLong(FLASHMEM_MCU_WDIVB,MCU_WDIVB) == 0) {
      OnLine_state = OnLine_stFlash_Write_WDIVC;
   }
   break;
case OnLine_stFlash_Write_WDIVC:
   if (FLASH_writeLong(FLASHMEM_MCU_WDIVC,MCU_WDIVC) == 0) {
      OnLine_state = OnLine_stFlash_Write_VARDIVA;
   }
   break;
case OnLine_stFlash_Write_VARDIVA:
   if (FLASH_writeLong(FLASHMEM_MCU_VARDIVA,MCU_VARDIVA) == 0) {
      OnLine_state = OnLine_stFlash_Write_VARDIVB;
   }
   break;
case OnLine_stFlash_Write_VARDIVB:
   if (FLASH_writeLong(FLASHMEM_MCU_VARDIVB,MCU_VARDIVB) == 0) {
      OnLine_state = OnLine_stFlash_Write_VARDIVC;
   }
   break;
case OnLine_stFlash_Write_VARDIVC:
   if (FLASH_writeLong(FLASHMEM_MCU_VARDIVC,MCU_VARDIVC) == 0) {
      OnLine_state = OnLine_stFlash_Write_VADIVA;
   }
   break;
case OnLine_stFlash_Write_VADIVA:
   if (FLASH_writeLong(FLASHMEM_MCU_VADIVA,MCU_VADIVA) == 0) {
      OnLine_state = OnLine_stFlash_Write_VADIVB;
   }
   break;
case OnLine_stFlash_Write_VADIVB:
   if (FLASH_writeLong(FLASHMEM_MCU_VADIVB,MCU_VADIVB) == 0) {
      OnLine_state = OnLine_stFlash_Write_VADIVC;
   }
   break;
case OnLine_stFlash_Write_VADIVC:
   if (FLASH_writeLong(FLASHMEM_MCU_VADIVC,MCU_VADIVC) == 0) {
      OnLine_state = OnLine_stFlash_AD7758Read_AVRMSOS;
   }
```

```
      break;
   case OnLine_stFlash_AD7758Read_AVRMSOS:
      if (AD7758_readInt(AVRMSOS) == 0x0) {
         OnLine_state = OnLine_stFlash_Write_AVRMSOS;
      }
      break;
   case OnLine_stFlash_Write_AVRMSOS:
      if (AD7758_isDataReady() == 0) {
         if (FLASH_writeInt(FLASHMEM_AVRMSOS,AD7758_getInt()) == 0) {
            OnLine_state = OnLine_stFlash_AD7758Read_BVRMSOS;
         }
      }
      break;
   case OnLine_stFlash_AD7758Read_BVRMSOS:
      if (AD7758_readInt(BVRMSOS) == 0x0) {
         OnLine_state = OnLine_stFlash_Write_BVRMSOS;
      }
      break;
   case OnLine_stFlash_Write_BVRMSOS:
      if (AD7758_isDataReady() == 0) {
         if (FLASH_writeInt(FLASHMEM_BVRMSOS,AD7758_getInt()) == 0) {
            OnLine_state = OnLine_stFlash_AD7758Read_CVRMSOS;
         }
      }
      break;
   case OnLine_stFlash_AD7758Read_CVRMSOS:
      if (AD7758_readInt(CVRMSOS) == 0x0) {
         OnLine_state = OnLine_stFlash_Write_CVRMSOS;
      }
      break;
   case OnLine_stFlash_Write_CVRMSOS:
      if (AD7758_isDataReady() == 0) {
         if (FLASH_writeInt(FLASHMEM_CVRMSOS,AD7758_getInt()) == 0) {
            OnLine_state = OnLine_stFlash_AD7758Read_AIRMSOS;
         }
      }
      break;
   case OnLine_stFlash_AD7758Read_AIRMSOS:
      if (AD7758_readInt(AIRMSOS) == 0x0) {
         OnLine_state = OnLine_stFlash_Write_AIRMSOS;
      }
      break;
   case OnLine_stFlash_Write_AIRMSOS:
      if (AD7758_isDataReady() == 0) {
         if (FLASH_writeInt(FLASHMEM_AIRMSOS,AD7758_getInt()) == 0) {
            OnLine_state = OnLine_stFlash_AD7758Read_BIRMSOS;
         }
      }
      break;
   case OnLine_stFlash_AD7758Read_BIRMSOS:
      if (AD7758_readInt(BIRMSOS) == 0x0) {
         OnLine_state = OnLine_stFlash_Write_BIRMSOS;
      }
      break;
   case OnLine_stFlash_Write_BIRMSOS:
      if (AD7758_isDataReady() == 0) {
         if (FLASH_writeInt(FLASHMEM_BIRMSOS,AD7758_getInt()) == 0) {
            OnLine_state = OnLine_stFlash_AD7758Read_CIRMSOS;
         }
      }
      break;
   case OnLine_stFlash_AD7758Read_CIRMSOS:
      if (AD7758_readInt(CIRMSOS) == 0x0) {
         OnLine_state = OnLine_stFlash_Write_CIRMSOS;
      }
      break;
   case OnLine_stFlash_Write_CIRMSOS:
      if (AD7758_isDataReady() == 0) {
         if (FLASH_writeInt(FLASHMEM_CIRMSOS,AD7758_getInt()) == 0) {
            OnLine_state = OnLine_stFlash_AD7758Read_APHCAL;
         }
      }
      break;
   case OnLine_stFlash_AD7758Read_APHCAL:
      if (AD7758_readChar(APHCAL) == 0x0) {
         OnLine_state = OnLine_stFlash_Write_APHCAL;
      }
      break;
   case OnLine_stFlash_Write_APHCAL:
```

```
      if (AD7758_isDataReady() == 0) {
         if (FLASH_writeChar(FLASHMEM_APHCAL,AD7758_getChar()) == 0) {
            OnLine_state = OnLine_stFlash_AD7758Read_BPHCAL;
         }
      }
      break;
   case OnLine_stFlash_AD7758Read_BPHCAL:
      if (AD7758_readChar(BPHCAL) == 0x0) {
         OnLine_state = OnLine_stFlash_Write_BPHCAL;
      }
      break;
   case OnLine_stFlash_Write_BPHCAL:
      if (AD7758_isDataReady() == 0) {
         if (FLASH_writeChar(FLASHMEM_BPHCAL,AD7758_getChar()) == 0) {
            OnLine_state = OnLine_stFlash_AD7758Read_CPHCAL;
         }
      }
      break;
   case OnLine_stFlash_AD7758Read_CPHCAL:
      if (AD7758_readChar(CPHCAL) == 0x0) {
         OnLine_state = OnLine_stFlash_Write_CPHCAL;
      }
      break;
   case OnLine_stFlash_Write_CPHCAL:
      if (AD7758_isDataReady() == 0) {
         if (FLASH_writeChar(FLASHMEM_CPHCAL,AD7758_getChar()) == 0) {
            OnLine_state = OnLine_stIdle;
         }
      }
      break;
   //Frequency Callibration
   case OnLine_stFreqCal_Init:
      if (AD7758_readInt(FREQ) == 0x0) {
         OnLine_state = OnLine_stFreqCal_ReadData;
      }
      break;
   case OnLine_stFreqCal_ReadData:
      if (AD7758_isDataReady() == 0x0) {
         OnLine_state = OnLine_stFreqCal_SendAD7758Data;
      }
      break;
   case OnLine_stFreqCal_SendAD7758Data:
      if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,FREQ,AD7758_getInt()) == 0x0) {
         OnLine_state = OnLine_stFreqCal_SendCalAck;
      }
      break;
   case OnLine_stFreqCal_SendCalAck:
      if (RS232_sendInt(PROTOCOL_COMPONENT_MCU,PROTOCOL_REGISTER_MCU_CURRENTCAL,PROTOCOL_VALUE_MCU_FREQ) == 0x0) {
         OnLine_state = OnLine_stIdle;
      }
      break;

   case OnLine_stPhaseCal_InitXWG:
      //Prior to calling this state, the following variables must be set up correctly:
      //    OnLine_Cal_lcycmode  - required value of AD7768 register
      //    OnLine_Cal_xWG       - used registers...
      //    OnLine_Cal_xWATTHR
      //    OnLine_Cal_currentCal- remembers the current callibration procedure
      // This callibration procedure is done according to the AD7758 datasheet.
      if (AD7758_writeInt(OnLine_Cal_xWG,0x00) == 0x0) {
         OnLine_state = OnLine_stPhaseCal_InitLCYCMODE;
      }
      break;
   case OnLine_stPhaseCal_InitLCYCMODE:      //Set LCYCMODE
      if (AD7758_writeChar(LCYCMODE,OnLine_Cal_lcycmode) == 0x0) {
         OnLine_state = OnLine_stPhaseCal_InitRead_LCYCMODE;
      }
      break;
   case OnLine_stPhaseCal_InitRead_LCYCMODE: //LCYCMODE
      if (AD7758_readChar(LCYCMODE) == 0x0) {
         OnLine_state = OnLine_stPhaseCal_Read_LCYCMODE;
      }
      break;
   case OnLine_stPhaseCal_Read_LCYCMODE:
      if (AD7758_isDataReady() == 0x0) {
         OnLine_dataChar = AD7758_getChar();
         OnLine_state = OnLine_stPhaseCal_Send_LCYCMODE;
      }
      break;
```

```
case OnLine_stPhaseCal_Send_LCYCMODE:
   if (RS232_sendChar(PROTOCOL_COMPONENT_AD7758,LCYCMODE,OnLine_dataChar) == 0x0) {
      OnLine_state = OnLine_stPhaseCal_InitRead_LINECYC;
   }
   break;
case OnLine_stPhaseCal_InitRead_LINECYC:  //LINECYC
   if (AD7758_readInt(LINECYC) == 0x0) {
      OnLine_state = OnLine_stPhaseCal_Read_LINECYC;
   }
   break;
case OnLine_stPhaseCal_Read_LINECYC:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stPhaseCal_Send_LINECYC;
   }
   break;
case OnLine_stPhaseCal_Send_LINECYC:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,LINECYC,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stPhaseCal_InitMASK;
   }
   break;
case OnLine_stPhaseCal_InitMASK:       //Set MASK
   if (AD7758_writeLong(MASK,0x01000) == 0x0) {
      OnLine_state = OnLine_stPhaseCal_InitRead_MASK;
   }
   break;
case OnLine_stPhaseCal_InitRead_MASK:  //MASK
   if (AD7758_readLong(MASK) == 0x0) {
      OnLine_state = OnLine_stPhaseCal_Read_MASK;
   }
   break;
case OnLine_stPhaseCal_Read_MASK:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataLong = AD7758_getLong();
      OnLine_state = OnLine_stPhaseCal_Send_MASK;
   }
   break;
case OnLine_stPhaseCal_Send_MASK:
   if (RS232_sendLong(PROTOCOL_COMPONENT_AD7758,MASK,OnLine_dataLong) == 0x0) {
      OnLine_state = OnLine_stPhaseCal_InitRead_FREQ;
   }
   break;
case OnLine_stPhaseCal_InitRead_FREQ:  //FREQ
   if (AD7758_readInt(FREQ) == 0x0) {
      OnLine_state = OnLine_stPhaseCal_Read_FREQ;
   }
   break;
case OnLine_stPhaseCal_Read_FREQ:
   if (AD7758_isDataReady() == 0x0) {
      OnLine_dataInt = AD7758_getInt();
      OnLine_state = OnLine_stPhaseCal_Send_FREQ;
   }
   break;
case OnLine_stPhaseCal_Send_FREQ:
   if (RS232_sendInt(PROTOCOL_COMPONENT_AD7758,FREQ,OnLine_dataInt) == 0x0) {
      OnLine_state = OnLine_stPhaseCal_FirstResetInt;
   }
   break;
case OnLine_stPhaseCal_FirstResetInt:     //Reset Sequence
   if (AD7758_readLong(RSTATUS) == 0x0) {
      OnLine_state = OnLine_stPhaseCal_ResetInt;
   }
   break;
case OnLine_stPhaseCal_ResetInt:
   if (AD7758_readLong(RSTATUS) == 0x0) {
      OnLine_state = OnLine_stPhaseCal_ResetIntWait;
   }
   break;
case OnLine_stPhaseCal_ResetIntWait:
   if (AD7758_isDataReady() == 0x0) {
      if ((AD7758_getLong() & 0x01000) == 0) {
         OnLine_state = OnLine_stPhaseCal_ResetInt;
      } else {
         OnLine_state = OnLine_stPhaseCal_InitRead_xWATTHR;
         if ((AD7758_getLong() & 0x040) != 0) {
            OnLine_flags |= OnLine_flagGainCalOverflow;
         } else {
            OnLine_flags &= ~OnLine_flagGainCalOverflow;
         }
```

```
            }
         }
         break;
      case OnLine_stPhaseCal_InitRead_xWATTHR:  //xWATTHR
         if (AD7758_readInt(OnLine_Cal_xWATTHR) == 0x0) {
            OnLine_state = OnLine_stPhaseCal_Read_xWATTHR;
         }
         break;
      case OnLine_stPhaseCal_Read_xWATTHR:
         if (AD7758_isDataReady() == 0x0) {
            OnLine_dataInt = AD7758_getInt();
            OnLine_state = OnLine_stPhaseCal_Send_xWATTHR;
         }
         break;
      case OnLine_stPhaseCal_Send_xWATTHR:
         if (RS232_sendInt(PROTOCOL_COMPONENT_MCU,OnLine_Cal_currentCal,OnLine_dataInt) == 0x0) {
            OnLine_state = OnLine_stIdle;
         }
         break;
   }
}
```

R.5.4 gsmMode.h

```
//*****************************************************************************
//
// gsmMode
//
// This header runs on an application level and is solely responsible for the
//    reporting energy values via gsm.
//
//
// History:
// v1.0 - 2004-07-20 Creation of Driver
//
//  R. Doorduin
//  PEG - University of Stellenbosch
//  March 2004
//*****************************************************************************

//GSM defines
#define GSM_PIN "9142"
#define GSM_NO "0846845943"
#define GSM_SERVER "0846845946"
#define GSM_ADMIN "0848163790"
#define GSM_SMSCenter "+27841000000"

// State Event Declerations
//-----------------------------------------------------------------
//Driver
#define gsmMode_stInit                          0x000
#define gsmMode_stIdle                          0x001
#define gsmMode_stInit_waitRS232off             0x002
#define gsmMode_stInitStart                     0x003
#define gsmMode_stInitSystem                    0x004
#define gsmMode_stInit_GSMoff                   0x005
#define gsmMode_waitGSMonSignal                 0x006
#define gsmMode_waitGSMoffSignal                0x007
#define gsmMode_gsmATE0                         0x008
#define gsmMode_gsmATE0_OK                      0x009
#define gsmMode_gsmCGMM                         0x00A
#define gsmMode_gsmCGMMlineReady                0x00B
#define gsmMode_gsmCGMM_OK                      0x00C
#define gsmMode_gsmPIN                          0x00E
#define gsmMode_gsmSMS_ServiceCenter            0x010
#define gsmMode_gsmSMS_Setup                    0x011
#define gsmMode_gsmSMS_SetupFormat              0x012
#define gsmMode_gsmCLIP                         0x013
#define gsmMode_gsmADT_Credit                   0x014
#define gsmMode_gsmADT_adminCallMe              0x015
#define gsmMode_gsmADT_listAllMessages          0x016
#define gsmMode_gsmADT_deleteAllMessages        0x017
#define gsmMode_stSMS_sendEnergyReport          0x018
#define gsmMode_stSMS_isSendEnergyReportSent    0x019

//Variables
int gsmMode_state = gsmMode_stInit;
```

```
//Flags
#define gsmMode_flagInit                BIT0
int gsmMode_flags                  = 0x00;     //Flags to allow the driver to keep track of user input

// GSM_driver
//
// The driver responsible for coordinating various tasks regarding GSM operation.
//
//---------------------------------------------------------------------------------------------------------
void GSM_driver(void) {
   switch (gsmMode_state) {
      //Initialization
      //-------------------------------------------------------
      case gsmMode_stInit:
         gsmMode_state = gsmMode_stInit_GSMoff;
         break;
      //GSM OFF
      case gsmMode_stInit_GSMoff:
         if (AT_sendCommand("AT+CPOF") == 0) {
            UART_setRS232off();
            gsmMode_state = gsmMode_stInit_waitRS232off;
         }
         break;
      case gsmMode_stInit_waitRS232off:
         if (UART_isRS232off() == 0) {
            //Start General Timers
            gsmMode_state = gsmMode_stInitStart;
         }
         break;
      case gsmMode_stInitStart:
         if (INIT_init() == 0) {
            gsmMode_state = gsmMode_stInitSystem;
         }
         break;
      //GSM ON
      case gsmMode_stInitSystem:
         if (INIT_isReady() != 0) {
            INIT_driver();
         } else {
            AT_setGSMonSignal();
            TIMER_startTimer(tmrGSM_On_Off_Pulse);
            gsmMode_state = gsmMode_waitGSMonSignal;
         }
         break;
      case gsmMode_waitGSMonSignal:
         if (TIMER_isExpired(tmrGSM_On_Off_Pulse,delayGSM_On_Off_Pulse) == 0) {
            TIMER_startTimer(tmrGSM_On_Off_Pulse);
            AT_resetGSMonSignal();
            gsmMode_state = gsmMode_waitGSMoffSignal;
         }
         break;
      case gsmMode_waitGSMoffSignal:
         if (TIMER_isExpired(tmrGSM_On_Off_Pulse,delayGSM_On_Off_Pulse) == 0) {
            TIMER_stopTimer(tmrGSM_On_Off_Pulse);
            gsmMode_state = gsmMode_gsmATE0;
         }
         break;
      //ECHO
      case gsmMode_gsmATE0:
         if (AT_sendCommand("ATE0") == 0) {
            gsmMode_state = gsmMode_gsmATE0_OK;
         }
         break;
      case gsmMode_gsmATE0_OK:
         if (AT_isLine("ATE0OK") == 0) {
            gsmMode_state = gsmMode_gsmCGMM;
         }
         break;
      //MODEL NAME
      case gsmMode_gsmCGMM:
         if (AT_sendCommand("AT+CGMM") == 0) {
            AT_lineReset();
            gsmMode_state = gsmMode_gsmCGMMlineReady;
         }
         break;
      case gsmMode_gsmCGMMlineReady:
         if (AT_isLineReady() == 0) {
            AT_lineReset();
            gsmMode_state = gsmMode_gsmCGMM_OK;
```

```
   }
   break;
case gsmMode_gsmCGMM_OK:
   if (AT_isLine("OK") == 0) {
      TIMER_startTimer(tmrGSM_WritePIN_delay);
      gsmMode_state = gsmMode_gsmPIN;
   }
   break;
//PIN
case gsmMode_gsmPIN:
   if (AT_sendCommandOK("AT+CPIN=" GSM_PIN) == 0) {
      gsmMode_state = gsmMode_gsmSMS_ServiceCenter;
   }
   break;
//SMS
case gsmMode_gsmSMS_ServiceCenter:
   if (AT_sendCommandOK("AT+CSCA=\"" GSM_SMSCenter "\",145") == 0) {
      gsmMode_state = gsmMode_gsmSMS_Setup;
   }
   break;
case gsmMode_gsmSMS_Setup:
   if (AT_sendCommandOK("AT+CNMI=0,1,1,1,0") == 0) {
      gsmMode_state = gsmMode_gsmSMS_SetupFormat;
   }
   break;
case gsmMode_gsmSMS_SetupFormat:
   if (AT_sendCommandOK("AT+CMGF=1") == 0) {
      gsmMode_state = gsmMode_gsmCLIP;
   }
   break;
//CLIP
case gsmMode_gsmCLIP:
   if (AT_sendCommandOK("AT+CLIP=1") == 0) {
      gsmMode_state = gsmMode_gsmADT_Credit;
   }
   break;
//MESSAGES
case gsmMode_gsmADT_Credit:
   if (AT_sendCommandOK("ATD*101#") == 0) {
      gsmMode_state = gsmMode_gsmADT_adminCallMe;
   }
   break;
case gsmMode_gsmADT_adminCallMe:
   if (AT_sendCommandOK("ATD*111*" GSM_ADMIN "#") == 0) {
      gsmMode_state = gsmMode_gsmADT_listAllMessages;
   }
   break;
case gsmMode_gsmADT_listAllMessages:
   if (AT_sendCommand("AT+CMGL=\"ALL\"") == 0) {
      gsmMode_state = gsmMode_gsmADT_deleteAllMessages;
   }
   break;
case gsmMode_gsmADT_deleteAllMessages:
   if (AT_sendCommandOK("AT+CMGD=1,4") == 0) {
      AT_lineReset();
      TIMER_startTimer(tmrWDTGSM);
      gsmMode_state = gsmMode_stIdle;
   }
   break;

//IDLE
// This state is responsible for correct state assignment
// according to the decoded RS232 protocol.
//-----------------------------------------------------
case gsmMode_stIdle:
   if (TIMER_isExpired(tmrWDTGSM,delayWDTGSM) == 0) {
      TIMER_startTimer(tmrWDTGSM);
      LEDs_toggleLED0();
   } else if (AT_isLineReady() == 0) {
      if (AT_isLine("RING") == 0) {
         AT_lineReset();
         //gsmMode_state = gsmMode_stSMS_sendEnergyReport;
      } else if (AT_parseLine() == 0) {
         if (strCmp("CLIP",AT_Cmd) == 0) {
            AT_lineReset();
            gsmMode_state = gsmMode_stSMS_sendEnergyReport;
         } else if (strCmp("CMTI",AT_Cmd) == 0) {
            AT_lineReset();
            gsmMode_state = gsmMode_stSMS_sendEnergyReport;
```

```
               } else if (strCmp("CMGS",AT_Cmd) == 0) {
                  AT_lineReset();
               }
            } else {
               AT_lineReset();
            }
         }
         break;

      //SMS REPORTS
      //-------------------------------------------------------
      case gsmMode_stSMS_sendEnergyReport:
         if (AT_SendEnergyReport(GSM_SERVER) == 0) {
            gsmMode_state = gsmMode_stSMS_isSendEnergyReportSent;
         }
         break;
      case gsmMode_stSMS_isSendEnergyReportSent:
         if (AT_isSendEnergyReportSent() == 0) {
            gsmMode_state = gsmMode_gsmADT_deleteAllMessages;
         }
         break;
   }
}
```

R.5.5 testStateEvent.h

Included for debugging and provides an easy starting point for testing of new routines.

```
/*****************************************************************
*  testStateEvent.h
*
*  This state event machine is only inteded for testing and debugging
*     purposes.
*
*  R. Doorduin
*  PEG - University of Stellenbosch
*****************************************************************/

//The state event machine is driven by the main procedure
char c;
int state = 0x0;
void test_StateEventMachine(void) {
   LEDs_toggleLED6();

   //Check State Mode
   switch (state) {
      //No State
      //---------------------------------------------------------------------
      case 0x00:
         UART_setRS232on();
         state = 0x01;
         break;
      case 0x01:     //RS232_Init_Wait
         if (UART_isRS232on() == 0) {
            TIMER_startTimer(tmrTestSE);
            state = 0x010;
         }
         break;
      case 0x010:
         if (FLASH_ClearSegA() == 0) {
            state = 0x011;
         }
         break;
      case 0x011:
         if (FLASH_ClearSegB() == 0) {
            state = 0x012;
         }
         break;
      case 0x012:
         if (FLASH_writeLong(FLASHMEM_energy_VA_C,0x030313233) == 0) {
            state = 0x013;
         }
         break;
      case 0x013:
         if (FLASH_writeInt(FLASHMEM_AIRMSOS,0x03839) == 0) {
            state = 0x02;
```

```
         }
         break;
      case 0x02:     //Idle
         if (UART_isCharReady() == 0) {
            LEDs_toggleLED1();
            c = UART_getChar();
            state = 0x04;
         } else if (TIMER_isExpired(tmrTestSE,delayTestSE) == 0) {
            TIMER_startTimer(tmrTestSE);
            LEDs_toggleLED0();
            //state = 0x04;
         } else {
            //state = 0x04;
         }
         break;
      case 0x04:     //Write special charchters out
         if (UART_writeLong(FLASH_readLong(FLASHMEM_energy_VA_C)) == 0) {
            state = 0x05;
         }
         break;
      case 0x05:     //Write special charchters out
         if (UART_writeInt(FLASH_readInt(FLASHMEM_AIRMSOS)) == 0) {
            state = 0x02;
         }
         break;
      default:
         state = 0x0;
   }
}
```