DESIGN AND DEVELOPMENT OF 3 PHASE POWER METER

By

SYAFIQ BASRI BIN SHAARI

FINAL REPORT

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

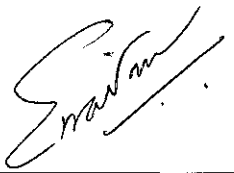
DESIGN AND DEVELOPMENT OF 3 PHASE POWER METER

by

Syafiq Basri Bin Shaari

A project dissertation report submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:



A.P. Dr. Irraivan Elamvazuthi
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

May 2011

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

Syafiq Basri Bin Shaari

ABSTRACT

In recent years, research into energy saving has been increasing in view of dealing with environmental problems and effectively using energy resources. Hence, embedded applications for energy saving and monitoring in industrial environment has gone up steadily. However, most applications are limited to measurement of power consumption of the whole facility rather than individual devices. Therefore, a 3 phase power meter has been designed specifically to measure power consumption on a single device. Different from the conventional power meters, this meter is designed to be plugged-in directly into the power supply socket meanwhile the load will be plugged-in onto the power meter socket. This idea will remove the need of wires that conventionally used and suitable for temporary or permanent installation. The project focused on using Programmable interface controller and linear isolated analog sensors to achieve safe power measurement procedure. The measurement result is expected to be accurate and acceptable.

ACKNOWLEDGEMENTS

I would like to show my highest gratitude to The Almighty God for giving me the chance and guiding me to complete this project. Throughout finishing the project, I have learnt a lot from other people. Their contributions to my project either directly or indirectly can never be paid. I want to take this opportunity to thank them and hopefully they can keep the good work up ahead.

My most gratitude goes to my supervisor, A.P. Dr. Irraivan Elamvazuthi for guiding me throughout the last two semesters. Without his support, help and advice, I would never be able to achieve this level. Last but not least, thank you to all electrical department lab technologists that provides me the equipments, guidance and help to do my project. Special thanks to Mr. Badrulnizam Bin Ni-A Rani for allowing me to use the Power Electronic laboratory to test my prototype. In addition, thank you to Ms. Siti Hawa Bt. Tahir for guiding me do complete the report according to its format. Only God can pay all of your kindness.

To my beloved family especially my parents, thank you for always believing in me, supporting me and making me believe to finish my study. Finally, thank you to all my friends that helped me either directly or indirectly upon completion of this project. Without their support it might be harder for me to finish this project on time. Only God knows how to pay all of your kindness.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

A/D	Analog to Digital Converter
AC	Alternating Current
CUT	Circuit under test
DC	Direct Current
DUT	Device under test
EEPROM	Electrically Erasable Programmable Read-Only Memory
FYP	Final Year Project
HEX	Hexadecimal
ID	Index
LCD	Liquid Crystal Display
PIC	Programmable Interface Controller Microcontroller
RMS	Root Mean Square
SD	Secure Digital
SPI	Serial Peripheral Interface Bus
TNB	Tenaga Nasional Berhad
USB	Universal Serial Bus

CHAPTER 1
INTRODUCTION

This section will discuss on the background of study, problem statement and objective and scope of study.

1.1 Background of Study

In electrical power system there are two power supply phases that commonly used. The power supplies phases are single phase and three phases. Single phase power supply is commonly used at household or small power consumption area. However the three phase power supply is used at industrial areas where the loads are consuming large power.

The electrical power supplier such as Tenaga Nasional Berhad (TNB) will charge the customer by metering the power consumptions by the facility. The power consumption metered by TNB is the total power consumption taken by the facility loads. In the bill there will be the total power consumption and its charge. There will be no details about the power consumption taken by a single load. So the customer unable to analyze the appliances power consumption efficiency individually. At household, the power consumption by individual appliance is not significant because the charge is considerably acceptable. In addition there are not much appliance that consumes a lot of power such as washing machine, air conditioner and refrigerator. However the analysis is considerably critical for industrial users because the power consumption will affect their monthly expenses.

Besides of using the installed TNB power meter, some facility installed embedded power metering systems. However this system is expensive and hard to expand if there is new appliance to be added. Other metering option is by using portable power meter. Due to its portability it can be used to measure individual appliance power consumption at any location.

By having individual power metering capability, power hungry appliance can be detected. Moreover, individual measurement also allows the user to compare power consumption of appliance with same function but from different manufacturer. This will allows the user to select the most efficient and less power consuming appliance. This approach will save future expenses due to electrical bill charges thus improve the facility profit.

1.2 Problem Statement

In most industrial area, three phase electricity is usually used. Sometimes, the user does not know how much exact power has been consumed by the device and is it the best device to be used in the facility. Because of this, a three phase power meter is needed to measure the power consumption. However, usually the facility has power meter that measure power consumption for whole part of the facility and not meant for a single device measurement.

In addition, most of available power meter uses cables and clamper to connect to the DUT. This will introduce untidiness of work area when we have lots of wires hanging. Moreover the clampers or wire clips was meant for temporary metering only and not useful for long term data logging.

1.3 Objective and Scope of Study

The objective of the project is to design a portable meter that is capable of measuring three phase voltage, current and power by using analog circuit with help of Programmable Interface Controller Microcontroller (PIC) with protective isolation. The meter is designed to be plugged-in directly into the power supply socket meanwhile the load will be plugged-in onto the power meter socket. This idea will remove the need of wires that conventionally used and suitable for temporary or permanent installation.

The scope of study of this project is as follows:

- Fundamental study of three phase power supply. The study would cover the voltage, current and power behavior in three phase system.
- Isolation study.

- Circuit design and simulation using computer software.
- Actual circuit testing in power laboratory.
- Programmable Interface Controller Microcontroller (PIC) programming.
- Final prototype design.

CHAPTER 2
LITERATURE REVIEW

In recent years, research into energy saving has been increasing in view of dealing with environmental problems and effectively using energy resources. Hence, embedded applications for energy saving and monitoring in industrial environment has gone up steadily. The research related to power meters has been done by [1] – [6]. However, most of the researches are limited to measurement of power consumption of the whole facility rather than individual devices. This section will discuss general information about power meters, power meter product examples, power calculation theory and technical description about the tools that will be used in the project.

2.1 Power Meters

Power meter is a device that capable to measure power qualities of a power system. Most power meter capable to measure real, reactive and apparent power. Some power meter also includes other parameters such as power factor, frequency, current and voltage. Not all power meter capable to measure all type power system configuration. Certain power meters capable to measure single phase power system only and some of them capable to measure three phase power system only. However there are also power meter that capable to measure both power systems.

2.1.1 Three Phase Power Meter

Three phase power meter is a device that can measure three phase AC power. Currently, there a lot of power meter that available in market. Most of the power meter has the capability to measure real power (P), reactive power (Q), and apparent power (S). Most of them allow the user to display measurements for each phase. Some of them also capable to measure harmonic in the device under test (DUT). More advance power meter even support data logging, computer communication and

external memory module.

For high power measurement, the power meter uses current sensor based on Hall Effect current sensor transducer to measure voltage and current on DUT. By using this technique they can be totally isolated from another high voltage electrical system which eliminates many safety concerns [7]. However, the cost for the transducer is very high which result in expensive product price.

In smaller power measurement, other alternative can be use such as precision current resistor [8] and opto-coupler. The precision current resistor is a resistor with low resistance so that it will not alter the DUT current flow. So, voltage drop across the resistor will be used to calculate the current flow in DUT. Meanwhile, according to [9], the opto-coupler uses a short optical signal from transmitter to receiver with without physical connection. This will keep them electrically isolated. The optical signal received by received will converted back into suitable electrical signal. Isolation of an opto-couple is depends on manufacturer but typically it is rated at 7500 Volt peak for 1 second. However, the limitation is the input current support is very low. The cost for opto-coupler with high input current is quite high than the low input current.

2.1.2 PITE 3551 Power Meter Tester

PITE 3551 is a portable power meter designed for field testing. According to [10], this meter is smart, portable and multi-functional for field-testing. This meter capable to measure power for power grid of high and low voltage, both single-phase and three-phase. The device features are as following [10]:

- Portable
- Low voltage & high voltage measurement
- Online activity measurement
- Test result playback: Convenient to overview testing result
- System management: Parameter setting and measurement calibration
- Memory: 16M byte Flash
- Communication port: USB port

Figure 1 PITE 3551 Power Meter Tester [10]

2.1.3 Reed Instruments KEW6300-03

Another available product is Reed Instruments KEW 6300-03 power meter. This meter has more advance features than PITE 3551. For example, it has both wire clips interface and also clampers to measure the DUT. In addition, it has larger LCD display and also support external memory module. Below are the specifications of this device [11]:

- Measures 4 types of wiring systems: 1 phase-2 wire; 1 phase-3 wire, 3 phase-3 wire, and 3 phase-4 wire
- Electric power and power factor for each phase can be confirmed
- Recording interval can be set between 1 second and 1 hour
- Internal non-volatile memory for non-stop recording up to 10 days
- Accepts removable compact flash memory up to 128MB for non-stop recording up to 5 years
- Direct communication with PC by USB connection

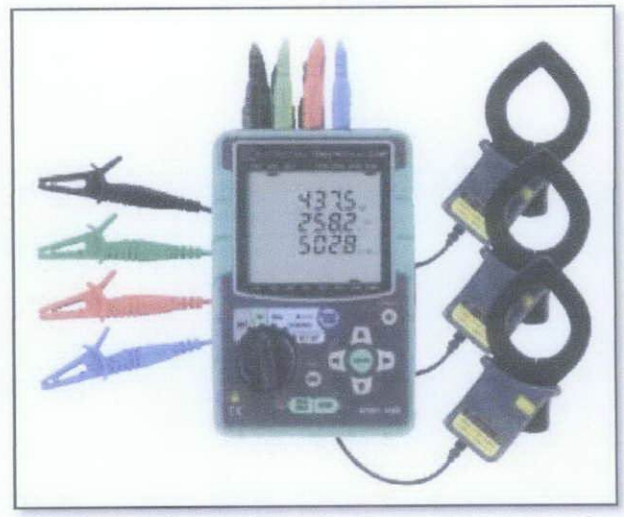


Figure 2 Reed Instruments KEW6300-03 [11]

2.2 Principles of Power Calculation

Basically the total instantaneous real power is the summation of instantaneous power of each phase as shown below [12]:

$$p_{3\emptyset} = v_{an}i_a + v_{bn}i_b + v_{cn}i_c \ \ where\ n = 0,1,2,3,\ldots \qquad (2.1)$$

The simplified version of the three phase power formula which taken from single phase analysis is [12]:

$$P_{3\emptyset} = 3|V_p||I_p|\cos\theta \qquad (2.2)$$

where the $\cos\theta$ represent the power factor of the system. The θ value is equal to $\theta_v - \theta_i$ which represents the angle difference between phase voltage and phase current or the impedance angle [12].

However the instantaneous measurement in equation 2.1 is not desirable in metering because the value will keep on changing over time. In addition, equation 2.2 cannot be implemented directly to the PIC. Thus, RMS value needs to be calculated. Below is the single phase RMS power calculation [13] - [14]:

$$P = \sqrt{\frac{1}{1+N}\sum_{n=0}^{N}(v_n i_n)^2} \qquad (2.3)$$

Thus, equation in 2.3 is suitable to be used in the power meter. So, the total RMS three phase real power can be calculated as follows:

$$P_{3\emptyset} = \sqrt{\frac{1}{1+N}\sum_{n=0}^{N}(v_{an}i_{an})^2} + \sqrt{\frac{1}{1+N}\sum_{n=0}^{N}(v_{bn}i_{bn})^2} + \sqrt{\frac{1}{1+N}\sum_{n=0}^{N}(v_{cn}i_{cn})^2} \quad (2.4)$$

However, the equation 2.4 does not provide information about the system power factor. In order to grab power factor value, apparent power (S) value needs to be calculated. Power factor (p.f) can be determined by following equation [12]:

$$\text{p.f} = \cos\emptyset = \frac{P_{RMS}}{S_{RMS}} \quad (2.5)$$

Apparent power for a single phase system can be determined by general equation in equation 2.6 meanwhile equation 2.7, 2.8 and 2.9 represent RMS apparent power formula for phase a, phase b and phase c respectively [12]:

$$S_{1\emptyset} = v_a i_a \quad (2.6)$$

$$S_{a\emptyset} = \sqrt{\frac{1}{1+N}\sum_{n=0}^{N}(v_{an})^2} \times \sqrt{\frac{1}{1+N}\sum_{n=0}^{N}(i_{an})^2} \quad (2.7)$$

$$S_{b\emptyset} = \sqrt{\frac{1}{1+N}\sum_{n=0}^{N}(v_{bn})^2} \times \sqrt{\frac{1}{1+N}\sum_{n=0}^{N}(i_{bn})^2} \quad (2.8)$$

$$S_{c\emptyset} = \sqrt{\frac{1}{1+N}\sum_{n=0}^{N}(v_{cn})^2} \times \sqrt{\frac{1}{1+N}\sum_{n=0}^{N}(i_{cn})^2} \quad (2.9)$$

So, the total apparent power is the summation of the apparent powers from all phases. By having power factor information, the reactive power (Q) can be calculated. Equation 2.10 represent the general equation for reactive power meanwhile equation 2.11 is reactive power equation derived from previous equations [12].

$$Q_n = |V_n||I_n|\sin\emptyset \quad (2.10)$$

$$Q_n = |V_n||I_n|\sin\left(\cos^{-1}\frac{P_{RMS}}{S_{RMS}}\right) \quad (2.11)$$

2.3 Technical Principle

The power meter that was developed in this project involved selection of components such PIC, Hall Effect current sensor and opto-coupler (as voltage sensor). This section will discuss on the basic operation of PIC, selected current sensor and linear analog opto-coupler.

2.3.1 Programmable Interface Controller Microcontroller

Microcontroller is a small computer chip that used in control application. PIC is one of the microcontroller families that are produced by Microchip Technology Inc. Another term for microcontroller is embedded controller, since most of the microcontrollers are built into (or embedded in) the devices they control [15]. Different from microprocessor; microcontroller already has all needed chip embedded onto it such as memory module, input and output module, registers, analog to digital converter and more.

The microcontroller can be programmed by several programming languages. The languages are assembly language, BASIC, C and PASCAL. Most popular and easier to learn is C language since it is a high level language. In theory, a single chip is sufficient to have a running microcontroller system. In practical applications, however, additional components may be required so the microcomputer can interface with its environment [15].

The function of the microcontroller basically executes user program that loaded in its memory. Under the control of this program, data is received from external devices (inputs), manipulated, and then sent to external devices (outputs) [15]. Figure 3 shows the Microchip PIC.

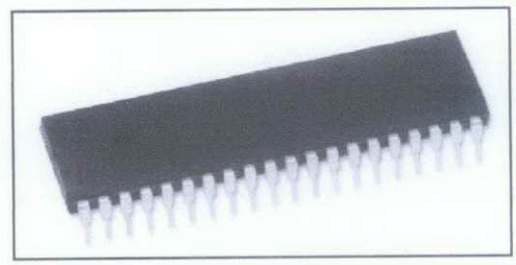


Figure 3 Microchip PIC [16]

2.3.2 Allegro ACS 712 Current Sensor

The Allegro ACS712 family of current sensor ICs provides economical and precise solutions for AC or DC current sensing [17]. It utilized low-offset linear Hall circuit with a copper conduction path located near the die to measure current flow. Current flow through the copper conduction path generates a magnetic field meanwhile a voltage that proportional to the current flow will be generated by the Hall. The close proximity of the magnetic signal to the Hall transducer provides the device accuracy [17]. A precise, proportional voltage is provided by the low-offset, chopper-stabilized BiCMOS Hall IC, which is programmed for accuracy at the factory [17]. According to the device datasheet, the internal resistance of the conductor path is 1.5 mΩ thus power loss is very low. The device conductor thickness is design to hold over current up to 5 times [17]. In addition, the terminals of the conductive path are electrically isolated from the signal leads (pins 1 and 2 through 3 and 4) [17]. By having this feature, this device can be operated in isolated condition without using opto-isolator technique. Figure 4 shows the Allegro ACS712 current sensor.

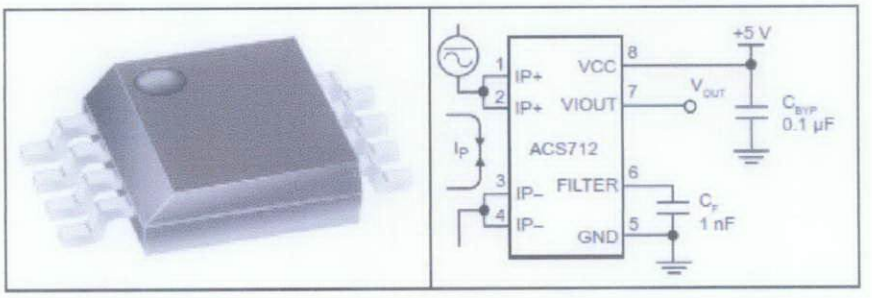


Figure 4 Allegro ACS 712 Current Sensor [17]

2.3.3 Agilent HCPL7520

The HCPL-7520 isolated linear current sensing IC family is designed for current sensing in low-power electronic motor drives [18]. In typical application, the load current is passing through a small resistor. The voltage drop across the resistor is sensed by this device. Across the optical isolation barrier at another side of the IC the output is generated. The output gives a voltage which is proportional to the current or voltage drop across the resistor. The HCPL7520 also was designed to ignore very high common-mode transient slew rates (of at least 10 kV/μs) [18]. In addition, according to [18], the high CMR capability of the HCPL-7520 isolation amplifier provides the precision and stability needed to accurately monitor motor current in high electrical noise environment. In addition, this device also can be used for general analog signal isolation applications. Figure 5 shows the Agilent HCPL 7520 opto-coupler.

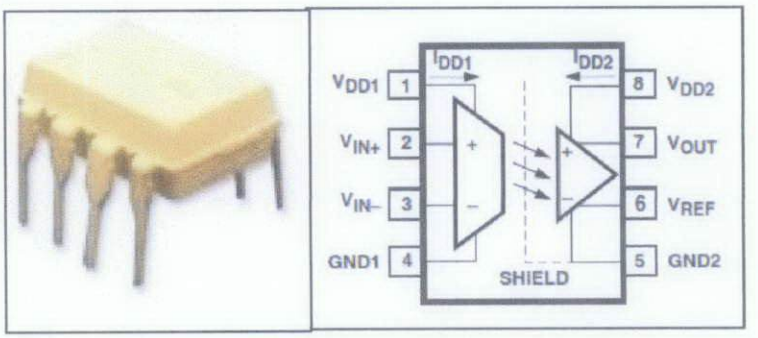


Figure 5 Agilent HCPL7520 [18]

CHAPTER 3
METHODOLOGY

This chapter discuss on the procedure identification of the project, tools and equipment used in the project, and finally the project design. The procedure identification part will discuss all 7 procedures taken to complete the project. Meanwhile tools and equipment part will discussed on all tools and equipment which consists of software and hardware. The last part will discuss all project design related subject consisting of software and hardware development.

3.1 Procedure Identification

Figure 6 shows the procedure identification flow that used through the project design and development process.

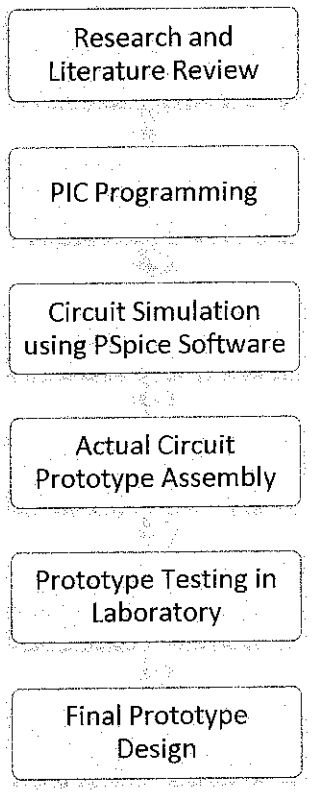


Figure 6 Procedure Identification Flow

3.1.1 Research and Literature Review

The research is done through various sources such as journals, internet, books, people and previous projects. The research mainly is about the power measurement fundamentals that consist of voltage and current behavior in three phase system.

3.1.2 PIC Programming

The source code for the PIC will be written in C language. The PIC will be programmed using any compatible C compiler for PIC such as microC for PIC software.

3.1.3 Circuit Simulation using PSpice Software

The circuit for voltage and current measurement are drawn and simulated using PSpice software. This to ensure that the resulting current and voltage fed into PIC are within the PIC ratings. This step also to ensure that circuit is working and safe to fabricate.

3.1.4 Actual Circuit Prototype Assembly

After the designed circuits passed the requirement needed during the simulation, the circuit prototypes are assembled on breadboard or veroboard. The circuits are PIC circuit, voltage measurement circuit and current measurement circuit. The measurement circuit will be the PIC input and the LCD display will be the output for the PIC. In the circuit, some buttons will be inputs for the PIC which allows the user to set some configuration into the PIC.

3.1.5 Prototype Testing in Laboratory

The assembled prototype will be tested in UTP power laboratory. The purpose of the test is to ensure that the circuit working properly with proper safety measures. The lab has the utility to provide three phase power supply. In additional of that, the lab also has wattmeter that can be use to compare power measurement with the prototype.

3.1.6 Final Prototype Design

The working prototype will be use as the reference to design the final prototype. The final prototype is planned to have proper casing and packaging. Electrical circuit will be transferred into a proper printed circuit board (PCB). The board will be design using Eagle PCB Maker software and will be fabricated inside or outside UTP. Moreover the PIC program will be updated to add any new features. Safety measures also are seriously taken into account to ensure that the prototype is safe to use.

3.2 Tools and Equipments

Basically, there are two categories of tools that will be used in this project. They are the software and hardware. Table 1 shows the list of software used during project design and development. Meanwhile Table 2 shows the list of hardware used during project design and development.

Table 1 Software List

Software Name	Function
Mikroelektronika MicroC	PIC source code editor and compiler.
Microchip PICKit2	PIC HEX programmer. This software will flash the compiled HEX file into the PIC memory.
Orcad PSPice	Circuit Simulation Software.
Cadsoft Eagle PCB	Software to design PCB.
LVDAM-EMS	Laboratory data acquisition software that retrieve data from metering module from data acquisition interface.

Table 2 Hardware List

Hardware Name	Function
PIC 18F458	Main project PIC. Work as the "brain" of the unit.
9 V Battery	Main DC Supply
Wires	General purpose connection.
Breadboard	To test temporary circuit.
4MHz Crystal Oscillator	Main PIC clock.
2 Lines LCD panel	To display the PIC output as the user interface.
Push Buttons	General purpose and as user interface inputs.
LEDs	General purpose indicator.
Voltage Regulator (7805)	To produce 5V output and as a main 5V supply for the circuit.
Voltage Regulator (LM1117-T3.3)	To produce 3.3V power supply to memory card module.
Resistors.	General purpose.
Allegro ACS 712 - Current Sensor	Isolated hall-effect current sensor. For current sensing purpose.
Agilent HCPL7520 - Linear Analog Opto-isolator	To isolate DUT from power meter unit. For voltage measurement purpose.
Capacitor	To eliminate noises and ripples from power supply.
SD-Card Holder (9 Pins)	To hold SD-Card. For logging data purpose.
KBU8A Rectifier (400V, 8A)	To convert bipolar AC signal to unipolar signal.

3.3 Project Design

Project design will discuss about the software and hardware design of the project. Software design (refer section 3.3.1) will explains all detail on PIC programming. Under hardware, it will discuss about circuit simulation using Orcad PSpice (refer section 3.3.2.1), PIC circuit design (refer section 3.3.2.2), measurement circuit design (refer section 3.3.2.3) and casing development (refer section 3.3.2.4). Finally after all software and hardware components has been assembled the prototype was tested in laboratory (refer section 3.3.3).

3.3.1 Software: PIC Programming

The PIC program is written in C language using MicroC Pro software. This C program is called source code. This source code is compiled using MicroC Pro to produce a HEX file. This HEX file is downloaded by PIC using MICROCHIP PICkit 2 Programmer Software. After that the PIC is ready to be installed into power meter circuit. The program flow is discussed further in section 4.1 .

Table 3 PIC Programming Equipments

No	Equipment	Function
1	Mikroelectronica mikroC compiler software	This software is use to write and compile the source code.
2	MICROCHIP PICkit 2 Programmer Software	This software is use to download the generated HEX file into the PIC memory.
3	USB ICSP PIC Programmer V2010 and ICSP Programmer Socket.	Hardware based programmer. The PIC will be put into this programmer. This programmer is connected to a PC. HEX file from PC is transferred into PIC through this programmer.

3.3.2 Hardware

3.3.2.1 Simulation

The objective of this simulation is to compare the differences of input current waveforms of the DUT (induction motor) with and without measurement elements. The PSpice simulation has 2 parts; the first part is the modeled per-phase induction motor equivalent circuit operating (acting as DUT) without any metering component. The second part is the simulation of the same circuit but with added measurement components.

The opto-coupler (A4N25) used in the simulation is the nearest equivalent model for incoming opto-coupler that will be used in the project. A resistor 130 micro-ohm is used to represent the resistance of the hall-effect sensor conductor that will be used to measure current in the project.

The equipment needed is only the Orcad PSpice Schematic Software. The version used in this simulation is 9.1. The first simulation is done base on circuit in Figure 7 . The second simulation is done based on circuit in Figure 8 . The transient simulation was done for 3 seconds for both simulations. The simulation traces is taken according to list in Table 4 .

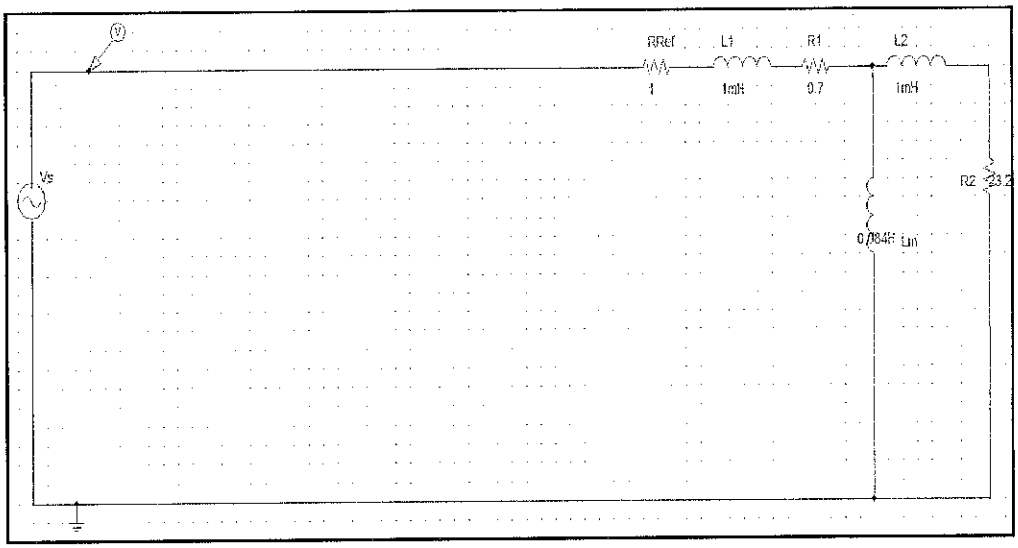


Figure 7 Part 1: Induction motor per-phase equivalent circuit

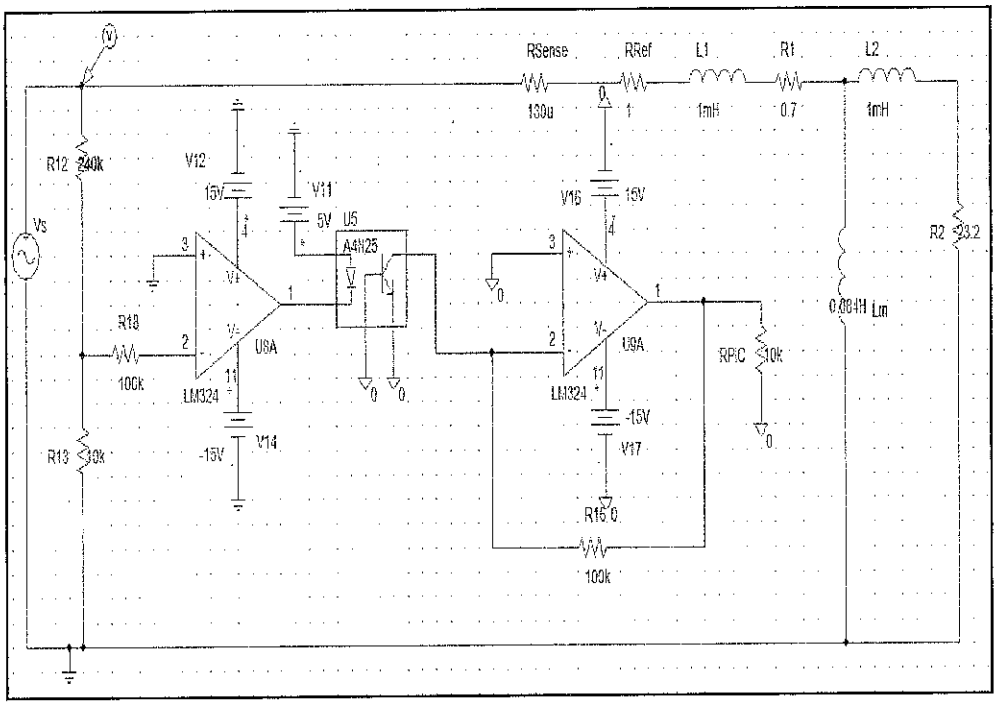


Figure 8 Part 2: Induction motor with measurement circuit

Table 4 Simulation Traces

Part	**Traces**
1	Current: RRef (Without measurement Circuit) Voltage: Load Voltage (Without measurement Circuit)
2	Current: RRef (With measurement Circuit) Voltage: Load Voltage (With measurement Circuit)

3.3.2.2 Main Circuit

Main circuit consists of PIC, SD card module, LCD display, A/D converter input channels and push buttons input. Table 6 shows the list of the equipment used to develop the main circuit. The circuit was assembled as in Figure 9 .

The main component of the PIC circuit is the PIC itself. The PIC need to be programmed before it is installed into the circuit. Refer section 3.3.1 for the programming part of the PIC. 24MHz crystal oscillator is selected as the PIC clock. The maximum frequency that PIC can support is up to 48MHz. The higher the clock frequency is better. This is because the frequency will determine the PIC instruction cycle speed. The faster the instruction speed cycle, the faster the instruction will be processed by PIC.

The main power supply for this circuit is 9V battery. The battery output voltage is regulated using LM7805 in order to produce 5V output. This 5V output voltage from the regulator is used to powers up the PIC.

The other voltage regulator, LM1117-T3.3 is used specifically for SD card module. LM1117-T3.3 will provide regulated 3.3V voltage supply to SD card module. This is because the voltage operating range for a SD card is 2.0V to 3.6V.

SD card holder is installed so that a SD can be inserted to it. This will let the PIC to communicate with the SD card using SPI protocol at port C (RC2 – RC5). This will support the logging feature for the meter. RC2, RC3 and RC5 are not connected directly to the SD card holder but through voltage divider system. The voltage divider is formed by 2.2kΩ and 3.3kΩ resistor. This will divide the signal voltage from PIC which is initially 5V to become 3.3V for high logic [15]. This will ensure that the SD card will receive signals within its operating range from PIC.

The Push Buttons is installed as the intermediate between user and PIC program. The push buttons will act as user command to the PIC. Four set of buttons act as directional buttons which consists of UP, DOWN, LEFT and RIGHT function. The fifth button act ac RETURN button. More details about the buttons are explained in section 3.3.1.

The main output is the LCD display which connected to port D of the PIC. The LCD will display meaningful and understandable outputs generated by PIC. The LCD is assigned to display desired output from user such as current, voltage, power, time, calibration and logging setup.

Table 6 Main Circuit Equipment List

No	**Equipment**	**Quantity**
1	PIC 18F4682	1
2	9 V Battery	1
3	Wires	-
4	Veroboard	1
5	24MHz Crystal Oscillator	1
6	2 Lines LCD Panel	1
7	Push Button	6
8	LED	1
9	LM7805 Voltage Regulator	1
10	LM1117-T3.3 Voltage Regulator	1
11	Resistors	Refer Figure 9
12	0.1uF Capacitor	4
13	SD Card Holder	1

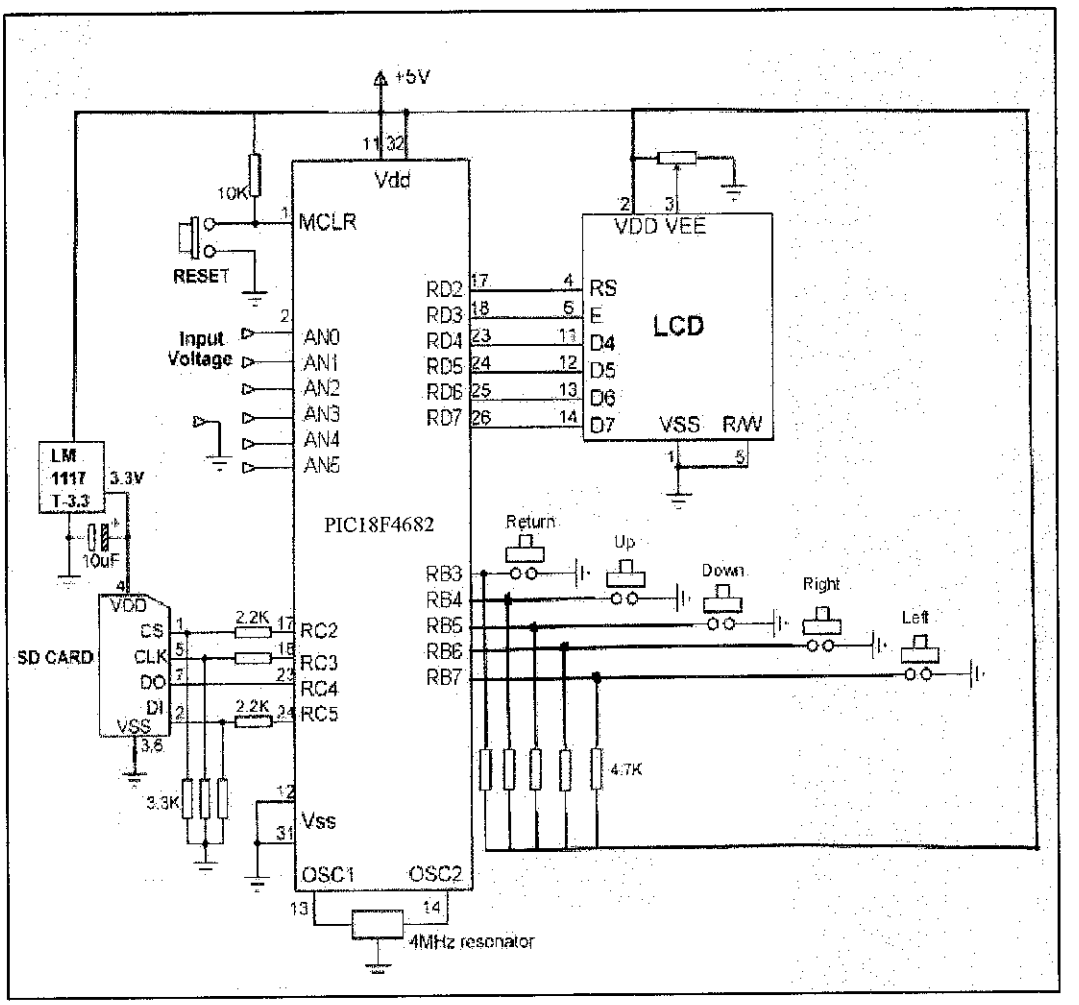


Figure 9 Main Circuit Diagram [15]

3.3.2.3 Measurement Circuit

The measurement circuit function is to provide isolated analog measurement of current and voltage in between on the power supply and the load. The measurement output of the sensors which is in form of analog signal is transmitter to PIC through PIC A/D channels. Table 7 shows the list of equipment that used to develop this circuit. Figure 10 shows the per phase circuit diagram for the measurement circuit.

The basic power measurement needs voltage and current as the parameter. Refer section 2.2 calculation derivation details. The measurement circuit consists of two different sensors. The sensors are HCPL7520 opto-coupler for voltage measurement and ACS712 Hall Effect sensor as current sensor. Both sensors provide isolation from high voltage and current to signaling side. This will ensure that both PIC circuit and the user are protected from over voltage or over current from power supply or load.

Current measurement principle is simple. The current from source is connected to ACS712 at pin no. 1 and no. 2. Then, the current will flow into the sensor conductor and flowing out and connect to load at pin no. 3 and no. 4. Refer section 2.3.2 for principle details. Output voltage is produce at pin no. 5 which is proportional to the current flow. This output voltage is connected to A/D channel of PIC.

Voltage measurement is possible by scaling down the high voltage from power supply by using voltage divide concept. The voltage divider is formed from a pair of resistors. However the scaled down output voltage from the voltage divider is not safe to enter to A/D channel of PIC because there is no isolation between it. Isolated linear analog opto-coupler HCPL7520 is used to solve this problem. The operating range for the HCPL7520 input voltage is 0 to 200mV. Since the input voltage from supply is an AC voltage, it needs to be rectified by using a bridge rectifier (KBU8A) so that the input voltage to the voltage divider network is in uni-polar. In order to ensure the voltage divider network gives 200mV voltage when the input supply voltage is at maximum, following calculations is done. Let the maximum supply voltage is 240V.

$$Vout = \frac{R2}{R1 + R2} \tag{3.1}$$

$$\frac{R2}{R1 + R2} = \frac{Vout}{Vmax} \tag{3.2}$$

$$\frac{R2}{R1 + R2} = \frac{200mV}{240V} \tag{3.3}$$

$$\frac{R2}{R1 + R2} = \frac{200}{240k} \tag{3.4}$$

Thus, R2 = 200Ω and R1 + R2 = 240kΩ. So, R1 is calculated as follows:

$$\begin{aligned} R1 &= 240k\Omega - 200\Omega \\ &= 239.8k\Omega \end{aligned} \tag{3.5}$$

However, there is no practical 239.8kΩ resistor. So, in this design the selected R1 value is 240kΩ. The HCPL7520 octo-coupler will produce an output voltage corresponding to the input voltage linearly. This output voltage is connected A/D channels of PIC.

The power supply for low side (output side) of ACS712 and HCPL7520 is from one LM7805 voltage regulator output which supplied by battery from PIC circuit in order to form a common ground between PIC and output of the measurement circuit. This common ground will complete the communication loop of A/D channels and measurement circuit output.

Meanwhile the power supply for high voltage side of the HCPL7520 opto-coupler is from one LM7805 voltage regulator. This voltage regulator is powered by a separate 9V battery. This to ensure that there will be no common point between PIC and high side of the measurement circuit. Thus it will provide complete isolation between the two circuits.

Since the project design and development is for three phase power system, the circuit in Figure 10 needs to be duplicated into three identical circuits on one board to measure phase A, phase B and phase C respectively.

Table 7 Measurement Circuit Equipment List

No	Equipment	Quantity
1	Agilent HCPL 7520 Opto-coupler	3
2	Allegro ACS712 Hall Effect Sensor	3
3	9 V Battery	1
4	Veroboard	1
5	LED	2
6	LM7805 Voltage Regulator	2
7	Resistors	Refer Figure 10
8	0.1uF Capacitor	4
9	KBU8A Rectifiers (400V, 8A)	3
10	Terminal Blocks	3

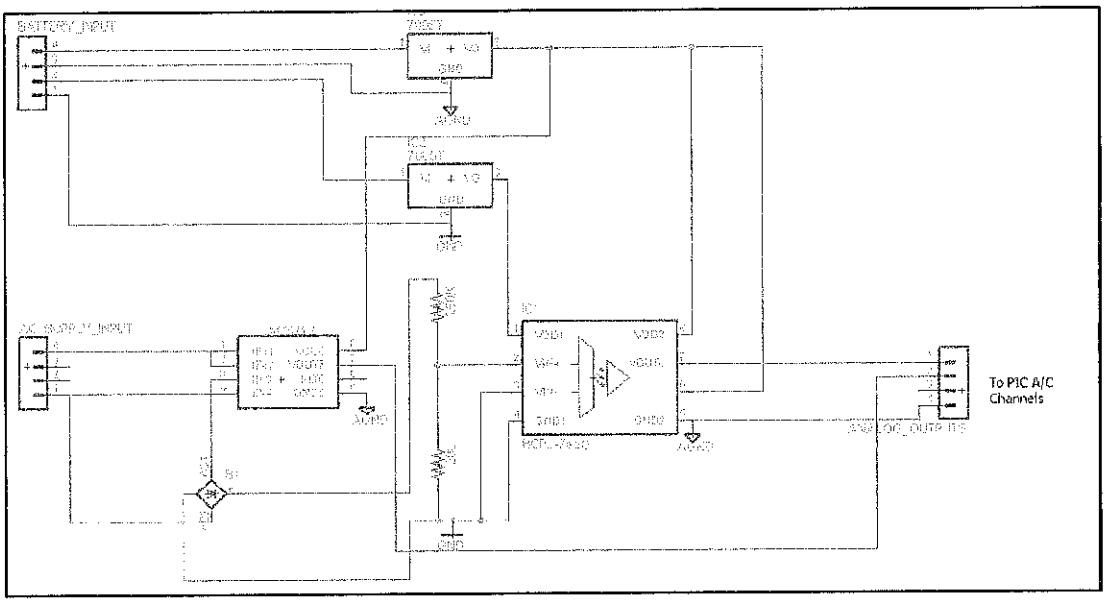


Figure 10 Measurement Circuit Diagram (Per-phase)

3.3.2.4 Test Prototype Casing Development

The completed circuits will be put into a casing to protect the user from touching high voltage and high current element in the circuit. The casing also provides the protection to the circuit from external element that may damage the circuit or cause short circuit. Table 8 shows the casing equipment list that use to develop the casing.

Table 8 Casing Equipment List

No	Equipment	Quantity
1	Plastic Case (15.0cm x 17.5cm x 5.0cm)	1
2	Plastic Case (6.5cm x 12.0cm x 4.0cm)	1
3	4.0 mm Sockets	7
4	Screws (M3)	8
5	Nut (M3)	8
6	Drill	1

3.3.3 Full Prototype Test

There are two type of test is done to the prototype. The first test is the functional test for all features available in the prototype. The second is the measurement test by the prototype on an actual load and three phase power supply in laboratory. Table 9 shows the list of equipments needed during the test. Table 10 shows the prototype features and its function.

The first test is done without connecting the prototype to the load or power supply. The tests are done by navigating to the each feature and see its response using the navigation buttons. Each feature should be doing its programmed function. If there is a feature that not functions, it is declared as logical error. Thus the PIC program needs to be analyzed to remove the error.

For the actual condition test, the prototype is connected as in Figure 11 . Figure 12 , Figure 13 and Figure 14 shows the actual equipment that used during test. The test is done by varying the supply voltage at three different levels (refer section 4.3 for the details). The measurement reading is taken from LVDAM-EMS software in computer and from prototype. Both measurement readings are compared. This comparison will determine the accuracy of the prototype. Refer section 4.3 for the result details.

Table 9 Equipment List for Full Prototype Test

No	Equipment	Quantity
1	Variable 3 phase power supply	1
2	3 phase synchronous motor	1
3	Wires	-
4	Prototype (Calibrated)	1
5	LVDAM-EMS (Version 3.0) Data Acquisition	1
6	LVDAM-EMS Data Acquisition Interface	1
7	Computer	1

Table 10 Prototype Features

No	Feature	Function
1	View	To view all measurement reading such as voltage, current and apparent power.
2	Calibration	To set sensors input relationship to the displayed output.
3	Time	To update and display current time and date. It is also necessary for logging feature time stamp.
4	Logging	To set the logging interval and log the reading taken by the prototype in to SD card.
5	Writing data into SD card	To write logged data into SD card through SPI communication protocol.
6	Writing configuration setting into EEPROM	To write and store all configurations into EEPROM. Stored data is logging setup, last operating time and calibration data.
7	Power Measurement	To measure power consumed by connected load or DUT.

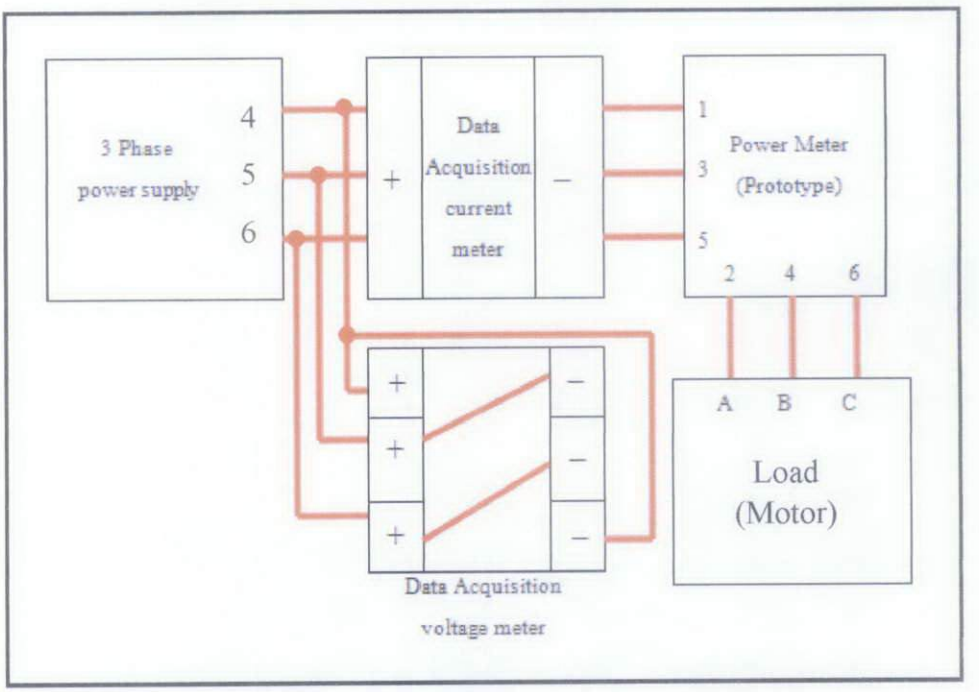


Figure 11 Prototype Test Setup

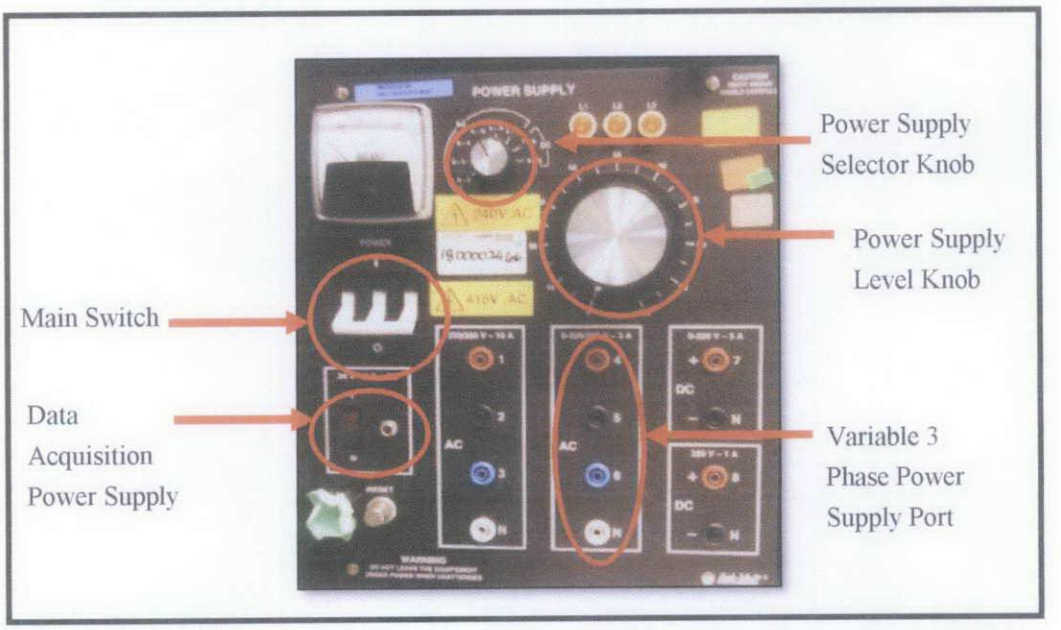


Figure 12 Power Supply

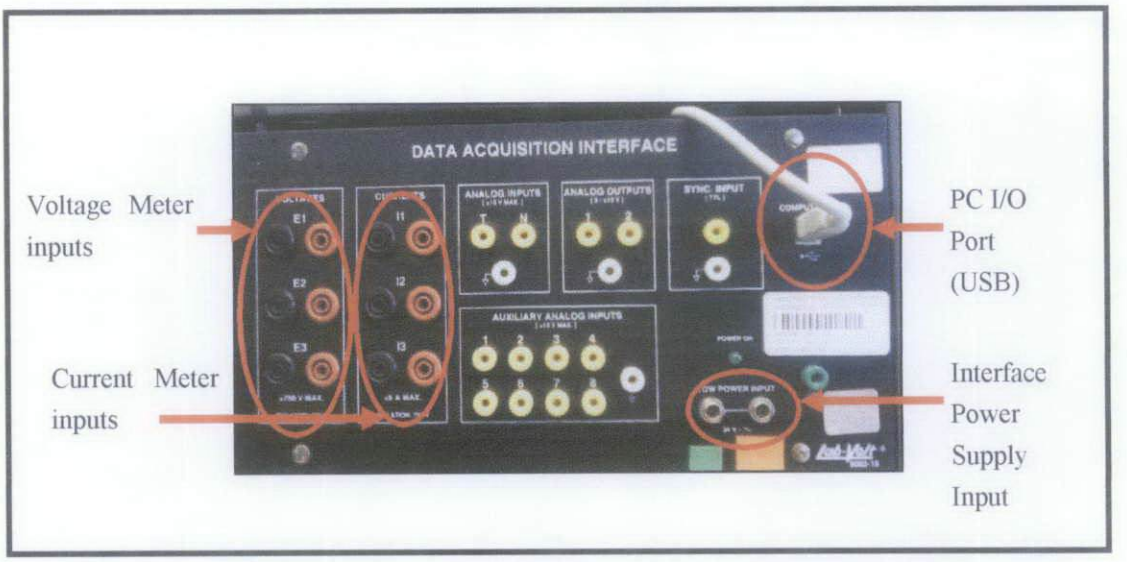


Figure 13 LVDAM-EMS Data Acquisition Interface

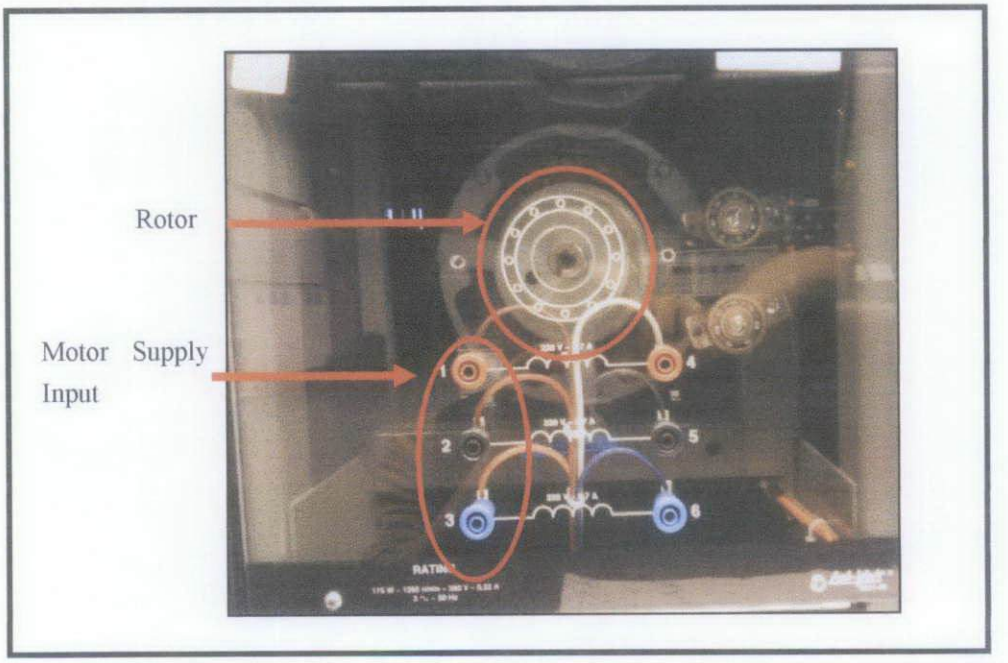


Figure 14 Four Pole 3 Phase Synchronous Motor

CHAPTER 4
RESULT AND DISCUSSION

4.1 Software: PIC Programming

This section will discuss about PIC programming result and its flow. Figure 15 shows the PDL language of the PIC program.

```
START
        Configure Port D as output
        Configure Port C as SPI module
        Configure Port A as input
        Configure Port B as Input
        Configure LCD
        Configure A/D converter
        Configure SPI Protocol
        Configure Timer0 interrupt
        Configure Data Logging
        Read configuration from EEPROM
        Configure Calibration Data
        Configure Current Time
        DO FOREVER
                Update Time
                Read analog data from channel 0, 1, 2, 3, 4 and 5.
                Calculate RMS values
                User's Directional Buttons Instruction.
                Check Log status and write to SD Card
                Go to associated MODE.
                Set User's Button direction value.
                Format the data according to display mode.
                Mode specific actions.
                Display the data on LCD.
                RETURN button instruction.
                Delay
        END
END
```

Figure 15 PDL Language of the PIC Program

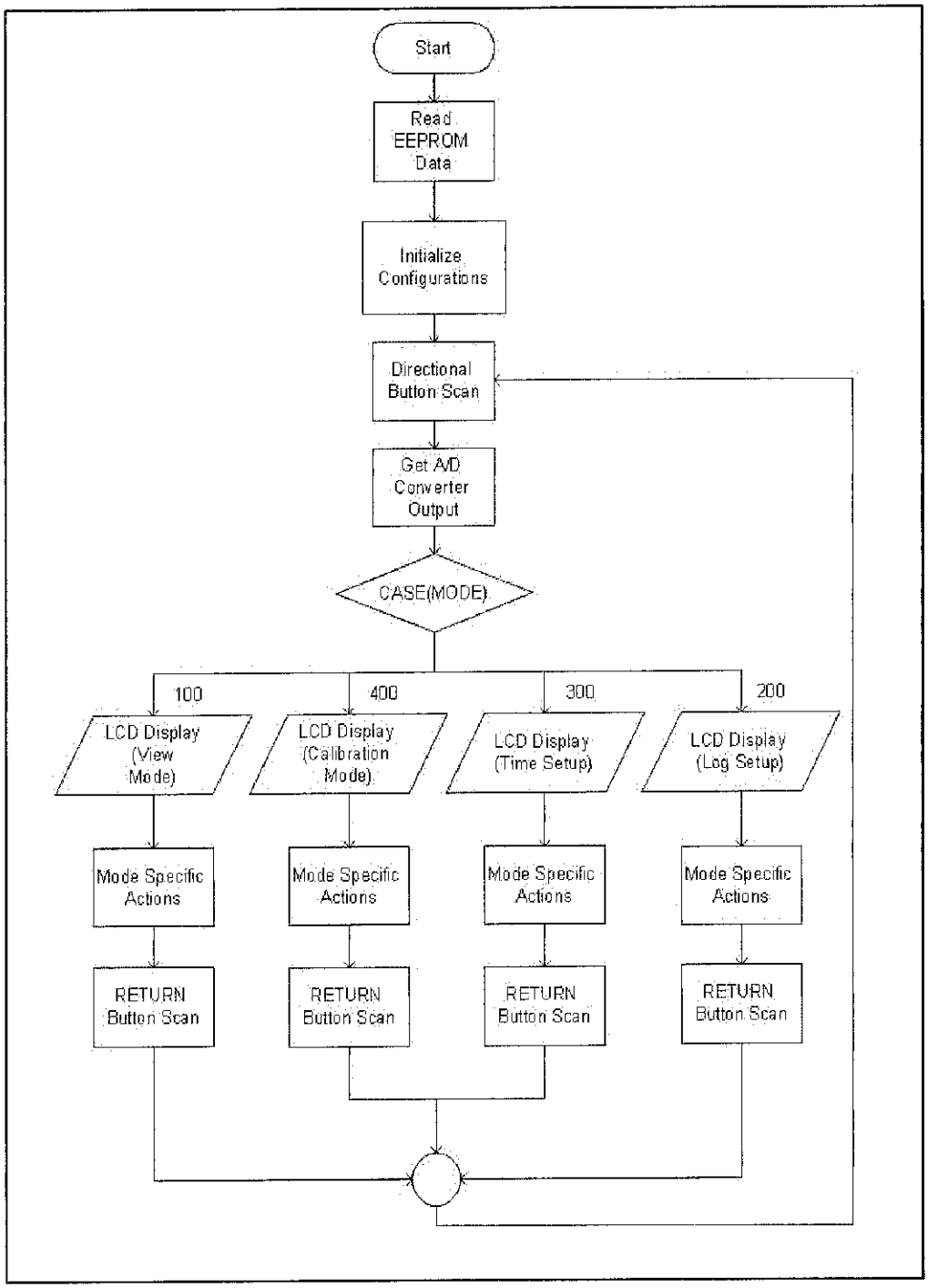


Figure 16 Program Flow

Figure 16 shows the actual program flow of the PIC program. From the program flow, there are 7 control strategies in the program. The control strategies are LCD display control, view mode control, log setup control, time setup control, calibration setup control, SD card module control, Timer Interrupt Control and EEPROM read and write control.

4.1.1 LCD Display Control

There are a lot of text and output to be displayed on the LCD. Thus a control strategy is planed so that the program can be written smoothly and less error. Basically each displayed text on the LCD is assigned to a unique ID which is called MODE ID. The ID consists of three digits. Table 11 shows the main ID distribution by modes. Based on the Table 11 , any display ID begins with 1 is under display mode. Second and third digit will represent other function based on their mode. For example, under View mode, if the ID is 11X, it means the display output is related to phase A values. For other IDs, refer Appendix C.

The MODE ID value is controlled by navigation buttons. So, the main function of the navigation button is basically is increasing or decreasing the MODE ID value.

Table 11 Display IDs

Mode	ID
View Mode	1XX
Log Setup Mode	2XX
Time Setup Mode	3XX
Calibration Setup Mode	4XX

4.1.2 View Mode Control

Under View Mode, it only displays the measurement readings that have been calculated during each program loop. It controls the display of voltage (V), current (I) and apparent power (S) measurement for all phases. It handles the value formatting such as adding floating point and converts it to suitable variable that LCD can display.

4.1.3 Log Setup Control

The log setup consists of three functions. The first function is to display the current logging setup such as log switch and log interval. The second function is the switch setup either to turn on or off the logging system. And finally is the function to set the logging interval in minutes. The PIC is programmed to be able to log from 1 minutes interval up to 240 minutes intervals. In order to control the display under this mode, the Up and Down button acting as directional button meanwhile Left and Right button act as value changer buttons. Left button used to decrease value meanwhile Right button used to increase value.

4.1.4 Time Setup Control

Under time setup mode, there are three main functions. The first function is to display the current time, the second function is to display current date and the last function is to set the time and date. Under this mode, Left and Right button act as directional button where capable to switch between functions. Meanwhile Up and Down button is used to change the value during time and date setup.

4.1.5 Calibration Setup Control

This mode will assign and convert the analog value from the sensors to its corresponding output value. The A/D converter is set to give value from 0 to 5000. For example, 0 is assigned to 0V meanwhile 5000 is assigned to 240V. By means of interpolation technique, the unknown corresponding voltage for A/D value between 0 and 5000 can be determined. The calibration mode is just assignment function only. The interpolation calculation is done by a special custom function called "calc_v", "calc_i" and "calc_s" where they will calculate for voltage, current and apparent power respectively. Under this mode, Left and Right button act as directional button where capable to switch between functions. Meanwhile Up and Down button is used to change the value during zero level and upper range level value trimming.

4.1.6 SD Card Module

SD card module consists of function to initialize the SD card prior to each logging intervals and the function of writing data into SD card. The SD card communication between PIC is done using SPI module that available at pin 3, 4 and 5 of port C in PIC. The function to initialization and writing is taken from the built in function in the MicroC Pro software. Table 12 shows the list of function that use to enable SD card module. Data writing into SD card is done at each logging intervals.

Table 12 SD Card Module Codes

Code	Function
Mmc_Fat_Init();	Initializes MMC/SD card, reads MMC/SD FAT16 boot sector and extracts necessary data needed by the library.
SPI1_Init_Advanced(MASTER_OSC_DIV64, DATA_SAMPLE_MIDDLE, CLK_IDLE_LOW, LOW_2_HIGH);	Configures and initializes SPI. SPI1_Init or SPI1_Init_Advanced needs to be called before using other functions of SPI Library.
unsigned short Mmc_Fat_Assign(char *filename, char file_cre_attr);	Assigns file for file operations (read, write, delete...). All subsequent file operations will be applied on an assigned file.
void Mmc_Fat_Write(char *fdata, unsigned data_len);	Writes requested number of bytes to the currently assigned file opened for writing.
void Mmc_Fat_Append();	Opens the currently assigned file for appending. Upon this function execution file pointers will be positioned after the last byte in the file, so any subsequent file write operation will start from there.

4.1.7 Timer Interrupt Control

The function of this interrupt is to update the time at every 1 seconds. This means at every one second, the interrupt routine will activate and update the current time. This is important for the logging system time stamping. The timer interrupt is enabled at TIMER0 of the PIC. The timer is controlled by assigning appropriate values into TMR0L and TMR0H registers. Following is the formula used to get the value:

$$Register\ Value = 65536 - \frac{Time}{4 \times \frac{1}{Oscillator} \times Prescaler} \tag{4.1}$$

$$Register\ Value = 65536 - \frac{1\ sec}{4 \times \frac{1}{24MHz} \times 128} = 18661 \tag{4.2}$$

4.1.8 EEPROM Read/Write Control

The EEPROM is used to store certain configuration of the power meter. The configurations are the time setup, log switch condition, log intervals, and calibration data. The PIC has 1024 bytes of data EEPROM. Each of the EEPROM addresses is capable to hold up to 8-bit of data. This means it can store decimal value from 0 to 254. However this data size is not sufficient for the power meter because the calibration data and year value is exceeding this range. So, a special control strategy needs to be done so that those data can be store into the EEPROM.

The solution taken is breaking up data in to small part (into 2 decimal digits) then store them separately into EEPROM. When the PIC wants to read the data from the EEPROM, the PIC will read the data from EEPROM addresses one by one and joint the data to form the original data. Thus the EEPROM data size limitation can be eliminated.

4.2 Hardware

This section will discuss on all hardware related design result. It consists of simulation result, PIC circuit assembly and test, measurement circuit assembly and test and casing assembly.

4.2.1 Simulation

Following are the simulations result that discussed from section 3.3.2.1 . The results consists of all traces waveform and the peak voltage comparison of all traces.

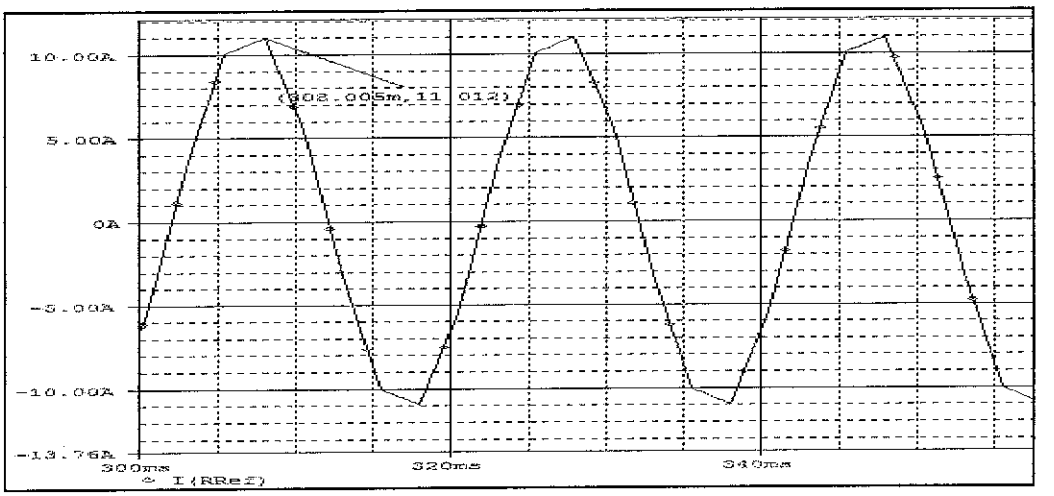


Figure 17 Current Waveform through R1

The Figure 17 shows the current waveform through R1 which is the reference current. This is the current flow to the load without any disturbance from prototype meter.

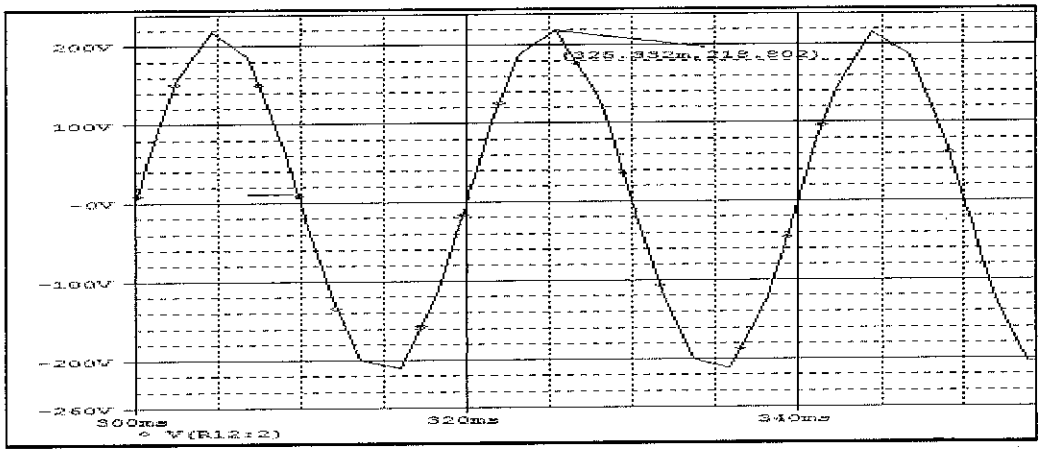


Figure 18 Load Voltage Waveform (Part 1)

Figure 18 shows the load voltage waveform across the load (DUT). This voltage waveform is the voltage drop across the load during normal condition where there is no disturbance from prototype.

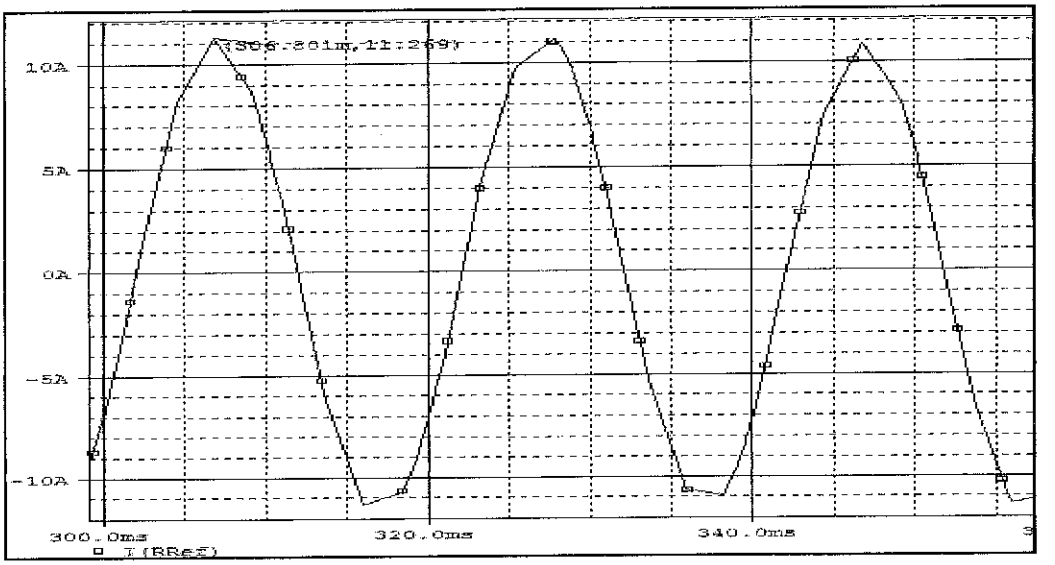


Figure 19 Current waveform at RRef (Part 2)

Figure 19 shows the current waveform across the RRef where the current across it represent the current flow into load. RRef represent the resistance of the Hall Effect current sensor conductor. This waveform peak will be compared with the waveform in Figure 17 . This waveform is produce when there is prototype connected in between the load and power supply.

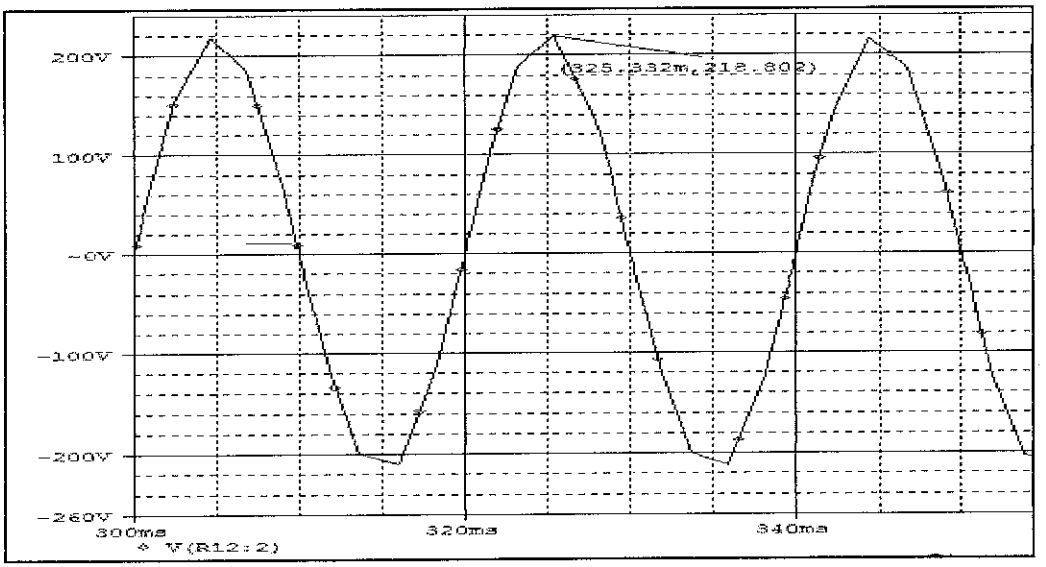


Figure 20 Load Voltage Waveform (Part 2)

Figure 20 shows the voltage waveform across the load. This waveform peak will be compared with the waveform in Figure 18 . This waveform is produce when there is prototype connected in between the load and power supply.

Table 13 Numerical Simulation Results

	Simulation Type		
	Without Measurement Circuit	**With Measurement Circuit**	**Error (%)**
Peak Load Voltage (V)	218.802	218.802	0%
Peak Load Current (A)	11.102	11.269	1.5%

Based on the results in Table 13 , the effect of the measurement circuit to the load current and load voltage is less than 2%. Usually in industry the acceptable error range is less than 5%. This means the simulation result is within the industrial requirements. In addition, the effect of prototype disturbance can be considered as insignificant.

4.2.2 Main Circuit

This section will discuss about the main circuit development and test result. Figure 21 shows the assembled PIC circuit with SD card module. In Table 14 is the test for A/D channels of the PIC to see its accuracy. The test is limited from 0V to 5V because the reference voltage for the A/D channel is from 0V to 5V only.

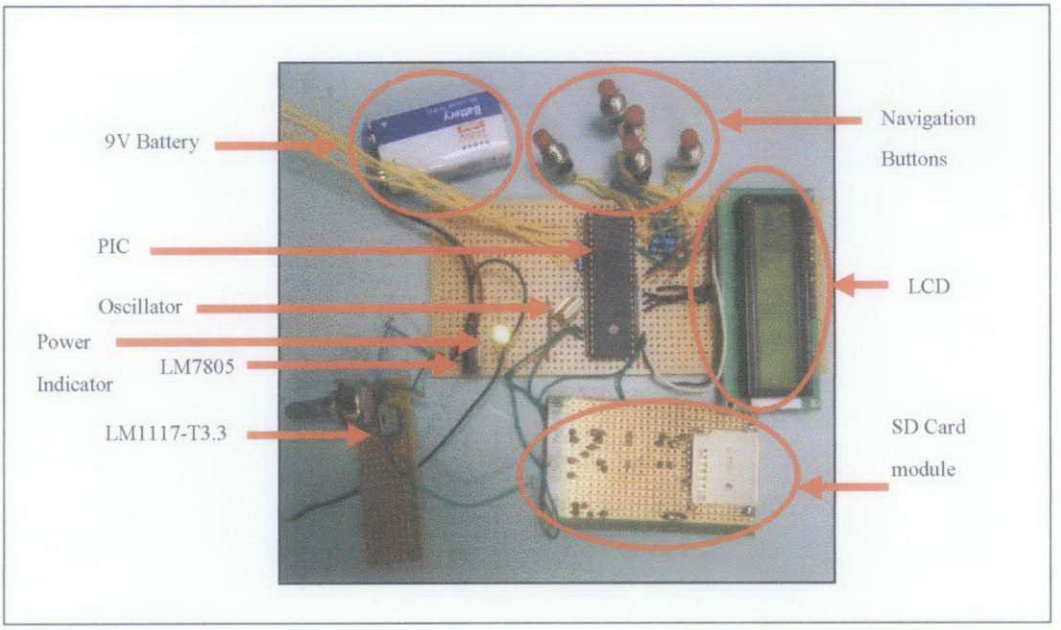


Figure 21 Main Circuit

Table 14 PIC A/D Measurement Result

Test Item	**Voltmeter(mV)**	**Project Prototype (mV)**	**% Difference (%)**
Battery 1.5 V	1549	1562	0.83925
Voltage Regulator Output (5 V)	5000	4995	0.1
LED voltage drop on the circuit	2030	2041	0.54187

Based on result on the Table 14 , the percentage different can consider very low which are less than 1% for all measurement. This means the built-in A/D converter of PIC can be consider precise enough to perform the measurement.

4.2.3 Measurement Circuit

This section shows the assembly result of the measurement circuit. Figure 22 shows the measurement circuit assembly. In the Figure 22 , there is no Hall Effect sensor label because the sensor is installed at the bottom side of the veroboard. The measurement circuit can be tested by using small DC supply. After being interfaced with PIC, the PIC need to be calibrated so that it can create input output relationship between measurement circuit output signals.

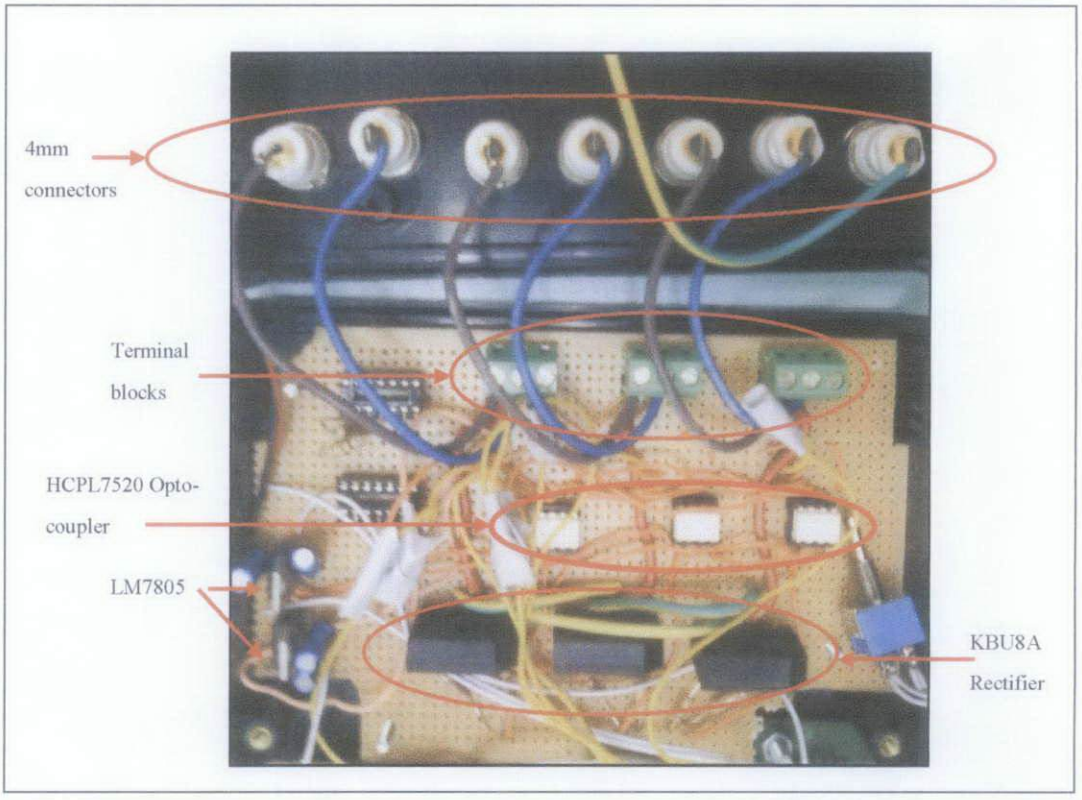


Figure 22 Measurement Circuit

4.2.4 Test Prototype Casing Development

Figure 23 shows the test prototype casing after development finished. The Figure 23 also shows the main parts at the prototype casing.

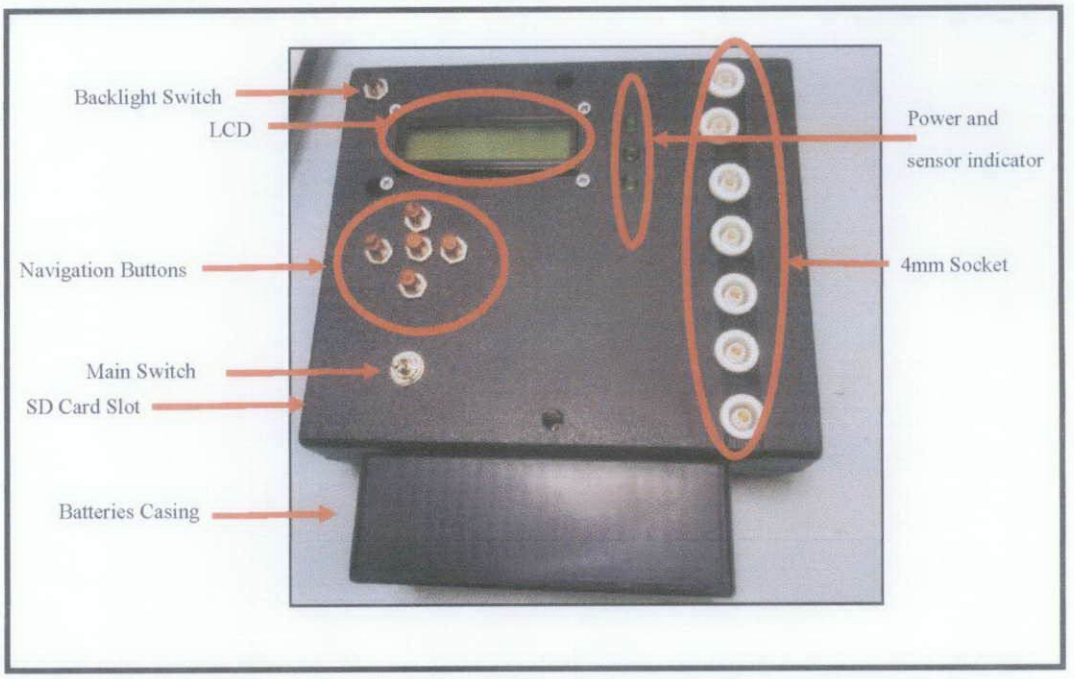


Figure 23 Test Prototype Casing

4.3 Full Prototype Test

This section discuss on full test on the test prototype. The test includes the feature tests and measurement tests which has be discussed in section 3.3.3 . Table 15 shows the test results for all prototype features. Table 16 , Table 17 and Table 18 shows the measurement test results for phase A, phase B and phase C respectively. PM in the tables represent power meter prototype and DA represent data acquisition interface meter value that retrieved from LVDAM-EMS software. Meanwhile Figure 24 shows the logged data into the SD card during the test. The data may include record before and during calibration and unset time stamp.

Table 15 Prototype Test By Features

No	Feature	Condition
1	View Mode	Pass
2	Time Setup	Pass
3	Calibration Setup	Pass
4	Logging Setup	Pass
5	Store Data In EEPROM	Pass
6	Data Logs In SD Card	Pass
7	Power Measurement	Pass

Based on Table 15 , all features that available in the prototype are working properly without any detectable bug or logical errors.

Table 16 Power Measurement Test (Phase A)

Phase A									
No	**Voltage (V)**			**Current (mA)**			**Apparent Power, S (VA)**		
	PM	DA	Error (%)	PM	DA	Error (%)	PM	DA	Error (%)
1	57.1	57.28	0.314246	0.12	0.13	7.692308	6.852	7.4464	7.982381
2	73.8	73.34	-0.62722	0.10	0.098	-2.04082	7.38	7.18732	-2.68083
3	105.34	106	0.622642	0.10	0.095	-5.26316	10.534	10.07	-4.60775

Table 16 shows the measurement result for phase A. Based on the Table 16 , the measurement error for voltage is low which is less than 1% meanwhile for current the error is a bit high (>5%).

Table 17 Power Measurement Test (Phase B)

Phase B									
No	**Voltage (V)**			**Current (mA)**			**Apparent Power, S (VA)**		
	PM	DA	Error (%)	PM	DA	Error (%)	PM	DA	Error (%)
1	57.36	57.04	-0.56101	0.12	0.129	6.976744	6.8832	7.35816	6.454875
2	73.16	73.68	0.705755	0.10	0.099	-1.0101	7.316	7.29432	-0.29722
3	106.17	106.18	0.009418	0.10	0.095	-5.26316	10.617	10.0871	-5.25324

Table 17 shows the measurement result for phase B. Based on the Table 16 , the measurement error for voltage is low which is less than 1% meanwhile for current the error is a bit high (>5%).

Table 18 Power Measurement Test (Phase C)

Phase C									
No	**Voltage (V)**			**Current (mA)**			**Apparent Power, S (VA)**		
	PM	DA	Error (%)	PM	DA	Error (%)	PM	DA	Error (%)
1	57.3	57.36	0.104603	0.12	0.129	6.976744	6.876	7.39944	7.074049
2	73.2	73.74	0.732303	0.1	0.099	-1.0101	7.32	7.30026	-0.2704
3	105.95	106.5	0.516432	0.1	0.095	-5.26316	10.595	10.1175	-4.71955

Table 18 shows the measurement result for phase C. Based on the Table 16 , the measurement error for voltage is low which is less than 1% meanwhile for current the error is a bit high (>5%).

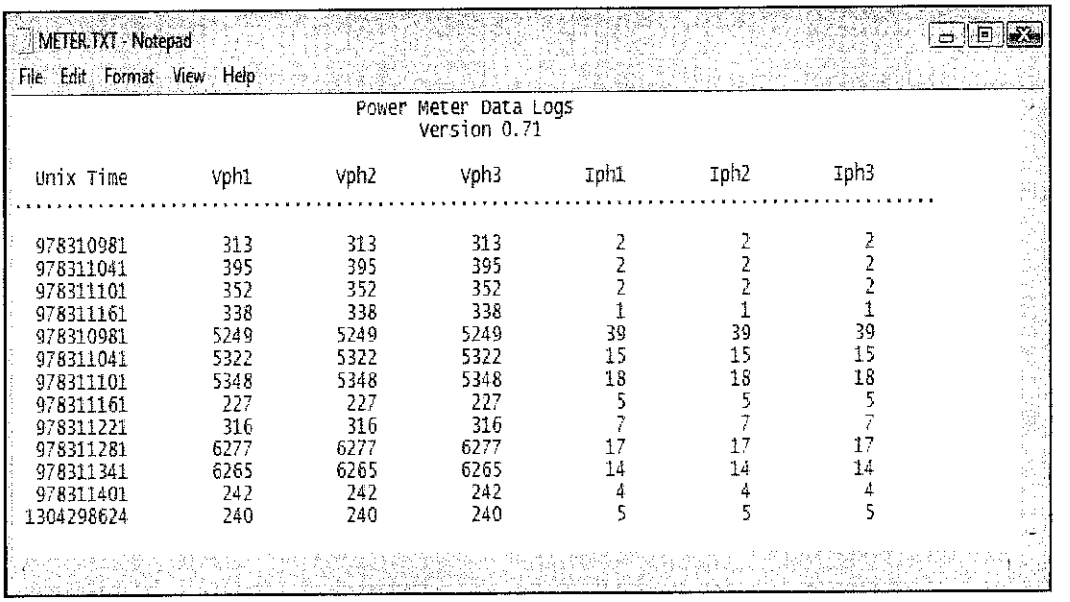


Figure 24 Logged Data from SD Card

Based on Figure 24 , the logged data is stored in SD card in text format with file name of METER.TXT. This file can be open using Notepad software that embedded in most of Windows operating system. The time stamp is recorded using EPOCH time format. EPOCH is a UNIX operating system time stamp. Thus, converter software is needed to convert the date into normal format. Other data are recorded without decimal points. Thus, all readings need to be multiplies with 10^{-2}.

CHAPTER 5
CONCLUSION AND RECOMMENDATION

5.1 Conclusion

The project carried out manages to achieve the project objective where to measure 3 phase power parameters such as voltages, currents and apparent powers with protective isolation integrated with Programmable Interface Controller (PIC).

As the conclusion, at power meter prototype works properly without noticeable logical bugs. However it will be working slightly slow if the RMS measurement is done. Sometimes the reading fluctuates due to noise and negative value. This can be eliminated by adding capacitors and calculation improvement in PIC program. In addition, the power meter is limited to measure up to 240V (phase to phase), 5 A and 1200 W.

Currently, the power meter is capable to measure three phase voltage, current and apparent power. Since the power meter input is instantaneous voltage and instantaneous current, real power, reactive power, power factor and frequency can be measure in the future.

5.2 Recommendations

Based on current project progress, some recommendations can be implemented to improve the project. The recommendations are:

- Do calibration with larger range for more precise reading.
- Use different sensor for different measurement range (Based on sensors sensitivity).
- Add filters to reduce noise between A/D converter and output from sensor.

REFERENCES

[1] Jishun Jiang and Lanlan Yu, "Design of a New Three-phase Multi-rate Watt-hour Meter Based on AT89S52,"isid, *2009 Second International Symposium on Computational Intelligence and Design,*vol. 1, pp. 416-419, 2009.

[2] Jishun Jiang; Lanlan Yu; "Design of a New Three-Phase Multi-Rate Watt-Hour Meter Based on Singlechip," *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on* , vol., no., pp.1-4, 11-13 Dec. 2009

[3] Pasdar, A.; Mirzakuchaki, S.; "Three phase power line balancing based on smart energy meters," *EUROCON 2009, EUROCON '09. IEEE* , vol., no., pp.1876-1878, 18-23 May 2009

[4] Shun-Yu Chan; Shang-Wen Luan; Jen-Hao Teng; Ming-Chang Tsai; , "Design and implementation of a RFID-based power meter and outage recording system," *Sustainable Energy Technologies, 2008. ICSET 2008. IEEE International Conference on* , vol., no., pp.750-754, 24-27 Nov. 2008

[5] Zheng Wenzheng; "Design and implementation on wireless power meter system based on GSM network," *Computer, Mechatronics, Control and Electronic Engineering (CMCE), 2010 International Conference on* , vol.2, no., pp.76-79, 24-26 Aug. 2010

[6] Chatterjee, A.; Sarkar, G.; Rakshit, A.; , "A Reinforcement-Learning-Based Fuzzy Compensator for a Microcontroller-Based Frequency Synthesizer/Vector Voltmeter", *Instrumentation and Measurement, IEEE Transactions on Instrumentation and Measurement*, vol.PP, no.99, pp.1-8, 2011

[7] LLC, P. P. (n.d.). *Measure AC and DC Current Amps using a Hall Effect Current Sensor Clamp*. Retrieved August 15, 2010, from Free Math Physics Engineering Lessons Simulations Freeware Shareware: http://sciencesshareware.com/how-to-measure-AC-DC-current-with-a-hall-effect-clamp-.htm

[8] LLC, P. P. (n.d.). *How to measure current and power using current sense resistor*. Retrieved August 15, 2010, from Free Math Physics Engineering

Lessons Simulations Freeware Shareware: http://sciencesshareware.com/bg-current-monitoring.htm

[9] *Optocoupler.* (2011). Retrieved April 26, 2011, from Electronic Repair Guide: http://www.electronicrepairguide.com/optocoupler.html

[10] PITE TECH. INC. (2009). *PITE.Tech.* Retrieved September 29, 2010, from http://pitetech.com/product_show.asp?bid=3&id=11

[11] Connection, T. E. (n.d.). *Reed Instruments KEW6300-03.* Retrieved August 15, 2010, from Test Equiptment Connection: http://www.testequipmentconnection.com/products/39773

[12] Saadat, H. (2002). *Power System Analysis* (2nd ed., ch. 2, pp. 14-42). McGraw Hill Primis Custom Publishing.

[13] *RF Cafe - RMS Voltage and Average Power Equation Formula.* (2011). Retrieved January 25, 2011, from RF Cafe: http://www.rfcafe.com/references/electrical/rms.htm

[14] UNSW. (2011). *AC Power, RMS and 3 Phase Circuits.* Retrieved January 25, 2011, from PhysClips: http://www.animations.physics.unsw.edu.au/jw/power.html

[15] Ibrahim, D. (2008). *Advanced PIC Microcontroller Projects in C From USB to ZIGBEE with the PIC 18F Series* . (pp. 1 – 2). Elsevier Ltd.

[16] Octopart. (2011). *PIC18F4682 I/P - Microchip - PIC18F4682IP.* Retrieved January 25, 2011, from Octopart Base: http://octopart.com/pic18f4682-i%2Fp-microchip-486747

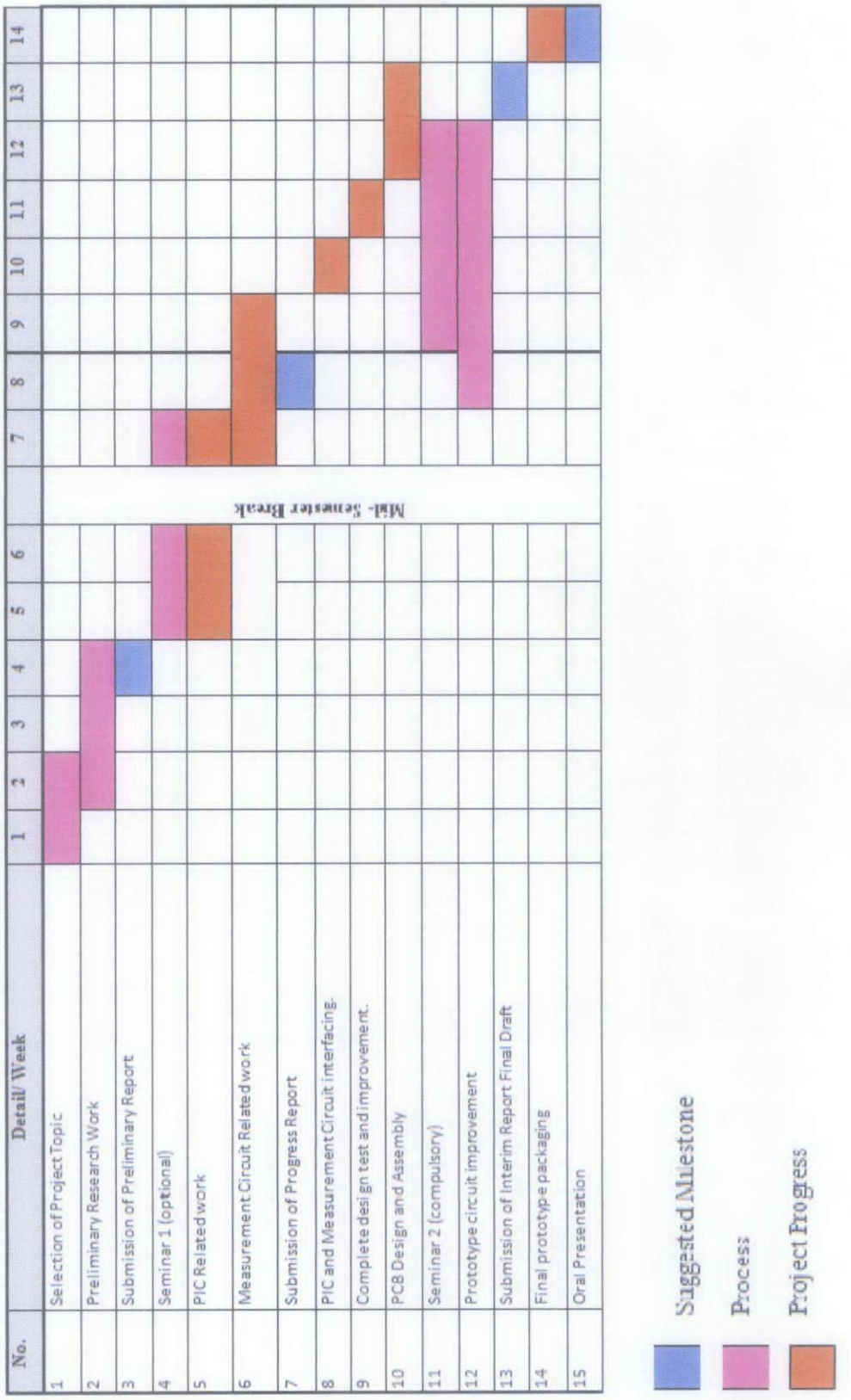
[17] Allegro MicroSystems, Inc. (2006). *Fully Integrated, Hall Effect-Based Linear Current Sensor with 2.1 kVRMS Voltage Isolation and a Low-Resistance Current Conductor.* (pp.1). Worcester, Massachusetts, U.S.A.

[18] Agilent Technologies, Inc. (2003, July 8). *Agilent HCPL-7520 Isolated Linear Sensing IC Data Sheet.* (pp.1).

APPENDICES

APPENDIX A

GANTT CHART FOR FYP 1



APPENDIX B

GANTT CHART FOR FYP 2

No.	Detail/ Week	1	2	3	4	5	6	7		8	9	10	11	12	13	14
1	Features addition and improvement into PIC program: - View Mode - Log Setup - Time Setup - Calibration Setup - Save configurations into EEPROM								Mid- Semester Break							
2	Test Sensor circuits															
3	Add SD-Card Module															
4	Submission of Progress Report															
5	Circuit Testing in Lab															
6	Final checking and modification															
7	PCB design and fabrication															
8	Casing design and fabrication															
9	Logging Data Reader Software Design															
10	Final prototype assembly															
11	Poster Exhibition															
12	Submission of Dissertation (soft bound)															
13	Oral Presentation															
14	Submission of Project Dissertation (Hard Bound)															

APPENDIX C

PIC SOURCE CODE

```
#include "timelib.h" // Time library header file

sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D7 at RD7_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D4 at RD4_bit;

// Pin direction For LCD
sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D7_Direction at TRISD7_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D4_Direction at TRISD4_bit;

// Pin direction for SPI Module
sfr sbit Mmc_Chip_Select at RC2_bit;
sfr sbit Mmc_Chip_Select_Direction at TRISC2_bit;

unsigned int mode=100; // Diplays mode variable
unsigned int uf,df,rf,lf; // Navigation buttons direction variable
unsigned long epoch;//Epoch time variable
unsigned short maxday; //Maximum day in a month variable holder
timestruct time; //Time variable

//Sensors calibration data
unsigned int usensv[3];
unsigned int lsensv[3];
unsigned int uactualv[3];
unsigned int lactualv[3];
unsigned int usensi[3];
unsigned int lsensi[3];
unsigned int uactuali[3];
unsigned int lactuali[3];
// Log variables
unsigned short logsw; // Logger switch
unsigned short int logint; // log interval
unsigned int logcounter; //Log conuter. Converted log intervals in seconds

unsigned char datatowrite[12]; // Variable holder for SD card datas
unsigned long av1, av2, av3, ai1, ai2, ai3; //A/D variables
signed double v1, v2, v3, i1, i2, i3; //RMS voltage and current
unsigned short mmc_stat; //SD card status (inserted or not)

//EEPROM Write function
void ee_write_byte(unsigned char address, unsigned short input){

  EEDATA = input;
  EEADR = address;
  // start write sequence as described in datasheet, page 91
  EECON1.EEPGD = 0;
  EECON1.CFGS = 0;
  EECON1.WREN = 1; // enable writes to data EEPROM
  INTCON.GIE = 0; // disable interrupts
  EECON2 = 0x55;
  EECON2 = 0x0AA;
  EECON1.WR = 1; // start writing
  while(EECON1.WR);
  PIR2.EEIF=0;
  EECON1.WREN = 0;
  INTCON.GIE = 1; // enable interrupts
}
// EEPROM Read function
void ee_read_byte(unsigned char address, unsigned char *_data){
  EEADR = address;
  EECON1.CFGS = 0;
  EECON1.EEPGD = 0;
  EECON1.RD = 1;
  *_data = EEDATA;
}
```

```
unsigned long calc_v(unsigned short phase, unsigned long
av) // voltage calculation based on calibration
{       signed long result,voltage,m;
        voltage = (av * 5000) >> 10;
        m=(uactualv[phase-1]-lactualv[phase-1]) * (voltage-
lsensv[phase-1]);
        result= (m / (usensv[phase-1]-lsensv[phase-1]));
        result= result + lactualv[phase-1];
        return result;
}

unsigned long calc_i(unsigned short phase, unsigned long
ai) // current calculation based on calibration
{       signed long result,current,m;
        current = (ai * 5000) >> 10;
        m=(uactuali[phase-1]-lactuali[phase-1]) * (current-
lsensi[phase-1]);
        result= (m / (usensi[phase-1]-lsensi[phase-1]));
        result = result + lactuali[phase-1];
        return result;
}

unsigned long calc_s(unsigned short phase, unsigned long
av, unsigned long ai) // S calculation based on calibration
{       signed long result,voltage,current;
        voltage=calc_v(phase,av);
        current=calc_i(phase,ai);
        result=voltage * current;
        result=result/100;
        return result;
}

void disp_long(unsigned long var) //Format converter (to
convert data so it is compatible with LCD)
{
 unsigned int i,j,a,b;
 unsigned char op[12],lcd[6];
 a=var/100;
 b=(var%100);
 LongToStr(a,op);              // Convert to string in "op"
 j=0;
 for(i=0;i<=11;i++)
 {
  if(op[i] != ' ')             // If a blank
  {
   lcd[j]=op[i];
   j++;
  }
 }
 Lcd_Out(2,1,lcd);   // Output to LCD
 LongToStr(b,op);              // Convert to string in "op"
 j=0;
 for(i=0;i<=11;i++)
 {
  if(op[i] != ' ')             // If a blank
  {
   lcd[j]=op[i];
   j++;
  }
 }
 if (b<10) lcd_out_cp(".0");
 else lcd_out_cp(".");
 lcd_out_cp(lcd);
}

void disp_long_cp(unsigned long var) //Floating point
display converter to LCD
{
 unsigned int i,j;
 unsigned char op[12],lcd[6];
 LongToStr(var,op);            // Convert to string in "op"
 j=0;
 for(i=0;i<=11;i++)
 {
  if(op[i] != ' ')             // If a blank
  {
   lcd[j]=op[i];
   j++;
  }
 }
 Lcd_Out_cp(lcd);   // Output to LCD
}

//Special EEPROM write handles
void sp_eeprom_write(unsigned short int start_address,
unsigned short int address_block, unsigned int user_data)
{
   unsigned short int temp;
   for(address_block=address_block;address_block>0;
address_block--)
   {
       temp=user_data%100;
       ee_write_byte(start_address + address_block - 1,
temp);
       user_data=user_data/100;
```

```
    }
}
//Special EEPROM read handles
unsigned int sp_eeprom_read(unsigned short int
start_address, unsigned short int address_block)
{
   unsigned short int temp,i;
   unsigned long result;
   result = 0;
   for(i = 0; i < address_block ; i++)
   {
      result=result * 100;
      ee_read_byte(start_address + i, &temp);
      result = result + temp;
   }
   return result;
}
//Log write to SD
unsigned log_write(void)
{
        logcounter=logint * 60;
        if(!Mmc_Init() && !Mmc_Fat_Init())
        {
         mmc_stat=Mmc_Fat_Assign("meter.txt",1);
         if (!mmc_stat)
         {
           Mmc_Fat_Assign("meter.txt",0x80);
           Mmc_Fat_Append();

           Mmc_Fat_Write("                              Power
Meter Data Logs\r\n",56);
           Mmc_Fat_Write("
Version 0.8\r\n\r\n",54);
           Mmc_Fat_Write("   Unix  Time          Vph1
Vph2      Vph3      Iph1      Iph2      Iph3",83);
           Mmc_Fat_Write("\r\n",2);

Mmc_Fat_Write("..........
..........\r\n\r\n",93);

        }

           Mmc_Fat_Append();
           longtostr(epoch,datatowrite);

           Mmc_Fat_Write(datatowrite,11);
           Mmc_Fat_Write(" ",1);

           longtostr(v1,datatowrite);
           Mmc_Fat_Write(datatowrite,11);

           Mmc_Fat_Write(" ",1);

           longtostr(v2,datatowrite);
           Mmc_Fat_Write(datatowrite,11);

           Mmc_Fat_Write(" ",1);

           longtostr(v3,datatowrite);
           Mmc_Fat_Write(datatowrite,11);

           Mmc_Fat_Write(" ",1);

           longtostr(i1,datatowrite);
           Mmc_Fat_Write(datatowrite,11);

           Mmc_Fat_Write(" ",1);

           longtostr(i2,datatowrite);
           Mmc_Fat_Write(datatowrite,11);

           Mmc_Fat_Write(" ",1);

           longtostr(i3,datatowrite);
           Mmc_Fat_Write(datatowrite,11);

           Mmc_Fat_Write("\r\n",2);

        }

}
//Timer interrupt (1 seconds)
void interrupt()
{
 if (INTCON.TMR0IF) {
  TMR0L = 0xE5;
  TMR0H = 0x48;
  INTCON.TMR0IF = 0;
  epoch++;
  time_epochtodate(epoch,&time);
  if (logcounter>0)
  {
   logcounter=logcounter-1;
  }
```

```
  }

}

// Main Function
void main()
{
    unsigned short bklight;
    unsigned char op[12];
    unsigned char i,j,lcd[6];
    unsigned int sample;
    unsigned long var1, counter;
    signed long actual_val;
    signed long temp_val[3];
    unsigned long sense[3];
    unsigned int my_delay;
    unsigned short vselect;

//IO Initializations
    PORTD = 0;
    LATD = 0;
    CMCON = 0x07;   //Disable comparator
    TRISD = 0;                  // PORTD are outputs
(LCD)

    TRISA = 0xFF;               // PORTA is input
    LATA = 0;
    TRISB = 0xFD;
    LATB = 0;
//Timer and interrupt initialization
    T0CON = 0x86;
    TMR0L = 0xE5;
    TMR0H = 0x48;
    INTCON = 0xA0;


    //  Configure LCD
    Lcd_Init();
    Lcd_Cmd(_LCD_CLEAR);
    PortB.F1=1;  //Turn On Backlight During Startup
    Lcd_Out(1,1,"POWERMETER-FYP2");
    Lcd_Out(2,1,"Version 0.8");

    //Initialize Configuration from eeprom
    ee_read_byte(0x00, &logsw);
    ee_read_byte(0x01, &logint);
    ee_read_byte(0x10, &time.hh);
    ee_read_byte(0x11, &time.mn);
    ee_read_byte(0x12, &time.ss);
    ee_read_byte(0x13, &time.md);
    ee_read_byte(0x14, &time.mo);
    time.yy = sp_eeprom_read(0x15, 2);

    //Get Calibration data from eeprom
    lsensv[0]=sp_eeprom_read(0x20, 2);
    lsensv[1]=sp_eeprom_read(0x22, 2);
    lsensv[2]=sp_eeprom_read(0x24, 2);
    lactualv[0]=sp_eeprom_read(0x26, 3);
    lactualv[1]=sp_eeprom_read(0x29, 3);
    lactualv[2]=sp_eeprom_read(0x2C, 3);

    usensv[0]=sp_eeprom_read(0x30, 2);
    usensv[1]=sp_eeprom_read(0x32, 2);
    usensv[2]=sp_eeprom_read(0x34, 2);
    uactualv[0]=sp_eeprom_read(0x36, 3);
    uactualv[1]=sp_eeprom_read(0x39, 3);
    uactualv[2]=sp_eeprom_read(0x3C, 3);

    lsensi[0]=sp_eeprom_read(0x40, 2);
    lsensi[1]=sp_eeprom_read(0x42, 2);
    lsensi[2]=sp_eeprom_read(0x44, 2);
    lactuali[0]=sp_eeprom_read(0x46, 3);
    lactuali[1]=sp_eeprom_read(0x49, 3);
    lactuali[2]=sp_eeprom_read(0x4C, 3);

    usensi[0]=sp_eeprom_read(0x50, 2);
    usensi[1]=sp_eeprom_read(0x52, 2);
    usensi[2]=sp_eeprom_read(0x54, 2);
    uactuali[0]=sp_eeprom_read(0x56, 3);
    uactuali[1]=sp_eeprom_read(0x59, 3);
    uactuali[2]=sp_eeprom_read(0x5C, 3);

    //Initialize time in epoch format
    epoch=time_datetoepoch(&time);

    //Configure A/D module
    ADCON1 = 0x09;                  // Use AN0 and
Vref=+5V

    //Initialize Log Counter
    logcounter=logint * 60;
```

```
    //SPI Module Initialize

SPI1_Init_Advanced(_SPI_MASTER_OSC_DIV16,_SPI
_DATA_SAMPLE_MIDDLE,_SPI_CLK_IDLE_LOW,_
SPI_LOW_2_HIGH);


    //Inintialization Delay
    delay_ms(2000);


    //Initialize Backlight
    bklight=0;
    PortB.F1=0;


    //  Program loop
    for(;;)
    {
//Navigation button function
      if (Button(&PORTB, 4, 1, 0)) mode=mode+uf;
      if (Button(&PORTB, 5, 1, 0)) mode=mode-df;
      if (Button(&PORTB, 6, 1, 0)) mode=mode-lf;
      if (Button(&PORTB, 7, 1, 0)) mode=mode+rf;


//Backlight button switch
      if (Button(&PORTB, 2, 1, 0))
      {
       if (bklight==0)
       {
        bklight=1;
        PortB.F1=1;
        while(Button(&PORTB, 2, 1, 0));
       }
       else
       {
        bklight=0;
        PortB.F1=0;
        while(Button(&PORTB, 2, 1, 0));
       }
      }
     //  RMS Calculation
      v1 = 0;
      v2 = 0;
      v3 = 0;
      i1 = 0;
      i2 = 0;
      i3 = 0;
      for(sample=0;sample<50;sample++)
      {
      av1 = ADC_Read(0);
      av2 = ADC_Read(1);
      av3 = ADC_Read(2);
      ai1 = ADC_Read(3);
      ai2 = ADC_Read(4);
      ai3 = ADC_Read(5);
      v1= v1 + pow(calc_v(1,av1),2);
      v2= v2 + pow(calc_v(2,av2),2);
      v3= v3 + pow(calc_v(3,av3),2);
      i1= i1 + pow(calc_i(1,ai1),2);
      i2= i2 + pow(calc_i(2,ai2),2);
      i3= i3 + pow(calc_i(3,ai3),2);
      }
      v1 = v1 / 50;
      v2 = v2 / 50;
      v3 = v3 / 50;
      i1 = i1 / 50;
      i2 = i2 / 50;
      i3 = i3 / 50;
      v1 = sqrt(v1);
      v2 = sqrt(v2);
      v3 = sqrt(v3);
      i1 = sqrt(i1);
      i2 = sqrt(i2);
      i3 = sqrt(i3);


     if (logcounter==0 && logsw==1) log_write(); // Log
counting. If zero write to SD card.


     switch (mode) // Modes Matrix selection
     {
     case 100: //View Mode
           uf=100;
           df=-300;
           rf=10;
           lf=-10;
           Lcd_Cmd(_LCD_CLEAR);
           Lcd_Out(1,1,"VIEW");
           Lcd_Out(2,1,"MODE");
           break;
     case 200: //Log Setup
           uf=100;
           df=100;
           rf=10;
           lf=-10;
           Lcd_Cmd(_LCD_CLEAR);
```

```
        Lcd_Out(1,1,"LOGGING");
        Lcd_Out(2,1,"SETUP");
        break;
case 300: //Time Setup
        uf=100;
        df=100;
        rf=10;
        lf=-10;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"TIME");
        Lcd_Out(2,1,"SETUP");
        break;
case 400: // Calibration Setup
        uf=-300;
        df=100;
        rf=10;
        lf=-10;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"CALIBRATION");
        Lcd_Out(2,1,"SETUP");
        break;
// Phase 1 Measurement
case 110: //S1
        uf=10;
        df=-20;
        rf=1;
        lf=-2;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"S1 = ");
        var1=calc_s(1,av1,ai1);
        disp_long(var1);
        Lcd_Out_cp("VA");
        if (Button(&PORTB, 3, 1, 0)) mode=100;
        break;
case 111: //V1
        uf=10;
        df=-20;
        rf=1;
        lf=1;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"V1 = ");
        //var1=calc_v(1,av1);
        disp_long(v1);
        Lcd_Out_cp("V");
        if (Button(&PORTB, 3, 1, 0)) mode=100;
        break;
case 112: //I1
        uf=10;
        df=-20;
        rf=-2;
        lf=1;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"I1 = ");
        //var1=calc_i(1,ai1);
        disp_long(i1);
        Lcd_Out_cp("A");
        if (Button(&PORTB, 3, 1, 0)) mode=100;
        break;
//PHASE 2
case 120: //S2
        uf=10;
        df=10;
        rf=1;
        lf=-2;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"S2 = ");
        var1=calc_s(2,av2,ai2);
        disp_long(var1);
        Lcd_Out_cp("VA");
        if (Button(&PORTB, 3, 1, 0)) mode=100;
        break;
case 121: //V2
        uf=10;
        df=10;
        rf=1;
        lf=1;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"V2 = ");
        //var1=calc_v(2,av2);
        disp_long(v2);
        Lcd_Out_cp("V");
        if (Button(&PORTB, 3, 1, 0)) mode=100;
        break;
case 122: //I2
        uf=10;
        df=10;
        rf=-2;
        lf=1;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"I2 = ");
        //var1=calc_i(2,ai2);
        disp_long(i2);
```

```
        Lcd_Out_cp("A");
        if (Button(&PORTB, 3, 1, 0)) mode=100;
        break;
//Phase 3
case 130: //S3
        uf=-20;
        df=10;
        rf=1;
        lf=-2;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"S3 = ");
        var1=calc_s(3,av3,ai3);
        disp_long(var1);
        Lcd_Out_cp("VA");
        if (Button(&PORTB, 3, 1, 0)) mode=100;
        break;
case 131: //V3
        uf=-20;
        df=10;
        rf=1;
        lf=1;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"V3 = ");
        //var1=calc_v(3,av3);
        disp_long(v3);
        Lcd_Out_cp("V");
        if (Button(&PORTB, 3, 1, 0)) mode=100;
        break;
case 132: //I3
        uf=-20;
        df=10;
        rf=-2;
        lf=1;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"I3 = ");
        //var1=calc_i(3,ai3);
        disp_long(i3);
        Lcd_Out_cp("A");
        if (Button(&PORTB, 3, 1, 0)) mode=100;
        break;

// Log Setup
case 210: //Status
        uf=20;
        df=-10;
        rf=0;
        lf=0;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Log Status:");
        if (logsw==1)
        {
         Lcd_Out(2,1,"SW:ON  INT:");
         disp_long_cp(logint);
        }
        else Lcd_Out(2,1,"SW:OFF");

        if (Button(&PORTB, 3, 1, 0))
        {
          ee_write_byte(0x01, logint);
          mode=200;
        }
        break;
case 220: //Log switch (on/off)
        uf=-10;
        df=-10;
        rf=0;
        lf=0;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Set Log Switch:");
        if (Button(&PORTB, 6, 1, 0))
        {
         logsw=0;
         ee_write_byte(0x00, 0x00);
        }
        if (Button(&PORTB, 7, 1, 0))
        {
         logsw=1;
         ee_write_byte(0x00, 0x01);
        }
        if (logsw==1) Lcd_Out(2,1,"ON");
        else Lcd_Out(2,1,"OFF");

        if (Button(&PORTB, 3, 1, 0))
        {
          ee_write_byte(0x01, logint);
          mode=200;
        }
        break;
case 230: //Log interval setup
        uf=-10;
        df=20;
        rf=0;
```

```
            lf=0;
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Set Interval:");

            my_delay=200;
            while (Button(&PORTB, 6, 1, 0) &&
(logint)>1)
            {
              Lcd_Cmd(_LCD_CLEAR);
              Lcd_Out(1,1,"Set Interval:");
              if (my_delay>50) my_delay=my_delay-10;
              else
              {
               my_delay=50;
              }
              logint--;
              Lcd_Cmd(_LCD_RETURN_HOME);
              Lcd_Cmd(_LCD_SECOND_ROW);
              disp_long_cp(logint);
              Vdelay_ms(my_delay);
            }
            my_delay=200;
            while (Button(&PORTB, 7, 1, 0) &&
(logint)<240)
            {
              Lcd_Cmd(_LCD_CLEAR);
              Lcd_Out(1,1,"Set Interval:");
              if (my_delay>50) my_delay=my_delay-10;
              else
              {
               my_delay=50;
              }
              logint++;
              Lcd_Cmd(_LCD_RETURN_HOME);
              Lcd_Cmd(_LCD_SECOND_ROW);
              disp_long_cp(logint);
              Vdelay_ms(my_delay);
            }

            Lcd_Cmd(_LCD_RETURN_HOME);
            Lcd_Cmd(_LCD_SECOND_ROW);
            disp_long_cp(logint);
            if (Button(&PORTB, 3, 1, 0))
            {
              ee_write_byte(0x01, logint);
              mode=200;
            }
            break;

          //Time Setup
          case 310: //Display time
            uf=0;
            df=-10;
            rf=0;
            lf=0;
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Current Time:");
            Lcd_Cmd(_LCD_RETURN_HOME);
            Lcd_Cmd(_LCD_SECOND_ROW);
            disp_long_cp(time.hh);
            LCD_Out_cp(":");
            disp_long_cp(time.mn);
            LCD_Out_cp(":");
            disp_long_cp(time.ss);
            if (Button(&PORTB, 3, 1, 0)) mode=300;
            break;
          case 320: //Display date
            uf=-10;
            df=-10;
            rf=0;
            lf=0;
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Current Date:");
            Lcd_Cmd(_LCD_RETURN_HOME);
            Lcd_Cmd(_LCD_SECOND_ROW);
            disp_long_cp(time.md);
            LCD_Out_cp("/");
            disp_long_cp(time.mo);
            LCD_Out_cp("/");
            disp_long_cp(time.yy);
            if (Button(&PORTB, 3, 1, 0)) mode=300;
            break;
          case 330: //Set hour
            uf=0;
            df=-10;
            rf=0;
            lf=0;
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Set Hour (24h):");

            while(Button(&PORTB, 6, 1, 0))
            {
            if (time.hh>0) time.hh--;
```

```
            else if (time.hh==0) time.hh=23;
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Set Hour (24h):");
            Lcd_Cmd(_LCD_RETURN_HOME);
            Lcd_Cmd(_LCD_SECOND_ROW);
            disp_long_cp(time.hh);
            epoch = Time_dateToEpoch(&time);
            delay_ms(100);
            }
            while(Button(&PORTB, 7, 1, 0))
            {
            if (time.hh<23) time.hh++;
            else if (time.hh==23) time.hh=0;
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Set Hour (24h):");
            Lcd_Cmd(_LCD_RETURN_HOME);
            Lcd_Cmd(_LCD_SECOND_ROW);
            disp_long_cp(time.hh);
            epoch = Time_dateToEpoch(&time);
            delay_ms(100);
            }
            Lcd_Cmd(_LCD_RETURN_HOME);
            Lcd_Cmd(_LCD_SECOND_ROW);
            disp_long_cp(time.hh);

            time.ss=0;
            epoch = Time_dateToEpoch(&time);
            break;
        case 340: //Set minutes
            uf=-10;
            df=-10;
            rf=0;
            lf=0;
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Set Minutes:");

            while(Button(&PORTB, 6, 1, 0) ||
Button(&PORTB, 7, 1, 0))
            {
            if ((Button(&PORTB, 6, 1, 0))&& time.mn>0)
time.mn--;
            else if ((Button(&PORTB, 6, 1, 0))&&
time.mn==0) time.mn=59;
            if ((Button(&PORTB, 7, 1, 0))&&
time.mn<59) time.mn++;
            else if ((Button(&PORTB, 7, 1, 0))&&
time.mn==59) time.mn=0;
            epoch = Time_dateToEpoch(&time);
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Set Minutes:");
            Lcd_Cmd(_LCD_RETURN_HOME);
            Lcd_Cmd(_LCD_SECOND_ROW);
            disp_long_cp(time.mn);
            delay_ms(100);
            }
            Lcd_Cmd(_LCD_RETURN_HOME);
            Lcd_Cmd(_LCD_SECOND_ROW);
            disp_long_cp(time.mn);

            time.ss=0;
            epoch = Time_dateToEpoch(&time);
            break;
        case 350: //Set Year
            uf=-10;
            df=-10;
            rf=0;
            lf=0;
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Set Year:");

            while(Button(&PORTB, 6, 1, 0) ||
Button(&PORTB, 7, 1, 0))
            {
            if ((Button(&PORTB, 6, 1, 0))&&
time.yy>2011) time.yy--;
            else if ((Button(&PORTB, 6, 1, 0))&&
time.yy==2011) time.yy=2038;
            if ((Button(&PORTB, 7, 1, 0))&&
time.yy<2038) time.yy++;
            else if ((Button(&PORTB, 7, 1, 0))&&
time.yy==2038) time.yy=2011;

            if (time.yy%4==0 && time.mo==2)
maxday=29;
            else if (time.yy%4>0 && time.mo==2)
maxday=28;
            else if (time.mo==4 || time.mo==6 ||
time.mo==9 || time.mo==11) maxday=30;
            else maxday=31;

            if (time.md>maxday) time.md=maxday;
            epoch = Time_dateToEpoch(&time);
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Set Year:");
            Lcd_Cmd(_LCD_RETURN_HOME);
            Lcd_Cmd(_LCD_SECOND_ROW);
            disp_long_cp(time.yy);
```

```
        delay_ms(100);
        }

        Lcd_Cmd(_LCD_RETURN_HOME);
        Lcd_Cmd(_LCD_SECOND_ROW);
        disp_long_cp(time.yy);

        time.ss=0;
        epoch = Time_dateToEpoch(&time);
        break;
    case 360: //Set month
        uf=-10;
        df=-10;
        rf=0;
        lf=0;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Set Month:");

        while(Button(&PORTB, 6, 1, 0) ||
Button(&PORTB, 7, 1, 0))
        {
        if ((Button(&PORTB, 6, 1, 0))&& time.mo>1)
time.mo--;
        else if ((Button(&PORTB, 6, 1, 0))&&
time.mo==1) time.mo=12;
        if ((Button(&PORTB, 7, 1, 0))&&
time.mo<12) time.mo++;
        else if ((Button(&PORTB, 7, 1, 0))&&
time.mo==12) time.mo=1;

        if (time.yy%4==0 && time.mo==2)
maxday=29;
        else if (time.yy%4>0 && time.mo==2)
maxday=28;
        else if (time.mo==4 || time.mo==6 ||
time.mo==9 || time.mo==11) maxday=30;
        else maxday=31;

        if (time.md>maxday) time.md=maxday;
        epoch = Time_dateToEpoch(&time);
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Set Month:");
        Lcd_Cmd(_LCD_RETURN_HOME);
        Lcd_Cmd(_LCD_SECOND_ROW);
        disp_long_cp(time.mo);
        delay_ms(100);
        }
        Lcd_Cmd(_LCD_RETURN_HOME);
        Lcd_Cmd(_LCD_SECOND_ROW);
        disp_long_cp(time.mo);

        time.ss=0;
        epoch = Time_dateToEpoch(&time);
        break;
    case 370: //Set day
        uf=-10;
        df=-10;
        rf=0;
        lf=0;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Set Day:");

        if (time.yy%4==0 && time.mo==2)
maxday=29;
        else if (time.yy%4>0 && time.mo==2)
maxday=28;
        else if (time.mo==4 || time.mo==6 ||
time.mo==9 || time.mo==11) maxday=30;
        else maxday=31;

        while(Button(&PORTB, 6, 1, 0) ||
Button(&PORTB, 7, 1, 0))
        {
        if ((Button(&PORTB, 6, 1, 0))&& time.md>1)
time.md--;
        else if ((Button(&PORTB, 6, 1, 0))&&
time.md==1) time.md=maxday;
        if ((Button(&PORTB, 7, 1, 0))&&
time.md<maxday) time.md++;
        else if ((Button(&PORTB, 7, 1, 0))&&
time.md==maxday) time.md=1;
        epoch = Time_dateToEpoch(&time);
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Set Day:");
        Lcd_Cmd(_LCD_RETURN_HOME);
        Lcd_Cmd(_LCD_SECOND_ROW);
        disp_long_cp(time.md);
        delay_ms(100);
        }
        Lcd_Cmd(_LCD_RETURN_HOME);
        Lcd_Cmd(_LCD_SECOND_ROW);
        disp_long_cp(time.md);

        time.ss=0;
        epoch = Time_dateToEpoch(&time);
        break;
    case 380: //Save time setup to EEPROM
```

```
        uf=0;
        df=0;
        rf=0;
        lf=0;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Saving...");
        delay_ms(1000);

        sp_eeprom_write(0x10,1,time.hh);
        sp_eeprom_write(0x11,1,time.mn);
        sp_eeprom_write(0x12,1,time.ss);
        sp_eeprom_write(0x13,1,time.md);
        sp_eeprom_write(0x14,1,time.mo);
        sp_eeprom_write(0x15,2,time.yy);

        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Complete!");
        delay_ms(1000);
        mode=310;
        break;

//Calibration Setup
case 410: //Voltage Calibration
        uf=0;
        df=-10;
        rf=20;
        lf=-10;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Voltage Sensor");
        Lcd_Out(2,1,"Calibration");
        if (Button(&PORTB, 3, 1, 0)) mode=400;
        break;
case 420: //Current Calibration
        uf=-10;
        df=0;
        rf=30;
        lf=-20;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Current Sensor");
        Lcd_Out(2,1,"Calibration");
        if (Button(&PORTB, 3, 1, 0)) mode=400;
        break;

//Zero Voltage Level Calibration
case 430:
        uf=0;
        df=-10;
        rf=1;
        lf=0;
        vselect=0;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Voltage Calib:");
        Lcd_Out(2,1,"Zero Level");
        if (Button(&PORTB, 3, 1, 0)) mode=410;
        break;
case 431:
        uf=0;
        df=0;
        rf=1;
        lf=0;
        switch(vselect)
        {
        case 0:
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Apply 0V to V1");
            Lcd_Out(2,1,"then press >");
            break;
        case 1:
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Apply 0V to V2");
            Lcd_Out(2,1,"then press >");
            break;
        case 2:
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Apply 0V to V3");
            Lcd_Out(2,1,"then press >");
            break;
        }
        if (Button(&PORTB, 3, 1, 0)) mode=430;
        break;
case 432:
        uf=0;
        df=0;
        rf=0;
        lf=0;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Wait until");
        Lcd_Out(2,1,"reading stable");
        delay_ms(1500);
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"then press >");
        delay_ms(1000);
```

```
        mode=433;
        break;
    case 433:
        uf=0;
        df=0;
        rf=1;
        lf=0;
        Lcd_Cmd(_LCD_CLEAR);
        switch(vselect)
        {
         case 0:
             actual_val=calc_v(1,av1);
             sense[0]=(av1 * 5000) >> 10;;
             break;
         case 1:
             actual_val=calc_v(2,av2);
             sense[1]=(av2 * 5000) >> 10;;
             break;
         case 2:
             actual_val=calc_v(3,av3);
             sense[2]=(av3 * 5000) >> 10;;
             break;
        }
        Lcd_Out(1,1,"Applied Voltage:");
        disp_long(actual_val);
        Lcd_Out_cp("V");
        if (Button(&PORTB, 3, 1, 0)) mode=430;
        break;
    case 434:
        uf=0;
        df=0;
        rf=1;
        lf=0;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Adjust Reading:");
        my_delay=200;
        counter=1;
        while (Button(&PORTB, 4, 300, 0) &&
(actual_val)<24000)
        {
          Lcd_Cmd(_LCD_CLEAR);
          Lcd_Out(1,1,"Adjust Reading:");
          if (my_delay>10) my_delay=my_delay-10;
          else
          {
           my_delay=200;
           counter=counter*10;
           actual_val=actual_val/counter;
           actual_val=actual_val * counter;
          }
          actual_val=actual_val+counter;
          if (actual_val>24000) actual_val=24000;
          disp_long(actual_val);
          Lcd_Out_cp("V");
          Vdelay_ms(my_delay);
        }
        my_delay=200;
        counter=1;
        while (Button(&PORTB, 5, 300, 0) &&
(actual_val)>0)
        {
          Lcd_Cmd(_LCD_CLEAR);
          Lcd_Out(1,1,"Adjust Reading:");
          if (my_delay>10) my_delay=my_delay-10;
          else
          {
           my_delay=200;
           counter=counter*10;
           actual_val=actual_val/counter;
           actual_val=actual_val * counter;
          }
          actual_val=actual_val-counter;
          if (actual_val<0) actual_val=0;
          disp_long(actual_val);
          Lcd_Out_cp("V");
          Vdelay_ms(my_delay);
        }
        disp_long(actual_val);
        Lcd_Out_cp("V");
        if (Button(&PORTB, 3, 1, 0)) mode=430;
        break;
    case 435:
        uf=0;
        df=0;
        rf=0;
        lf=0;
        switch(vselect)
        {
         case 0:
         case 1:
             temp_val[vselect]=actual_val;
             vselect++;
```

```
            mode=431;
            break;
        case 2:
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Please Wait");
            lsensv[0]=sense[0];
            lsensv[1]=sense[1];
            lsensv[2]=sense[2];
            lactualv[0]=temp_val[0];
            lactualv[1]=temp_val[1];
            lactualv[2]=temp_val[2];
            Lcd_Out(2,1,"Saving....");

            sp_eeprom_write(0x20,2,lsensv[0]);
            sp_eeprom_write(0x22,2,lsensv[1]);
            sp_eeprom_write(0x24,2,lsensv[2]);
            sp_eeprom_write(0x26,3,lactualv[0]);
            sp_eeprom_write(0x29,3,lactualv[1]);
            sp_eeprom_write(0x2C,3,lactualv[2]);

            delay_ms(2000);
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Complete");
            delay_ms(1000);
            mode=430;
            break;
    }
    break;

//Upper Voltage Level Calibration
case 440:
    uf=-10;
    df=0;
    rf=1;
    lf=0;
    vselect=0;
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Out(1,1,"Voltage Calib:");
    Lcd_Out(2,1,"Upper Level");
    if (Button(&PORTB, 3, 1, 0)) mode=410;
    break;
case 441:
    uf=0;
    df=0;
    rf=1;
    lf=0;
    switch(vselect)
    {
     case 0:
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Apply URV to V1");
        Lcd_Out(2,1,"then press >");
        break;
     case 1:
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Apply URV to V2");
        Lcd_Out(2,1,"then press >");
        break;
     case 2:
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Apply URV to V3");
        Lcd_Out(2,1,"then press >");
        break;
     }
    if (Button(&PORTB, 3, 1, 0)) mode=440;
    break;
case 442:
    uf=0;
    df=0;
    rf=0;
    lf=0;
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Out(1,1,"Wait until");
    Lcd_Out(2,1,"reading stable");
    delay_ms(1500);
    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Out(1,1,"then press >");
    delay_ms(1000);
    mode=443;
    break;
case 443:
    uf=0;
    df=0;
    rf=1;
    lf=0;
    Lcd_Cmd(_LCD_CLEAR);
    switch(vselect)
    {
     case 0:
        actual_val=calc_v(1,av1);
        sense[0]=(av1 * 5000) >> 10;;
        break;
```

```
            case 1:
                actual_val=calc_v(2,av2);
                sense[1]=(av2 * 5000) >> 10;;
                break;
            case 2:
                actual_val=calc_v(3,av3);
                sense[2]=(av3 * 5000) >> 10;;
                break;
            }
            Lcd_Out(1,1,"Applied Voltage:");
            disp_long(actual_val);
            Lcd_Out_cp("V");
            if (Button(&PORTB, 3, 1, 0)) mode=440;
            break;
        case 444:
            uf=0;
            df=0;
            rf=1;
            lf=0;
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Adjust Reading:");
            my_delay=200;
            counter=1;
            while (Button(&PORTB, 4, 300, 0) &&
(actual_val)<24000)
            {
              Lcd_Cmd(_LCD_CLEAR);
              Lcd_Out(1,1,"Adjust Reading:");
              if (my_delay>10) my_delay=my_delay-10;
              else
              {
               my_delay=200;
               counter=counter*10;
               actual_val=actual_val/counter;
               actual_val=actual_val * counter;
              }
              actual_val=actual_val+counter;
              if (actual_val>24000) actual_val=24000;
              disp_long(actual_val);
              Lcd_Out_cp("V");
              Vdelay_ms(my_delay);
            }
            my_delay=200;
            counter=1;
            while (Button(&PORTB, 5, 300, 0) &&
(actual_val)>0)
            {
              Lcd_Cmd(_LCD_CLEAR);
              Lcd_Out(1,1,"Adjust Reading:");
              if (my_delay>10) my_delay=my_delay-10;
              else
              {
               my_delay=200;
               counter=counter*10;
               actual_val=actual_val/counter;
               actual_val=actual_val * counter;
              }
              actual_val=actual_val-counter;
              if (actual_val<0) actual_val=0;
              disp_long(actual_val);
              Lcd_Out_cp("V");
              Vdelay_ms(my_delay);
            }

            disp_long(actual_val);
            Lcd_Out_cp("V");
            if (Button(&PORTB, 3, 1, 0)) mode=440;
            break;
        case 445:
            uf=0;
            df=0;
            rf=0;
            lf=0;
            switch(vselect)
            {
             case 0:
             case 1:
                temp_val[vselect]=actual_val;
                vselect++;
                mode=441;
                break;
             case 2:
                Lcd_Cmd(_LCD_CLEAR);
                Lcd_Out(1,1,"Please Wait");
                usensv[0]=sense[0];
                usensv[1]=sense[1];
                usensv[2]=sense[2];
                uactualv[0]=temp_val[0];
                uactualv[1]=temp_val[1];
                uactualv[2]=temp_val[2];
                Lcd_Out(2,1,"Saving....");

                sp_eeprom_write(0x30,2,usensv[0]);
```

```
            sp_eeprom_write(0x32,2,usensv[1]);
            sp_eeprom_write(0x34,2,usensv[2]);
            sp_eeprom_write(0x36,3,uactualv[0]);
            sp_eeprom_write(0x39,3,uactualv[1]);
            sp_eeprom_write(0x3C,3,uactualv[2]);

            delay_ms(2000);
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Complete");
            delay_ms(1000);
            mode=440;
            break;
        }
        break;

//Zero Current Level Calibration
case 450:
        uf=0;
        df=-10;
        rf=1;
        lf=0;
        vselect=0;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Current Calib:");
        Lcd_Out(2,1,"Zero Level");
        if (Button(&PORTB, 3, 1, 0)) mode=420;
        break;
case 451:
        uf=0;
        df=0;
        rf=1;
        lf=0;

        switch(vselect)
        {
         case 0:
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Apply 0A to I1");
            Lcd_Out(2,1,"then press >");
            break;
         case 1:
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Apply 0A to I2");
            Lcd_Out(2,1,"then press >");
            break;
         case 2:
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Apply 0A to I3");
            Lcd_Out(2,1,"then press >");
            break;
         }
        if (Button(&PORTB, 3, 1, 0)) mode=450;
        break;
case 452:
        uf=0;
        df=0;
        rf=0;
        lf=0;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Wait until");
        Lcd_Out(2,1,"reading stable");
        delay_ms(1500);
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"then press >");
        delay_ms(1000);
        mode=453;
        break;
case 453:
        uf=0;
        df=0;
        rf=1;
        lf=0;
        Lcd_Cmd(_LCD_CLEAR);

        switch(vselect)
        {
         case 0:
            actual_val=calc_i(1,ai1);
            sense[0]=(ai1 * 5000) >> 10;;
            break;
         case 1:
            actual_val=calc_i(2,ai2);
            sense[1]=(ai2 * 5000) >> 10;;
            break;
         case 2:
            actual_val=calc_i(3,ai3);
            sense[2]=(ai3 * 5000) >> 10;;
            break;
         }

        Lcd_Out(1,1,"Applied Current:");
        disp_long(actual_val);
```

```
            Lcd_Out_cp("A");
            if (Button(&PORTB, 3, 1, 0)) mode=450;
            break;
        case 454:
            uf=0;
            df=0;
            rf=1;
            lf=0;
            Lcd_Cmd(_LCD_CLEAR);
            Lcd_Out(1,1,"Adjust Reading:");
            my_delay=200;
            counter=1;
            while (Button(&PORTB, 4, 300, 0) &&
(actual_val)<24000)
            {
              Lcd_Cmd(_LCD_CLEAR);
              Lcd_Out(1,1,"Adjust Reading:");
              if (my_delay>10) my_delay=my_delay-10;
              else
              {
               my_delay=200;
               counter=counter*10;
               actual_val=actual_val/counter;
               actual_val=actual_val * counter;
              }
              actual_val=actual_val+counter;
              if (actual_val>24000) actual_val=24000;
              disp_long(actual_val);
              Lcd_Out_cp("A");
              Vdelay_ms(my_delay);
            }
            my_delay=200;
            counter=1;
            while (Button(&PORTB, 5, 300, 0) &&
(actual_val)>0)
            {
              Lcd_Cmd(_LCD_CLEAR);
              Lcd_Out(1,1,"Adjust Reading:");
              if (my_delay>10) my_delay=my_delay-10;
              else
              {
               my_delay=200;
               counter=counter*10;
               actual_val=actual_val/counter;
               actual_val=actual_val * counter;
              }
              actual_val=actual_val-counter;
              if (actual_val<0) actual_val=0;
              disp_long(actual_val);
              Lcd_Out_cp("A");
              Vdelay_ms(my_delay);
            }
            disp_long(actual_val);
            Lcd_Out_cp("A");
            if (Button(&PORTB, 3, 1, 0)) mode=450;
            break;
        case 455:
            uf=0;
            df=0;
            rf=0;
            lf=0;
            switch(vselect)
            {
             case 0:
             case 1:
                 temp_val[vselect]=actual_val;
                 vselect++;
                 mode=451;
                 break;
             case 2:
                 Lcd_Cmd(_LCD_CLEAR);
                 Lcd_Out(1,1,"Please Wait");
                 lsensi[0]=sense[0];
                 lsensi[1]=sense[1];
                 lsensi[2]=sense[2];
                 lactuali[0]=temp_val[0];
                 lactuali[1]=temp_val[1];
                 lactuali[2]=temp_val[2];
                 Lcd_Out(2,1,"Saving....");

                 sp_eeprom_write(0x40,2,lsensi[0]);
                 sp_eeprom_write(0x42,2,lsensi[1]);
                 sp_eeprom_write(0x44,2,lsensi[2]);
                 sp_eeprom_write(0x46,3,lactuali[0]);
                 sp_eeprom_write(0x49,3,lactuali[1]);
                 sp_eeprom_write(0x4C,3,lactuali[2]);

                 delay_ms(2000);
                 Lcd_Cmd(_LCD_CLEAR);
                 Lcd_Out(1,1,"Complete");
                 delay_ms(1000);
                 mode=450;
                 break;
```

```
        }
        break;

//Upper Current Level Calibration
case 460:
        uf=-10;
        df=0;
        rf=1;
        lf=0;
        vselect=0;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Current Calib:");
        Lcd_Out(2,1,"Upper Level");
        if (Button(&PORTB, 3, 1, 0)) mode=420;
        break;
case 461:
        uf=0;
        df=0;
        rf=1;
        lf=0;
        switch(vselect)
        {
         case 0:
              Lcd_Cmd(_LCD_CLEAR);
              Lcd_Out(1,1,"Apply URV to I1");
              Lcd_Out(2,1,"then press >");
              break;
         case 1:
              Lcd_Cmd(_LCD_CLEAR);
              Lcd_Out(1,1,"Apply URV to I2");
              Lcd_Out(2,1,"then press >");
              break;
         case 2:
              Lcd_Cmd(_LCD_CLEAR);
              Lcd_Out(1,1,"Apply URV to I3");
              Lcd_Out(2,1,"then press >");
              break;
         }
        if (Button(&PORTB, 3, 1, 0)) mode=460;
        break;
case 462:
        uf=0;
        df=0;
        rf=0;
        lf=0;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Wait until");
        Lcd_Out(2,1,"reading stable");
        delay_ms(1500);
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"then press >");
        delay_ms(1000);
        mode=463;
        break;
case 463:
        uf=0;
        df=0;
        rf=1;
        lf=0;
        Lcd_Cmd(_LCD_CLEAR);
        switch(vselect)
        {
         case 0:
              actual_val=calc_i(1,ai1);
              sense[0]=(ai1 * 5000) >> 10;;
              break;
         case 1:
              actual_val=calc_i(2,ai2);
              sense[1]=(ai2 * 5000) >> 10;;
              break;
         case 2:
              actual_val=calc_i(3,ai3);
              sense[2]=(ai3 * 5000) >> 10;;
              break;
         }

        Lcd_Out(1,1,"Applied Current:");
        disp_long(actual_val);
        Lcd_Out_cp("A");
        if (Button(&PORTB, 3, 1, 0)) mode=460;
        break;
case 464:
        uf=0;
        df=0;
        rf=1;
        lf=0;
        Lcd_Cmd(_LCD_CLEAR);
        Lcd_Out(1,1,"Adjust Reading:");
        my_delay=200;
        counter=1;
        while (Button(&PORTB, 4, 300, 0) &&
(actual_val)<24000)
```

```
        {
         Lcd_Cmd(_LCD_CLEAR);
         Lcd_Out(1,1,"Adjust Reading:");
         if (my_delay>10) my_delay=my_delay-10;
         else
         {
          my_delay=200;
          counter=counter*10;
          actual_val=actual_val/counter;
          actual_val=actual_val * counter;
         }
         actual_val=actual_val+counter;
         if (actual_val>24000) actual_val=24000;
         disp_long(actual_val);
         Lcd_Out_cp("A");
         Vdelay_ms(my_delay);
        }
        my_delay=200;
        counter=1;
        while (Button(&PORTB, 5, 300, 0) &&
(actual_val)>0)
        {
          Lcd_Cmd(_LCD_CLEAR);
          Lcd_Out(1,1,"Adjust Reading:");
          if (my_delay>10) my_delay=my_delay-10;
          else
          {
           my_delay=200;
           counter=counter*10;
           actual_val=actual_val/counter;
           actual_val=actual_val * counter;
          }
          actual_val=actual_val-counter;
          if (actual_val<0) actual_val=0;
          disp_long(actual_val);
          Lcd_Out_cp("A");
          Vdelay_ms(my_delay);
        }
        disp_long(actual_val);
        Lcd_Out_cp("A");
        if (Button(&PORTB, 3, 1, 0)) mode=460;
        break;
    case 465:
        uf=0;
        df=0;
        rf=0;
        lf=0;
        switch(vselect)
        {
         case 0:
         case 1:
             temp_val[vselect]=actual_val;
             vselect++;
             mode=461;
             break;
         case 2:
             Lcd_Cmd(_LCD_CLEAR);
             Lcd_Out(1,1,"Please Wait");
             usensi[0]=sense[0];
             usensi[1]=sense[1];
             usensi[2]=sense[2];
             uactuali[0]=temp_val[0];
             uactuali[1]=temp_val[1];
             uactuali[2]=temp_val[2];
             Lcd_Out(2,1,"Saving....");

             sp_eeprom_write(0x50,2,usensi[0]);
             sp_eeprom_write(0x52,2,usensi[1]);
             sp_eeprom_write(0x54,2,usensi[2]);
             sp_eeprom_write(0x56,3,uactuali[0]);
             sp_eeprom_write(0x59,3,uactuali[1]);
             sp_eeprom_write(0x5C,3,uactuali[2]);

             delay_ms(2000);
             Lcd_Cmd(_LCD_CLEAR);
             Lcd_Out(1,1,"Complete");
             delay_ms(1000);
             mode=460;
             break;
        }
        break;


    }
   Delay_ms(300);
  }
}
```