DEVELOPMENT OF PC BASED HOME ENERGY MANAGEMENT SYSTEM

MUHAMAD KHAIRI BIN CHE YUSOFF

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **DEVELOPMENT OF PC BASED HOME ENERGY MANAGEMENT SYSTEM**

SESI PENGAJIAN: **2007/2008**

Saya **MUHAMAD KHAIRI BIN CHE YUSOFF (850212-03-5469)**
(HURUF BESAR)

mengaku membenarkan tesis (Sarjana Muda/~~Sarjana~~ /~~Doktor Falsafah~~)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Kolej Universiti Kejuruteraan & Teknologi Malaysia.
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan (√)

☐	**SULIT**	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)
☐	**TERHAD**	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)
√	**TIDAK TERHAD**	

Disahkan oleh:

_____ (TANDATANGAN PENULIS)

_____ (TANDATANGAN PENYELIA)

Alamat Tetap:

KAMPUNG TELUK BAYU
17600 JELI
KELANTAN

MOHD SYAWAL JADIN
(Nama Penyelia)

Tarikh: **27 NOVEMBER 2007**

Tarikh: : **27 NOVEMBER 2007**

CATATAN:
* Potong yang tidak berkenaan.
** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

"I hereby declare that I have read this thesis and in
My opinion this thesis is sufficient in terms of scope and
Quality for the award of the Degree of
Bachelor of Electrical & Electronic Engineering"

Signature : .....
Supervisor : Mr. Mohd Shawal Jadin
Date : 27 November 2007

DEVELOPMENT OF PC BASED HOME ENERGY MANAGEMENT SYSTEM

MUHAMAD KHAIRI BIN CHE YUSOFF

This thesis is submitted as partial fulfillment of the requirements for the award of the Bachelor Degree of Electrical Engineering (Electronics)

Faculty of Electrical & Electronics Engineering
Universiti Malaysia Pahang

NOVEMBER, 2007

Signature : _____

Author : MUHAMAD KHAIRI BIN CHE YUSOFF

Date : 27 November 2007

Dedicate to my beloved family and friends
who always give me a courage to finish this thesis.

Also, to those people who have been supportive through all this time.
Thank you for the kindness and advices that have been given.

God bless you all,-amin-

ACKNOWLEDGEMENT

Alhamdulillah, I am really grateful to the creator Allah S.W.T because of His regards I finally finish this final year project. Without His blessing, it is difficult for me to overcome and face all problems while completing this project. I also would like to express thousand of thank to my supervisor, Mr. Mohd Shawal Jadin who give highly encouragement, supporting and guideline in order to finish this task.

Not forgetting to my beloved parents that always prays for me and give me strength with unlimited effort. They always remind and give lots of motivation about patient and ask me to never give up. Thank you mum and my father, may Allah bless you always.

Beside that, thank you very much to my entire friend who always shares ideas and co-operation in order to finish this project. I wish you all best of luck.

Lastly, thank you for those who are involved directly or indirectly and your co-operation will never be forgotten.

Thank you

Muhamad Khairi Che Yusoff

ABSTRACT

The usage of electrical power at home is something that always occurred in our life and it was a necessary to manage it in a systematic way to ensure the system operate in intelligent mode. In United State when users at home turn on the electrical appliances in peak time which contribute to the exceeding of limit power drop causing them to be charge in expensive rate. The purpose of this project is to design a control system at home to meet the objectives of monitoring the consumption's rate of power, voltage level, current level, and turn on or off appliance through a computer. Methodology in implement this system could be categorized into software development and hardware design. The software develops based on graphical user interface in MATLAB 7.1 which is interface with hardware that used MC68HC11 as controller through serial communication. The hardware involves three stages which are signal conditioning circuit, controller and interfacing circuit. Operation of the hardware is started from signal conditioning circuit where voltage and current convert into voltage representation in range of 0V to 5V in order to fit the analog to digital converter (ADC) port at microcontroller. Voltage level of 240 Vrms need to be step down using voltage transformer and current measured by current transformer which is producing small voltage level and required amplification. Then microcontroller computes the power consumption and rms value of current and voltage and display at liquid crystal display (LCD). At the same time the microcontroller send the results of conversion of ADC to personel computer (PC) through serial communication. The application of organizing the energy could be seem when appliance also able to be control from GUI and also from external switch. Besides that, warning indicator at LCD exhibit when voltage and current level exceed certain limit and the limiting level could be determine from entering rms value at GUI. With the ability of manage of some certain characteristics of home's electrical power supply it will produce a systematic system in organizing our energy.

ABSTRAK

Penggunaan tenaga elektrik sentiasa wujud dalam kehidupan seharian dan adalah suatu perkara yang mustahak untuk menguruskannya supaya beroperasi secara pintar. Di negara seperti Amerika Syarikat, pemilik sesebuah rumah akan dikenakan kadar bayaran yang berlainan apabila rumah mereka menggunakan tenaga lebih daripada yang ditetapkan dalam waktu puncak. Wlaubagaimanapun, syarikat yang mengendalikan pengurusan tenaga elektirk di Malaysia iaitu Tenaga Nasional Berhad (TNB) tidak mengenakan peraturan sedemikian rupa tetapi akan mengenakan saman sekiranya sesebuah industri menyebabkan "power factor" rendah daripada 0.8. Tujuan utama alat ini dibangunkan adalah untuk memantau pengunaan kuasa elektrik, voltan, arus dan membolehkan pengguna sama ada membuka atau menutup suis kepada alatan elektrik tertentu dengan menggunakan komputer. Kaedah dalam pembangunan sistem ini boleh dibahagikan kepada dua bahagian utama iaitu pembangunan terhadap perkakasan dan perisian. Pembangunan perisian adalah berdasarkan kepada kemudahan graphical user interface (GUI) dalam MATLAB 7.1 yang mana di sambung kepada chip MC68HC11 sebagai alat pengawal melalui pengantaraan data secara siri. Bahagian perkakasan adalah terdiri daripada chip pengawal, litar penyesuaian isyarat dan litar untuk sambungan kepada peralatan elektrik. Operasi sistem ini adalah bermula dari litar penyesuaian yang mana output daripada pengesan arus dan voltan akan di masukkan kepada pin ADC untuk melakukan penukaran kepada isyarat digital. Chip pengawal akan melakukan pengiraan untuk mendapatkan nilai kuasa dan kemudiannya nilai berkenan akan dipaparkan pada LCD dan komputer. Oleh hal yang demikian, penguna mampu untuk menguruskan penngunaa kuasa elektrik di rumah mereka sekiranya maklumat yang tepat dapat diberikan kepada mereka. Selain itu terdapat beberapa ciri tambahan telah reka kepada sistem ini supaya beroperasi dengan lebih sistematik.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

PC	*- Personnel Computer*
GUI	*- Graphical User Interface*
PLC	*- Programmable Logic Controller*
SCI	*- Serial Communication Interface*
ADC	*- Analog to Digital Converter*
LCD	*- Liquid Crystal Display*
RMS	*- Root Mean Square*
KWh	*- kilowatt-hour*
RAM	*- Random Access Memory*
ROM	*- Read Only Memory*
EPROM	*- Erasable Programmable ROM*
EEPROM	*- Electrical Erasable Programmable ROM*
CPU	*- Central Processing Unit*
I/O	*- Input/output*
IC	*- Integrated Circuit*
PCB	*- Printed Board Circuit*
SPI	*- Serial Peripheral Interface*
AC	*- Alternate Current*
DC	*- Direct Current*
CT	*- Current Transformer*
VT	*- Voltage Transformer*
GUIDE	*- Graphical User Interface Development Environment*
ASCII	*- American Standard Code International Interchange*
CGRAM	*- Character Generator RAM*
NC	*- Normally Closed*
NO	*- Normally Open*

LIST OF APPENDIXES

CHAPTER 1

INTRODUCTION

1.1 Background

Development a personnel computer (PC) based energy management system associated with automatic meter reading and various load control functions for load management of home customers. Automatic meter reading configured by a programmable logic controller and power meter supports the function of recording 20 power parameters retrieved from an academic substation. Load control functions; which consist of demand control, timer control, cycling on/off control and direct control, are designed to effectively reduce peak load and to save electrical energy. Several types of software program coded by Visual Basic, programmable logic controller (PLC) ladder language, and partial Diamond package, are designed to integrate all hardware equipment in energy management system. The entire system has been installed and successfully operated since October 1997. Statistical data collected from 1997/10 to 1999/10 indicate that the proposed effect for peak load cutting and energy saving may be justified

Development of PC based Home Energy Management System is a control system that embedded with intelligent to manage and monitor the usage of energy at home. Basically, this project is designed to be interface with home electrical

appliances based on development of Graphical User Interface (GUI) in Matlab 7.1 and microcontroller MC68HC11. The system is expected to become an intelligent management system that lead to energy saving in mean of low in cost and also to keep the life span of the appliances. For general view of this system, by plugging a plug of load at the provided socket at the system, it allows the users to see consumption rate for different appliances in their house. For example, when a heater is connected to the socket of this system, the LCD display will display the power dissipation.

The knowledge of power draw of each appliance becomes vital when it come to the usage of appliances in demand time. As example, in several countries such as US, the utility's company will charge different rate if power consumption for a home in peak time is exceed certain limit. By adding a meter which is displaying power dissipation for summation of several appliances at same time will lead the users to disable certain appliance when total power consumption is exceeded the limit. Thus, user definitely can avoid the unnecessary charge due to exceeding the rate. Besides that, user could find the suitable range of time to use the appliances which is consuming major power and also be able to turn on or off appliances either through GUI and using external switch in purpose of limiting power consumption.

The system required the designing on hardware and software where it could be separate into three main parts. Its involve software development, control unit (microcontroller) and appliances interfacing circuit. For advance features of this system, the control unit will able to stand-alone running without connected to the PC. This is important because the usage of PC for 24 hours in a day is not efficient, as we know PC also draw a lot of power and furthermore the continuous running of PC would give bad effect to itself.

1.2 Objective

The objective of this project is:

(i) To create a device that constantly monitors the power consumption by each appliance and communicates the data to the user in easy manner by developing a GUI system.

(ii) To allow the user to control the appliances through a PC and also the system able to work independently base on the program that been set in control unit to be automatically react to the data such in order to manipulate the energy consumption patterns.

1.3 SCOPE OF PROJECT

(i) Develop a GUI using Matlab 7.1 for hardware and software interfacing.

(ii) Designing on microcontroller MC68HC11 in **expanded mode** as control unit and optimize the usage of features that exist in that microcontroller such as serial communication (SCI) and analog to digital converter (ADC).

(iii) Designing a signal conditioning circuit which is consisting of voltage sensor and current sensor.

(iv) Designing appliance interfacing circuit which is interface with appliances that apply the switching concept based on relay that controlled by microcontroller.

(v) Designing LCD display as a meter to view that measurement of voltage (rms), current (rms) and power consumption.

CHAPTER 2

LITERATURE REVIEW

2.0 Introduction

Development of PC based home energy management system is a development of a control system to monitor the power usage at home. This system could be classified as supervisory control system where the PC is used to monitor the usage of energy at home. However the PC only receive the measured value of voltage and current, then user can interface with PC to determine the set point to be sent to the microcontroller. From here user was able to manage the usage of energy at home in systematic ways and the system can supply a homeowner with the necessary information that allows the homeowner to adjust the usage of individual appliances. As referring to the below statement:

A PC based energy management system associated with automatic meter reading and various load control functions for load management (Ming Yuan Cho, Cha Win Huang, May 2001). **[1]**

Thus, the development of this project is coincidence with the statement where the system is function to be as meter for current, voltage and power reading and especially for management of various load at home.

In certain countries, the utility's company determined different rate of charge per kilowatt-hour (kWh) when homeowner is consuming energy during periods of high demand. As referring to this statement:

Manitoba Hydro provides electrical energy at a fixed rate, but in many other countries utility companies charge variable rates for energy, charging more per kilowatthour (kWh) during periods of high demand. **[2]**

This condition will cause the electricity bill become high. In order to avoid this situation, user must consuming energy during off peak periods instead of during periods of high demand. Through this step, consumers can save money and reduce the load placed on utility companies. Thus, creating a device that encourages off-peak consumption will not only save money for consumers, but on a larger scale, will also reduce the risk of blackouts due to the reduction of the peak loads placed on utility companies. Besides that, it will also be able to allow user intervention such as turning the appliance on or off over the network through an easy to use GUI.

2.1 Analog Circuit

Under this analog circuit, most of the analog and electronic component will be discuss. It consists of:

2.1.1 Microcontroller
2.1.2 LCD display
2.1.3 Voltage transformer
2.1.4 Current transformer
2.1.5 Rectifier

2.1.1 Microcontroller

Many things should be considered before choosing Microcontroller as the controller. Generally there are two types of controller which are Microprocessor and Microcontroller. General purpose microprocessor such as Intel's x86 family (8086, 80286, 80386, 80486 and Pentium) or Motorola's 680x0 family (68000, 68010, etc) contain no random access memory (RAM), read only memory (ROM) or input/output (I/O) on the chip itself. They require these devices externally to make them functioning. A microcontroller consists of central processing unit (CPU), ROM,

embedded together in a single chip. It is an ideal for many applications. Figure 2.1 show the diagram of microprocessor and microcontroller.

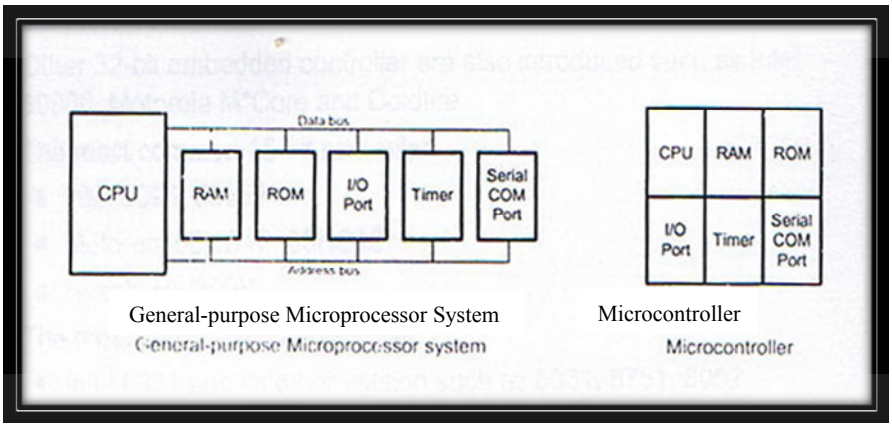


Figure 2.1: Microprocessor versus microcontroller

Both chip offer their own advantages and disadvantages. For microprocessor, it a high performance integrated circuit (IC), more flexibility and can easily expended. However, it's more expensive as it required additional ICs and used large space. While for microcontroller, it is a high integration IC with small space for printed board circuit (PCB). Its cost is cheaper compare to microprocessor. It also has special architecture with low power consumption. However, it has limited expansion due to extra features offer in microcontroller. In this project, microcontroller is used due to its advantages compare to microprocessor. Figure 2.2 show the block diagram of a Typical Microcontroller shown in a Single-Chip Mode.

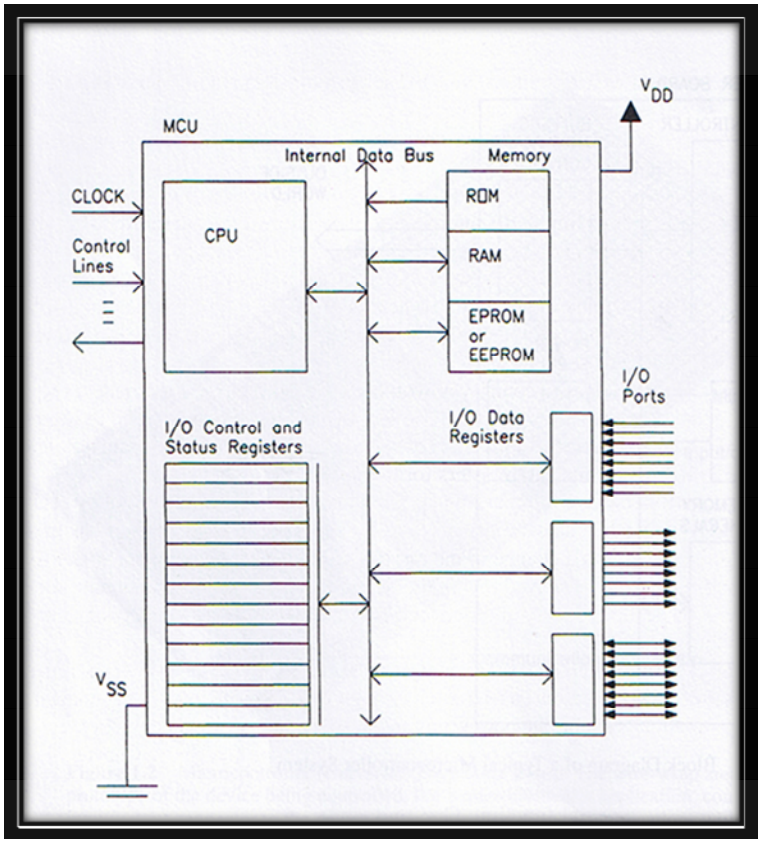


Figure 2.2: Block diagram of a Typical Microcontroller in a Single-Chip Mode.

2.1.1.1 Motorola MC68HC11

The most commonly microcontroller used nowadays are 8-bit and 16-bit microcontroller. Other 32-bit embedded controller are also introduced such as Intel 80960, Motorola M*Core and Coldfire. The most common 16-bit controller is Intel 8096, 80251, Motorola 68HC12, 68HC16 and Hitachi H8/300H. For 8-bit microcontroller, commonly used are Intel 8051 and its other version such as 8031 8751, 8052, Microchip PIC such as P1C16F84 and BASIC STAMP and Motorola 68H 005, 68HC08, 68HC1 1 and Zilog Z8.

The most commonly microcontroller used nowadays are 8-bit and 16-bit microcontroller. Other 32-bit embedded controller are also introduced such as Intel 80960, Motorola M*Core and Coldfire. The most common 16-bit controller is Intel 8096, 80251, Motorola 68HC12, 68HC16 and Hitachi H8/300H. For 8-bit

microcontroller, commonly used are Intel 8051 and its other version such as 8031 8751, 8052, Microchip PIC such as P1C16F84 and BASIC STAMP and Motorola 68H 005, 68HC08, 68HC1 1 and Zilog Z8.

When choosing correct microcontroller, several things also need to be considered such as meet the computing needs for the task at cost efficiency, software availability, wide availability and reliable sources. For this project, 8-bit Motorola MC68HC11 family microcontroller is used as the main controller for the system. Commonly this model more synonyms with name 68HC11. This model of microcontroller offer several features which meets the requirement for the task at the lowest cost. 68HC11 offers various subsystems such as ADC, interrupts, timers and so on. Besides, it uses simple assembly language because the processor uses the Von Neumann architecture. Table 2.0 shows the version of MC68HC11.

Table 2.0: Version of Motorola MC68HC11

Part Number	EPROM	ROM	EE-PROM	RAM	CONFIG[2]	Comments
MC68HC11A8	—	—	512	256	$0F	Family Built Around This Device
MC68HC11A1	—	—	512	256	$0D	'A8 with ROM Disabled
MC68HC11A0	—	—	—	256	$0C	'A8 with ROM and EEPROM Disabled
MC68HC811A8	—	—	8K + 512	256	$0F	EEPROM Emulator for 'A8
MC68HC11E9	—	12K	512	512	$0F	Four Input Capture/Bigger RAM 12K ROM
MC68HC11E1	—	—	512	512	$0D	'E9 with ROM Disabled
MC68HC11E0	—	—	—	512	$0C	'E9 with ROM and EEPROM Disabled
MC68HC811E2	—	—	2K[1]	256	$FF[3]	No ROM Part for Expanded Systems
MC68HC711E9	12K	—	512	512	$0F	One-Time Programmable Version of 'E9
MC68HC11D3	—	4K	—	192	N/A	Low-Cost 40-Pin Version
MC68HC711D9	4K	—	—	192	N/A	One-Time Programmable Version of 'D3
MC68HC11F1	—	—	512[1]	1K	$FF[3]	High-Performance Non-Multiplexed 6B-Pin
MC68HC11K4	—	24K	640	768	$FF	> 1 Mbyte memory space, PWM, C_S, 84-Pin
MC68HC711K4	24K	—	640	768	$FF	One-Time Programmable Version of 'K4
MC68HC11L6	—	16K	512	512	$0F	Like 'E9 with more ROM and more I/O, 64/68
MC68HC711L6	16K	—	512	512	$0F	One-Time Programmable Version of 'L4

2.1.1.2 Motorola 68HC11 Architecture

The MicroStamp11 module is built around the Motorola 68HC11 micro-controller IC. In order to program the MicroStamp11, user need to have a closer look at the 68HC11's architecture. The 68HC11's basic architectural blocks are shown in figure 2.3. This figure explicitly shows the peripheral subsystems in the Motorola 68HC11 micro-controller and it shows which pins those subsystems are tied to.

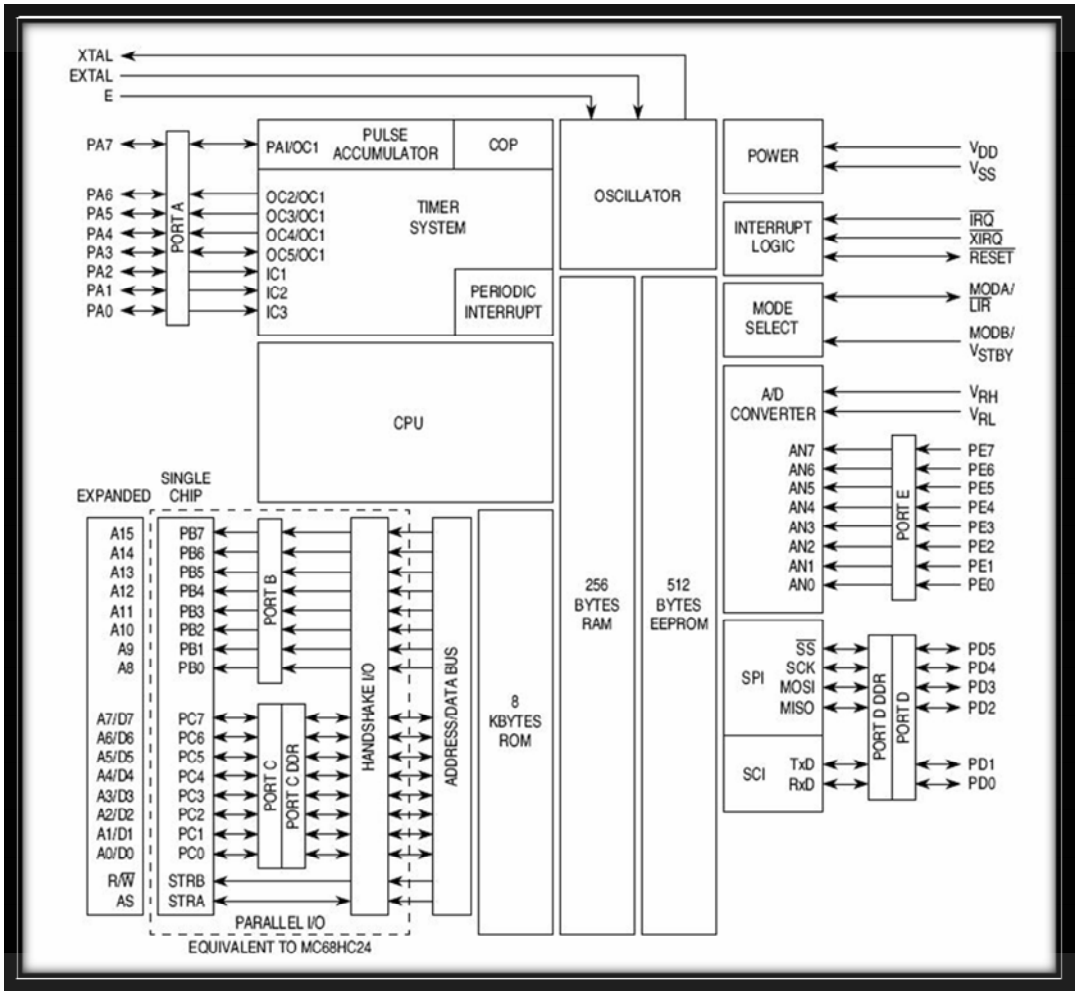


Figure 2.3: 68HC11 Architecture

From Figure 2.3, we see that the 68HC11 has a number of pins. Some of these pins are used to control the micro-controller's operating mode, clock logic, special interrupts, or power. The majority of the pins, however, have been organized into four 8-bit I/O ports. These ports have the logical names PORTA, PORTB,

PORTC, PORTD and PORTE. It is through these five ports that the 68HC11 channels most of its interactions with the outside world.

Table 2.1 is showed the descriptions of each port while Figure 2.4 shows the pin configuration for MC68HC11 microcontroller.

Table 2.1: Description of each port at MC68HC11

Port	Function
Port A	Parallel I/O or timer/counter
Port B	Output port or upper address (A8 - M5) in expanded mode
Port C	I/O port or lower address (A0 - A7) and data bus (DO - D7) in expanded mode
Port D	6-bits I/O port or serial communication interface (SCI) and serial peripheral interface (SPI)
Port E	Input port or 8-channels input analog for ADC

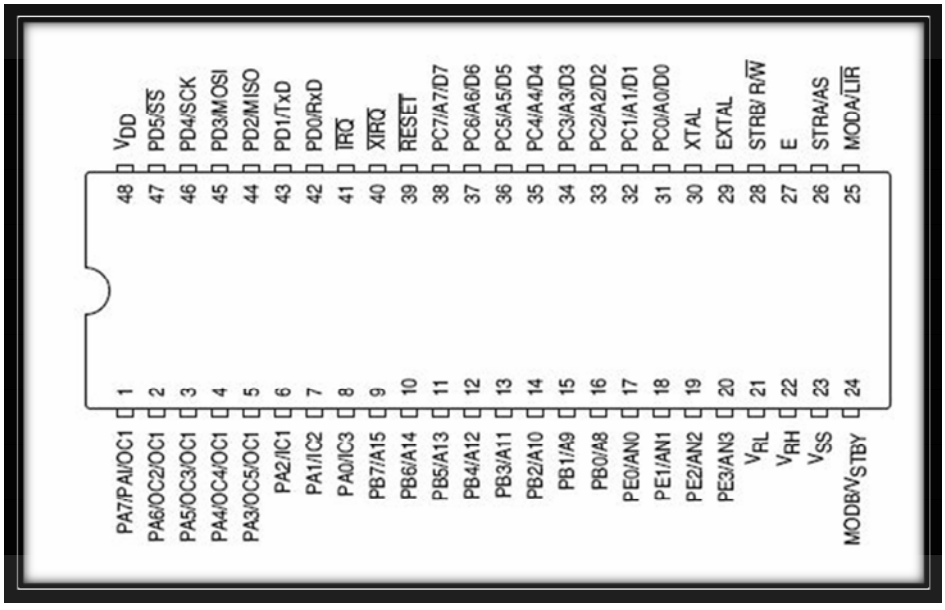


Figure 2.4: 48-Pin DIP Pin Assignments

As mentioned earlier, a micro-controller is often distinguished by the fact that its input/output devices are directly mapped into RAM. This is also true of the I/O ports in the 68HC11. The logical names for the I/O ports are associated with absolute addresses in RAM and these addresses are in turn tied to hardware registers. When

an input pin, for example, is set to a high logical level, then that logic level directly sets the value in the port's hardware register. Since that hardware register is mapped directly into the micro-controller's address space, a program can then directly read that register's value by accessing memory.

The I/O ports and other device pins are connected to special subsystems in the 68HC11. The subsystems shown in figure 2.3 are briefly described below:

a) **Erasable programmable read only memory (EPROM):** Some versions of the 68HC11 have as much as 4 kilo-bytes of internal Electrical erasable programmable read only memory **(**EEPROM). If your program is sufficiently small, then your micro-controller system would not need external memory chips and could be operated in single-chip mode.

b) **RAM:** The version of the 68HC11 in your MicroStamp11 has 256 bytes of internal RAM. As mentioned above, some of these bytes are mapped into hardware registers that are used to control the micro-controller. In reality the MicroStamp11 programmer only has 192 bytes of RAM that can be used for program variables.

c) **Serial Peripheral Interface (SPI):** This subsystem allows the 68HC11 to communicate with synchronous serial devices such as serial/parallel slave devices.

d) **Serial Communication Interface (SCI):** This subsystem allows the 68HC11 to communicate with asynchronous serial devices. The SCI interface is used to communicate with laptop computers.

e) **Parallel I/O Interface:** This subsystem is generally used to provide the 68HC11 with a way of writing digital data in parallel to an external device. The usual parallel device is a memory device. Recall that the 68HC11 has a very limited amount of internal program memory. If we need to augment the EEPROM in the micro-controller with additional memory, we use the parallel I/O interface to address, read, and write data to this external memory chip. When we do this we usually operate the chip in so-called **expanded mode**.

Running the chip in expanded mode greatly reduces the number of I/O Ports available to the system. This is because PORTB and PORTC are connected to the memory chip and hence are unavailable for other external devices. Since the MicroStamp11 uses an external memory chip, it is running the 68HC11 in expanded mode and hence only PORTA and PORTD can be used by the programmer for interfacing with the external world.

f) **Mode Selection System:** This subsystem selects whether the 68HC11 runs in expanded or single-chip mode. In single chip mode, the 68HC11 allows the user to have complete control over all four I/O ports. In expanded mode, the 68HC11 uses ports B and C to address, read, and write to external memory; hence the programmer can only use PORTA and PORTD. In the MicroStamp11 module, the chip is usually in expanded mode.

g) **Clock logic:** An important feature of micro-controllers is that they work in **real-time**. By real-time, we mean that instruction executions are completed by specified time deadlines. This means that the micro-controller needs a clock. The clock logic subsystem provides the real-time clock for the 68HC11. The rate of the clock is determined by a crystal that is connected to the clock logic pins. The MicroStamp11 has a crystal on the module, so these pins are not available to the programmer.

h) **Interrupt Logic:** Micro-controllers must be able to respond quickly to asynchronous events. The interrupt logic subsystems provide three pins that can be used to trigger hardware interrupts. Hardware interrupts automatically transfers software execution to a specified memory address in response to the hardware event (such as the pin's logic state going low). We say that this interrupt is generated *asynchronously* because the event can occur between ticks of the system's real-time clock. Hardware interrupts provide a means for assuring that micro-controllers respond in a timely manner to external events.

i) **Timer Interrupts:** This subsystem generates interrupts that are associated with an internal timer. Remember that the 68HC11 executes instructions in step with a clock tick provided by the clock logic subsystem. With each tick of the clock, an internal register called a timer is incremented. This timer is

memory mapped to an address in RAM with the logical name TCNT. SO at any instant you can fetch the current count (time) on the timer by simply reading TCNT.

There are two types of interrupts associated with TCNT. An **input-compare** (IC) interrupt is generated with a specified input pin changes state. When the IC interrupt occurs, then the value in TCNT is stored in an input-compare register. This register is also memory mapped so the programmer can easily read the clock tick when the input event occurred. Input compare events are often used to make very precise timing measurements.

The other type of timer interrupt is called an **output compare** (OC) interrupt. The output compare event occurs when TCNT matches the value stored in an output compare register. The output compare register is also memory mapped, so its value can be easily set by the programmer. Output compare events are often used to force the micro-controller to respond to timed events.

2.1.2 LCD Display

LCD (commonly abbreviated LCD) is a thin, flat display device made up of any number of color or monochrome pixels arrayed in front of a light source or reflector. It is often utilized in battery-powered electronic devices because it uses very small amounts of electric power.

LCDs with a small number of segments, such as those used in digital watches and pocket calculators, have individual electrical contacts for each segment. An external dedicated circuit supplies an electric charge to control each segment. This display structure is unwieldy for more than a few display elements.

The VCM162A dot-matrix LCD controller and driver LSI displays alphanumeric, Japanese kana characters, and symbols. It can be configured to drive a dot-matrix liquid crystal display under the control of a 4- or 8- bit microprocessor. Since all the functions such as display RAM, character generator, and liquid crystal

driver, required for driving a dot-matrix liquid crystal display are internally provided on one chip, a minimal system can be interfaced with this controller/driver.

A single VCM162A can display up to one 8-character line or two 8-character lines. The VCM162A has pin function compatibility with the HD44780U which allows the user to easily replace an LCD-II with a VCM162A. The VCM162A character generator ROM is extended to generate 208 5×8 dot character fonts and 32 5×10 dot character fonts for a total of 240 different character fonts. The low power supply (2.7V to 5.5V) of the VCM162A is suitable for any portable battery-driven product requiring low power dissipation. Shown in Figure 2.5 is a (16 x 2) lines (8 characters) of VCM162A LCD display.

Figure 2.5: 2 lines (8 character) VCM162A LCD display

2.1.3 Voltage Transformer

Transformers convert alternate current (AC) electricity from one voltage to another with little loss of power. Transformers work only with AC and this is one of the reasons why mains electricity is AC. Step-up transformers increase voltage, step-down transformers reduce voltage. Most power supplies use a step-down transformer to reduce the dangerously high mains voltage (230V in UK) to a safer low voltage.

The input coil is called the primary and the output coil is called the secondary. There is no electrical connection between the two coils; instead they are

linked by an alternating magnetic field created in the soft-iron core of the transformer. The two lines in the middle of the circuit symbol represent the core. Transformers waste very little power so the power out is (almost) equal to the power in. Note that as voltage is stepped down current is stepped up.

The ratio of the number of turns on each coil, called the turn's ratio, determines the ratio of the voltages. A step-down transformer has a large number of turns on its primary (input) coil which is connected to the high voltage mains supply, and a small number of turns on its secondary (output) coil to give a low output voltage.

2.1.4 Current Transformer (CT)

A CT is designed to monitor current flow in a circuit, such as to detect excessive power consumption and provide a warning signal or disconnect power supply, by utilizing the strength of the magnetic field around the conductor to form an induced current that can then be applied across a resistance to form a proportional voltage. CTs are currently used to measure either common-mode current or differential-mode current which is flowing in an electrical circuit conducting substantial electrical power. A CT is used when it is desired to introduce electrical signals from a commercial power line into an analog IC . Current transformers are typically used in three-phase machines to measure the current in each of the three-phase neutral leads of the main stator coils. Conventional AC current transformers fall into two categories; magnetic core current transformers and air coupled current transformers. [3]

Current sensors are used in the electric power industry to measure current flowing in electrical systems. Electrical current sensing is used in many applications. In particular, current sensors may be used in electrical switchgear such as circuit breakers, and switches to determine when a fault has occurred in the electrical system. Current sensors for detecting direct current (DC) have been widely used in a variety of fields, such as, home electric appliances (air conditioners, automatic washing machines, sewing machines, etc.), industrial equipment and transportation equipment. There are two common methods used to sense the current. The first method detects and measures the magnetic field produced by the current flowing in a

conductor (inductive systems). A second method of current sensing uses a `shunt` resistance in the current path to generate a voltage across the shunt resistance in proportion to the current flowing. A variety of sensors are used to measure the amount of current flowing through a conductor. Eddy current sensors are known and widely used in a variety of applications to measure characteristics of moving, electrically conductive objects. A common use of eddy current sensors is in fans and turbines, where the sensors are used to measure parameters related to blade status. A Hall Effect current sensor measures current flowing through a conductor and provides an output signal proportional to the level of current. Hall Effect current sensors offer several advantages over traditional current transformers such as a more compact size, higher current levels for a given size, and a larger frequency bandwidth. AC current sensors typical comprise inductive elements into which current is induced by the changing magnetic field surrounding an AC current conductor.

2.1.5 Rectifier

There are several ways of connecting diodes to make a rectifier to convert AC to DC. The bridge rectifier is the most important and it produces full-wave varying DC. A full-wave rectifier can also be made from just two diodes if a centre-tap transformer is used, but this method is rarely used now that diodes are cheaper.

A bridge rectifier can be made using four individual diodes, but it is also available in special packages containing the four diodes required. It is called a full-wave rectifier because it uses the entire AC wave (both positive and negative sections). 1.4V is used up in the bridge rectifier because each diode uses 0.7V when conducting and there are always two diodes conducting, as shown in the diagram below. Bridge rectifiers are rated by the maximum current they can pass and the maximum reverse voltage they can withstand (this must be at least three times the supply root mean square (RMS) voltage so the rectifier can withstand the peak voltages).

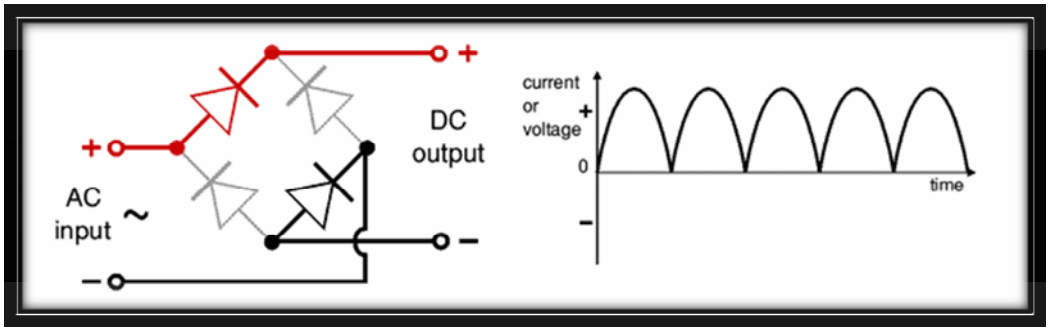


Figure 2.6: Bridge rectifier circuit and output voltage

Smoothing is performed by a large value electrolytic capacitor connected across the DC supply to act as a reservoir, supplying current to the output when the varying DC voltage from the rectifier is falling. The Figure 2.7 shows the unsmoothed varying DC (dotted line) and the smoothed DC (solid line). The capacitor charges quickly near the peak of the varying DC, and then discharges as it supplies current to the output.

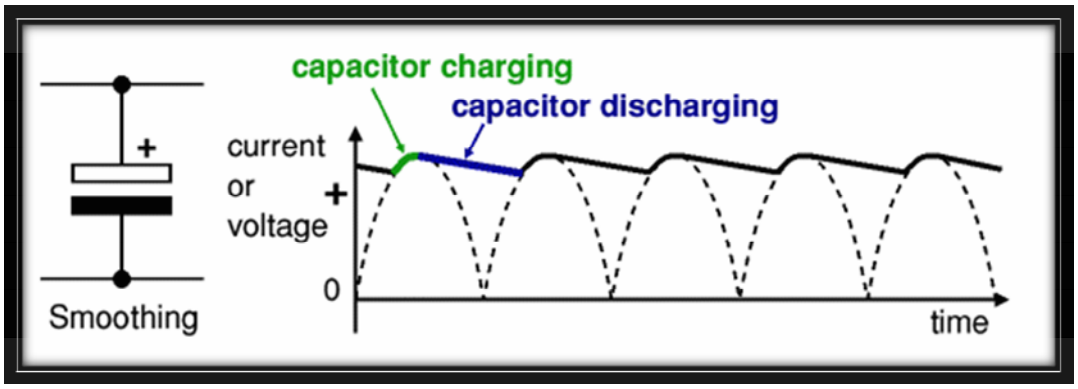


Figure 2.7: The unsmoothed varying DC and the smoothed DC

Smoothing is not perfect due to the capacitor voltage falls a little as it discharges, giving a small ripple voltage. For many circuits a ripple which is 10% of the supply voltage is satisfactory and the equation below gives the required value for the smoothing capacitor. A larger capacitor will give fewer ripples. The capacitor value must be doubled when smoothing half-wave DC.

2.1.6 Regulator

Voltage regulator ICs are available with fixed (typically 5, 12 and 15V) or variable output voltages. They are also rated by the maximum current they can pass. Negative voltage regulators are available, mainly for use in dual supplies. Most regulators include some automatic protection from excessive current (overload protection) and overheating (thermal protection).

Many of the fixed voltage regulator ICs has 3 leads and look like power transistors, such as the 7805 +5V 1A regulator as shown in Figure 2.8. They include a hole for attaching a heatsink if necessary.

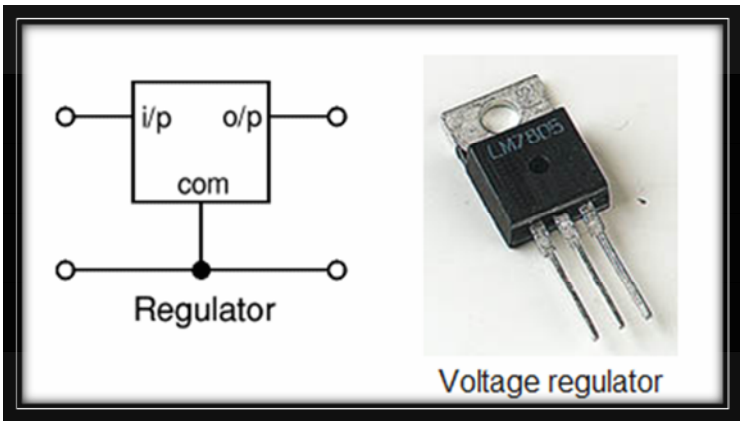


Figure 2.8: Voltage regulator

2.3 Software development

2.3.1 Introduction

Development of PC based home energy management system means that this project must include the development of any software to be interface with hardware. Thus, MATLAB is seemed to be useful to interface with external hardware. There was a specific function available in MATLAB to develop interfacing between PC and hardware. In MATLAB this function is call as GUI where it development can be done in GUIDE.

2.3.2 GUI

A GUI is a human-computer interface that uses windows, icons and menus and which can be manipulated by a mouse and a good GUI can make programs easier to use by providing them with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus, and so forth. In medical Simulink (MATLAB), it will compare the process of implementing a Pharmaco-kinetic/ Pharmaco-dinamic model of a biological system in traditional package. During simulation, a scope block automatically produces a time-course of the concentration of drug in the model compartments over a specified period.

A window is a (usually) rectangular portion of the monitor screen that can display its contents (e.g., a program, icons, a text file or an image) seemingly independently of the rest of the display screen. A major feature is the ability for multiple windows to be open simultaneously. Each window can display a different application, or each can display different files (e.g., text, image or spreadsheet files) that have been opened or created with a single application. The GUI components can be menus, toolbars, push buttons, radio buttons, list boxes, and sliders -- just to name a few. In MATLAB, a GUI can also display data in tabular form or as plots, and can group related components.

2.3.3 MATLAB GUI

The following GUI's MATLAB programs have been developed for the computational aids in the electrical engineering topics outlined in the menu at left. These GUI programs with point-and-click features are designed for ease of use. These programs together with the traditional hand-written problems can help students to develop a stronger intuition and deeper understanding of these topics.

The following aims to present some code which is repeatedly used and establishes what is expected from objects. The common code as below:

1. Radio Buttons

Where a group of buttons is inter-related, the following code should do:

```
set(handles.option1, 'Value', 1);
set(handles.option2, 'Value', 0);
set(handles.option3, 'Value', 0);
```

The '**option1'** is the source of the callback. For the other options, the only part which needs changing is the latter one where the state of the button is defined by a zero or a one.

2. **Check Boxes**

A naive yet useful implementation of those will involve an element *state* which holds some information about the state of the checkbox or its meaning.

```
if (get(handles.state,'Value') == 0),
set(handles.checkbox, 'Value', 0);
set(handles.state, 'String', '0');
else
set(handles.checkbox, 'Value', 1);
set(handles.state, 'String', '1');
end
```

3. **Drop-down Menus**

In the following, the menu object needs to first be identified using:

```
contents = get(hObject,'String');
```

Subsequently, the menu entry needs to be checked for extraction and setting of information.

```
if (strcmp(contents{get(hObject,'Value')},'MenuEntry1')),
set(handles.data, 'String', 'Data1');
elseif (strcmp(contents{get(hObject,'Value')},'MenuEntry2')),
set(handles.data, 'String', 'Data2');
else
msgbox('Error with menu callback. Parameter passed is not recognised.');
end
```

4. **Sliders**

The following code will fetch the quantised value of the slider and assign it to a new object called *slider_value*.set(handles.slider_value, 'String', num2str(ceil(get(handles.slider,**)**

For items that need to be ticked and un-ticked, the following can be useful.

```
set(handles.menuitem1, 'Checked', 'off');
set(handles.menuitem2, 'Checked', 'off');
set(handles.menuitem3, 'Checked', 'off');
set(handles.menuitem4, 'Checked', 'on');
set(handles.menu_selection, 'String', 'item4') [5]
```

For GUI editing, more efficient work on the code can be done by opening all relevant files in advance. I personally write M-files to open all relevant windows at the start. Set up a file which includes the following lines:

```
edit <m-file1> <m-file2>...
guide <fig-file1> <fig-file2>...
```

MATLAB GUI is a very powerful tool when used correctly. It takes a lot of experimenting with and a good background in programming. We also must have a good understanding of MATLAB and be able to use the MATLAB language and commands to create routines.

2.3.4 Creating GUIs with GUIDE

MATLAB implements GUIs as figure windows containing various uncontrolled objects. You must program each object to perform the action you intend it to do when a user activates the component. In addition, you must be able to save and run your GUI. All of these tasks are simplified by GUIDE, the MATLAB graphical user interface development environment.

GUIDE, the MATLAB Graphical User Interface development environment, provides a set of tools for creating GUIs. These tools greatly simplify the process of lying out and programming a GUI. GUIDE is displayed when GUI is opened in the Layout Editor, which is the control panel for all of the GUIDE tools. The Layout Editor will enables to lay out a GUI quickly and easily by dragging components, such as push buttons, pop-up menus, or axes, from the component palette into the layout area. The Figure 2.9 shows the Layout Editor.

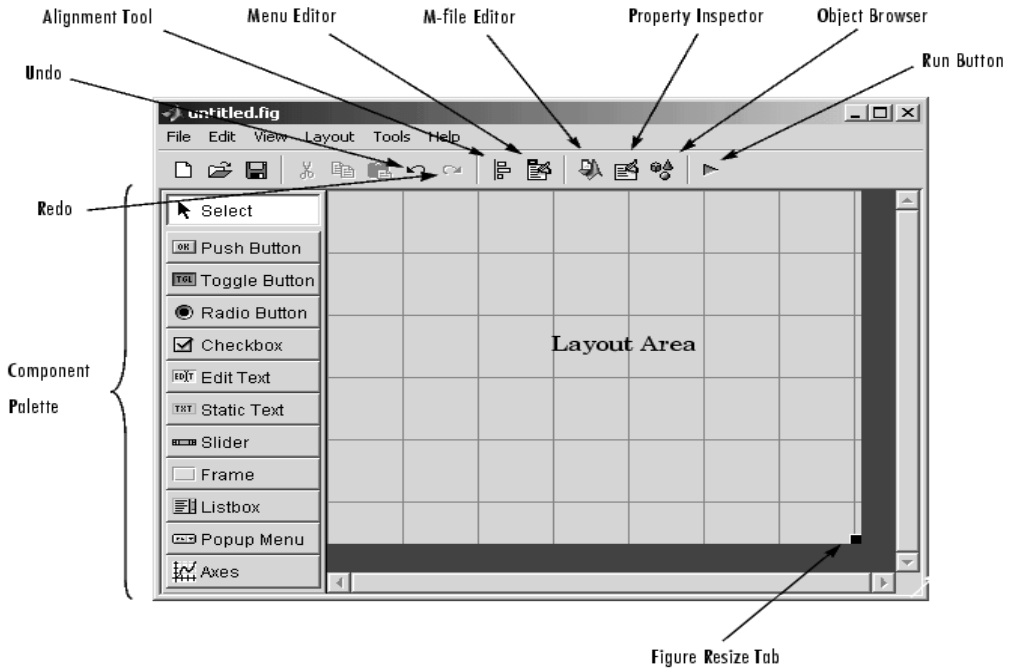


Figure 2.9: GUI layout

Once GUI layout of and set each component's properties, using the tools in the Layout Editor, GUI can be programmed with the M-file Editor. Finally, press the **Run** button on the toolbar, the functioning GUI appears outside the Layout to see the result.

- **GUI Development Environment**

Creating a GUI involves two basic tasks. They are laying out the GUI components and programming the GUI components. GUIDE primarily is a set of layout tools. However, GUIDE also generates an M-file that contains code to handle the initialization and launching of the GUI. This M-file provides a framework for the implementation of the *callbacks* -- the functions that execute when users activate components in the GUI.

- **GUIDE Generated Files**

While it is possible to write an M-file that contains all the commands to lay out a GUI, it is much easier to use GUIDE to lay out the components interactively. When you save or run the GUI, GUIDE automatically generates two files:

- A FIG-file -- a file with a `.fig` file name extension, which contains a complete description of the GUI figure and all of its children (uicontrols and axes), as well as the values of all object properties. You make changes to the FIG-file by editing the GUI in the Layout Editor.
- An M-file -- a file with a `.m` file name extension, which contains the functions that run and control the GUI and the callbacks. This file is referred to as the *GUI M-file*.

Note that the M-file does not contain the code that lays out the uicontrols; this information is saved in the FIG-file.

- **Features of the GUI M-file**

GUIDE simplifies the process of creating GUIs by automatically generating the GUI M-file directly from your layout. GUIDE generates callbacks for each component in the GUI that requires a callback. Initially, GUIDE generates just a function definition line for each callback. You can add code to the callback to make it perform the operation you want. The M-file contains two other functions where you might also need to add code:

- Opening function -- performs tasks before the GUI becomes visible to the user, such as creating data for the GUI. GUIDE names this function `my_gui_OpeningFcn`, where `my_gui` is the name of the GUI.
- Output function -- outputs variables to the command line, if necessary. GUIDE names this function `my_gui_OutputFcn`, where `my_gui` is the name of the GUI.

CHAPTER 3

METHODOLOGY

3.0 Introduction

The Figure 3.0 shows the overall view of the system. Appliance interfacing circuit functions to interface with load where the voltage and current will be step down to match the internal device. The loads being connected to the 240 Vrms sources through switching technique which is used relays to be turn on or off manually or automatically. The current and voltage then will be sent to control unit and convert into digital signal by ADC. The data from the ADC port will be record for every one milliseconds and then save it in random access memory (RAM). This process will take for one time interval and at the end of that interval, the microcontroller calculate the power average used in that interval and then save it in EEPROM. However this system running in two modes, the first mode is the control unit of the system will able to stand-alone running without connected to the PC and the second one is the system support by the PC.

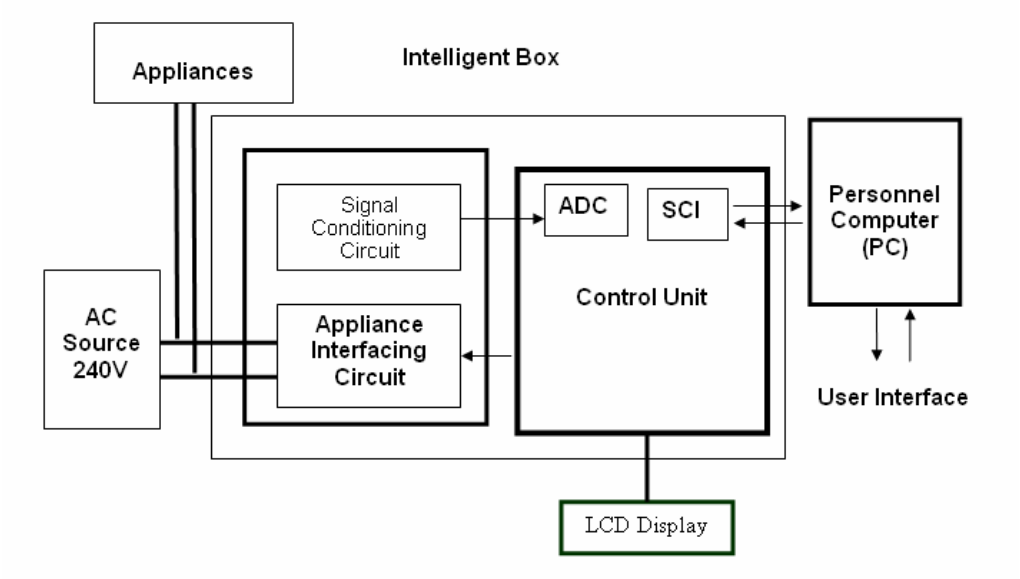
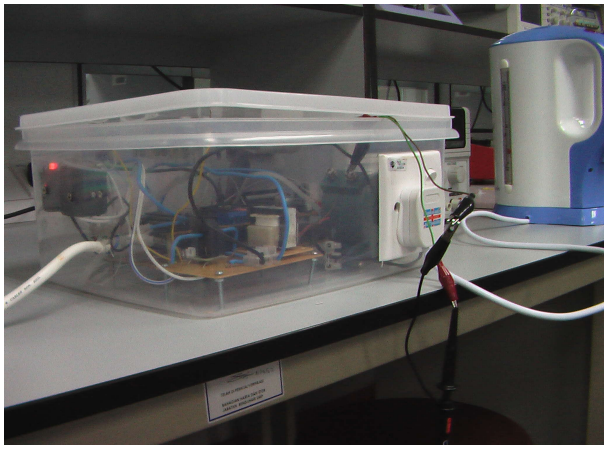
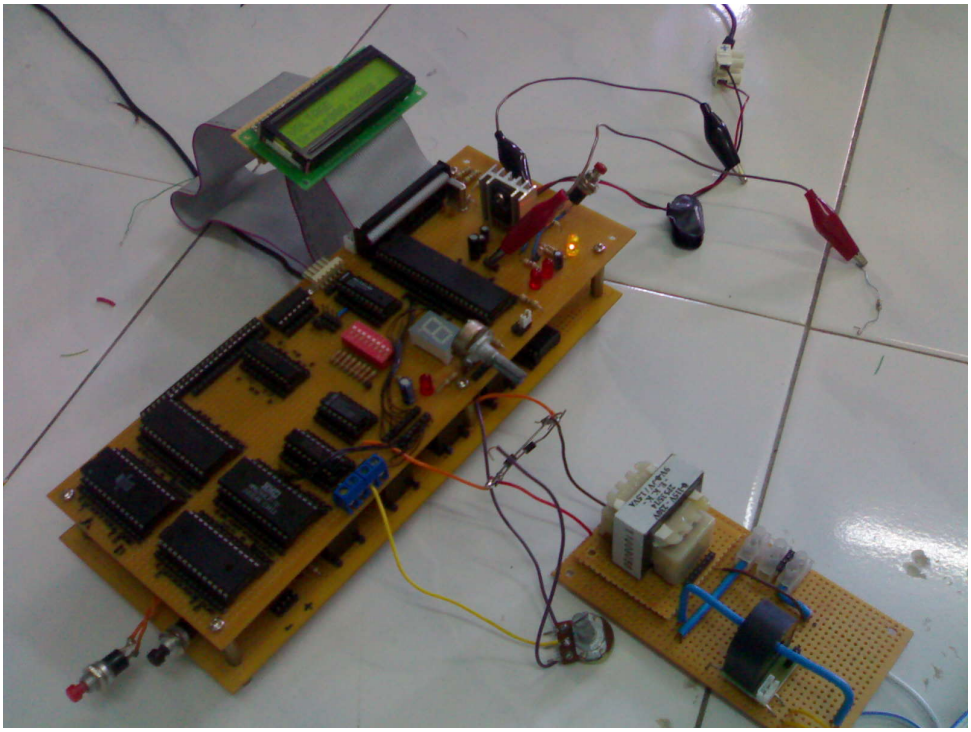


Figure 3.0: Overall view of system design

All of the system design was embedded in a box that called as Intelligent Box. The construction of the intelligent box includes the interfacing appliance circuit and control unit circuit. This box also equipped with an electrical socket, LCD display and some other external features. The sockets will be connected by appliances plug and the data of power dissipation is taken by recording the level of voltage and current. This system also to be expected able to deal with only one loads at one time that supporting by a socket with source of 240 Vrms, 50 Hz. However, more loads could be connected to this socket by using an extension sockets. Figure 3.1 shows the system which been connected to a heater.



(a)



(b)

Figure 3.1: Both figure (a) and (b) show real view of system where the circuit is embedded in intelligent box

3.1 Control Unit - MC68HC11 as controller

In order to design a control unit, some matter must be put into consideration. The consideration should be study based on I/O ports needed, is that microcontroller support ADC and SCI have high volume in storage. Thus, designing controller in this project is based on the development of microcontroller MC68HC11 which is a product of Motorola. Before proceeding into designing this system, the features of microcontroller is need to be revealed. Thus working with MC68HC11 datasheet is quite important to master the basic operation.

Since this project required a controller which is needed to be interfaced with a lot of hardware and also the development of assembly language required high volume of storage device. Thus the designing must be done in **expanded mode** to

match the system operating of this project. Before that, a basic design of circuit needed to be establishing to run a MC68HC11 which is consist of power supply module, reset circuit, and clock circuit.

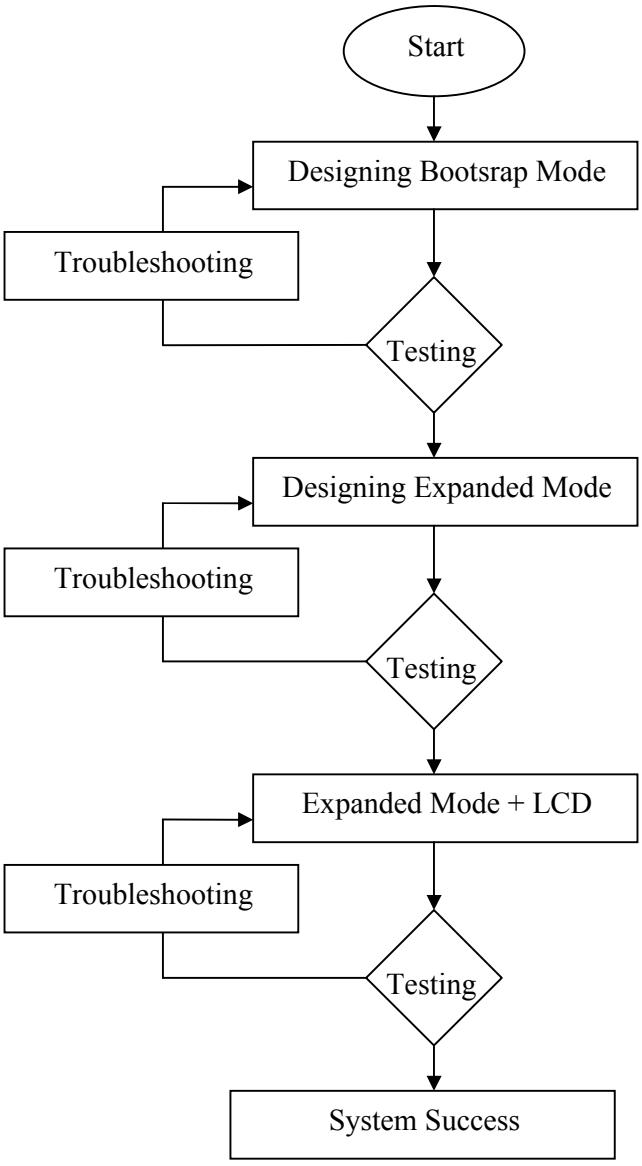


Figure 3.2: The flow of stage in designing control unit

The Figure 3.2 is shown the flow in designing the MC68HC11 as controller. The design is starting by establishing MC68HC11 in bootstrap mode. Then a simple program loaded into memory in the microcontroller to functions a seven segment. Before that, we need to know either circuit working properly or not. This could be done by initialization of circuit using software Wp11. If the circuit able to be initializes its mean that circuit is running properly. Besides of initialization function,

this software is used to load program into EEPROM in microcontroller. A certain program being develop in assembly language and being simulate by simulator in PC. Simulator is important to detect error in developed program since when a user want to convert any error assembly file into machine code will cause the file to be in zero size. Thus in order to avoid unnecessary thing, the developed assembly language should be simulate using THRsim11 software. The develop program need to be save in the format of assembly language (**.asm**) since the assembler only recognize this type of files. The assembler is software to convert the files in assembly format into machine code which is called as as11.exe assembler. . It simply could be done just by dragging file assembly into it as11.exe icon. However, there is a lot of assembler available nowadays and easily could be found at internet but this assembler is only developing to support any 68 families of Motorola's chip and is upward compatible. The output of this method will create a file in format .S19 which is needed to be load into EEPROM. The function of Wp11 is basically to load machine code in format .S19 to microcontroller.

3.1.1 Minimum Design of a MC68HC11 (Bootstrap Mode)

Basically, microcontroller unit has internal CPU, memory and registers. Externally, it has pins for I/O and bus signals. The I/O pins are grouped in sets of eight called ports. For the references see figure 2 where it shows the pin assignments. The MC68HC11 can be operated in different modes. They are single chip, expanded, boot-strap, and special test.

Table 3.0: The table for selecting the mode of MC68HC11

MODB	MODA	Mode Selected
1	0	Single Chip
1	1	Expanded Multiplexed
0	0	Special Bootstrap
0	1	Special Test

The mode selected is determined by how pins MODA and MODB are connected at the time of reset. To set it become bootstrap mode, both pins MODA and MODB must be grounded to get logic '0'. The bootstrap mode is considered a special operating mode as distinguished from the normal single-chip operating mode. This is a very versatile operating mode since there are essentially no limitations on the special purpose program that can be loaded into the internal RAM. The figure 3.3 show the simplified block diagram for bootstrap mode configuration and the Figure 3.4 shows the complete circuit for a bootstrap mode configuration.

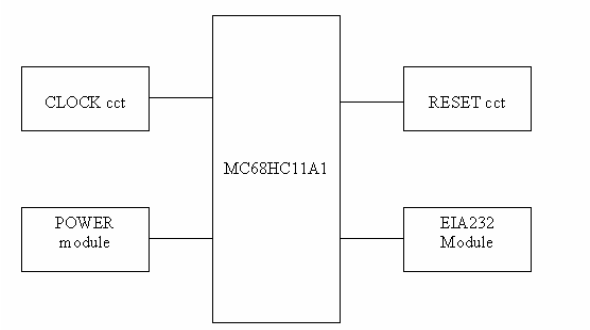


Figure 3.3: A simplified block diagram for bootstrap mode

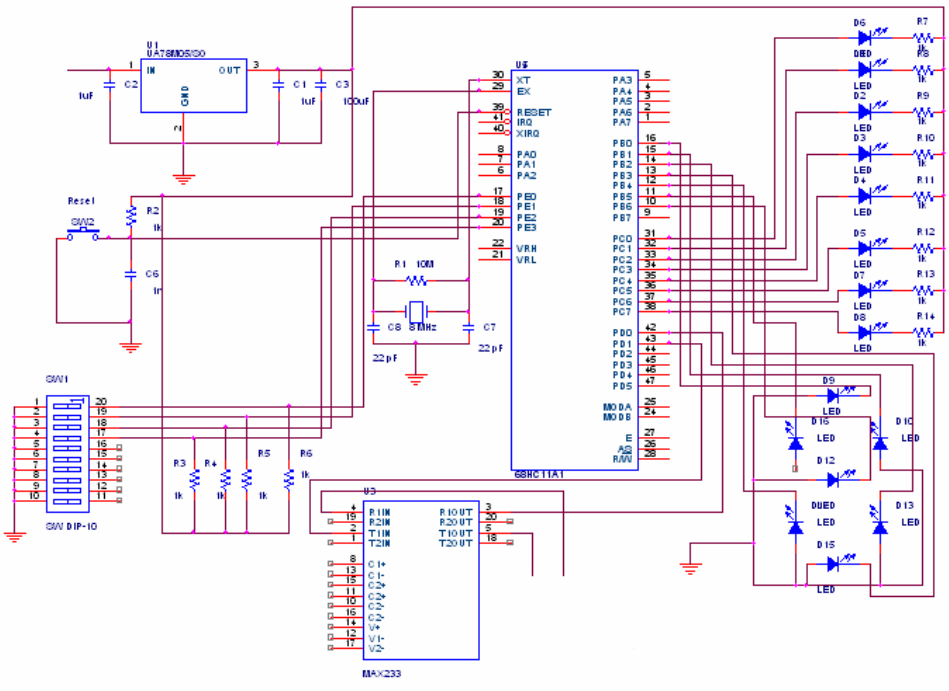


Figure 3.4: Schematic circuit for bootstrap mode configuration

3.1.2 Power Supply Module

MC68HC11 is must be design based on it specification, this is important to ensure the system could operate properly and more important is to avoid permanent damage to the microcontroller. Thus, the main purpose of power supply module is to be as power source to the microcontroller which fulfills the criteria of the MC68HC11. The Figure 3.5 illustrates the circuitry for the power supply module. This circuit functions to supply dc source at fixed voltage level at 5V and avoid the over current from entering microcontroller. The condition could be achieved by using IC regulator 7805 which provide fixed 5V although the load is changed.

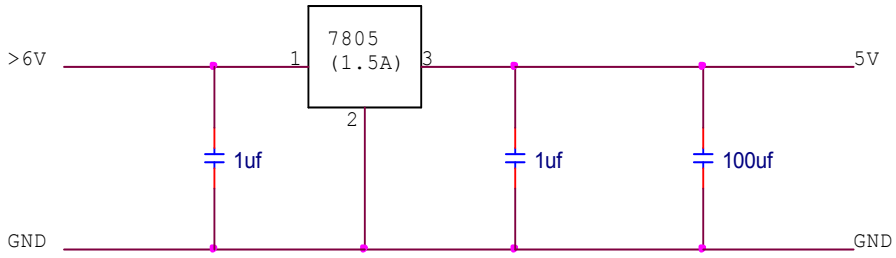


Figure 3.5: Power supply module

3.1.3 Crystal Driver and External Clock Input (XTAL, EXTAL)

As shown in Figure 3.6, these two pins provide the interface for either a crystal or a CMOS compatible clock to control the internal clock generator circuitry. The frequency applied to these pins shall be four times higher than the desired E clock rate. The XTAL pin is normally left without terminated when using an external CMOS compatible clock input to the EXTAL pin. However, a 10K to 100K load resistor to ground may be used to reduce RFI noise emission. The XTAL output is normally intended to drive only a crystal. The XTAL output may be buffered with a high-input-impedance buffer such as the 74HC04, or it may be used to drive the EXTAL input of another M68HC11. In all cases take extra care in the circuit board layout around the oscillator pins. Load capacitances shown in the oscillator circuits include all stray layout capacitances.

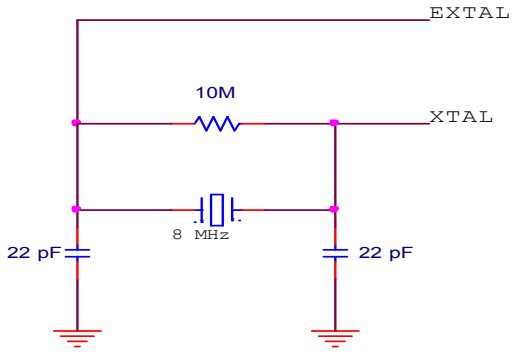


Figure 3.6: Clock circuit

3.1.4 Reset (RESET)

This active low bidirectional control signal is used as an input to initialize the MC68HC11A1/A8 to a known start-up state and as an open-drain output to indicate that an internal failure has been detected in either the clock monitor or computer operating properly (COP) watchdog circuit. This reset signal is significantly different from the reset signal used on other Motorola MCUs. The Figure 3.7 shows the connection for reset circuit.

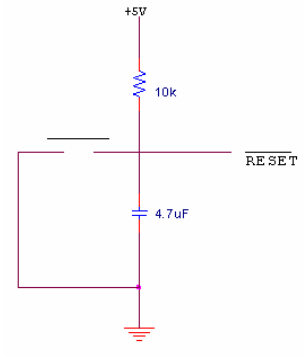


Figure 3.7: Reset circuit

3.1.5 MAX233 as medium to interface to PC

Connection of Rx and Tx is the basic connection in designing bootstrap mode since user need to load program into internal memory of microcontroller from PC. The Rx and Tx is serial data transfer and receive to another devices. The usage of

MAX233 is important because data being transfer in binary form and represented by voltage level as logic 1 and 0. Thus the MAX233 is vital to ensure the noise that occurs during transmission do not give any effect to the voltage level. The usage of MAX 233 can guarantee the accuracy of data transfer can last long for 100 meters. The configuration is shown in Figure 3.8.

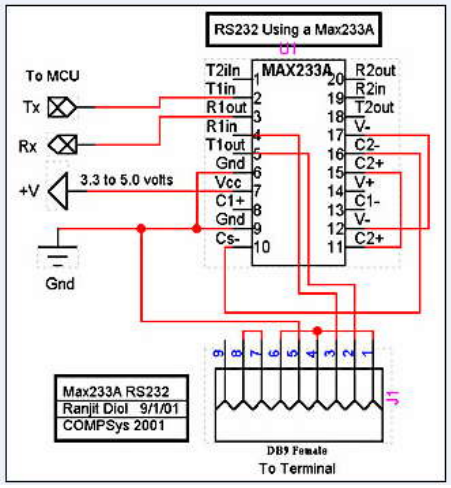


Figure 3.8: MAX233 Interface

3.1.6 Design of a MC68HC11 in expanded mode configuration

After succession in designing bootstrap mode, the mode of operation of MC68HC11 is should be change into expanded mode configuration. The first step before continuing with this mode, user must check the ability of MC68HC11 to be run in this mode. As problem arose in designing this configuration, the serial number of this controller must be check. This is because there are only a few types able to operate in expanded mode. The serial number that starting with ZVZFX#### always the right choice to run expanded mode.

For basic concept of an expanded mode configuration is the port-B and port-C operate as data and address bus. Data bus from D_0 to D_7 and address bus from A_0 to A_8 is shared port-C however both of them being separate by latch which is control by pin AS at microcontroller. Meanwhile for port-C is working as upper byte for addressing. The overall circuit is shown in Figure 3.0. The expanded mode configuration basically used external EEPROM to save the machine code into them.

Since the EEPROM in the microcontroller negligible when it comes to expanded mode, so that why external memory is needed. When it comes to expanded mode, job become more complicate since the process of loading program into EEPROM required a specific device which is supported by software. Not as simple as burning program using Wp11, it required some sort of setting and also the burning process also take a long time to complete.

The basic configuration also include of an encoder to select the external device to be interface with microcontroller. The IC 78LS138 function based on the three upper bits of addresses bus to select the external component. However the EEPROM that function as medium to save the developed program must be activate by the pin Y_7. This is because the address register for RESET is $EFFE where it lies in the range of $E000 to $FFFE which is must be selected by activating pin Y_7. The configuration of expanded mode is shown in Figure 3.0. In order to test expanded mode configuration, a developed program to run a seven segment is quite good to be implement since it only required simple program and simple connection.

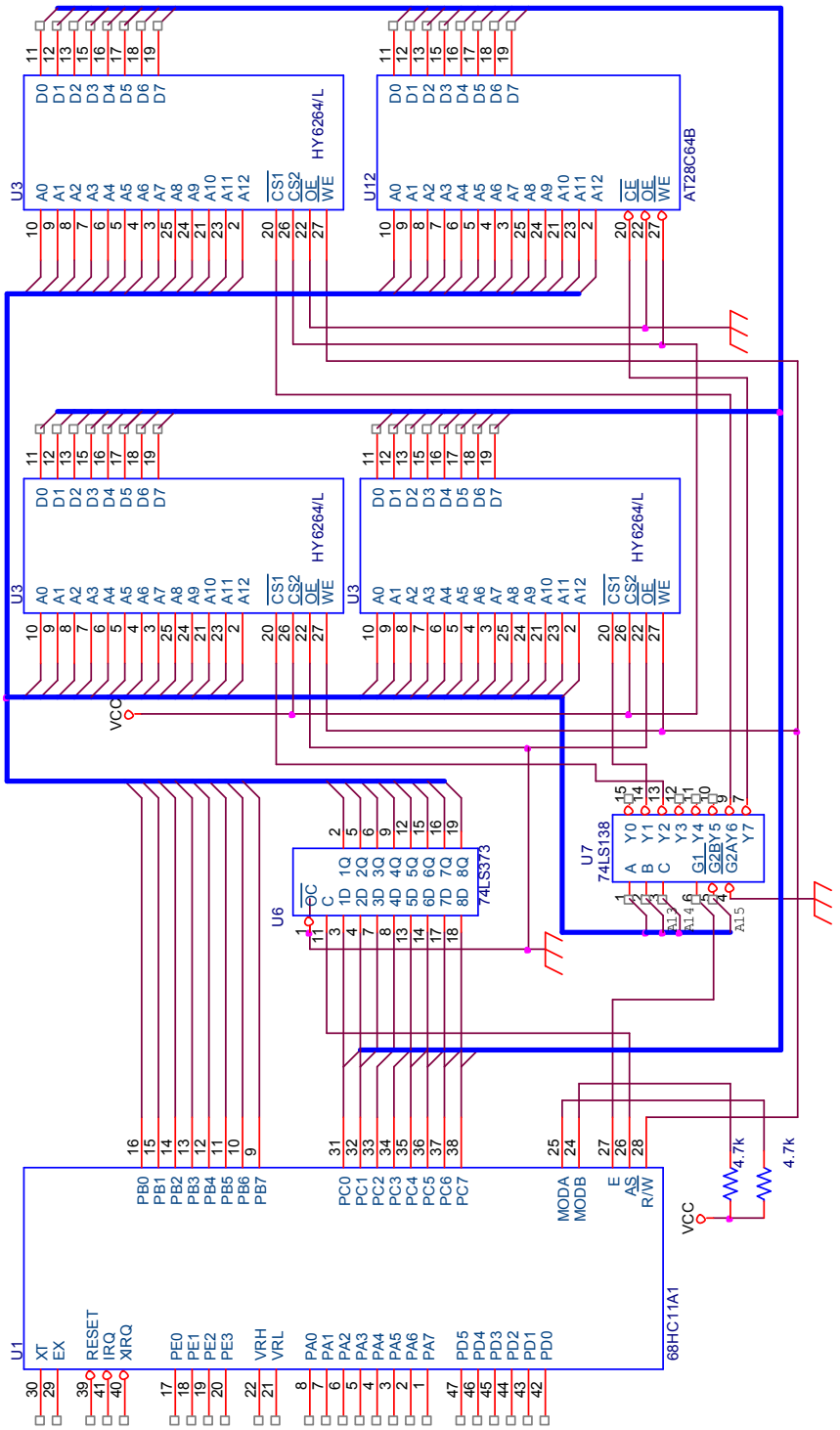


Figure 3.9: Expanded mode configuration

3.1.7 LCD display

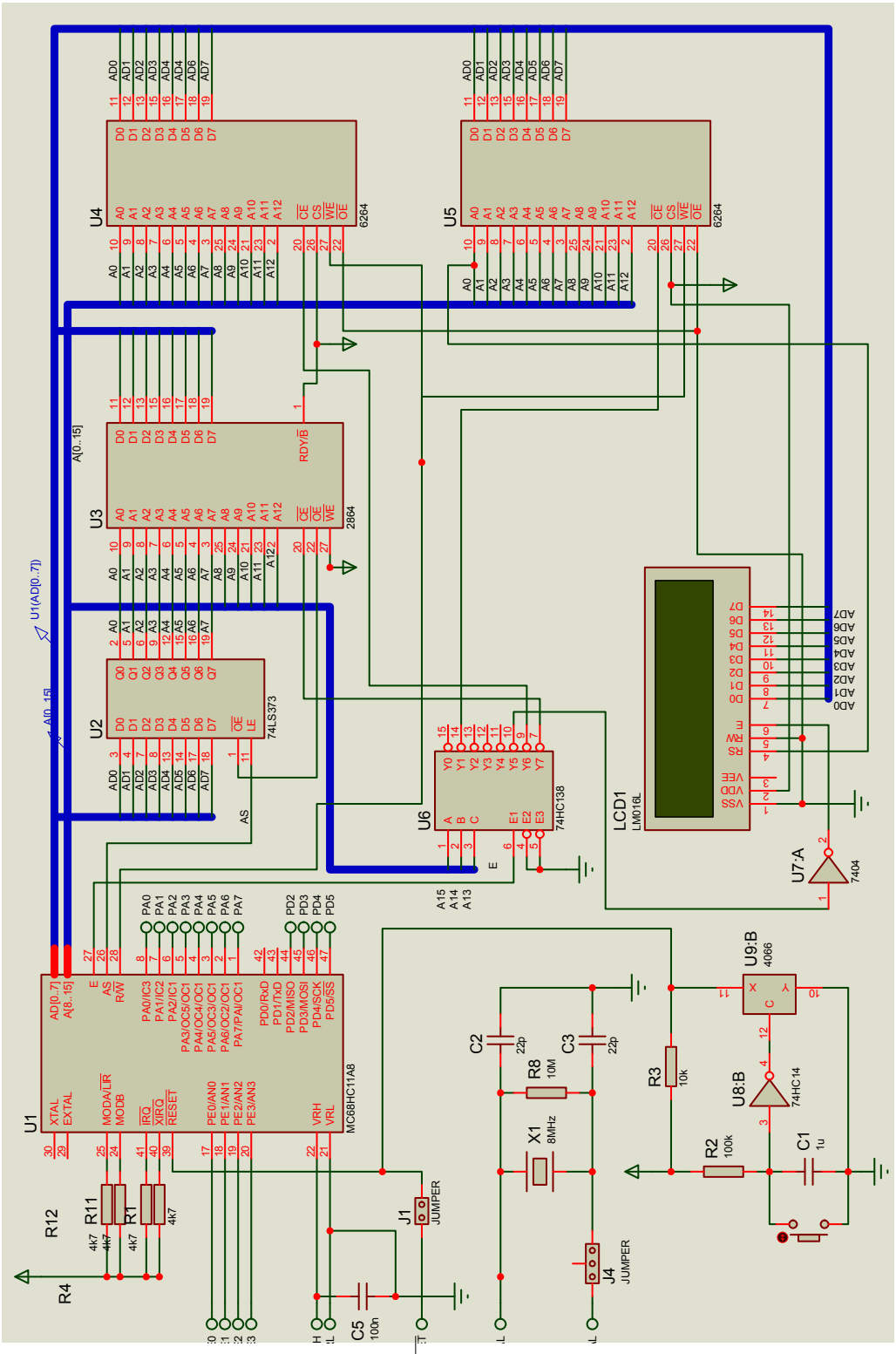


Figure 3.10: Expanded mode configuration with LCD

The schematic diagram in Figure 3.10 is show the connection of LCD to the expanded mode configuration. The LCD display model VCM162A is a double line display which is supported by back light. Table 3.1 is shown the pin assignment for LCD.

Table 3.1: Pin assignment for LCD (VCM162A)

No.	Symbol	Level	Function
1	Vss	–	Power Supply (0V, GND)
2	Vcc	–	Power Supply for Logic
3	VEE	–	Power Supply for LCD Drive
4	RS	H / L	Register Select Signal
5	R/W	H / L	H : Read L : Write
6	E	H, H→L	Enable Signal (No pull-up Resister)
7	DB0	H / L	Data Bus Line / Non-connection at 4-bit operation
8	DB1	H / L	Data Bus Line / Non-connection at 4-bit operation
9	DB2	H / L	Data Bus Line / Non-connection at 4-bit operation
10	DB3	H / L	Data Bus Line / Non-connection at 4-bit operation
11	DB4	H / L	Data Bus Line
12	DB5	H / L	Data Bus Line
13	DB6	H / L	Data Bus Line
14	DB7	H / L	Data Bus Line
15	LED CATHODE	–	LED Cathode Terminal
16	LED ANODE	–	LED Anode Terminal

Before continuing with designing of a LCD, it was vital to master the knowledge about LCD. This is because it was quite hard to understand the operation of LCD display. However once a user start working with it LCD, they will found out that its operation is quite simple and easy.

In dealing with LCD, the user must know some basic functions of LCD in interfacing with processor. First, user must know how to send command to LCD to determine pattern of operation. Then user also needs to know how to send any ASCII code through data bus to display character at LCD screen. Besides that, the function of every pins of LCD must be known in order to be connected to microcontroller.

Table 3.2: Table of command for the LCD

R/S	R/W	D7	D6	D5	D4	D3	D2	D1	D0	Instruction/Description
4	5	14	13	12	11	10	9	8	7	Pins
0	0	0	0	0	0	0	0	0	1	Clear Display
0	0	0	0	0	0	0	0	1	*	Return Cursor and LCD to Home Position
0	0	0	0	0	0	0	1	ID	S	Set Cursor Move Direction
0	0	0	0	0	0	1	D	C	B	Enable Display/Cursor
0	0	0	0	0	1	SC	RL	*	*	Move Cursor/Shift Display
0	0	0	0	1	DL	N	F	*	*	Set Interface Length
0	0	0	1	A	A	A	A	A	A	Move Cursor into CGRAM
0	0	1	A	A	A	A	A	A	A	Move Cursor to Display
0	1	BF	*	*	*	*	*	*	*	Poll the "Busy Flag"
1	0	D	D	D	D	D	D	D	D	Write a Character to the Display at the Current Cursor Position
1	1	D	D	D	D	D	D	D	D	Read the Character on the Display at the Current Cursor Position

Table 3.2 shows the table of command for the LCD. The symbol "*" is represent the "don't care condition" since that bit can be either "1" or "0". The bit descriptions for the different commands are:

Set Cursor Move Direction:

```
ID - Increment Cursor After Each Byte Written to Display if set
S - Shift Display when Byte Written to Display
```

Enable Display/Cursor:

```
D - Turn Display On(1)/Off(0)
C - Turn Cursor On(1)/Off(0)
B - Cursor Blink On(1)/Off(0)
```

Move Cursor/Shift Display

```
SC - Display Shift On(1)/Off(0)
RL - Direction of Shift Right(1)/Left(0)
```

Set Interface Length

```
DL - Set Data Interface Length 8(1)/4(0)
N - Number of Display Lines 1(0)/2(1)
F - Character Font 5x10(1)/5x7(0)
```

Poll the "Busy Flag"

```
BF - This bit is set while the LCD is processing
```

Move Cursor to CGRAM/Display

A - Address

Read/Write ASCII to the Display

D - Data

The interface of LCD is in parallel bus which is allowing simple and fast reading/writing of data to and from the LCD. The waveform of the LCD data write is shown in Figure 3.11.

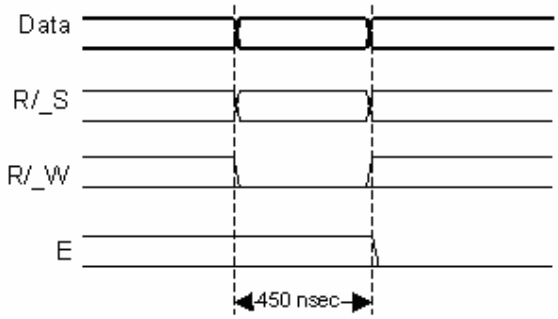


Figure 3.11: Data write waveform

This waveform will write an ASCII byte out to the LCD's screen. The American standard code for information interchange (ASCII) to be displayed is eight bits long and is sent to the LCD either four or eight bits at a time. If four bit mode is used, two "nibbles" of data (sent high four bits and then low four bits with an "E" clock pulse with each nibble) are sent to make up a full eight bit transfer. The "E" clock is used to initiate the data transfer within the LCD. Sending parallel data as either four or eight bits are the two primary modes of operation. While there are secondary considerations and modes, deciding how to send the data to the LCD is most critical decision to be made for an LCD interface application. Eight bit mode is best used when speed is required in an application and at least ten I/O pins are available. Four bit mode requires a minimum of six bits. To wire a microcontroller to an LCD in four bit mode, just the top four bits (DB4-7) are written to. The "R/S" bit is used to select whether data or an instruction is being transferred between the microcontroller and the LCD. If the Bit is set, then the byte at the current LCD "Cursor" Position can be read or written. When the Bit is reset, either an instruction

is being sent to the LCD or the execution status of the last instruction is read back (whether or not it has completed).

Reading data back is best used in applications which required data to be moved back and forth on the LCD (such as in applications which scroll data between lines). The "Busy Flag" can be polled to determine when the last instruction that has been sent has completed processing. In most applications, the "R/W" is always line to ground because there is no condition to read back. This simplifies the application because when data is read back, the microcontroller I/O pins have to be alternated between input and output modes. The "R/W" pin tie to ground and just wait the maximum amount of time for each instruction (4.1 microseconds for clearing the display or moving the cursor/display to the "home position", 160 microseconds for all other commands). As well as making my application software simpler, it also frees up a microcontroller pin for other uses. Different LCDs execute instructions at different rates and to avoid problems later on (such as if the LCD is changed to a slower unit), and the recommend step is just using the maximum delays given above.

In terms of options, the "F" bit in the "Set Interface Instruction" should always be reset (equal to "0") because the LCD is designed to run in 5 x 8 pixels.

Before sending commands or data to the LCD module, the module must be initialized. For eight bit mode, this is done using the following series of operations:

1. Wait more than 15 ms after power is applied.
2. Write 0x030 to LCD and wait 5 ms for the instruction to complete
3. Write 0x030 to LCD and wait 160 us for instruction to complete
4. Write 0x030 AGAIN to LCD and wait 160 µs or Poll the Busy Flag
5. Set the Operating Characteristics of the LCD
    - Write "Set Interface Length"
    - Write 0x010 to turn off the Display
    - Write 0x001 to Clear the Display
    - Write "Set Cursor Move Direction" Setting Cursor Behavior Bits
    - Write "Enable Display/Cursor" & enable Display and Optional Cursor

In describing how the LCD should be initialized in four bit mode, the operation is specifying by writing to the LCD in terms of nibbles. This is because initially, just single nibbles are sent (and not two, which make up a byte and a full instruction). When a byte is sent, the high nibble is sent before the low nibble and the "E" pin is toggled each time four bits is sent to the LCD. To initialize in four bit mode:

1. Wait more than 15 ms after power is applied.
2. Write 0x03 to LCD and wait 5 ms for the instruction to complete
3. Write 0x03 to LCD and wait 160 µs for instruction to complete
4. Write 0x03 AGAIN to LCD and wait 160 µs (or poll the Busy Flag)
5. Set the Operating Characteristics of the LCD
    - Write 0x02 to the LCD to Enable Four Bit Mode

    All following instruction/Data Writes require two nibble writes.

    - Write "Set Interface Length"
    - Write 0x01/0x00 to turn off the Display
    - Write 0x00/0x01 to Clear the Display
    - Write "Set Cursor Move Direction" Setting Cursor Behavior Bits
    - Write "Enable Display/Cursor" & enable Display and Optional Cursor

Once the initialization is complete, the LCD can be written to with data or instructions as required. Each character to display is written like the control bytes, except that the "R/S" line is set. During initialization, by setting the "S/C" bit during the "Move Cursor/Shift Display" command, after each character is sent to the LCD, the cursor built into the LCD will increment to the next position (either right or left). Normally, the "S/C" bit is set (equal to "1") along with the "R/L" bit in the "Move Cursor/Shift Display" command for characters to be written from left to right (as with a "Teletype" video display).

Cursors can be turned on as a simple underscore at any time using the "Enable Display/Cursor" LCD instruction and setting the "C" bit. However, it was not recommend using the "B" ("Block Mode") bit as this causes a flashing full

character square to be displayed. The Figure 3.12 is show how to move the LCD cursor.

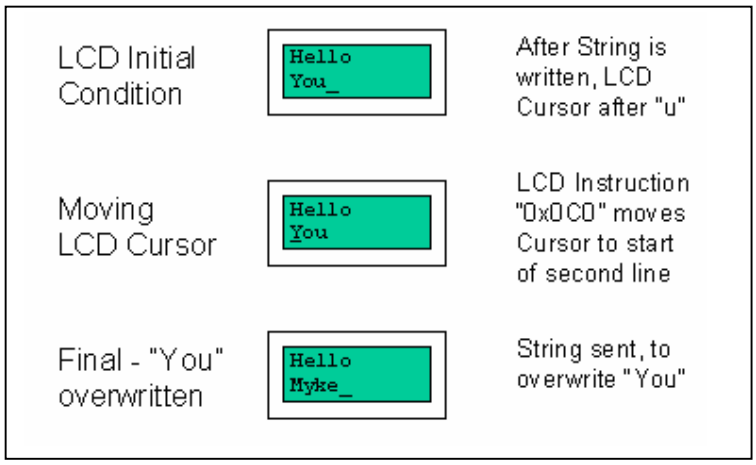


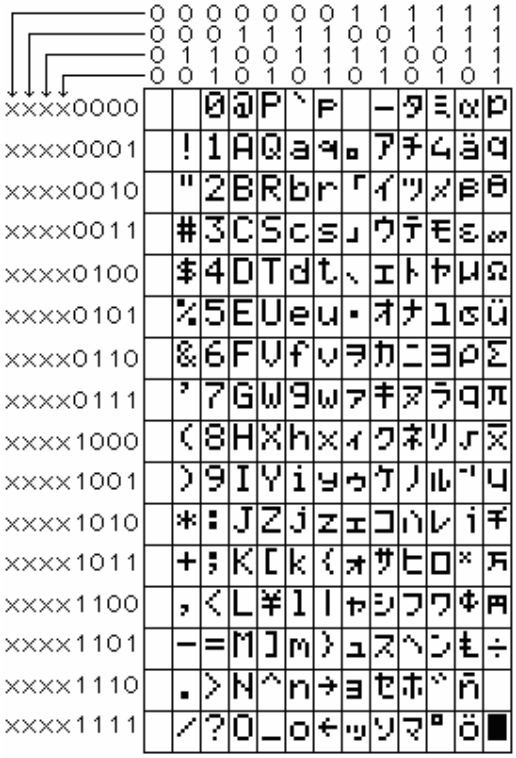
Figure 3.12: Moving the LCD cursor

The LCD can be thought of as a "Teletype" display because in normal operation, after a character has been sent to the LCD, the internal "Cursor" is moved one character to the right. The "Clear Display" and "Return Cursor and LCD to Home Position" instructions are used to reset the Cursor's position to the top right character on the display.

To move the Cursor, the "Move Cursor to Display" instruction is used. For this instruction, bit 7 of the instruction byte is set with the remaining seven bits used as the address of the character on the LCD the cursor is to move to. These seven bits provide 128 addresses, which matches the maximum number of LCD character addresses available. The table above should be used to determine the address of a character offset on a particular line of an LCD display.

It was vital to know that LCD only understands the ASCII code. Thus when the 8-bit data send to LCD, the LCD will defined the data based on drawing certain pattern. This could be illustrating as shown in Table 3.3. Eight programmable characters are available and use codes 0x000 to 0x007. They are programmed by pointing the LCD's "Cursor" to the Character Generator RAM ("CGRAM") Area at eight times the character address. The next eight characters written to the RAM are each line of the programmable character, starting at the top.

Table 3.3: The pattern that display at LCD based on 8-bit data



The pattern could be representing as box with eight squares multiply five as is shown in Figure 3.13. Each LCD character is actually eight pixels high, with the bottom row normally used for the underscore cursor. Using this box, user can draw in the pixels that define the special character and then use the bits to determine what the actual data codes are.

For the "Animate" applications, user can use "character" rotation for the animations. This means that instead of changing the character each time, it simply just display a different character at the same place. Doing this means that only two bytes (moving the cursor to the character and the new character to display) have to be sent to the LCD. If animation was accomplished by redefining the characters, then ten characters would have to be sent to the LCD (one to move into the CGRAM space, the eight defining characters and an instruction returning to display RAM). If multiple characters are going to be used or more than eight pictures for the animation, then you will have to rewrite the character each time.

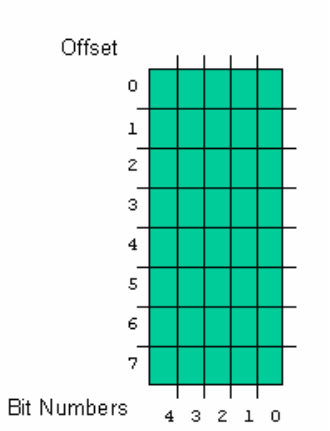


Figure 3.13: Creating custom LCD character by using character box

When user defined character, the line information is saved in the LCD's "CGRAM" area. These sixty four bytes of memory is accessed using the "Move Cursor into CGRAM" instruction in a similar manner to that of moving the cursor to a specific address in the memory with one important difference.

This difference is that each character starts at eight times its character value. This means that user definable character 0 has it's data starting at address 0 of the CGRAM, character 1 starts at address 8, character 2 starts at address 0x010 (16) and so on. To get a specific line within the user definable character, its offset from the top (the top line has an offset of 0) is added to the starting address. In most applications, characters are written to all at one time with character 0 first. In this case, the instruction 0x040 is written to the LCD followed by all the user-defined characters.

Another aspect that should be take into consideration is about the enabling the LCD at pin E. When high logic input given to this pin it means that the LCD is enabled. In expanded mode configuration, IC 74LS138 is used to select component that needed to be enabled. However the output of it IC is logic low when that output pin is selected. Thus, an inverter is useful to convert back the signal into logic high in order to activate the LCD.

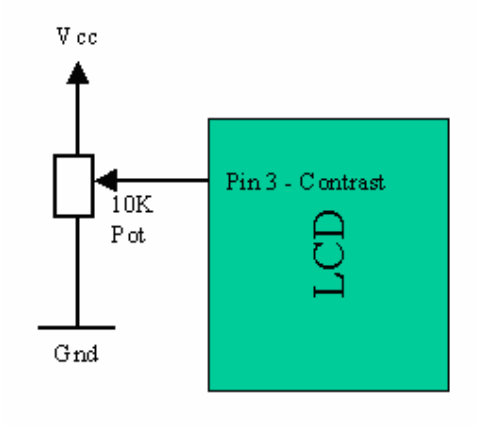


Figure 3.14: Circuit LCD contrast

The last aspect of the LCD to discuss is how to specify a contrast voltage to the display. Typically, a potentiometer is wired as a voltage divider. As shown in Figure 3.14. Potentiometer will provide an easily variable voltage between Ground and Vcc, which will be used to specify the contrast (or "darkness") of the characters on the LCD screen. You may find that different LCDs work differently with lower voltages providing darker characters in some and higher voltages do the same thing in others.

3.1.7 Data management in microcontroller

The designing of control unit is focusing on receiving data of voltage and current in digital form, calculating power consumption, the establishment the system based on application of real time, storing large amounts of data in non-volatile memory (EEPROM) and interfacing with computer through serial communication interface port. Thus designing on microcontroller will be at optimization level to ultimate all the facilities in order to create or provide a lot of features to this system. The measured that taken from sensor is sampled and then quantized into binary form from $0 to $255 since system is used is 8-bit ADC. In microcontroller, the data then being manipulate to be view at LCD and send to the PC. Follow are the equation to illustrate the data manipulation to obtain the voltage and current in rms value and also for the power drop.

- **Current Calculation**

$$Io = \frac{2Ip}{\pi} \qquad \textbf{(3.1)}$$
$$Io(\max) = 13.5A$$

$$Ip = \frac{Io\pi}{2} \qquad \textbf{(3.2)}$$
$$Irms(\max) = 14.985A \cong 15A$$

$$Irms = \frac{Ip}{\sqrt{2}} = \frac{\frac{Io\pi}{2}}{\sqrt{2}} = \frac{Io\pi}{2\sqrt{2}} = 1.11Io \qquad \textbf{(3.3)}$$

Where,

Io = the rectified current in dc level

Ip = peak current

Irms = real value of current in rms

Since the maximum voltage out from current sensor is represent maximum where ***Io (max) =13.5 A*** as shown in equation 3.0**.** The equation 3.3 is the relation between the data taken from ADC with Io.

$$Io = \frac{ADR2}{255} \times Io(\max)$$
$$Io = \frac{ADR2}{255} \times 13.5A$$

$$Irms = \frac{\pi}{2\sqrt{2}} \times \frac{ADR1}{255} \times 13.5$$
$$= \left[\frac{\pi}{2\sqrt{2}} \times 13.5\right] \times \frac{ADR1}{255}$$
$$= 14.99 \times \frac{ADR1}{255}$$
$$\cong 15 \times \frac{ADR1}{255} \qquad \textbf{(3.4)}$$

After substitute equation 3.3 into equation 3.2 results in equation 3.4. Thus the value of current taken from ADR will be converting by the assembly language base on this equation.

- **Voltage Calculation**

$$Vo = \frac{2Vp}{\pi} \quad \textbf{(3.5)}$$
$$Vo(\max) = 225V$$

$$Vp = \frac{Vo\pi}{2} \quad \textbf{(3.6)}$$

$$Vrms = \frac{Vp}{\sqrt{2}} \quad \textbf{(3.7)}$$
$$Vrms(\max) = 249.75V \cong 250V$$

$$Vrms = \frac{\frac{Vo\pi}{2}}{\sqrt{2}}$$
$$Vrms = \frac{Vo\pi}{2\sqrt{2}}$$
$$Vrms = 1.11Vo$$

$$Vo = \frac{ADR1}{255} \times Vo(\max)$$
$$Vo = \frac{ADR1}{255} \times 225$$
$$Thus,$$
$$Vrms = \frac{\pi}{2\sqrt{2}} \times \frac{ADR1}{255} \times 225$$
$$Vrms = 0.98 \times ADR1$$
$$Vrms = \frac{98}{100} \times ADR1 \quad \textbf{(3.8)}$$

Where,

Vo = the voltage in dc level after rectified by full bridge rectifier

Vp = the peak voltage

ADR2 = the voltage after digitized by 8-bit ADC 2 in decimal value

Vrms = real voltage in rms

The equation 3.8 is the simplified equation to obtain the value of voltage in rms. This equation is in purpose of development of assembly language in microcontroller.

- **Power Calculation**

$$
\begin{aligned}
P &= Irms \times Vrms \\
&= \left(\left[\frac{\pi}{2\sqrt{2}} \times 13.5 \right] \times \frac{ADR1}{255} \right) \times \left(\left[\frac{\pi}{2\sqrt{2}} \times 225 \right] \times \frac{ADR1}{255} \right) \\
&= \frac{\pi}{2\sqrt{2}} \times 13.5 \times \frac{1}{255} \times \frac{\pi}{2\sqrt{2}} \times 225 \times \frac{1}{255} \times ADR1 \times ADR1 \\
&= \frac{ADR1 \times ADR1}{17.35}
\end{aligned}
\qquad \textbf{(3.9)}
$$

Where, all parameters could be refered to the equation 3.1 to 3.8

3.1.8 Development of assembly language

The development of assembly language is not being described briefly in this chapter. However it could be referred into appendix to know more about the development of assembly language. The assembly language was being developed by helped of a simulator which is THRSim11 where user was able to test the developed program easily. Besides that, user also can use certain features that available in this software such as LCD display, seven segment, and etc. Actually without helped of this simulator it will be very hard to test the program either to check the error or it functionality.

3.2 Signal conditioning circuit

Signal conditioning circuit is the place where the signal conversion process happened. Signal conversion is a step referring to the modification that must be made to the control signal to properly interface with the next stage control. The devices that perform signal conversion are often called as transducers or sensors. In general, a sensor used to measure some variables in a process control-application and the selection of sensor is based mainly on required measurement specification. Usually the output of a sensor always represented in voltage. For example the temperature sensor (LM35) converts the heat into electrical signal (voltage) where the heat is proportional to the voltage output. A corollary of to the signal-conversion process is the development of special electronic devices that provide a high-energy output under the control of a low-energy input.

In designing this project, two measured variables are defined which is current and voltage. Thus selection of sensors had been made of that both elements which are lead to a signal conditioning circuit that includes two main circuits which is voltage sensor and current sensor circuit. The Figure 3.15 shows the overall view of the signal conditioning circuit.

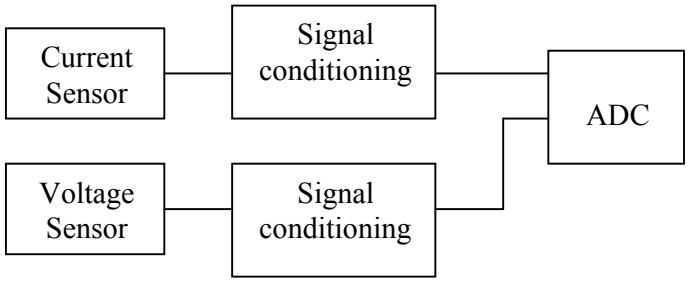


Figure 3.15: The overall circuit for signal conditioning circuit

3.2.1 Current Sensor

A current transformer (CT) is a type of instrument transformer designed to provide a current in its secondary winding proportional to the current flowing in its primary. They are commonly used in metering and protective relaying in the

electrical power industry where they facilitate the safe measurement of large currents, often in the presence of high voltages. The CT safely isolates measurement and control circuitry from the high voltages typically present on the circuit being measured. In order to design a safe system, it was just suitable to use CT for purpose of measuring current coming into a house since it involving high current around 0 to 15 Ampere.

CTs are toroidal in shape and utilize Ampere's Law. Basically, if any current passing through a conductor or wire, it will produce magnetic field around itself. If a CT in toroidal shape putting across the wire, it causes deduction of magnetic field by coil in the CT. Then this condition result in inducing a current in magnetic coil. The magnetic field induces an equal opposing magnetic field and therefore a current in each turn of wire in the CT. The total current induce in the windings of the CT is equivalent to total current flowing through the CT divided by the number of windings. A CT with a 200:5 ratio was selected so that when the outputs from both energized lines coming into the house were combined a maximum 5Amps (AC) of current flowed into signal conditioning circuit.

Nowadays, current transformer has been the standard for precise current measurement in instrumentation and other high reliability equipment applications. They are accurate, easy to implement, and reliable under harsh environmental and thermal conditions. In electronic systems applications such as switch-mode power supplies, current transformers are generally used for control, circuit-protection, and monitoring features. With the increasing availability of Off-The-Shelf (OTS) current transformers, a simple guideline can greatly help in the selection of proper and cost-effective components for many applications.

The selection of a current transformer must begin with the definition and verification of certain factors such as size, frequency, function, and the range of current being sampled – the accuracy and effectiveness will essentially be dependent on these parameters. Aside from the possibility of compromising the transformer's accuracy, using a current transformer above the manufacturer's rated current specification may saturate the transformer and may cause circuit failures due to an uncontrolled rise in operating temperature. On the other hand, a current transformer

that is rated much higher than the “sample current” might be restrictively too large and expensive for its purpose. Typically, selecting a current-transformer that is rated approximately 30% above the expected maximum of the “sample current” is a prudent starting point.

In market, current transformers are available in two different configurations which is split-core and solid-core where each of them having its own advantages and disadvantages. Both type of current transformer is as shown in Figure 3.1(a) and Figure 3.16. From Figure 3.17, it can be seen that the ring of the CT is solid and therefore requires that the conductor to be monitored must be put through the center of the CT. In market, solid-core CT is generally inexpensive and available in a multitude of shapes and sizes.

Figure 3.16: Solid Core Current Transformer

Figure 3.17: Solid Core Current Transformer

Split-core current transformers as the name implies have a split core. That is, the CT may be placed around a service conductor without re-routing the service conductor. This feature makes them much more convenient for installation. However, split-core current transformers are generally much more expensive than solid-core counterparts and also difficult to be found in common shop. For prototyping purposes, a solid-core current transformer had been chose as

manufactured by Talema model ASM-10 as shown in figure below. It has current range from 1 to 10 Ampere with output voltage tolerance ±10%. It has a 200:5 Amp turns ratio and is rated for 5 VA.

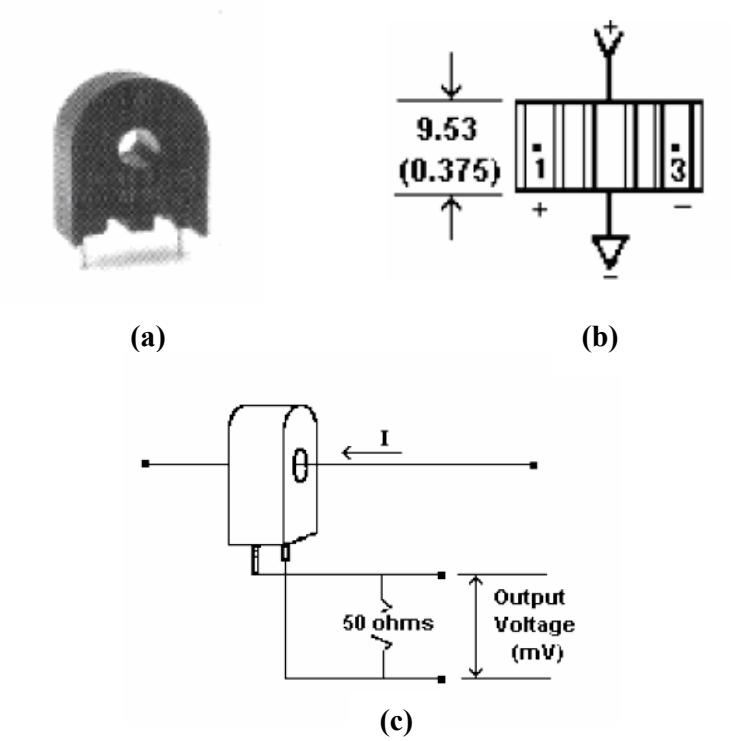


Figure 3.18: **(a)** Solid Core CT used for prototype
(b) The structure of CT
(c) The CT connected in parallel with a 50 ohm resister

3.2.2 The Primary/Secondary Turns Ratio

Off the shelf current-transformers commonly have turns ratios ranging from 1:10 to 1:1000. The higher the turns ratio (r = Nsec/Npri), the higher the resolution of the current measurement. However, care must be taken as too high of a turns ratio will necessitate an increase in distributed capacitance and leakage inductance which may decrease the transformer's accuracy and capability to operate at higher frequencies (due to self-resonance). However, if the number of turns is too low (lower inductance), the output signal may distort or "droop" (in positively sloped

unipolar input signal) which may also cause instability in the control circuit and inaccuracies in measurements.

3.2.3 Inductance and Excitation Current

The current transformer's secondary inductance will determine the fidelity of the output signal. The value of inductance is inversely proportional to the excitation current – which is then subtracted to the "sensed current." The excitation current should be several times less than the magnitude of the sample current (a maximum of 10% is ideal for most SMPS applications) – this will ensure the maximum error tolerance of the transformer. For instance, if a circuit has to maintain a maximum of 10% loss for a sample current of 1 A to 20 A at 100 kHz, the excitation current must be set to a maximum of 100 mA (10% of the minimum sample current value). A 1A sample current will yield an error of 10% while a 20 A sample will yield an error of 0.5%. In case the excitation current is not specified in the manufacturer's data sheet, it can be calculated with the equation:

Where e is the set output voltage (V), L is the inductance (H), and $\left|dI/dt\right|$ is the excitation current w/ respect to time (A/s).

$$e = -L\frac{dI}{dt}; \tag{4.0}$$

$$\left|\frac{dI}{dt}\right| = \frac{e}{L} \tag{4.1}$$

3.2.4 The Output Voltage and "Burden Resistor"

The output voltage (Vo) should be set as low as practically possible to minimize the insertion loss. Regarding to the Figure 3.19, it shows the typical response of CT when certain level of current through it core. Assuming 0.5 V is the optimum secondary output voltage in a circuit and the output current is 20 A, a 1:100

ratio transformer will yield a secondary current of ≅ 200 mA. In Figure 3.20, it shows the connection of burden resistor with output of CT.

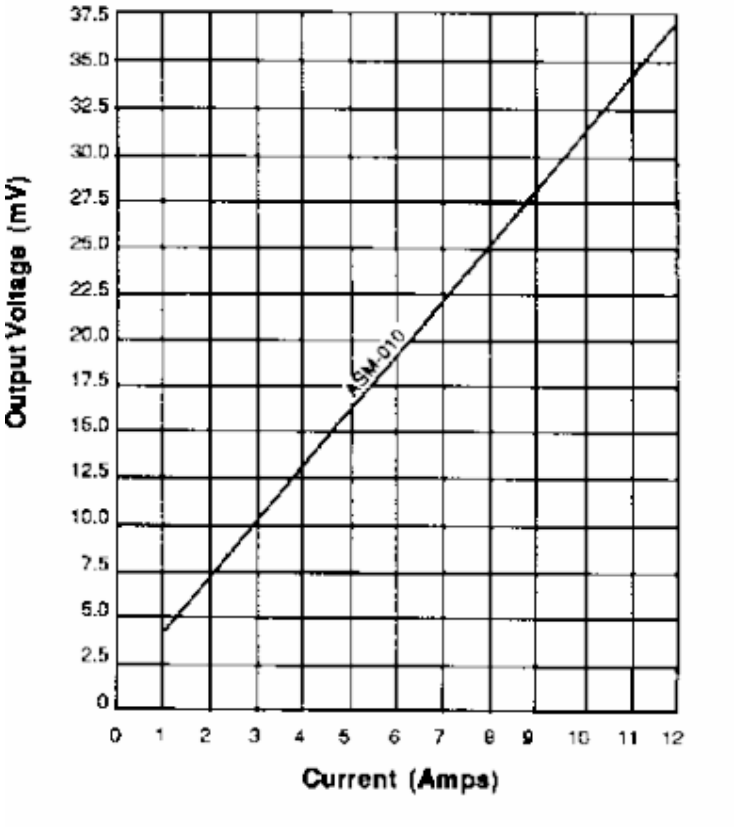


Figure 3.19: Graph show the typical response of CT

$$Ro = \frac{Vo}{Is} = \frac{0.5}{0.200} = 2.5\ \Omega$$

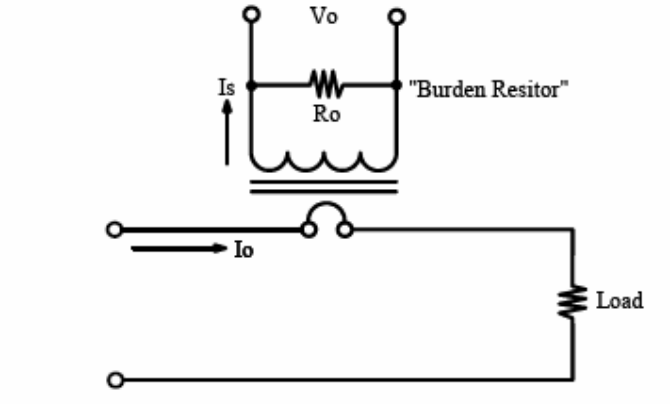


Figure 3.20: Current Sense Circuit

Error Level Approximation:

For 10% Error (neglecting coupling losses), the primary excitation current must be less than 10% of the minimum input current – a maximum of 100 mA in this case;

$$\begin{aligned} \text{Iexc (SEC)} &\cong 1/f * e/L \\ &\cong 1/100\ \text{kHz} * 0.1/(2.5\ \text{mH}) \\ &\cong 0.4\ \text{mA} \end{aligned} \quad \textbf{(4.2)}$$

This yields an approximate current of 40 mA on the primary winding, 60 mA less than the 100 mA maximum. Thus, the burden resistor calculation:

$$Ro = Vo/Isec = 0.1V / (1A/100) = 10\Omega \quad \textbf{(4.3)}$$

OTS components are inexpensive and instantly available, but as discussed in this article, there are functional limitations on their usage. There are applications where specific recommendations or even full customization may be required. It is therefore advisable to procure these components from reputable manufacturers that have strong engineering, manufacturing, and customer service capabilities.

3.2.5 Signal Conversion of Voltage sensor

The purpose of the conditioning circuit is to convert an AC current signal of 0 to 20 Amps (rms) into a dc voltage of 0 to 5 volts. This circuit must limit the voltage to the microcontroller to not exceed $5V_{DC}$. A small 2Ω resistor was put parallel to the current source to obtain a maximum voltage of $10V_{AC}$. Then, a voltage divider was utilized to obtain a maximum $5V_{AC}$ signal. The resistors were chosen so that the output voltage would be the same magnitude as the current entering the circuit. The configuration is shown in Figure 3.21.

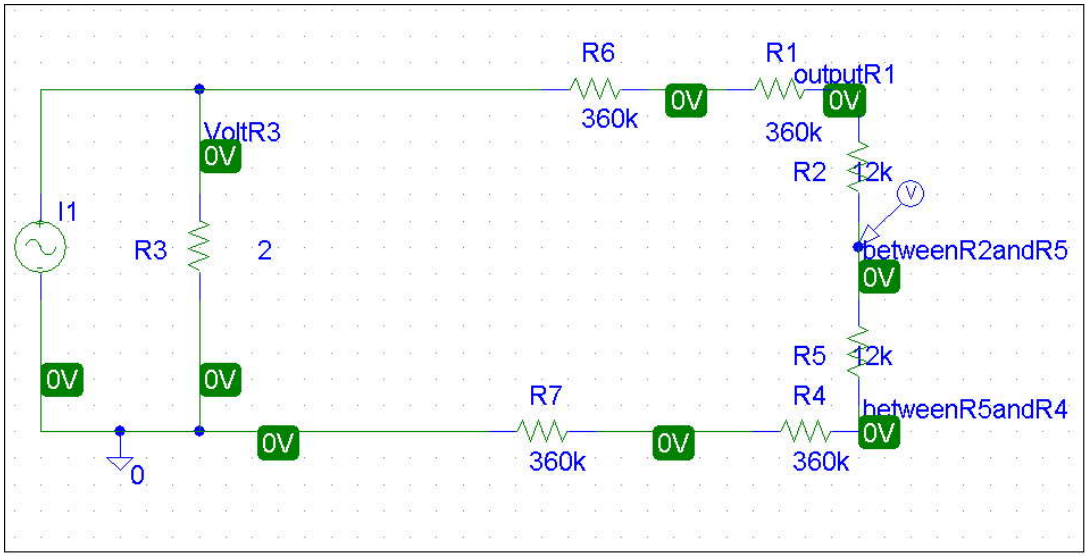


Figure 3.21: Signal conditioning circuit for first testing

However, this circuit provided an AC voltage and offered no voltage limiting on the output. So, a unity gain configured operational-amplifier was used to clip the voltage at 5V using a 5-volt positive voltage rail and the signal was then put through a bridge rectifier to convert the AC signal to a DC analog signal that could connect straight to the ADC pins. The configuration is shown in Figure 3.22.

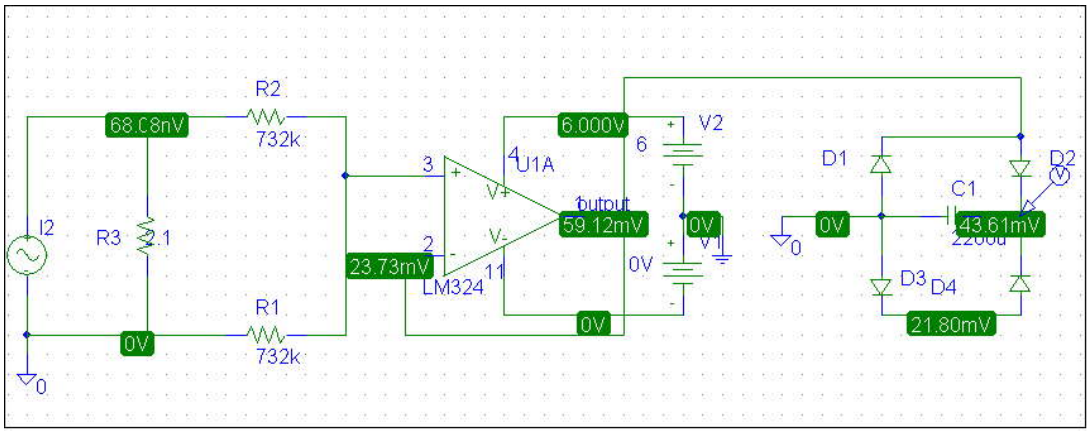


Figure 3.22: Signal conditioning circuit for second testing

Problems that arose: First, the small 2Ω resistor parallel to the CT must be able to dissipate a large amount of power (approximately 50Watts maximum). Five 10Ω, 10Watt resistors were put in parallel to obtain the same 2Ω resistance and split the power dissipation among them. Second, the voltage rails on the op-amp were

clipping the voltage off above 0 volts and below 5 volts. However, the voltage rails must be bipolar (a negative voltage must be applied to the negative rail) or the op-amp will not work properly. To remedy this, two more 1.5-volt batteries were used to obtain a (-3) - volt negative voltage rail for the op-amp. Further testing with this same configuration revealed that the output voltage never exceeding 4 V_{DC}. Since a full range of 0 to 5 volts (DC) is desirable, the each component of the circuit is tested separately.

The 2Ω-load resistance and voltage divider were tested as driven by the CT. The output voltage corresponded directly to the input current. A signal generator was used to test the unity-gain-configured op-amp, and the output of the op-amp was found to correspond directly to its input while limited by the voltage rails.

The bridge rectifier was tested using a signal generator, and it was found that the experimental output voltage was greater than the expected output voltage. After some analysis, the team concluded that this inconsistency was due to DC bias currents inherent to the signal generator. A large 2200 micro-Farad capacitance was placed in line with the AC input of the bridge rectifier to pass the low 60 Hz input but block any DC components, and the testing was repeated. This time, measured output voltages were found to be approximately 1.2 volts lower than the peak of the input voltages. Since this voltage drop is inherent to the diodes of the bridge full-wave rectifier, a design change was made to minimize the voltage error caused by this anomaly.

The team decided to swap the order of the unity-gain op-amp and full-wave bridge rectifier components so that the magnitude of the AC voltage could be increased. The 2Ω resistance was also replaced with a 4Ω resistance, and the initial voltage divider was removed to increase the voltage driving the rectifier. This leads to a larger DC voltage out of the rectifier for which 1.2 volts is a much smaller percentage. A voltage divider is then used to obtain a 0 to 5-volt DC output, which is fed into the unity-gain op-amp to prevent the output to the PIC from exceeding 5 V_{DC}.

This circuit was tested as driven by the current transformers. The output DC voltage was found to correspond directly to the input AC current with a range from 0

V_{DC} to 4.993 V_{DC}. However, when the output of the op-amp was connected to the A/D pin of the controller, it was found that the op-amp did not supply enough current to drive the ADC converter. Instead, the output voltage of the op-amp was being drained to ground when connected to the ADC pin.

An operational amplifier with more available output current may be necessary to resolve this issue. Meanwhile, the unity-gain op-amp has been removed from the circuit to allow more current to drive the ADC pin without significantly affecting the output voltage of the circuit. The conditioning circuit currently being used is shown Figure 3.23. It too, was tested as driven by the current transformers. Its output DC voltage also corresponded directly to the input AC current. However, the circuit as shown here does not yet have any voltage limiting for the output.

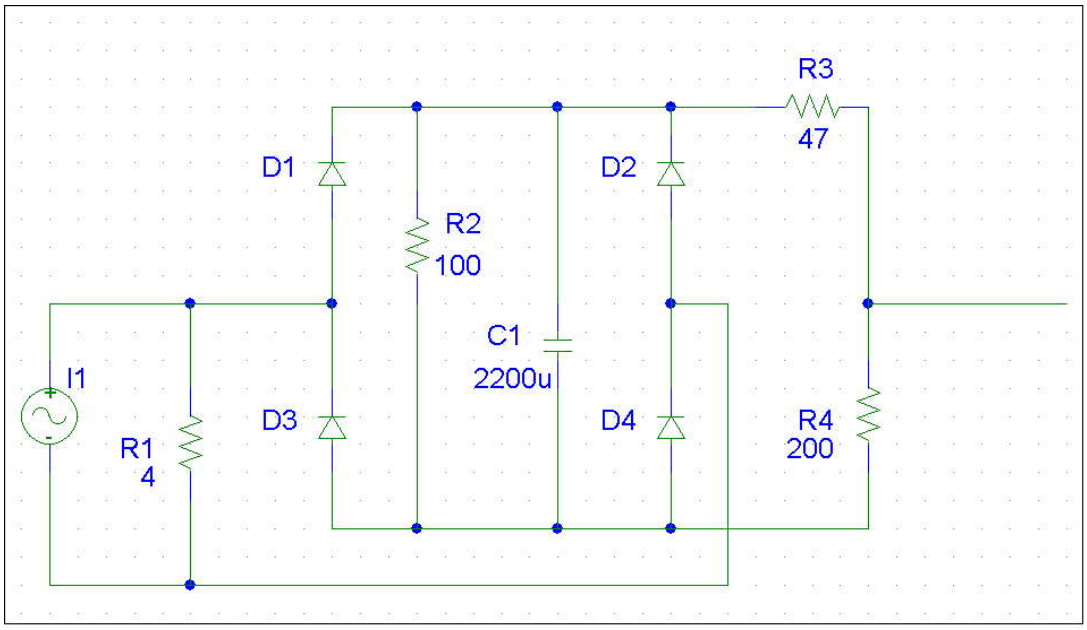


Figure 3.23: Signal conditioning circuit for third testing

The conditioning circuit was revamped to solve all of the prior issues including but not limited to heat dissipation, accuracy, diode-voltage drop, and the necessity for calibration. The following circuit as shown in Figure 3.24 was designed, implemented, and tested to solve all of these design problems.

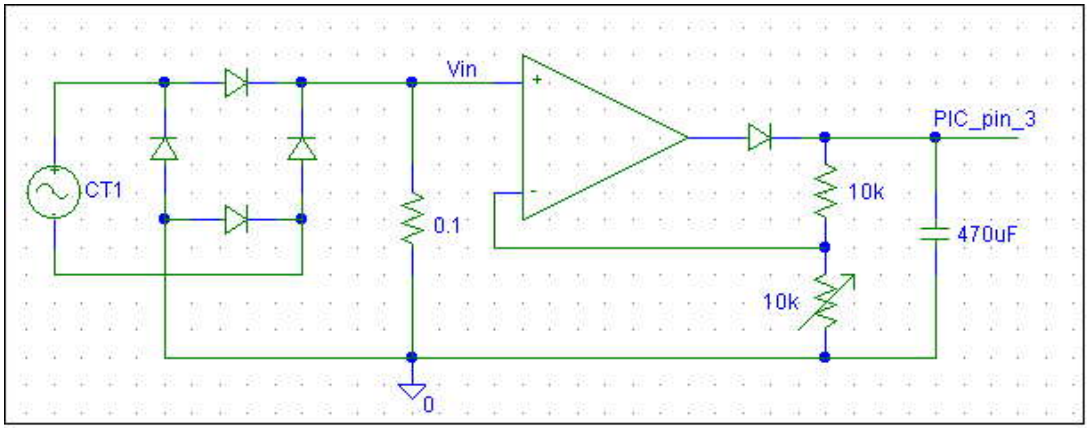


Figure 3.24: The successful circuit for signal conditioning purpose

This circuit is duplicated so that each current transformer has its own conditioning circuit. Placing the bridge rectifier in series with the CT solves the previous problem of diode voltage drop affecting the input to the circuit. Since the CT acts as a dependent current source, the same current created by the CT is driven through the 0.1 ohm load resistance. Thus, the voltage drop across the load resistance depends only on the current produced by the CT. This design does however require the use of power diodes for the bridge rectifier capable of conducting currents as high as 5 amps (worst case). The new design allows for the use of a single-rail op amp. A power op amp (TLV 4112) was selected due to the current requirements for driving the input pin of the microcontroller's A/D converter. The feedback loop of the op-amp is connected in a positive feedback configuration with a variable resistance. This allows calibration of the output voltage due to variations in resistance values and even CT winding ratios. The 470uF capacitance, diode, and choice of resistance values are designed to obtain an RC time constant of at least 5 times the cycle time (1/60 seconds). The diode on the output is necessary due to the low output resistance characteristic inherent of op amps.

For packaging and installation purposes; the CT, bridge rectifier, and load resistance are all packaged together. This solves a couple of potential user problems. First, if the CT is placed around a current carrying conductor without a conducting path, currents will be produced inside the core material and material breakdown is possible (i.e. the CT could fail before the circuits were ever connected). Secondly, this eliminates the need to pay attention to what direction the CTs are facing when

installed. Either orientation of a CT still produces the same waveform at the output (across the load resistance). Only the polarity of the leads from the CT to the circuit must be kept the same, and that can be accomplished through connector choice. For this, we chose to use RCA style connectors.

3.26 Voltage sensor

A transformer is a device that changes AC electric power into AC at another voltage level through the action of a magnetic field. It consist two or more coils of wire wrapped around a common ferromagnetic core. These coils are usually not directly connected. The only common connection between the coils is the common magnetic flux present within the core. One of the transformer winding is connected to a AC electrical power and the second (perhaps third) transformer winding supplies electrical power to load. The winding which is connected to the power source is called as primary winding and the winding connected to load is called as secondary winding and if any present of third winding; it is called as tertiary winding. Figure 3.25 illustrate the basic structure of a transformer.

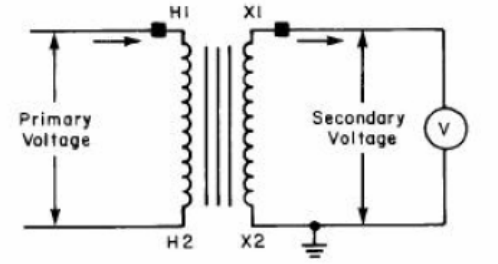


Figure 3.25: The basic structure of voltage transformer

The application of transformer in this part quite important since the measured voltage is around 240 Vrms. Since the value of the range of voltage which is connected to ADC pins is varied from 0 V to 5 V. Its mean that a step down process is required to change the voltage level to be present 0 Vrms to 240 Vrms into range of 0 V to 5 V in DC voltage as an input to ADC pin. In order to design a perfect voltage sensor, the selection of voltage transformer is a necessary. Thus, the specification of a voltage transformer must fit with the design requirements. Since

then, a voltage transformer with rating 240:12 manufactured by Telectron model T-1201H.

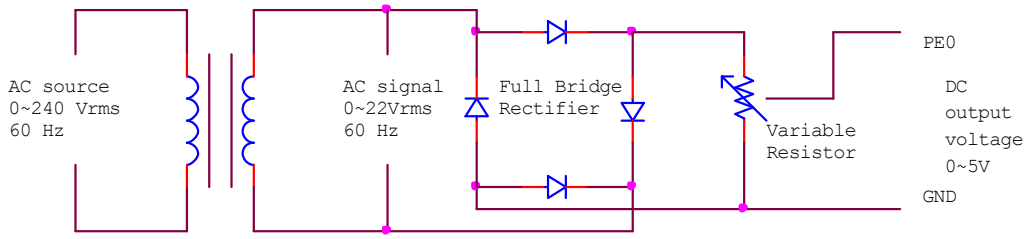


Figure 3.26: The diagram of voltage sensor

The Figure 3.26 is shown the simple idea on designing voltage sensor. As illustrated, the input winding is connected to source 240 Vrms which is in parallel with power supply to the load. Then the transformer stepped down the voltage level of 240 Vrms to 12 Vrms at output winding. Based on the output voltage, it is still in AC level and must converted in DC level. In order to obtain DC level of the secondary winding, a full bridge rectifier must be constructed where the bridge circuit is constructed by using four diodes model IN4006. However the output of bridge it is still excluded of the range 0 to 5 V that necessitate to be fed to ADC port because it may up to 12 V in DC. Just in case, the usages of diodes contribute to the voltage dissipation across them. However the voltage drop across every diode is negligible since the output of voltage is still high. Since the voltage level after bridge is still out of the range, a voltage divider technique is suitable to get the voltage in required range and it could be achieve by applying a potentiometer. However the usage of potentiometer is not fairly good since it do not operate well in high current. Thus, an approaching step was been taken in solving this problem. It was that the function of potentiometer is replaced by using tungsten's wire. This conductor is well known in industry and always been used as heat element in heater, soldering, and so on. The ability of the conductor to maintain it resistance although the current is really high make it useful in designing as voltage divider.

The output of voltage divider however is still not suitable to be directly fed into ADC port because it high in current which could give permanent damage to the port. Thus a resistor should be put in series with output of voltage divider to reduce the current flow into microcontroller. The calculation of the output voltage is as

below where equation 4.4 is for finding turns ratio and equation 4.5 is to find the dc level current after bridging by a full wave rectifier circuit.

$$N_1 / N_2 = V_1 / V_2 = 240/12 = 20 \quad \textbf{(4.4)}$$

$$V_{rms} = \frac{V_m}{\sqrt{2}}$$

$$V_m = V_{rms}\sqrt{2}$$

$$V_o = \frac{1}{\Pi}\int_0^{\Pi} V_m \sin(\varpi t) d(\varpi t) \quad \textbf{(4.5)}$$

$$= \frac{2V_m}{\Pi} = \frac{2V_{rms}\sqrt{2}}{\Pi}$$

$$= \frac{2(12)\sqrt{2}}{\Pi} = 10.8$$

The output of the voltage divider is could be obtain from equation as follow:

$$V_{R2} = \frac{R_2}{R_1 + R_2} \times V_o = \frac{R_2}{R_1 + R_2} \times 10.8$$

Where,

N1 = the number of winding at primary

N2 = the number of winding at secondary

Vm = the peak voltage

V_{R2} = the output of voltage divider

R_1 = resistor 1

R_2 = resistor 2

3.3 Appliance interfacing circuit – switch

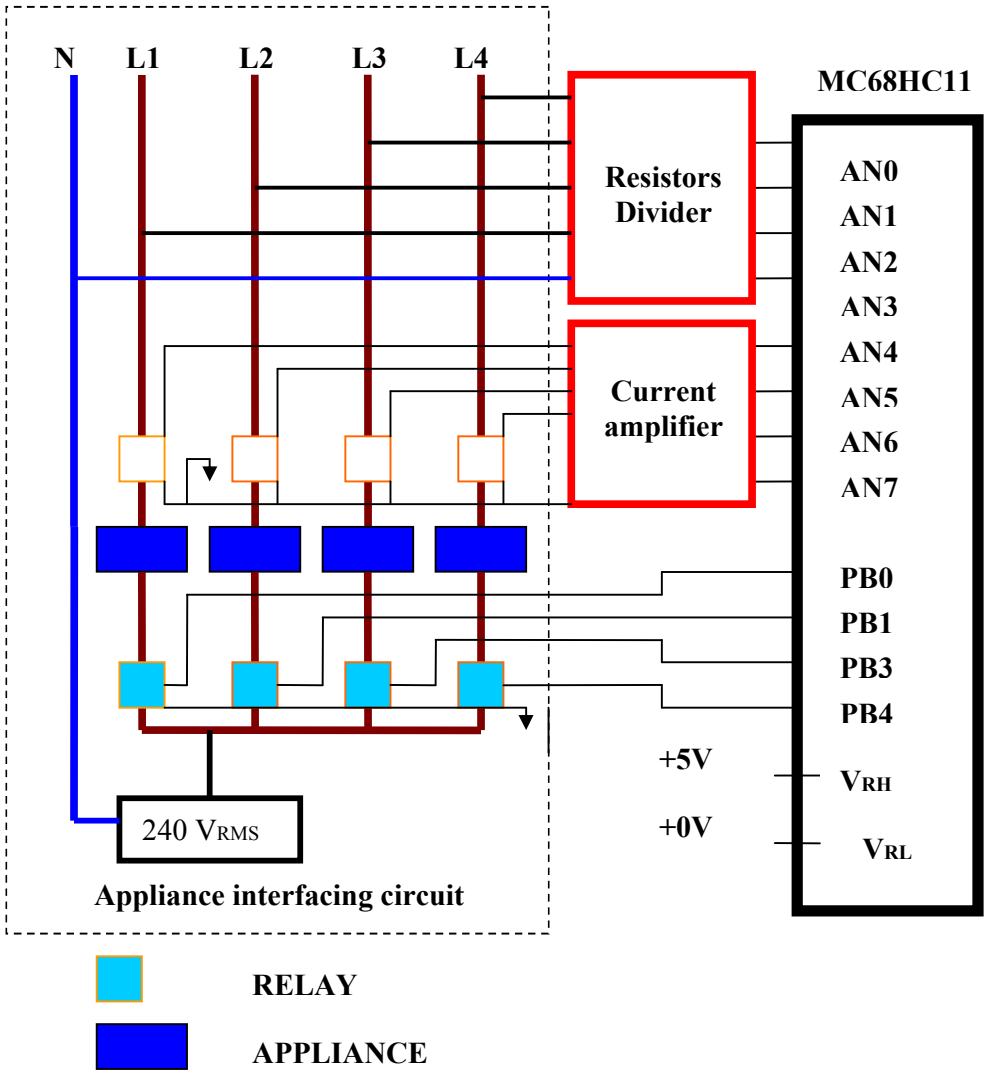
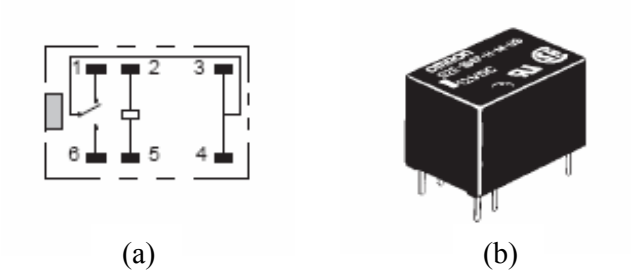


Figure 3.27: The basic concept of interfacing the appliances

Referred to the figure 3.27, the appliance interfacing circuit contained the devices for switching purpose. In this project, the voltage source of 240 Vrms sometimes need to disable and some other time need to enable. However deals with high voltage reveal to high risk. Furthermore, this source has to enable or disable through digital control which is only deal with electronic circuit. Thus, high voltage must be isolate from digital circuit. In solving this problem, relays seemed to be useful in isolating high voltage and also for digital control purpose. The structure of a relay is shown in figure below. As shown in figure a relay normally has a magnetic

coil, a normally closed (NC) and a normally open (NO) contactor. When certain voltage level consume to magnetic coil, the coil will be energized and produce a magnetic field which will pull a contactor that stay in normally closed condition to normally open state. The special of this device is we only need a small voltage to energize a coil but the contactor able to support until 240 Vrms for AC source and 48 V for DC source. However users still needs make necessary reference to manufacture datasheet to know the boundary of operation.



(a) (b)

Figure 3.28: **(a)** A relay
(b) Structure of a relay

As shown in figure above, relay is used to operate as switch at life (L) wire for each load. However in designing interfacing circuit, five relays are needed to cover switching process for a load. It means that, if there is more sockets provide at this system so more relays are required. In this project, the source of 240 Vrms connected to an outlet socket which is become as main supply to the appliance. There is only one outlet socket that used as prototype. However, system still could design to add another socket to make the system able to measure current and voltage for many loads at same time. In this case, system needs to be adding more current transformer because each load must be measured by different CT. The option to design with many outlet sockets is not efficient since it will consume high coast and required the usage a lot of ADC and output port at microcontroller.

The design of circuit of interfacing circuit is as shown in Figure 3.29. Basically this system was designed that capable to be on or off through clicking button at control panel in GUI. Besides that, switching also could be done by pushing external push button. The special design of this switching is that when load being on by external switch but it still able to be off by PC's and also same condition if load

being on by PC's command. The basic idea of switching from pc is here; when a button being clicked at control panel (GUI), it sends certain character to microcontroller. In microcontroller, it will compare that data that are taken to recognize the type of instruction. If instruction on the load, it will send PA3 with logic '1'. Then this level of voltage energized coil in the relay caused contactor to change it current state from NC to NO. However the current from microcontroller is not enough to drive a relay, thus external source must be giving to the coil by using switching concept of transistor. There is one more thing should be take care, the output of microcontroller may last longer, thus a hold circuit is vital to ensure 240 Vrms is always on to the load start from beginning.

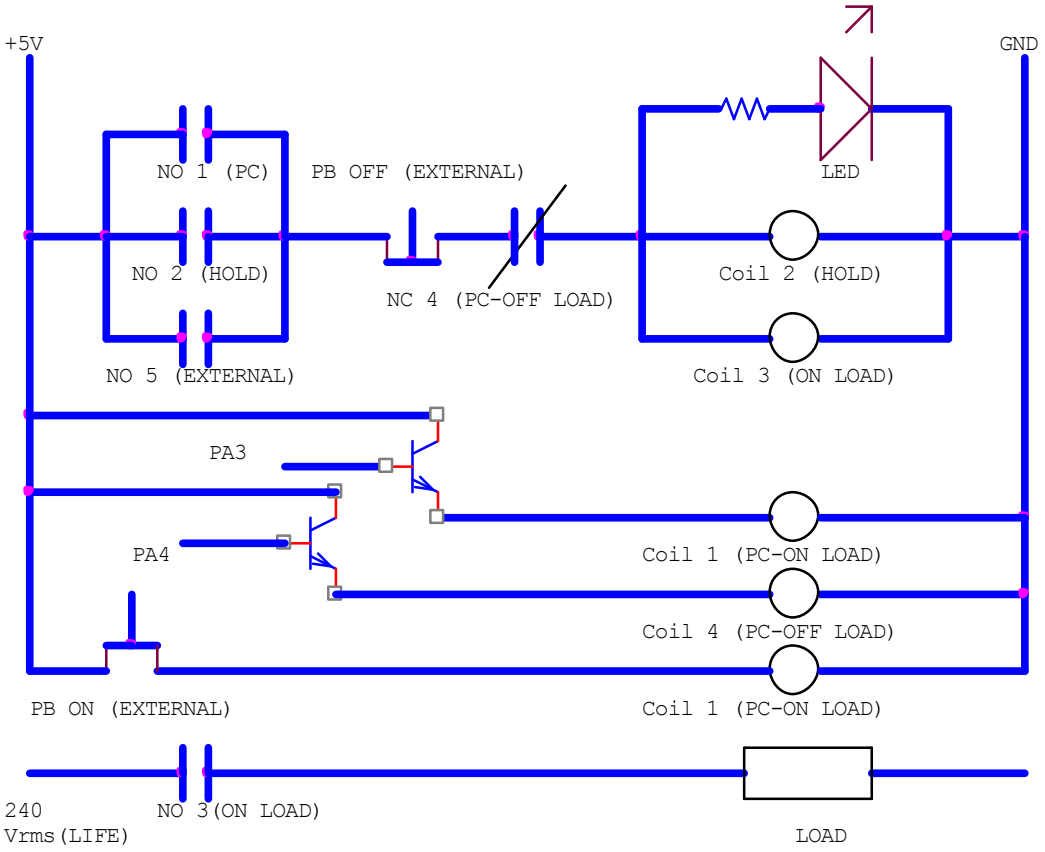


Figure 3.29: Physical ladder diagram for the controlling purpose

3.4 LCD display as a meter

The basic purpose of the designing this circuit is to monitor the power dissipation of a house. Based on the main design of this project, the management of this supposed to be done by a PC. Development of software is necessary to become as an energy meter where data manipulation is done via control panel at GUI. However it was impossible for a PC to be working for 24 hours in a day. Thus, an external meter is an excellent solution to this matter. A LCD (liquid crystal display) display was been choosing as replacement to the PC as display meter.

It is desirable for the display to show the power drop of the electrical appliances at instant of time. Besides that, LCD is not only function as power meter, but it also designed to display the instant value of current and voltage. By adding this features the system is seem to be more practical to be as controller to organize the usage of energy at home. The functions of LCD are listed as follows:

- External guidance to user
- Voltmeter
- Current meter
- Power meter
- Status of operation
- Warning sign

As the system started to operate, LCD will exhibit the initial welcoming display to the user. Then LCD will show instruction which is asked the user to select the mode of operation of the system. The selection of mode is based on displaying either to show current, voltage or power. The selection can be done in two ways either user can choose the mode by selecting external switch which is available outside of the intelligent box or through GUI.

3.5 Development of MATLAB to create GUI

3.5.1 Introduction

Development of PC is defined as development of certain software which is provides the features of Graphical User Interface (GUI). Thus, the software that was used to create GUI is MATLAB version 7.1. MATLAB was been chosen because working with MATLAB is more easy compare to another software. In order to make connection between PC and hardware, MATLAB supposed to be able to communicate with in bilateral with microcontroller. In achieving this objective, command or coding for transmits and receive is should be develop. Generally, interfacing using GUI always been connected to the developing of panel which is for controlling purpose meanwhile for intention of displaying the measurement of current, voltage and power. The signal from current and voltage sensor will be display on GUI using microcontroller as the interface medium. It could be representing in graph form which is easily design in MATLAB.

3.5.2 Step to create MATLAB GUI Development Environment

1. Type GUIDE at the command window. The command window is as illustrate in Figure 3.30.

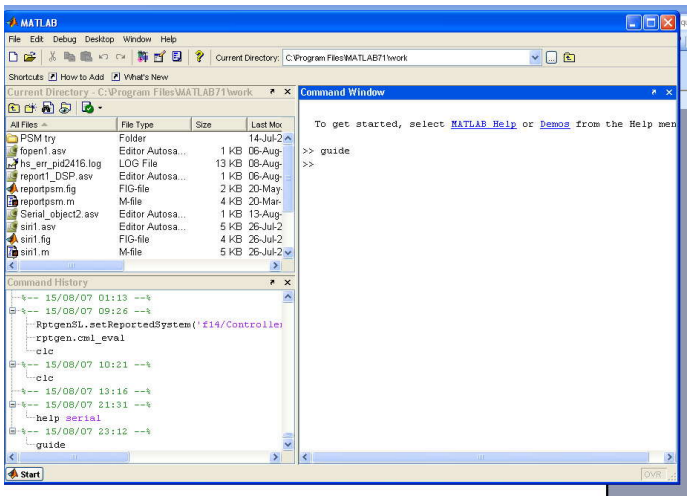


Figure 3.30: The command window

2. When user clicked on the tool bar as direct by arrow. The GUIDE start dialog will pop up and select the pre-built templates. Use the default blank GUI template. It was as shown in Figure 3.31.

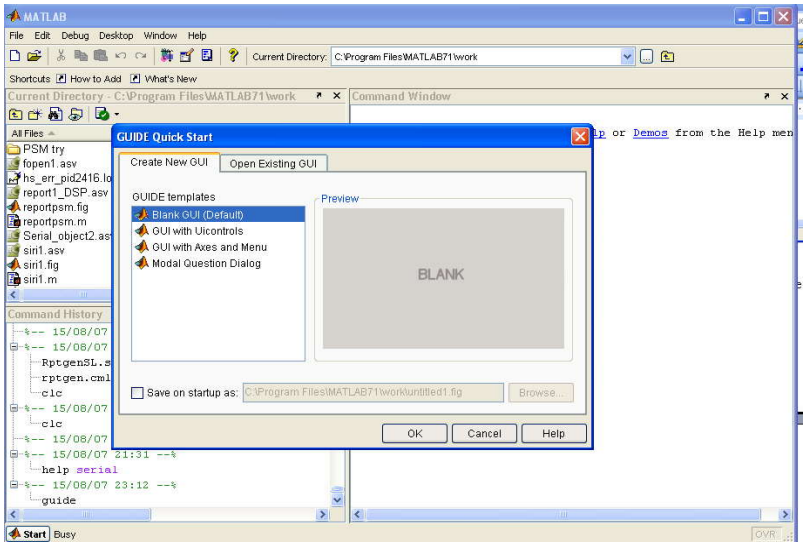


Figure 3.31: The GUIDE start dialog is pop up

3. The GUIDE layout editor that appears for design the layout of GUI.

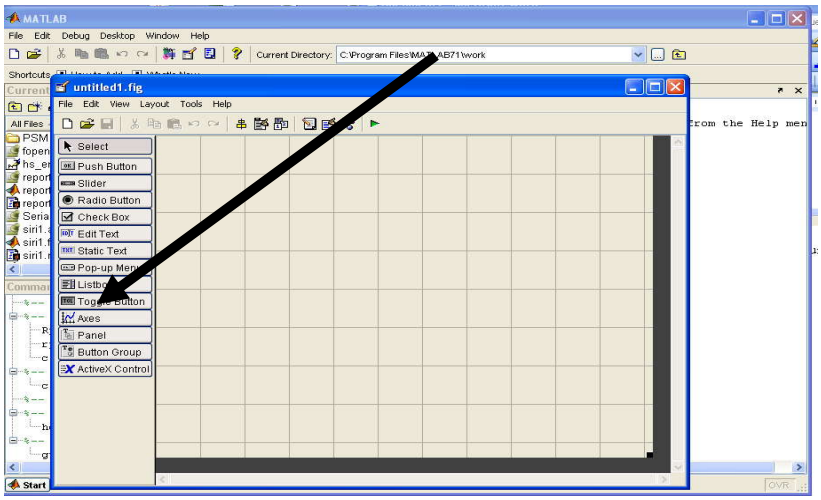


Figure 3.32: Creating GUI using components available in the pallet

4. Resize the GUI's layout is vital since the size of layout will be same as GUI run.

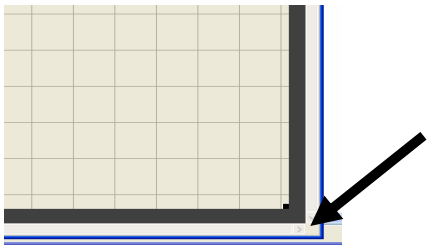


Figure 3.33: Resize the layout

5. Add push button, pop up menu and axis from the pallet to the layout by left click and drag on the layout.

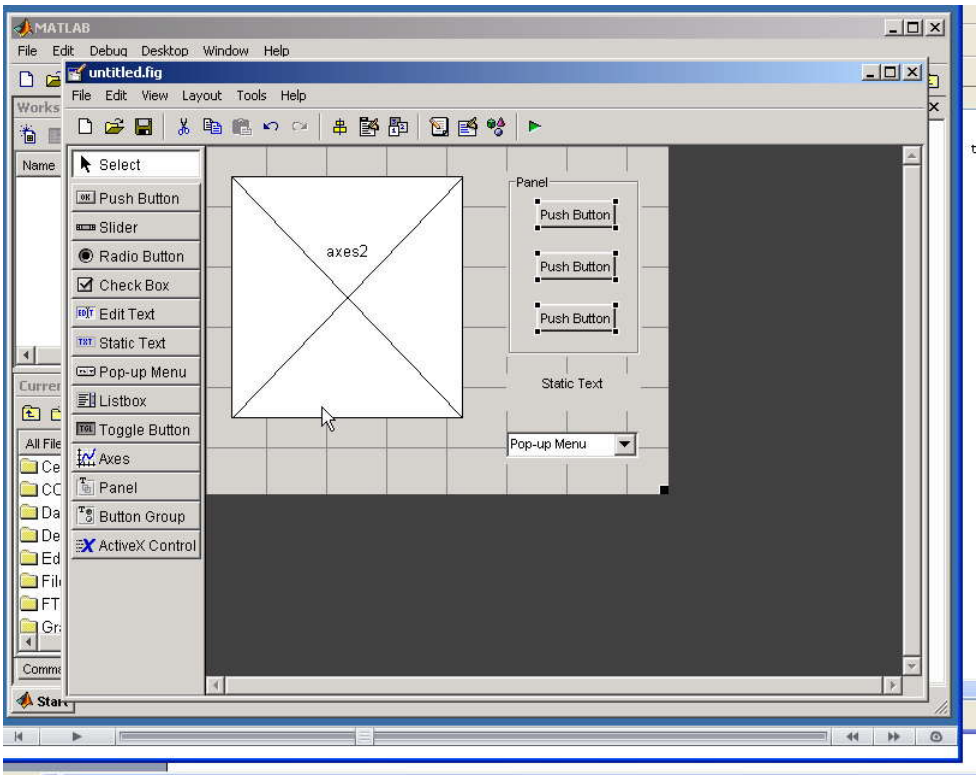


Figure 3.34: Selecting any menu from layout editor to create GUI

6. Open the Property Inspector at View > Property Inspector to change the name of pushbutton, Panel and other function name.

7. Run the GUI by click the green button

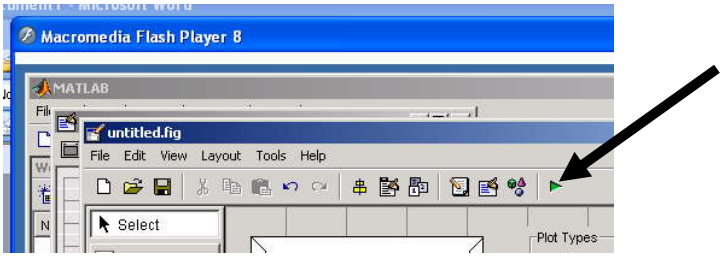


Figure 3.35: Icon to run the GUI

8. After click the green button, save the GUI M-file layout, the automatic generate code will display at M-file. Click the F symbol too.

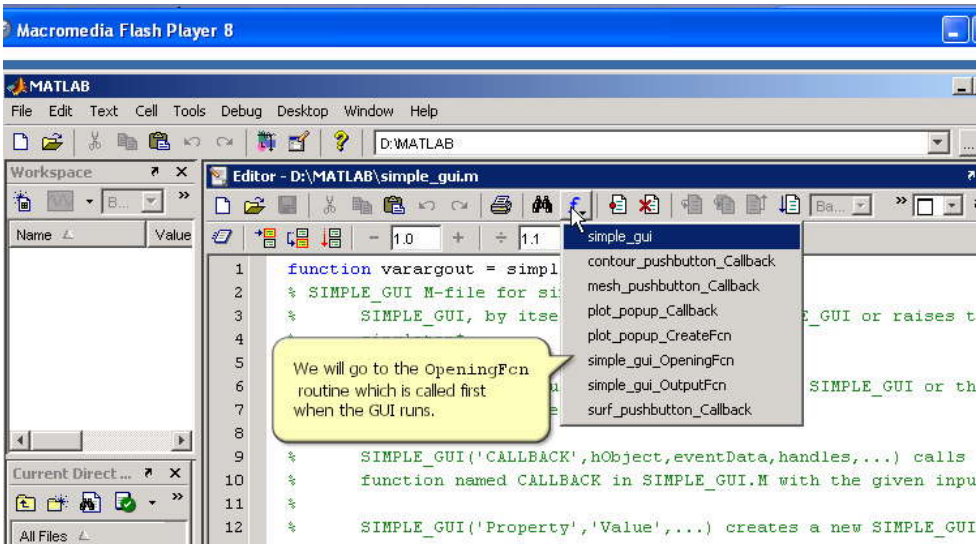


Figure 3.36: The icon to show the functions of each button

9. In the function, data can be load to do the function. Then syntax for MATLAB GUI Development Environment can be developed.

3.5.3 Development of MATLAB Coding (GUI)

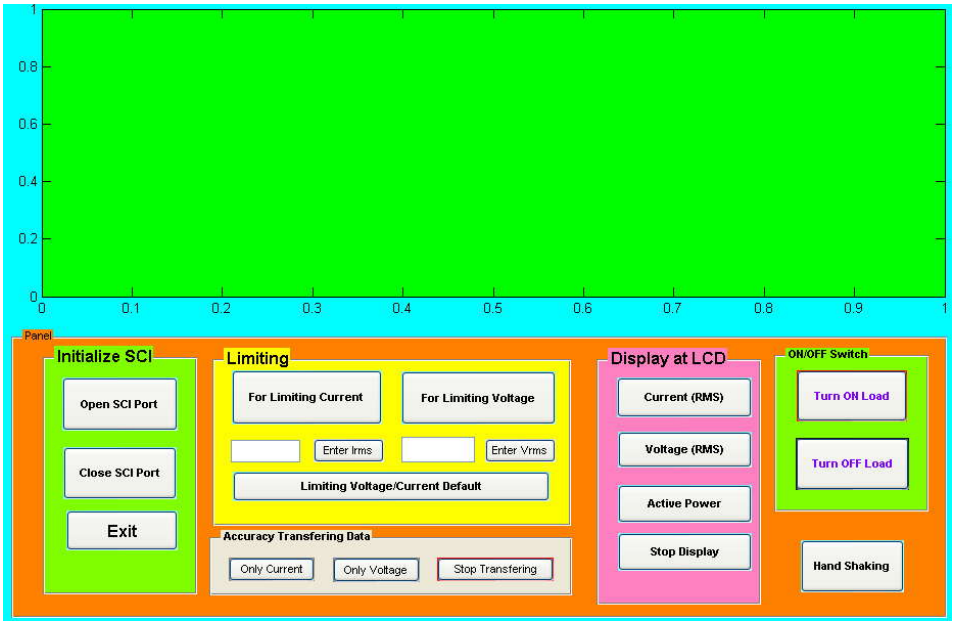


Figure 3.37: Overall view of GUI

3.5.4 Initialize GUI

In MATLAB, GUI can be develop by create the coding in M-File. The initialize coding must be creating first, before create the main coding. The purpose to create the initialize coding is to assign port which is available and enable the port in computer. The function of the port is to transfer and receive data. It will be linker between hardware and software in computer. Below is the initialize coding in M-File:

```
sci=serial('COM1')
handles.open_port=sci
guidata(hObject, handles);
```

The *s=serial('COM1')* is to define the parameter in MATLAB that representing the data either for transmit or receive from serial port. However user need to defined the serial port that is being used. If user does not know which port is existed, user can justify which port by looking at window in device manager. However, there will be some sort of problem. If certain port is already used by

another program, clash will occurred and the serial port object will not be able to connect to the device.

The `handles.op=s command` defined as when GUI is running, the M-file creates a handles structure that contains all the data for GUI objects, such as controls, menus, and axes. The handles structure is passed as an input to each callback. The handles structure can be use to share data between callbacks and access GUI data.

The `guidata(hObject, handles)` is command that will stores the variable data as GUI data. If *object_handle* is not a figure handle, then the object's parent figure is used and data can be any of MATLAB variables, but is typically a structure, which is enables to add new fields as required.

3.5.5 Open and Close Coding for GUI

Figure 3.38: Panel for open and close serial port

After GUI already running, user must click the button 'open SCI Port' to give command to Matlab to open serial port. The callback for that button consist of coding as below where coding `fopen(sci)` is for specifically to open serial port.

```
sci=handles.open_port
fopen(sci)
```

Below is the coding to close the port where coding `fclose(sci)` is for specifically to closed certain serial port.

```
sci=handles.open_port
fclose(sci)
```

3.5.5 Coding for transmit command to microcontroller

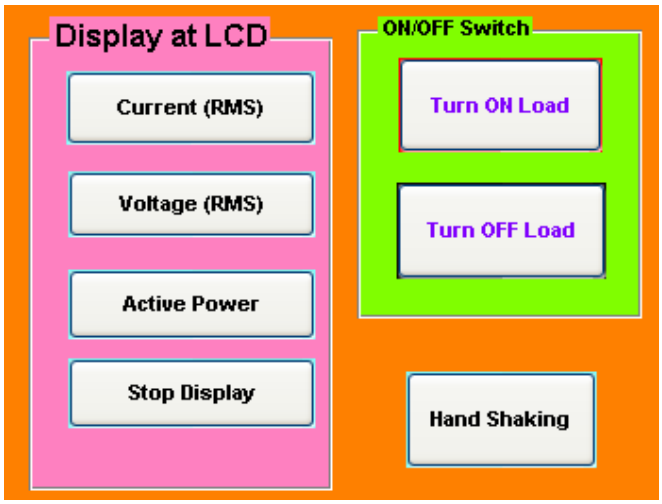


Figure 3.39: Panel for selecting mode operation of controller

Basic concept of giving command to controller for controlling intention is by transmitting character to the controller. Thus every time user clicked the button on GUI's panel; PC will send certain character as representing a command to the controller. In Figure 3.39 is shown the panel for switching purpose. From this panel, user able to turn on or off the power supply to load, select the mode of display at LCD either displaying value of current or voltage or power. However, the function of GUI will be denied if the external switch is already been setting. This could be referred to the LCD display either it will write down 'BUSY!!! CHECK EXTERNAL SWITCH' or 'ALREADY RUNNING'. Below is one of example of a command that exist in this project. The MATLAB will cause PC to transmit character 'A' to controller in order to turn on power supply. Here is the M-File:

```
sci=handles.open_port
fprintf(sci,'%c','A')
```

The coding *fprintf(sci,'%c','A')* is a command in MATLAB to send data through serial port and the format of transmitting is could be change by replacing certain variables in the middle of character in the column.

3.5.6 Coding for receiving data and plotting graph

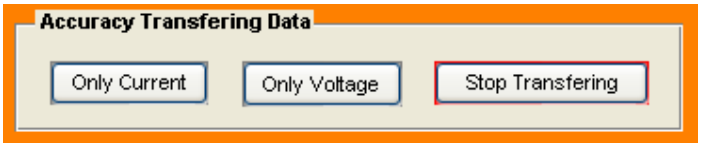


Figure 3.40: Panel for plotting the graph

In order to display desirable parameters, the GUI was been designed with specific button to represent as a command to load data from serial port and display it at GUI panel. The Figure 3.40 is shown the panel for plotting graph. The measured value of voltage and current from sensor is shown in graph form. The panel was designed by two buttons for completing this purpose which is function to load voltage and current data and then plot both of the graphs. Below is the coding in M-File for both of buttons:

- Coding to plot the graph for current.

```
sci=handles.open_port
fprintf(sci,'%c','T')
out=fread(sci)
y=out/255*15
plot(y)
```

- Coding to plot the graph for voltage.

```
sci=handles.open_port
fprintf(sci,'%c','R')
out=fread(sci)
y=0.98*out
plot(y)
```

The function of *sci=handles.op* coding is to retrieve data. Data will callback by using this coding; it will define again the data. While, the function of *out=fread(sci)* is to read binary data from the device connected to *obj*, and returns the data to A. Data will be read in 10 second. The GUI Plot button callback creates a plot of the run data and adds a legend. The data to plot is passed to the callback in the

handles structure, which also contains the gain settings used when the simulation ran. When a user clicks on the only current button at accuracy data transfer's panel, the callback executes the following steps:

- PC sends certain character to microcontroller where character 'T' as sign to apply data of current and character 'R' as sign to apply voltage's data.
- Then microcontroller will react by transferring data based on inquiry from PC.
- MATLAB collects the data for each run selected in the Results list, including two variables (time vector and output vector) and a color for each result run to plot.
- Generates a string for the legend from the stored data.
- Creates the figure and axes for plotting and saves the handles for use by the Close button callback.
- Plots the data, adds a legend, and makes the figure visible.

3.5.7 Coding for limiting current and voltage

The system also designs to able to make limiting for current and voltage. This feature is about to make the system to react in intelligent manner towards the condition of current and voltage. It means that when current or voltage is exceeding the limit, the system going to warn the user about unwilling condition.

This system function when user click the button either for limiting current or voltage. It shall cause the microcontroller to display either 'LIMITED CURRENT' or 'LIMITED VOLTAGE'. After display that statement, the controller is about to wait the data from PC which is mean that the data is the value of current or voltage that need to be limit. Thus in the range of time, user must enter either value of current or voltage in root mean square (rms). Then user must click the enter button to run a process of transmitting the rms value to the controller in hexadecimal form. In controller the data taken from serial communication data register (SCDR) to be calculated and then display it at second line of LCD. The display current or voltage at LCD supposed to be same as the data that user determined at the GUI panel.

However if the user do not set any value of limited current or voltage, the controller will use the default value to run the limiting purpose. At GUI panel, users also able to set the current or voltage into default value. This could be done by clicking the button available at that panel. The view of panel is shown in Figure 3.41. The coding for this command is as follows:

- Coding for limiting voltage

```
sci=handles.open_port
voltage_rms =
str2double(get(handles.Enter_Limited_Voltage,'String'));
voltage_hex=voltage_rms/250*255
v=dec2hex(voltage_hex)
fprintf(sci,'%c','v')
```

- Coding for limiting voltage

```
sci=handles.open_port
current_rms =
str2double(get(handles.Enter_Limited_Current,'String'));
current_calculated=current_rms/15*255
fprintf(sci,'%s','current_calculated')
```

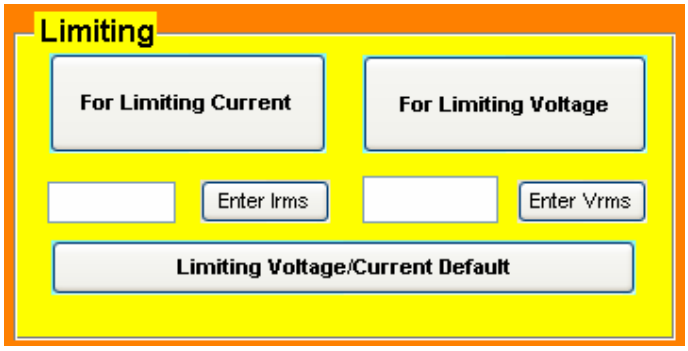


Figure 3.41: Panel for limiting current and voltage

CHAPTER 4

RESULTS AND ANALYSIS

4.0 Introduction

The successful of this project is determined in it capability to measure the power consumption of certain appliance. However the project was do not able to be complete based on the early objective since the voltage and current sensor is not working properly. However the major design of this project is including the controller, appliance interfacing circuit, LCD as display meter and development of software is quite successful. The circuit is needed to be test to know the result and the functionality of this system. Thus in completing this circuit, the voltage and current sensor must be replaced with another component as intention to generate input to the ADC.

In order to replace the function of current and voltage sensor, two potentiometer is useful to produce dc voltage level vary from 0 to 5V. However the voltage level that been consume to the voltage reference of ADC at microcontroller is 0 to 4.5V. It means that the highest voltage level is must not exceed this 4.5V. This technique is used voltage divider concept to vary the power supply of 4.5 volt from power supply module. In experimenting to this project, the varied resistance from potentiometer pin is measured by a multi meter to obtain the value of voltage consume to ADC pin. Then the accuracy of the output that been display at GUI and LCD screen is could be determined.

4.1 Current measurement

The monitoring of current was been done in two method, first is by displaying at LCD and second one is obtain it from GUI's graph at PC. Thus, several data and measurement have been done by varying a potentiometer to obtain a voltage in range of 0 to 4.5V. When certain voltage level set to ADC pin of current, the user just need to select the external switch to display at LCD. However if the user was monitoring the system from a PC, they must run the GUI to interface with the hardware. By clicking the button of "only current" at accuracy data transfer's panel, GUI will plot a graph to show the current level in rms. Once a user clicks this button, MATLAB will take the reading from serial port in about 10 seconds and just after that the graph will be plotted. Thus it will make this system not working in real time however the value of instant current is still available at LCD. Besides of acquire data reading for current, user also able select the displaying mode at LCD external switch via GUI panel to view current.

The Figure 4.1(a) is show the value of the current at zero level. This condition happened when the voltage divider from potentiometer is set to 0V. In Figure 4.1(b) is showed the current in rms at zero level at LCD when the output of voltage divider is set to 0V. The spike is exist in the display graph and this condition is not proper because it will cause the system to react by giving warning at LCD which is view "WARNIG CURRENT IS HIGH". Based on the comparison between the results display at GUI's graph and LCD meter, they are seem to be equal. Then the accuracy of reading could be determined by equation 4.1. The accuracy of reading is shown by taking four samples of measurement.

$$Irms = \frac{V_{R1}}{4.5} \times Irms(\max imum)$$

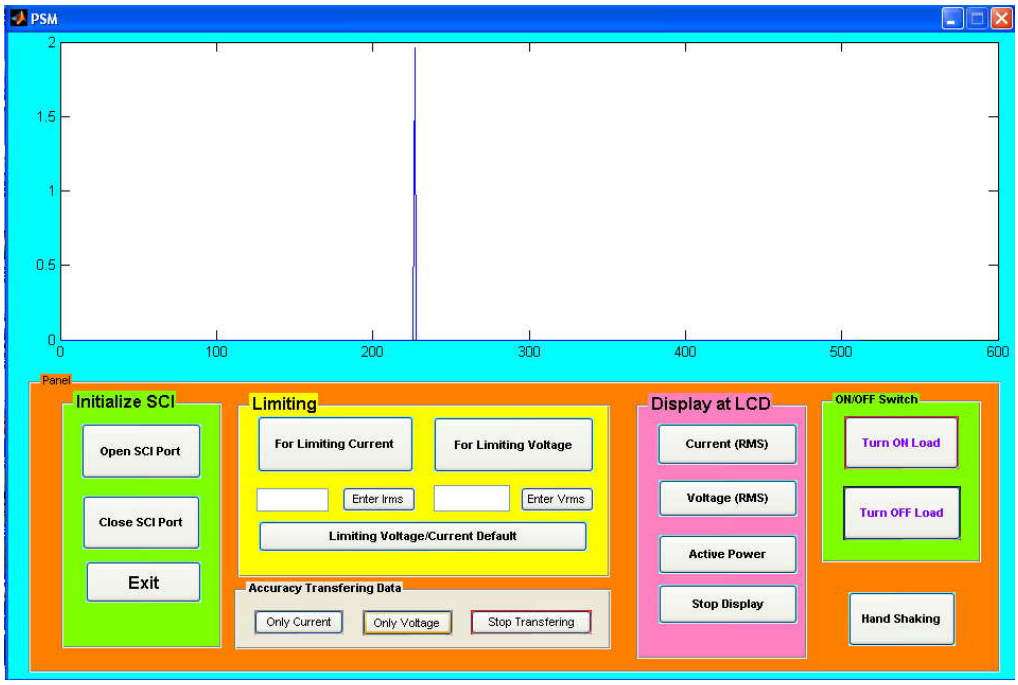
$$Irms = \frac{V_{R1}}{4.5} \times 15A \qquad \textbf{(4.1)}$$

Where,

V_{R1} = output of the potentiometer in voltage

Irms = level of current in rms value

- **Sample 1**



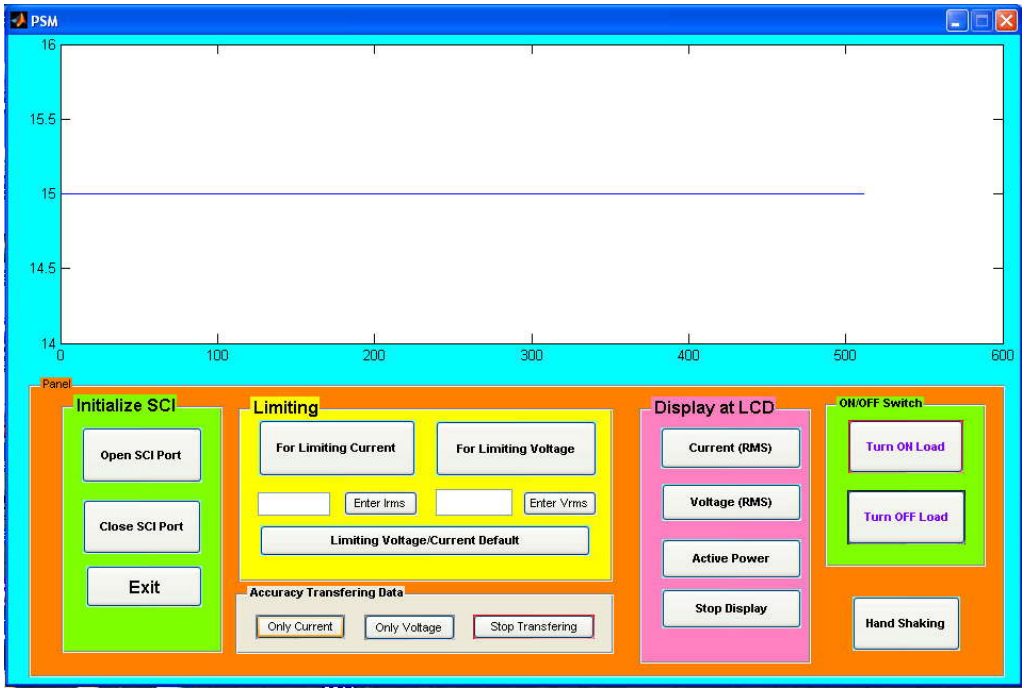
(a)

(b)

Figure 4.1: **(a)** The graph from GUI show the minimum scale of current
(b) LCD display and voltage divider show zero reading

- **Sample 2**

The Figure 4.2(a) is showing the current right now is at maximum of current level. The full scale of current is at 15A and hence here the graph shows a perfect straight line.



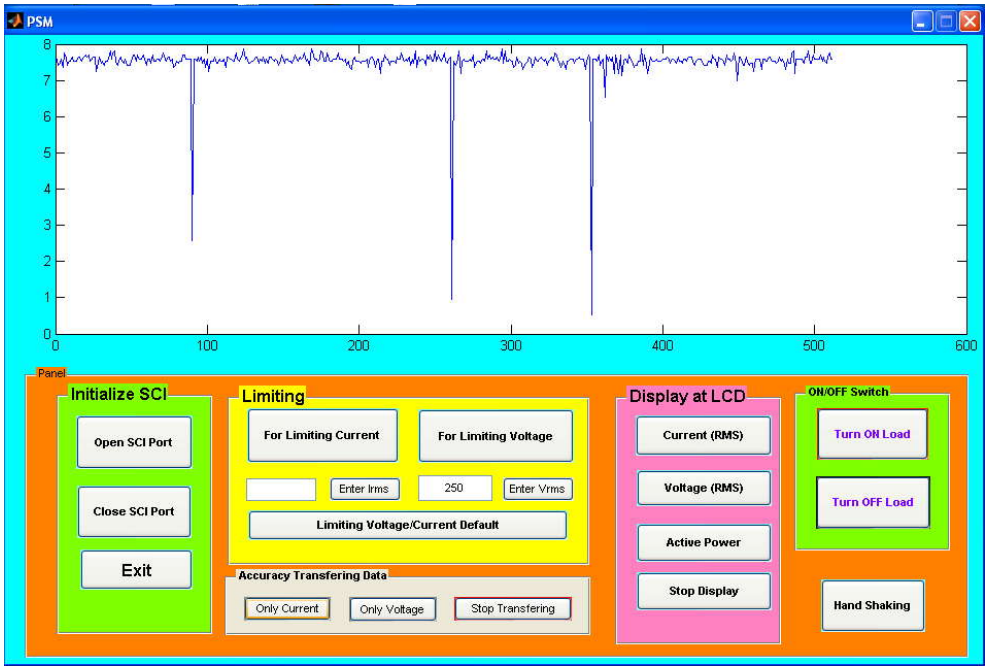
(a)

(b)

Figure 4.2: **(a)** The graph from GUI show the full scale current (15A)
(b) The LCD display the value of current exactly equal with graph from GUI an the multi meter show 4.54 V for maximum output of voltage divider

- **Sample 3**

The Figure 4.3(a) and Figure 4.3(b) is showing the current level is quite same and based on the output of the voltage divider, the output at GUI and LCD could be found by using equation 4.1. Based on the calculation it should be 7.533A and it mean that the measurement display on GUI and LCD is accurate.



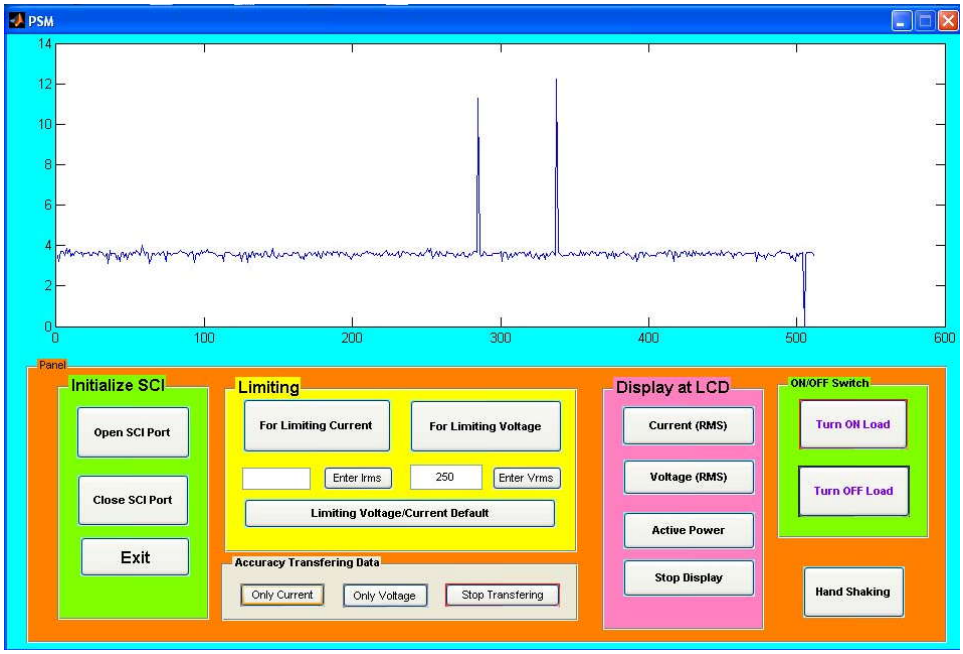
(a)

(b)

Figure 4.3: **(a)** The graph from GUI show the current at level about 7 A.
(b) The LCD shows the level of current at 7.588 A while voltage divider from potentiometer is 2.26 V

- **Sample 4**

The Figure 4.4(a) and Figure 4.4(b) is also showing the current level is quite same. Based on the calculation on equation 4.1 the display value should be 3.83A and it mean that the measurement display on GUI and LCD is near to accurate.



(a)

(b)

Figure 4.4: (a) The graph shows the level of current about over 3.5A
(b) V_{R1} is 1.15V and the current display at LCD is 3.5321A

4.2 Voltage measurement

The measurement of voltage is goes in the same method as measuring current. The different is only when it comes to measure the real voltage from the output of voltage divider. The equation is based on the equation 4.2 where VRI is the output of voltage divider. In determined the accuracy of measured value, four samples had been taken.

$$Vrms = \frac{V_{R1}}{4.5} \times Vrms(\max imum)$$

$$Vrms = \frac{V_{R1}}{4.5} \times 250V \qquad \textbf{(4.2)}$$

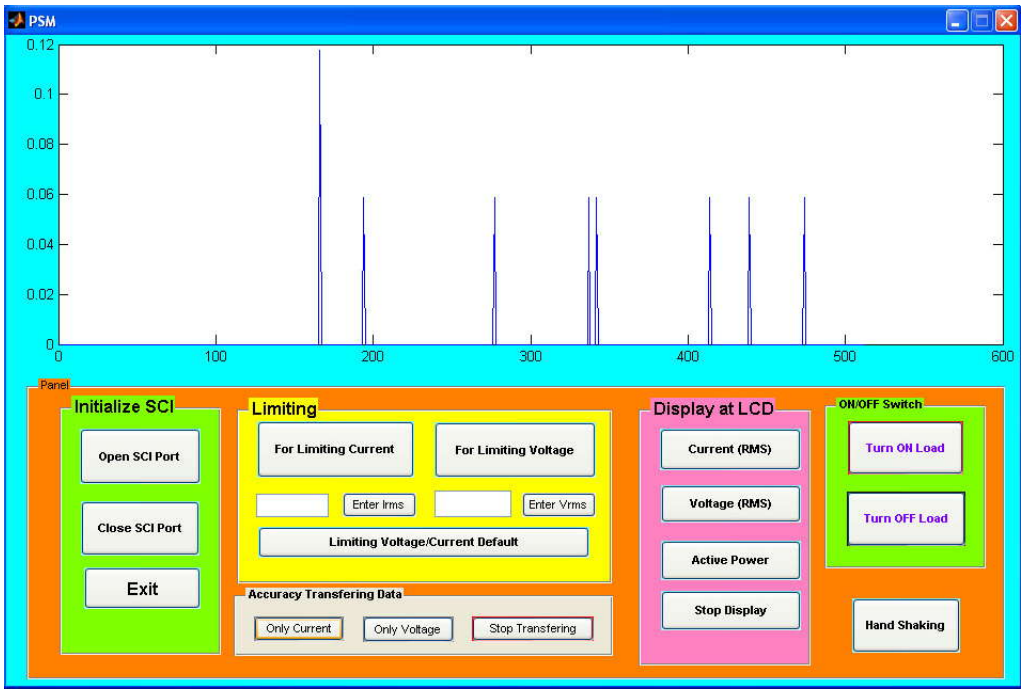
Where,

Vrms = real voltage level in rms value

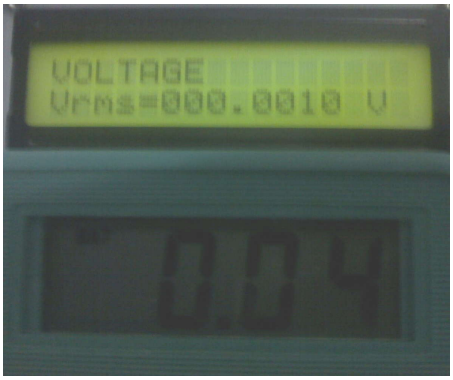
V_{R1} = output of the potentiometer

- **Sample 1**

The Figure 4.5(a) and 4.5(b) is showing the voltage right now is at minimum of current level. The lowest scale of voltage is at 0V and hence here the graph shows a little spike.



(a)

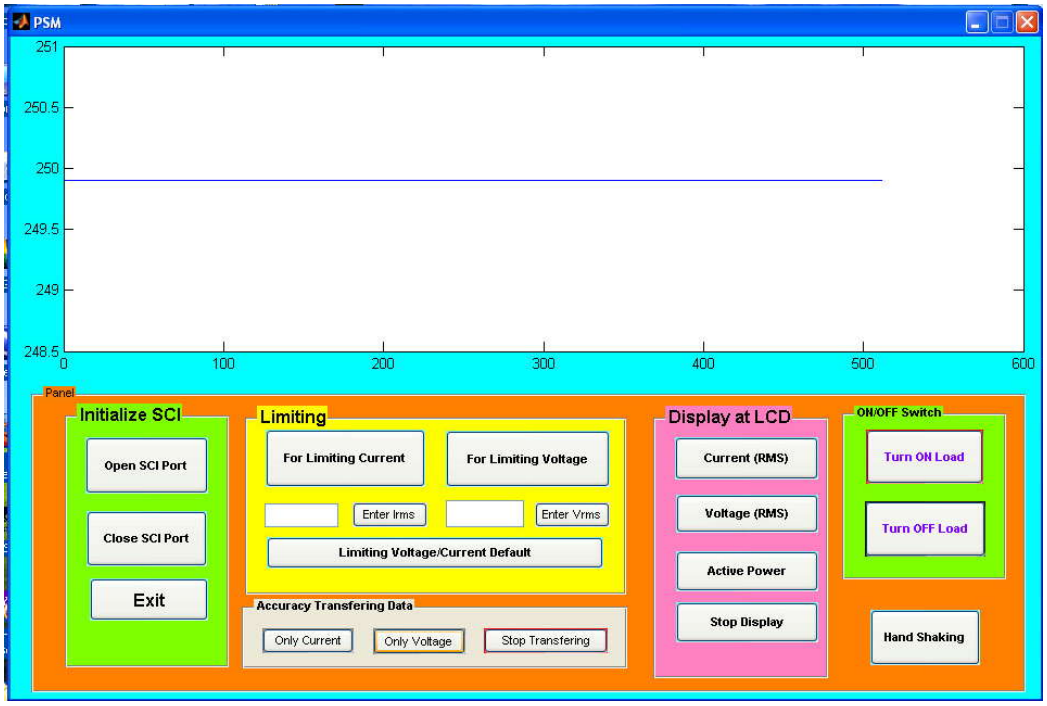


(**b**)

Figure 4.5: (a) The graph show the minimum scale of voltage (0V)
(b) The LCD and voltage divider is also in 0 V

- **Sample 2**

The Figure 4.6(a) and 4.6(b) is showing the voltage right now is at minimum of current level. The lowest scale of voltage is at 0V and hence here the graph shows a little spike.



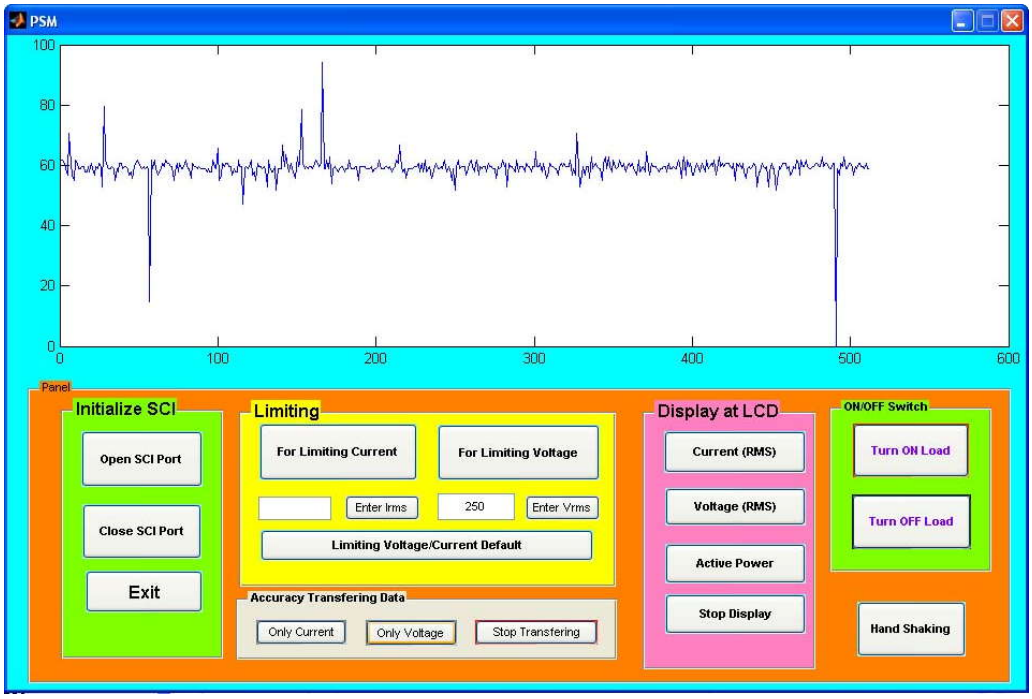
(a)

(b)

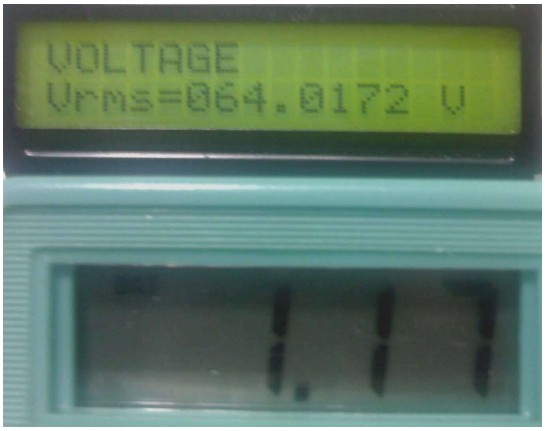
Figure 4.6: (a) The graph show the full scale of voltage 250 V
(b) The LCD show the full scale of voltage 255 V

- **Sample 3**

The Figure 4.7(a) and Figure 4.7(b) is also showing the current level is quite same. Based on the calculation on equation 4.2 the display value should be 65V and it mean that the measurement display on GUI and LCD is near to accurate.



(a)

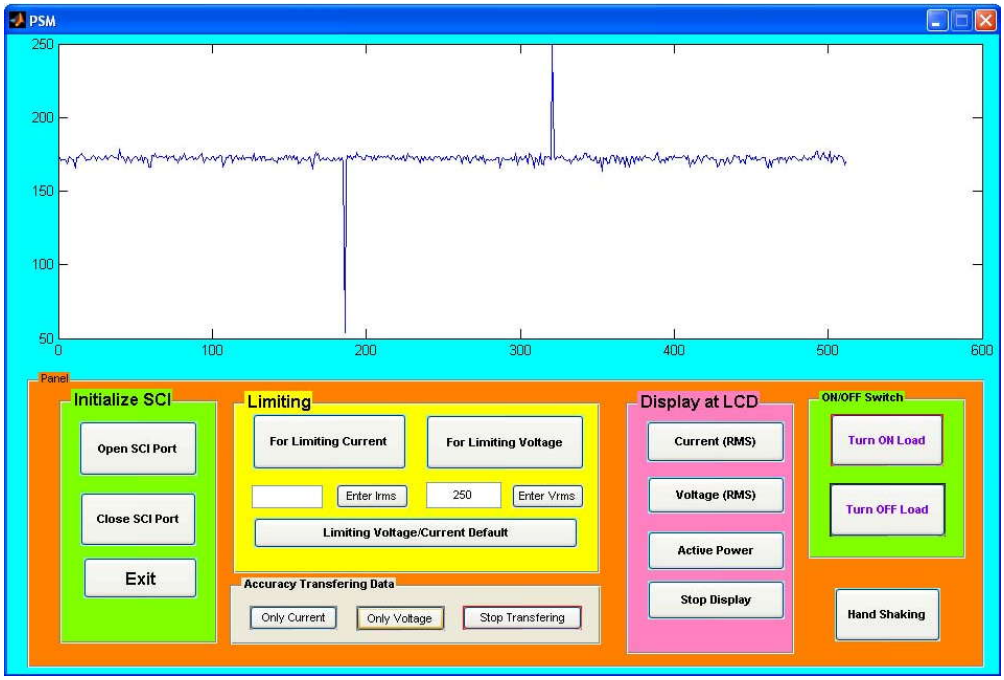


(b)

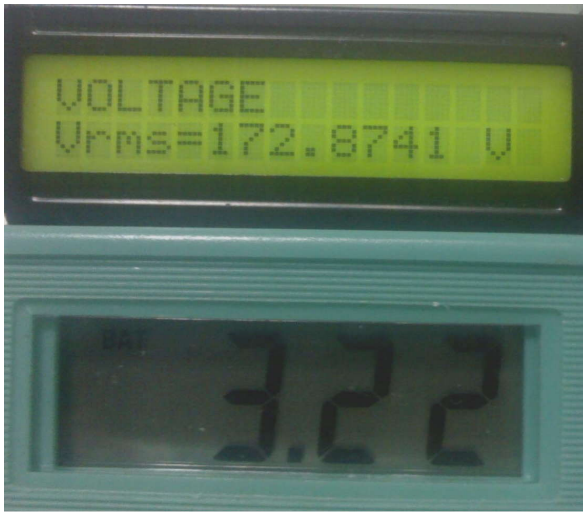
Figure 4.7: (a) The graph show the voltage of 64 V
(b) The LCD show the 64.01 V and V_{R1} is 1.17 V

- **Sample 4**

The Figure 4.8(a) and Figure 4.8(b) is showing the voltage level is quite same and based on the output of the voltage divider, the output at GUI and LCD could be found by using equation 4.2. Based on the calculation it should be 178.89 V and it mean that the measurement display on GUI and LCD is accurate**.**



(a)



(b)

Figure 4.8: (a) The graph show the voltage is about 170 V
(b) The LCD show the voltage of 172.87 V and V_{R1} is 3.22 V

4.3 Power measurement

There is problem in obtaining the power drop for combination of current and voltage. This is because the ADC system is does not able to work properly. Based on the experiment that be done, when user switch on the system for a long range of time it will cause ADC to operate improperly. As a result the measured value is become unstable and cause system work in accurate condition. In figure 4.9 is show the basic experiment that could be done which is include zero level of power. The equation to calculate the power is shown in equation 4.3.

$$P(reactive) = Irms \times Vrms \quad \textbf{(4.3)}$$

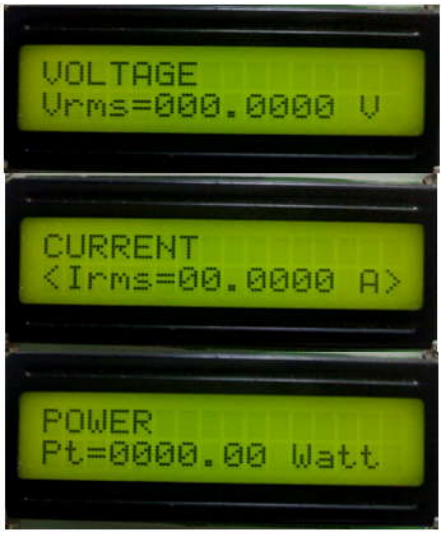


Figure 4.9: The LCD show the combination of measurement to test the accuracy power's measurement

4.4 Features

- Turn on and off load

The switching for power supply to the load can be achieved by two methods. First user can push the external start button. From here, the switching apply concept of hold circuit by using a relay. It means that the supply of 240 Vrms is on by energized magnetic coil and vice versa for instead situation. The second method is by clicking button at GUI panel. The concept of switching is still same but the function of push button being replaced by controller. The attractive of this method is the controller will display "TURN ON LOAD" or "TURN OFF LOAD" as shown in Figure 4.10. However, the combination of two switching method will cause conflict between them, thus this system was designed to has a coordination between then. Once a user already switched on certain external switch, they will not be able to control it from PC. However the LCD will display some identification telling that an error is happened. This could be illustrated in Figure 4.11. The LCD will display "ALREADY RUNNING" if same switch is selected and if difference switch is selected it will display "BUSY PLEASE CHECK EXTERNAL SWITCH".

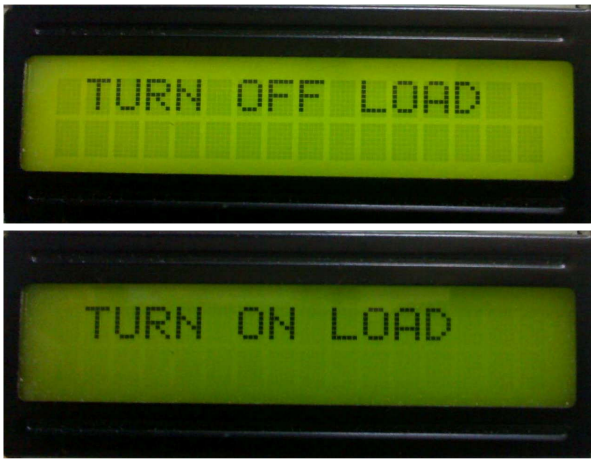


Figure 4.10: The LCD show the sign ON or OFF for load

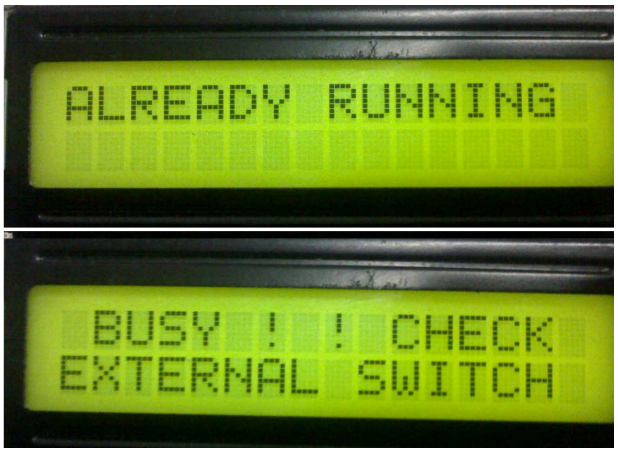


Figure 4.11: The LCD show the sign if conflict occurred between GUI and external switch

- Warning signal if current or voltage is over limit

The system was designed to make limiting to the current and voltage. When this system is turn on, the default value will be loaded to determine the first set point. This set point is that the current and voltage must not exceed the 10 A and 245 V respectively. This condition was set to properly interface for home user. If this condition is been hit, the controller will display warning signal at LCD as shown in Figure 4.12.

Figure 4.12: The LCD show the warning sign if current or voltage is over the limit

However the owner can change this set point by entering the desired value at GUI. At GUI, there was a panel which been specified for limiting purpose. The Figure 4.13 shows the limiting panel at GUI. When a user clicked either button for limiting current or limiting voltage, the LCD will display either "LIMITED CURRENT" or "LIMITED VOLTAGE". This could be pictured as shown in Figure 4.14. After LCD write this sign, controlled will loop to wait for any input from PC. The function of edit text at limiting panel is to entering the desired value for limiting purpose. It means that user can write down any number between 0 to 15 for current limiting and 0 to 250 for voltage limiting. After keep in value either voltage or current, user must click on button beside of edit text box for transmitting command.

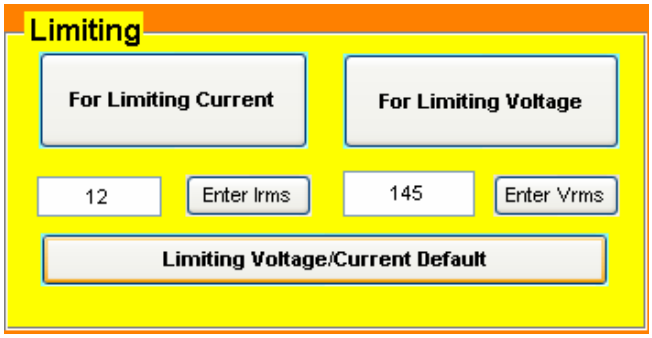


Figure 4.13: Limiting panel at GUI

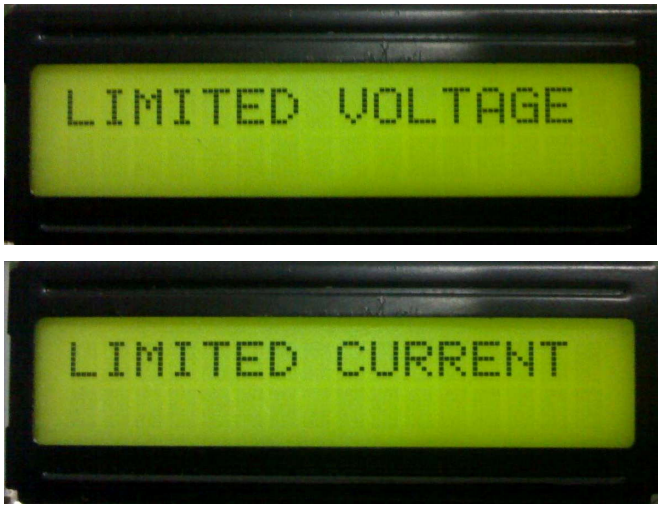


Figure 4.14: The LCD show limit sign at controller

After controller receives data from PC, it will make calculation to define the data taken and display the value at the second line at LCD display. The Figure 4.15 shows the operation. The inserted value at edit text box is supposed to be same as value display at second line at LCD.

Figure 4.15: LCD shows limit voltage or current same as been display

When a user set the limit the voltage or current to certain level, they can easily set the default value just by clicking button 'limiting voltage/current default'. Then LCD will display the sign of "DEFAULT LIMITED". It was as display in Figure 4.16.

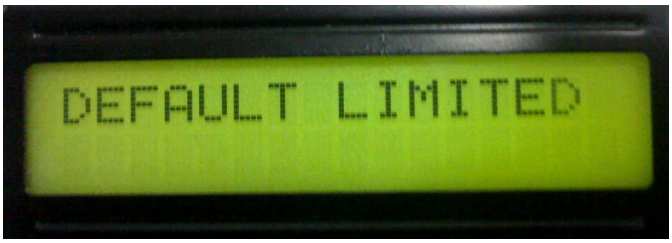


Figure 4.16: The LCD show sign of limited current and voltage is default

- **Sign showing certain data being send to PC**

When GUI acquired to plot the graph of data that are taken, it was only one data could be delivered to PC at certain time. This is important to keep the accuracy of data because user needs to make analysis on them. Thus this accuracy data transfer was been designed for this purpose. The Figure 4.17 is shown the LCD views the status of operation.

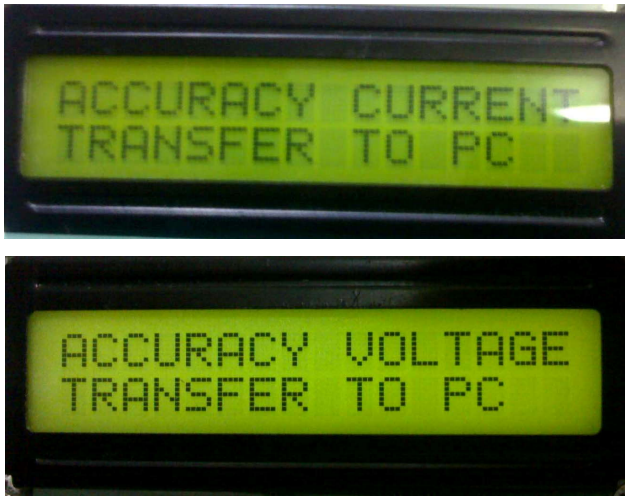


Figure 4.17: LCD shows sign of accuracy data transfer

CHAPTER 5

CONCLUSION AND RECOMEDATION

5.1 CONCLUSION

As a conclusion, this project bring a lot of meaning to the student because from here student able to mastering a lot of knowledge. Not just that, the implementation of the theory into designing hardware is quite challenge. The scope that are given is seemed to be little big and a lot of effort and commitment were needed to finished it. The most challenging parts are in designing expanded mode configuration and development of assembly language where it taken long time to be accomplished. The implementation of this project was a not a complete achievement because the sensor is not operating well as desired. However the system still could be tested as been discussed in chapter 4. Thus this system is still could be develop either to design or buy a great sensor which is can operate properly.

In Europe, this system was already been commercialized since it was really help the homeowner in saving the energy at home and indirectly reduce the electricity bill. However this system is not just limit to the home usage only, it may apply to industry if user wants to control certain component either machine or valve. The benefit of this system is for sure to give enough information about energy usage to homeowner and the attractive way of guiding by LCD make them more intelligent. Thus, the development of this system hopes to be useful for human being.

5.2 RECOMMENDATION

In future, this system is still could be develop into different design technique. It could be list as follows:

- The designer can change the way of interfacing with PC such as using local area network (LAN), internet and short massage system (sms) using hand phone to control the controller
- Add the function of switching by using wireless control system
- In order to tackle the problem of expanded mode, user can use PCB to replace the wrapping technique

5.3 COST AND COMMERCIALISATION

The total coast in developing this project could be referred to the appendix A. There estimation cost for overall components is about RM 282.72. However the most difficult component to be found is current transformer since it was not widely used by user. Others things that should be take into consideration are to find a great MC68HC11 since there are only a few of them that can used in expanded configuration. Thus, while buying this component user must check it ability first.

As stated in previous discussion, certain utility companies charge different rate to the homeowner if their house consumed energy exceed the rate in peak interval. The potential of this project to be commercial is quite high since homeowner may find solution in order to reduce the energy usage in the range of time. This could be verified when certain industries already developed this system to be sold to the homeowner. Besides that, the application of this system is not just limited to the house usage. Foe example, the concept of this system is could be applied into industries such as in power generation station. The engineer need to deal with high voltage, they can use a current transformer to isolate them from dangerous in order to measure current or voltage. The application of GUI is usable when they want to monitor the current, voltage and power from the generator. By sitting on a chair and interface with PC will make their job easier.

REFERENCES

[1] Ming Yuan Cho, Cha Win Huang, (May 2001). *Development of PC based energy management system for electrical energy saving of high voltage customer*. Industrial and Commercial Power Systems Technical Conference, 2001.
Available at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=966505

[2] Smart Home: PC Based Energy Management System - Ashan De Silva, Beau Maryniuk , Brian Chan

APPENDICES

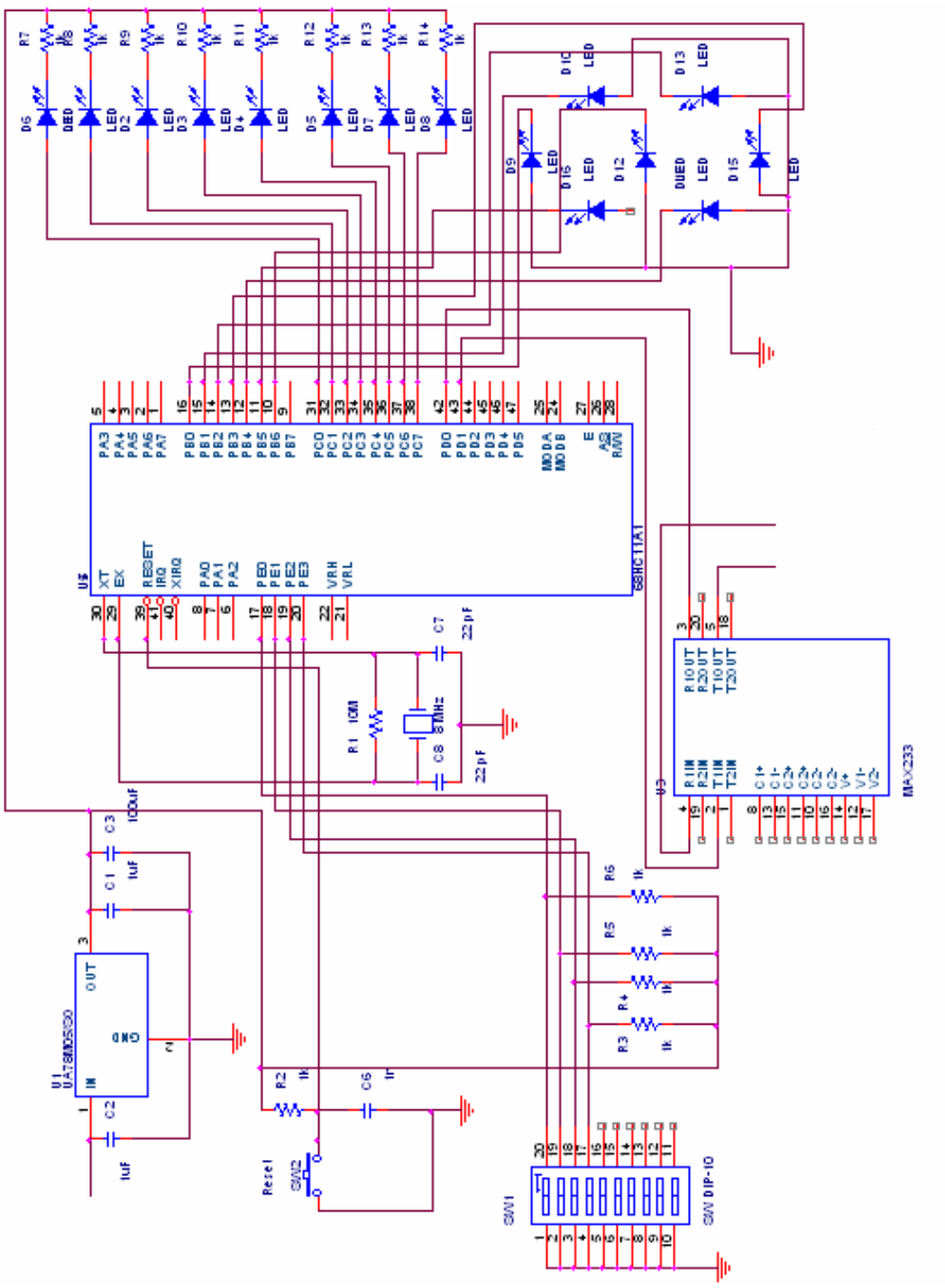
APPENDIX A

Project Cost

Bil	Bahan/Komponen	Spesifikasi	Anggaran Harga / unit	Kuantiti	Anggaran Harga
1	MICRO.C	MC68HC11	RM40.00	1	RM 40.00
2	REGULATOR	7805	RM1.00	1	RM 1.00
3	PCB HEADER		RM0.80	20	RM 16.00
4	I.C BASE	OVERALL		10	RM 10.00
	PLUG		RM 3.00	1	RM 3.00
	CRYSTAL	8MHz	RM1.20	1	RM 1.20
	STRIP BOARD	10” x 4”	RM4.50	2	RM 9.00
	RESET SWITCH		RM0.50	3	RM 1.50
	MAX233		RM9.50	1	RM 9.50
	IDE CABLE		RM 4.00	1	RM 4.00
	RELAY		RM 2.50	5	RM 0.72
	EEPROM		RM 12.00	1	RM 12.00
	CASING		RM 9.00	1	RM 9.00
	VOLTAGE TRANSFORMER		RM10.00	1	RM 10.00
	RAM		RM 8.00	3	RM 16.00
	STAND		RM 0.80	10	RM 8.00
	SOCKET		RM 4.90	1	RM 4.90
	SWITCH		RM 1.80	4	RM 0.08
	DB9	FEMALE	RM 0.60	1	RM 0.60
	DB9 COVER		RM 0.60	1	RM 0.60
	HEAT SINK		RM 0.70	1	RM 0.70
	LCD DISPLAY		RM 65.00		RM 65.00
	WARPING WIRE		RM 15.00	1	RM 15.00
	CURRENT TRANSFORMER		RM 45	1	RM 45
TOTAL ESTIMATION PRICE					**RM 282.72**

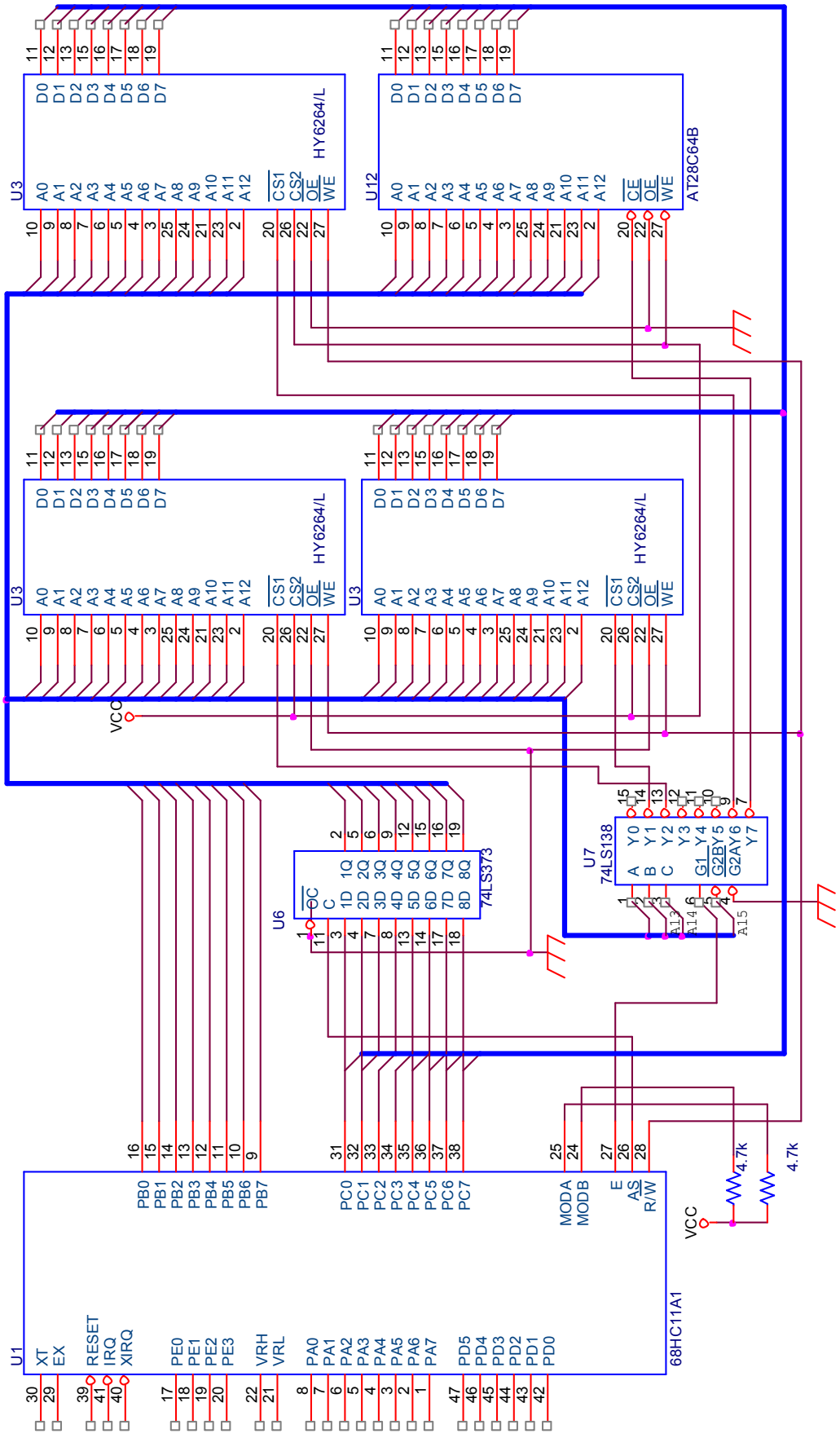
APPENDIX B

Circuit Diagram for Bootstrap Mode


R7 1k
R8 1k
R9 1k
R10 1k
R11 1k
R12 1k
R13 1k
R14 1k
D6 LED
D7 LED
D2 LED
D3 LED
D4 LED
D5 LED
D7 LED
D8 LED
D10 LED
D13 LED
D9 LED
D11 LED
D12 LED
D15 LED
U5
PA3 PA4 PA5 PA6 PA7
PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7
PC0 PC1 PC2 PC3 PC4 PC5 PC6 PC7
PD0 PD1 PD2 PD3 PD4 PD5
MODA MODB
E AS RW
XT EX RESET IRQ XIRQ
PA0 PA1 PA2
PE0 PE1 PE2 PE3
VRH VRL
68HC11A1
C7 22pF
C8 22pF
R1 10M
8MHz
U3
R1OUT R2OUT T1OUT T2OUT
R1IN R2IN T1IN T2IN
C1+ C1- C2+ C2- C2- C2- V+ V1- V2-
MAX233
C3 10uF
C1 1uf
U1
UA78M05CD
OUT GND IN
C2 1uf
R6 1k
R5 1k
R4 1k
R3 1k
R2 1K
C6 1f
Reset
SW2
SW1
SW DIP-10

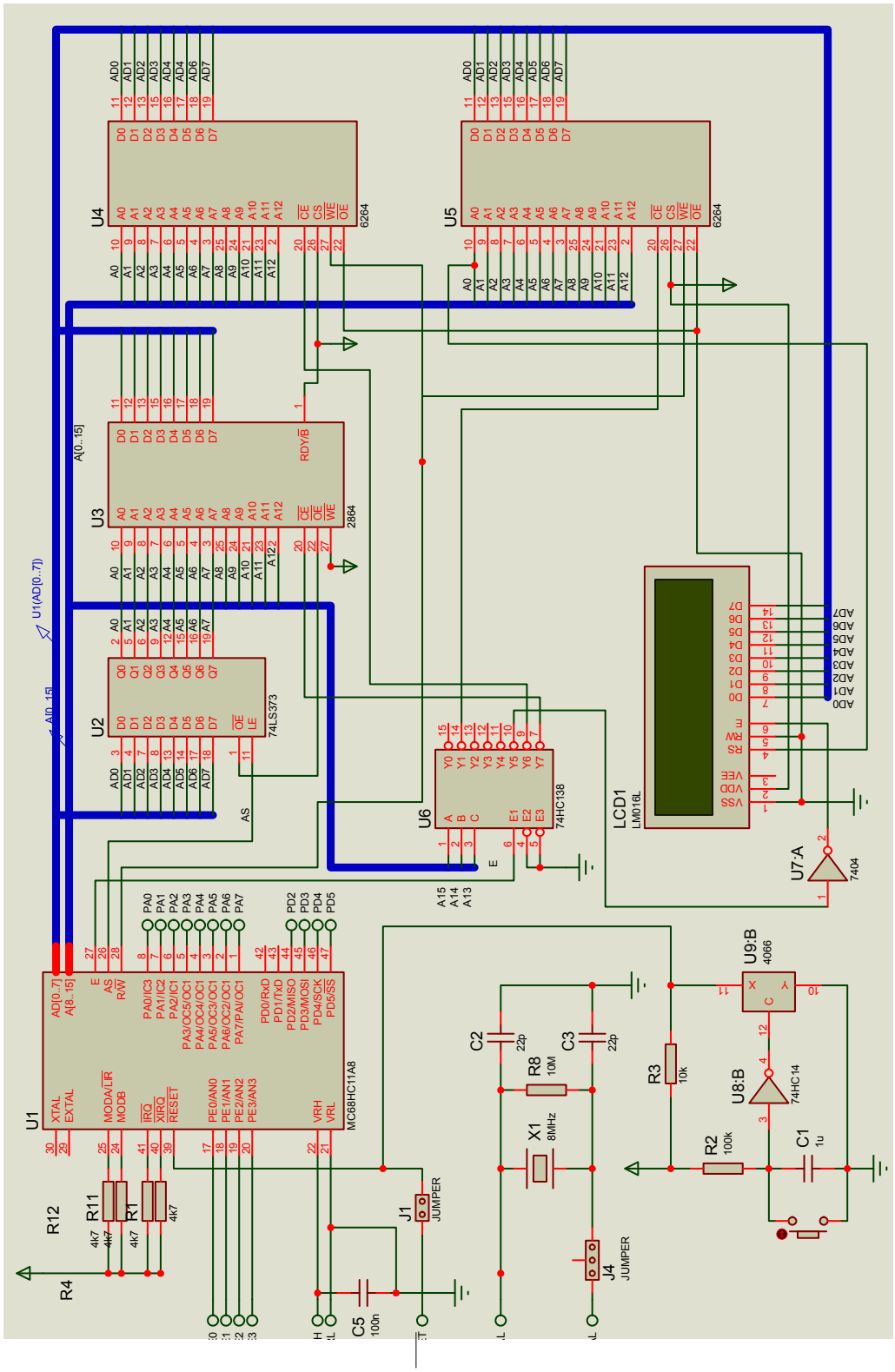
APPENDIX C

Circuit Diagram for basic Expanded Mode


U1
68HC11A1
XT
EX
RESET
IRQ
XIRQ
PE0
PE1
PE2
PE3
VRH
VRL
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PD5
PD4
PD3
PD2
PD1
PD0
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
MODA
MODB
E
AS
R/W
U3
HY6264/L
U12
AT28C64B
U6
74LS373
U7
74LS138
VCC
4.7k
4.7k

APPENDIX D

Circuit Diagram for Expanded Mode with LCD


U1
MC68HC11A8
U2
74LS373
U3
2864
U4
6264
U5
6264
U6
74HC138
LCD1
LM016L
U7:A
7404
U8:B
74HC14
U9:B
4066
X1
8MHz
C1
1u
C2
22p
C3
22p
C5
100n
R1
4k7
R2
100k
R3
10k
R4
4k7
R8
10M
R11
4k7
R12
4k7
J1
JUMPER
J4
JUMPER
A[0..15]
AD[0..7]
AS
E

APPENDIX E

Assembly language for microcontroller

```
REGS      EQU  $1000
OPTION         EQU  $39
ADCTL          EQU  $30
ADR1      EQU  $31
ADR2      EQU  $32
ADR3      EQU  $33
ADR4      EQU  $34
SCCR1          EQU  $2C
SCCR2          EQU  $2D
BAUD      EQU  $2B
SCSR      EQU  $2E
SCDR      EQU  $2F
COMAND         EQU  $A000
CHAR      EQU  $A001
PORTA          EQU  $0
PACTL          EQU  $26
SWITCH         EQU  $0
R_SWITCH  EQU  $20

          ORG  $E000
          LDS  #$DFFF
          LDX  #REGS
          LDAA #$00
          STAA $D950
          STAA $D200
          STAA $D201
          LDAA $D202
          LDAA #$FF
          STAA R_SWITCH

*****************INITIALIZE LCD*****************

INI_LCD LDAA   #$01
          JSR  SND_CMD
          LDAA #$02
          JSR  SND_CMD
          LDAA #$38
          JSR  SND_CMD
          LDAA #$06
          JSR  SND_CMD
          LDAA #$0C
          JSR  SND_CMD

******************INITIALIZE ADC************

INI_ADC   LDAA #$80
          STAA OPTION,X
          JSR  DELAY1
          LDAA #$30
          STAA ADCTL,X
```

```
****************** INITIALIZE SCI *********


INI_SCI   LDAA #$30
          STAA BAUD,X
          CLRA
          STAA SCCR1,X
          LDAA #$0C
          STAA SCCR2,X

***************WELCOMING DISPLAY DATA*********

     LDY  #DATA
NEXT1     LDAA 0,Y
          BEQ  TERJUN
          JSR  WRITE1
          INY
          BRA  NEXT1


TERJUN    LDAA #$BF
          JSR  SND_CMD

          LDY  #NAMA
NEXT21    LDAA 0,Y
          BEQ  TROJUN
          JSR  WRITE1
          INY
          BRA  NEXT21
TROJUN    JSR  DELAY4
          LDAA #$01
          JSR  SND_CMD
          BRA  MAIN



***************INITILIAZE PORTA AS I/O******

          BCLR PACTL,X   $C0

****************** MAIN PROGRAM **********

     MAIN LDY  #$2000
          LDAA #$00
          STAA $D200
          STAA $D201
          STAA $D202
          LDAA #170
          STAA $D800
          LDAA #250
          STAA $D801
```

```
LOOP      CLRA
SCN_CCF   BRCLR     ADCTL,X $80    SCN_CCF
          LDAA ADR1,X
          STAA $D700
          LDAA ADR3,X
          STAA $D701
          STAA 0,Y
          PSHY
          JSR  Tx_Rx
          JSR  CNVTER1
          JSR  DISPLAY

          LDAA $D700
          CMPA $D800
          BHS  WARN1
PATAH1    LDAA $D701
          CMPA $D801
          BHS  WARN2
PATAH2    PULY
          INY
          BRA  LOOP

****************************************


WARN1     LDAA #$00
          STAA $D950
          STAA $D200
          STAA $D201
          STAA $D202

          PSHX
          PSHY
          LDAA #$01
          JSR  SND_CMD
          LDY  #DATA6
REPT1     LDAA 0,Y
          BEQ  WENT1
          INY
          JSR  WRITE1
          BRA  REPT1

WENT1     LDAA #$BF
          JSR  SND_CMD
          LDY  #DATA7
REPT2     LDAA 0,Y
          BEQ  WENT2
          INY
          JSR  WRITE1
          BRA  REPT2
```

```
WENT2     PULY
          PULX
          JSR  DELAY4
          LDAA #$01
          JSR  SND_CMD
          JMP  PATAH1


WARN2     LDAA #$00
          STAA $D950
          STAA $D200
          STAA $D201
          STAA $D202
          PSHX
          PSHY
          LDAA #$01
          JSR  SND_CMD
          LDY  #DATA8
REPT3     LDAA 0,Y
          BEQ  WENT3
          INY
          JSR  WRITE
          BRA  REPT3

WENT3     LDAA #$BF
          JSR  SND_CMD
          LDY  #DATA9
REPT4     LDAA 0,Y
          BEQ  WENT4
          INY
          JSR  WRITE
          BRA  REPT4
WENT4     PULY
          PULX
          JSR  DELAY4
          LDAA #$01
          JSR  SND_CMD
          JMP  PATAH2

*********************************************


DISPLAY   BRCLR     SWITCH,X  $01  ANTARA1
          BRCLR     SWITCH,X  $02  ANTARA2
          BRCLR     SWITCH,X  $04  ANTARA3
          BRCLR     R_SWITCH  $01  ANTARA1
          BRCLR     R_SWITCH  $02  ANTARA2
          BRCLR     R_SWITCH  $04  ANTARA3
          PSHX
          PSHY
          JSR       NO_OPR
```

```
          PULY
          PULX
          RTS

ANTARA1   JMP  DISPLY1
ANTARA2   JMP  DISPLY2
ANTARA3   JMP  DISPLY3

*********************NO OPERATION**********

NO_OPR    LDAA $D200
          CMPA #'C'
          BEQ  GIKE
          LDAA $D201
          CMPA #'C'
          BEQ  GIKE
          LDAA $D202
          CMPA #'C'
          BEQ  GIKE

          LDAA $D950
          CMPA #'C'
          BEQ  GO_TO1

          LDAA #$02
          JSR  SND_CMD
          BRA  JEMP
GIKE LDAA #$01
          JSR  SND_CMD
JEMP LDY  #DATA2
SEMULE    LDAA 0,Y
          BEQ  GO_TO
          JSR  WRITE1
          INY
          BRA  SEMULE

GO_TO     LDAA #$BF
          JSR  SND_CMD
          LDY  #DATA5
SEMULE1   LDAA 0,Y
          BEQ  GO_TO1
          JSR  WRITE1
          INY
          BRA  SEMULE1
GO_TO1    LDAA #$CF
          JSR  SND_CMD
          LDAA #' '
          JSR  WRITE
          LDAA #$CF
          JSR  SND_CMD
          LDAA #'!'
```

```
          JSR  WRITE
          LDAA #$CF
          JSR  SND_CMD
          LDAA #' '
          JSR  WRITE
          LDAA #$CF
          JSR  SND_CMD
          LDAA #'!'
          JSR  WRITE

          LDAA #'C'
          STAA $D950
          RTS

*******************************************

DISPLY1   LDAA #$00
          STAA $D950
          STAA $D201
          STAA $D202
          LDAA $D200
          CMPA #'C'
          BEQ  ADR_70

          LDAA #$01
          JSR  SND_CMD
          LDY  #DATA1
NEXT2     LDAA 0,Y
          BEQ  KE1
          JSR  WRITE1
          INY
          BRA  NEXT2

KE11 LDAA #$00
          STAA $D950
          STAA $D201
          STAA $D200
          STAA $D202

KE1  LDAA #$BF
          JSR  SND_CMD
          LDAA #'<'
          JSR  WRITE1
          LDAA #'I'
          JSR  WRITE1
          BRA  STRT
ADR_70    LDAA #$C6
          JSR  SND_CMD
          BRA  STEP

STEP      JMP  LANGKAH
```

```
STRT      JMP  START

DISPLY2   LDAA #$00
          STAA $D950
          STAA $D200
          STAA $D202
          LDAA $D201
          CMPA #'C'
          BEQ  ADR_68

          LDAA #$01
          JSR  SND_CMD
          LDY  #DATA3
NEXT3     LDAA 0,Y
          BEQ  KE2
          JSR  WRITE1
          INY
          BRA  NEXT3

KE22      LDAA #$00
          STAA $D950
          STAA $D201
          STAA $D200
          STAA $D202

KE2       LDAA #$BF
          JSR  SND_CMD
          LDAA #'V'
          JSR  WRITE1
          BRA  START
ADR_68    LDAA #$C5
          JSR  SND_CMD
          BRA  LANGKAH


DISPLY3   LDAA #$00
          STAA $D950
          STAA $D200
          STAA $D201
          LDAA $D202
          CMPA #'C'
          BEQ  ADR_67

          LDAA #$01
          JSR  SND_CMD
          LDY  #DATA4
NEXT4     LDAA 0,Y
          BEQ  KE3
          JSR  WRITE1
          INY
          BRA  NEXT4
```

```
KE3       LDAA #$BF
          JSR  SND_CMD
          LDAA #'P'
          JSR  WRITE1
          LDAA #'t'
          JSR  WRITE
          BRA  START1
ADR_67    LDAA #$C3
          JSR  SND_CMD
          BRA  LANGKAH



START     LDAA #'r'
          JSR  WRITE1
          LDAA #'m'
          JSR  WRITE1
          LDAA #'s'
          JSR  WRITE1
START1    LDAA #'='
          JSR  WRITE1

          LDAA $D901
          CMPA #'C'
          BEQ  M_VLT

LANGKAH   BRCLR     SWITCH,X  $02  M_VLT
          BRA  TO_1

M_VLT     LDAA $D060
          JSR  WRITE1
TO_1      LDAA $D061
          JSR  WRITE1
          LDAA $D062
          JSR  WRITE1

          LDAA $D900
          CMPA #'C'
          BEQ  CRT_VLT
          LDAA $D901
          CMPA #'C'
          BEQ  CRT_VLT

          BRCLR     SWITCH,X  $01  CRT_VLT
          BRCLR     SWITCH,X  $02  CRT_VLT
          BRCLR     R_SWITCH  $01  CRT_VLT
          BRCLR     R_SWITCH  $02  CRT_VLT
          BRA  TO_2

CRT_VLT   LDAA #'.'
```

```
          JSR  WRITE1
TO_2      LDAA $D063
          JSR  WRITE1
          LDAA $D064
          JSR  WRITE1

          BRCLR     SWITCH,X  $04  PWR
          BRA  TO_3
PWR       LDAA #'.'
          JSR  WRITE1
TO_3      LDAA $D065
          JSR  WRITE1
          LDAA $D066
          JSR  WRITE1

          LDAA $D900
          CMPA #'C'
          BEQ  CRNT
          LDAA $D901
          CMPA #'C'
          BEQ  VOLT

          BRCLR     SWITCH,X  $01  CRNT
          BRCLR     SWITCH,X  $02  VOLT
          BRCLR     SWITCH,X  $04  POWR
          BRCLR     R_SWITCH  $01  CRNT
          BRCLR     R_SWITCH  $02  VOLT
          BRCLR     R_SWITCH  $04  POWR


CRNT      LDAA $D200
          CMPA #'C'
          BEQ  UTAMA
          LDAA #' '
          JSR  WRITE1
          LDAA #'A'
          JSR  WRITE1
          LDAA #'>'
          JSR  WRITE1

          LDAA $D900
          CMPA #'C'
          BEQ  HOME1

          LDAA #'C'
          STAA $D200
          JMP  UTAMA
HOME1     RTS

VOLT      LDAA $D201
          CMPA #'C'
```

```
          BEQ  UTAMA
          LDAA #' '
          JSR  WRITE1
          LDAA #'V'
          JSR  WRITE1

          LDAA $D901
          CMPA #'C'
          BEQ  HOME2

          LDAA #'C'
          STAA $D201
          JMP  UTAMA
HOME2     RTS

POWR      LDAA $D202
          CMPA #'C'
          BEQ  UTAMA
          LDAA #' '
          JSR  WRITE1
          LDAA #'W'
          JSR  WRITE1
          LDAA #'a'
          JSR  WRITE1
          LDAA #'t'
          JSR  WRITE1
          LDAA #'t'
          JSR  WRITE1
          LDAA #'C'
          STAA $D202
          JMP  UTAMA

UTAMA     RTS



************************************

Tx_Rx     BRSET     SCSR,X $20 RECEIVE
          BRSET     SCSR,X $80 EMIT
          BRA  Tx_Rx
EMIT      JMP  TRNSMIT

******** RECEIVE CHARACTER & DISPLAY OFF OR ON LOAD *****

RECEIVE   LDAA SCDR,X
          STAA $D000
          CMPA #'A'
          BEQ  N_LOAD
          CMPA #'B'
          BEQ  F_LOAD
```

```
            CMPA #'C'
            BEQ  SAV1
            CMPA #'D'
            BEQ  A_SAVE2
            CMPA #'E'
            BEQ  DFAULT
            CMPA #'1'
            BEQ  PERE
            CMPA #'2'
            BEQ  VLTG
            CMPA #'3'
            BEQ  WATTT
            CMPA #'S'
            BEQ  TOP
            CMPA #'T'
            BEQ  ADRC1
            CMPA #'R'
            BEQ  ADRV1
            CMPA #'M'
            BEQ  PC_REQ
            RTS

TOP         JMP  STOP
PERE        JMP  AMPERE
A_SAVE2     JMP  SAVE2
DFAULT      JMP  DEFAULT
N_LOAD      JMP  ON_LOAD
F_LOAD      JMP  OFFLOAD
SAV1        JMP  SAVE1
WATTT       JMP  WATT
VLTG        JMP  VOLTG
ADRC1       JMP  ADRC
ADRV1       JMP  ADRV


PC_REQ      PSHY
            LDAA #$01
            JSR  SND_CMD
            LDY  #DATA20
SEMME       LDAA 0,Y
            BEQ  GOTKW
            JSR  WRITE1
            INY
            BRA  SEMME
GOTKW       LDAA #$BF
            JSR  SND_CMD
            LDY  #DATA21
SEMUME      LDAA 0,Y
            BEQ  ASAM
            JSR  WRITE1
            INY
```

```
          BRA  SEMUME

ASAM      CLRA
SCCF      BRCLR     ADCTL,X $80    SCCF
          LDAA ADR1,X
          STAA $D300
          LDAB ADR3,X
          STAA $D301
PCORDR    BRCLR     SCSR,X    $20  PCORDR
          LDAA SCDR,X
          CMPA #'X'
          BEQ  TRS_C
          CMPA #'Y'
          BEQ  TRS_V
          CMPA #'Z'
          BEQ  STQP
          BRA  SCCF

TRS_C     BRCLR     SCSR,X    $80  TRS_C
          LDAA $D300
          STAA SCDR,X
          BRA  ASAM

TRS_V     BRCLR     SCSR,X    $80  TRS_V
          LDAA $D301
          STAA SCDR,X
          BRA  ASAM

STQP LDAA #$00
          STAA $D950
          STAA $D200
          STAA $D201
          LDAA $D202
          LDAA #$01
          JSR  SND_CMD
          PULY
          RTS



ADRC      PSHY
          LDAA #$01
          JSR  SND_CMD
          LDY  #DATA16
SEMM      LDAA 0,Y
          BEQ  GOTK
          JSR  WRITE1
          INY
          BRA  SEMM
GOTK      LDAA #$BF
          JSR  SND_CMD
```

```
          LDY  #DATA17
SEMUM     LDAA 0,Y
          BEQ  ASA
          JSR  WRITE1
          INY
          BRA  SEMUM

ASA       CLRA
SN_CCF    BRCLR     ADCTL,X $80    SN_CCF
          LDAA ADR1,X
          JSR  DELAY1
          BRSET     SCSR,X    $80  MIT1
          BRSET     SCSR,X    $20  RVE1
          BRA  ASA
MIT1 STAA SCDR,X
          BRA  ASA
RVE1 LDAA SCDR,X
     CMPA #'N'
     BEQ  FIN
     BRA  ASA
FIN  LDAA #$00
          STAA $D950
          STAA $D200
          STAA $D201
          LDAA $D202
          LDAA #$01
          JSR  SND_CMD
          PULY
          RTS



ADRV PSHY
          LDAA #$01
          JSR  SND_CMD
          LDY  #DATA18
SEMM1     LDAA 0,Y
          BEQ  GOTK1
          JSR  WRITE1
          INY
          BRA  SEMM1
GOTK1     LDAA #$BF
          JSR  SND_CMD
          LDY  #DATA19
SEMUM1    LDAA 0,Y
          BEQ  ASA1
          JSR  WRITE1
          INY
          BRA  SEMUM1

ASA1      CLRA
```

```
SN1_CCF   BRCLR     ADCTL,X $80    SN1_CCF
          LDAA ADR3,X
          JSR  DELAY1
          BRSET     SCSR,X $80     MIT2
          BRSET     SCSR,X $20     RVE2
          BRA  ASA1
MIT2      STAA SCDR,X
          BRA  ASA1
RVE2      LDAA SCDR,X
          CMPA #'N'
          BEQ  FIN1
          BRA  ASQ
ASQ       JMP  ASA
FIN1      LDAA #$00
          STAA $D950
          STAA $D200
          STAA $D201
          LDAA $D202
          LDAA #$01
          JSR  SND_CMD
          PULY
          RTS



STOP      LDAA #%11111111
          STAA R_SWITCH
          LDAA #$00
          STAA $D950
          RTS


AMPERE    BRCLR     SWITCH,X  $01  TELLC
          BRCLR     SWITCH,X  $02  TELL1
          BRCLR     SWITCH,X  $04  TELL1
          LDAA #%11111110
          STAA R_SWITCH
          BRA  ASAL1


VOLTG     BRCLR     SWITCH,X  $01  TELL1
          BRCLR     SWITCH,X  $02  TELLC
          BRCLR     SWITCH,X  $04  TELL1
          LDAA #%11111101
          STAA R_SWITCH
          BRA  ASAL1
```

```
WATT      BRCLR     SWITCH,X  $01  TELL1
          BRCLR     SWITCH,X  $02  TELL1
          BRCLR     SWITCH,X  $04  TELLC
          LDAA #%11111011
          STAA R_SWITCH
          BRA  ASAL1


TELLC     PSHY
          LDAA #$01
          JSR  SND_CMD
          LDY  #DATA15
ULG  LDAA 0,Y
          BEQ  ASAL
          JSR  WRITE1
          INY
          BRA  ULG



TELL1     PSHY
          LDAA #$01
          JSR  SND_CMD
          LDY  #DATA13
SEM1      LDAA 0,Y
          BEQ  GOT1
          JSR  WRITE
          INY
          BRA  SEM1
GOT1      LDAA #$BF
          JSR  SND_CMD
          LDY  #DATA14
SEMU1     LDAA 0,Y
          BEQ  ASAL
          JSR  WRITE
          INY
          BRA  SEMU1

ASAL      PULY
          JSR  DELAY4
ASAL1     LDAA #$00
          STAA $D950
          STAA $D200
          STAA $D201
          LDAA $D202
          LDAA #$01
          JSR  SND_CMD
          RTS
```

```
ON_LOAD   PSHY
     LDAB #$08
     STAB PORTA,X
     JSR  DELAY2
     LDAB #$00
     STAB PORTA,X
     JSR  DELAY2
     LDAA #$01
     JSR  SND_CMD
     LDY  #ON
AGAIN1    LDAA 0,Y
     BEQ  RETURN
     JSR  WRITE
     INY
     BRA  AGAIN1
RETURN    LDAA #$01
     JSR  SND_CMD
     PULY
     LDAA #$00
     STAA $D200
     STAA $D201
     STAA $D202
     LDAA #$00
     STAA $D950
     RTS

OFFLOAD   PSHY
          LDAB #$10
          STAB PORTA,X
          JSR  DELAY2
          LDAB #$00
          STAB PORTA,X
          JSR  DELAY2
          LDAA #$01
          JSR  SND_CMD
          LDY  #OFF
AGAIN2    LDAA 0,Y
          BEQ  BACK
          JSR  WRITE
          INY
          BRA  AGAIN2
BACK      LDAA #$01
          JSR  SND_CMD
          PULY
          LDAA #$00
          STAA $D200
          STAA $D201
          STAA $D202
          LDAA #$00
          STAA $D950
```

```
          RTS

SAVE1     PSHY
     LDAA #'C'
     STAA $D900
     LDAA #$01
     JSR  SND_CMD
     LDY  #DATA10
KIH  LDAA 0,Y
     BEQ  SCAN1
     JSR  WRITE1
     INY
     BRA  KIH

SCAN1     BRCLR     SCSR,X $80 SCAN1
          LDAA #'I'
          STAA SCDR,X

SCAN2     BRCLR     SCSR,X $20 SCAN2
          LDAB SCDR,X
          STAB $D800
          JSR  SET1
          JSR  KE11
          JSR  DELAY4
          LDAA #$01
          JSR  SND_CMD
          PULY
          LDAA #$00
          STAA $D200
          STAA $D201
          STAA $D202
          LDAA #$00
          STAA $D900
          LDAA #$00
          STAA $D950
          RTS

SAVE2     PSHY
          LDAA #'C'
          STAA $D901
          LDAA #$01
          JSR  SND_CMD
          LDY  #DATA11
KIH1      LDAA 0,Y
          BEQ  SCAN3
          JSR  WRITE1
          INY
          BRA  KIH1

SCAN3     BRCLR     SCSR,X $80 SCAN3
          LDAA #'J'
```

```
          STAA SCDR,X
SCAN4     BRCLR     SCSR,X $20 SCAN4
     LDAB SCDR,X
     STAB $D801
     JSR  SET2
     JSR  KE22
     JSR  DELAY4
     LDAA #$01
     JSR  SND_CMD
     PULY
     LDAA #$00
     STAA $D200
     STAA $D201
     STAA $D202
     LDAA #$00
     STAA $D901
     LDAA #$00
     STAA $D950
     RTS

DEFAULT   PSHY
          LDAA #$01
          JSR  SND_CMD
          LDY  #DATA12
KIH2 LDAA 0,Y
          BEQ  SCAN5
          JSR  WRITE
          INY
          BRA  KIH2

SCAN5     LDAA #170
          STAA $D800
          LDAA #250
          STAA $D801

          LDAA #$01
          JSR  SND_CMD
          PULY
          LDAA #$00
          STAA $D200
          STAA $D201
          STAA $D202
          LDAA #$00
          STAA $D950
          RTS


************* TRANSMIT CHARACTER ****************

TRNSMIT   LDAA #'I'
          STAA SCDR,X
```

```
EMIT1     BRCLR     SCSR,X $80 EMIT1
          LDAA $D700
          STAA SCDR,X
EMIT2     BRCLR     SCSR,X $80 EMIT2
          LDAA #'V'
          STAA SCDR,X
EMIT3     BRCLR     SCSR,X $80 EMIT3
          LDAA $D701
          STAA SCDR,X
          RTS

*********************************************

WRITE     STAA CHAR
          BSR  WDAT
          RTS

WRITE1    STAA CHAR
          BSR  WDAT1
          RTS

WDAT      LDAA #$01
          BSR  SND_CMD
          RTS

WDAT1     LDAA #$01
          BSR  SNDcmd
          RTS

DELAY1    PSHA
          PSHX

          LDAA #1
REPEAT1   LDX  #$28
REPT11    DEX
          BNE  REPT11
          DECA
          BNE  REPEAT1

          PULX
          PULA
          RTS

SND_CMD   STAA COMAND         ;STORE COMMAND
          BSR  DELAY2
          RTS

SNDcmd    STAA COMAND         ;STORE COMMAND
          BSR  DELAY3
          RTS
```

```
DELAY2    PSHX
          LDX  #$FFFF
REPEAT2   DEX
          BNE  REPEAT2
          PULX
          RTS

DELAY3    PSHX
          LDX  #$1111
REPEAT3   DEX
          BNE  REPEAT3
          PULX
          RTS

DELAY4    PSHA
          PSHX

          LDAA #$15
R1        LDX  #$FFFF
RPT11     DEX
          BNE  RPT11
          DECA
          BNE  R1

          PULX
          PULA
          RTS


***************CONVERTER***********************

CNVTER1   BRCLR     SWITCH,X  $01  CRT_CVT
          BRCLR     SWITCH,X  $02  VL_CVT
          BRCLR     SWITCH,X  $04  PWR_CVT
          BRCLR     R_SWITCH  $01  CRT_CVT
          BRCLR     R_SWITCH  $02  VL_CVT
          BRCLR     R_SWITCH  $04  PWR_CVT
          PSHX
          PSHY
          JSR  NO_OPR
          PULY
          PULX
          RTS

CRT_CVT   PSHX
          LDAA $D700
          BRA  HIT1
SET1      PSHX
          LDAA $D800
HIT1      LDAB #15
          MUL
```

```
          LDX  #255
          IDIV
          STD  $D001
          STX  $D003
          LDX  #255
          FDIV
          STD  $D005
          STX  $D007
          CLRA
          LDAB $D007
          BRA  BIT7_1

VL_CVT    PSHX
          LDAA $D701
          BRA  HIT2
SET2      PSHX
          LDAA $D801
HIT2      LDAB #98
          MUL
          LDX  #100
          IDIV
          STD  $D009
          STX  $D00B
          LDX  #100
          FDIV
          STD  $D00D
          STX  $D00F
          CLRA
          LDAB $D00F
          BRA  BIT7_1

PWR_CVT   PSHX
          LDAA $D700
          LDAB $D701
          MUL
          LDX  #1735
          IDIV
          STD  $D011
          STX  $D013
          LDX  #1735
          FDIV
          STD  $D015
          STX  $D017
          CLRA
          LDAB $D017
          BRA  BIT7_1


******* LAST NUMBER*************************
```

```
BIT7_1    BITB #$80
          BEQ  BIT6_1
          ADDA #0
BIT6_1    BITB #$40
          BEQ  BIT5_1
          ADDA #0
BIT5_1    BITB #$20
          BEQ  BIT4_1
          ADDA #5
BIT4_1    BITB #$10
          BEQ  BIT3_1
          ADDA #5
BIT3_1    BITB #$08
          BEQ  BIT2_1
          ADDA #3
BIT2_1    BITB #$04
          BEQ  BIT1_1
          ADDA #6
BIT1_1    BITB #$02
          BEQ  BIT0_1
          ADDA #8
BIT0_1    BITB #$01
          BEQ  CONVRT1
          ADDA #0

CONVRT1   CLRB
          CMPA #9
          BLS  ADJUST1
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST1
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST1
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST1
          ADDB #1
          SUBA #10
ADJUST1   ADDA    #$30
          STAA    $D066

*********************************************
***********SECOND LAST***********************

          TBA
```

```
          CLRB
          BRCLR     SWITCH,X  $01  LCRNT_1
          BRCLR     SWITCH,X  $02  LVOLT_1
          BRCLR     SWITCH,X  $04  LPOWR_1
          BRCLR     R_SWITCH  $01  LCRNT_1
          BRCLR     R_SWITCH  $02  LVOLT_1
          BRCLR     R_SWITCH  $04  LPOWR_1

LCRNT_1   LDAB $D007
          BRA  NEW_1
LVOLT_1   LDAB $D00F
          BRA  NEW_1
LPOWR_1   LDAB $D017
          BRA  NEW_1

NEW_1     BITB #$80
          BEQ  BIT6_2
          ADDA #0
BIT6_2    BITB #$40
          BEQ  BIT5_2
          ADDA #0
BIT5_2    BITB #$20
          BEQ  BIT4_2
          ADDA #2
BIT4_2    BITB #$10
          BEQ  BIT3_2
          ADDA #2
BIT3_2    BITB #$08
          BEQ  BIT2_2
          ADDA #1
BIT2_2    BITB #$04
          BEQ  BIT1_2
          ADDA #5
BIT1_2    BITB #$02
          BEQ  BIT0_2
          ADDA #7
BIT0_2    BITB #$01
          BEQ  CONVRT2
          ADDA #4

CONVRT2   CLRB
          CMPA #9
          BLS  ADJUST2
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST2
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST2
```

```
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST2
          ADDB #1
          SUBA #10
ADJUST2   ADDA    #$30
          STAA    $D065

*********************************************
**********THIRD LAST*************************


          TBA
          CLRB
          BRCLR     SWITCH,X  $01  LCRNT_2
          BRCLR     SWITCH,X  $02  LVOLT_2
          BRCLR     SWITCH,X  $04  LPOWR_2
          BRCLR     R_SWITCH  $01  LCRNT_2
          BRCLR     R_SWITCH  $02  LVOLT_2
          BRCLR     R_SWITCH  $04  LPOWR_2

LCRNT_2   LDAB $D007
          BRA  NEW_2
LVOLT_2   LDAB $D00F
          BRA  NEW_2
LPOWR_2   LDAB $D017
          BRA  NEW_2

NEW_2     LDAB $D007
          BITB #$80
          BEQ  BIT6_3
          ADDA #0
BIT6_3    BITB #$40
          BEQ  BIT5_3
          ADDA #5
BIT5_3    BITB #$20
          BEQ  BIT4_3
          ADDA #1
BIT4_3    BITB #$10
          BEQ  BIT3_3
          ADDA #6
BIT3_3    BITB #$08
          BEQ  BIT2_3
          ADDA #3
BIT2_3    BITB #$04
          BEQ  BIT1_3
          ADDA #1
BIT1_3    BITB #$02
          BEQ  BIT0_3
          ADDA #0
```

```
BIT0_3    BITB #$01
          BEQ  CONVRT3
          ADDA #0

CONVRT3   CLRB
          CMPA #9
          BLS  ADJUST3
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST3
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST3
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST3
          ADDB #1
          SUBA #10
ADJUST3   ADDA    #$30
          STAA    $D064

*********************************************
*********FURTH LAST*****************************

          TBA
          CLRB
          BRCLR     SWITCH,X  $01  LCRNT_3
          BRCLR     SWITCH,X  $02  LVOLT_3
          BRCLR     SWITCH,X  $04  LPOWR_3
          BRCLR     R_SWITCH  $01  LCRNT_3
          BRCLR     R_SWITCH  $02  LVOLT_3
          BRCLR     R_SWITCH  $04  LPOWR_3

LCRNT_3   LDAB $D007
          BRA  NEW_3
LVOLT_3   LDAB $D00F
          BRA  NEW_3
LPOWR_3   LDAB $D017
          BRA  NEW_3

NEW_3     LDAB $D007
          BITB #$80
          BEQ  BIT6_4
          ADDA #5
BIT6_4    BITB #$40
          BEQ  BIT5_4
          ADDA #2
BIT5_4    BITB #$20
```

```
          BEQ  BIT4_4
          ADDA #0
BIT4_4    BITB #$10
          BEQ  BIT3_4
          ADDA #0
BIT3_4    BITB #$08
          BEQ  BIT2_4
          ADDA #0
BIT2_4    BITB #$04
          BEQ  BIT1_4
          ADDA #0
BIT1_4    BITB #$01
          BEQ  BIT0_4
          ADDA #0
BIT0_4    BITB #$40
          BEQ  CONVRT4
          ADDA #0

CONVRT4   CLRB
          CMPA #9
          BLS  ADJUST4
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST4
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST4
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST4
          ADDB #1
          SUBA #10
ADJUST4   ADDA    #$30
          STAA    $D063

*********************************************

************FIFTH LAST***********************

          CLRA
          BRCLR     SWITCH,X  $01  LCRNT_4
          BRCLR     SWITCH,X  $02  LVOLT_4
          BRCLR     SWITCH,X  $04  LPOWR_4
          BRCLR     R_SWITCH  $01  LCRNT_4
          BRCLR     R_SWITCH  $02  LVOLT_4
          BRCLR     R_SWITCH  $04  LPOWR_4

LCRNT_4   LDAB $D004
```

```
           BRA  NEW_4
LVOLT_4    LDAB $D00C
           BRA  NEW_4
LPOWR_4    LDAB $D014
           BRA  NEW_4

NEW_4      BITB #$80
           BEQ  BIT6_5
           ADDA #8
BIT6_5     BITB #$40
           BEQ  BIT5_5
           ADDA #4
BIT5_5     BITB #$20
           BEQ  BIT4_5
           ADDA #2
BIT4_5     BITB #$10
           BEQ  BIT3_5
           ADDA #6
BIT3_5     BITB #$08
           BEQ  BIT2_5
           ADDA #8
BIT2_5     BITB #$04
           BEQ  BIT1_5
           ADDA #4
BIT1_5     BITB #$02
           BEQ  BIT0_5
           ADDA #2
BIT0_5     BITB #$01
           BEQ  CONVRT5
           ADDA #1

CONVRT5    CLRB
           CMPA #9
           BLS  ADJUST5
           ADDB #1
           SUBA #10
           CMPA #9
           BLO  ADJUST5
           ADDB #1
           SUBA #10
           CMPA #9
           BLO  ADJUST5
           ADDB #1
           SUBA #10
           CMPA #9
           BLO  ADJUST5
           ADDB #1
           SUBA #10
           CMPA #9
           BLS  ADJUST5
           ADDB #1
```

```
ADJUST5   ADDA    #$30
          STAA    $D062

*************************************************
*****SIXTH LAST*********************************

          TBA
          CLRB
          BRCLR     SWITCH,X  $01  LCRNT_5
          BRCLR     SWITCH,X  $02  LVOLT_5
          BRCLR     SWITCH,X  $04  LPOWR_5
          BRCLR     R_SWITCH  $01  LCRNT_5
          BRCLR     R_SWITCH  $02  LVOLT_5
          BRCLR     R_SWITCH  $04  LPOWR_5

LCRNT_5   LDAB $D004
          BRA  NEW_5
LVOLT_5   LDAB $D00C
          BRA  NEW_5
LPOWR_5   LDAB $D014
          BRA  NEW_5

NEW_5     BITB #$80
          BEQ  BIT6_6
          ADDA #2
BIT6_6    BITB #$40
          BEQ  BIT5_6
          ADDA #6
BIT5_6    BITB #$20
          BEQ  BIT4_6
          ADDA #3
BIT4_6    BITB #$10
          BEQ  BIT3_6
          ADDA #1
BIT3_6    BITB #$08
          BEQ  BIT2_6
          ADDA #0
BIT2_6    BITB #$04
          BEQ  BIT1_6
          ADDA #0
BIT1_6    BITB #$02
          BEQ  BIT0_6
          ADDA #0
BIT0_6    BITB #$01
          BEQ  CONVRT6
          ADDA #0

CONVRT6   CLRB
          CMPA #9
          BLS  ADJUST6
          ADDB #1
```

```
          SUBA #10
          CMPA #9
          BLO  ADJUST6
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST6
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST6
          ADDB #1
          SUBA #10
ADJUST6   ADDA    #$30
          STAA    $D061

*********************************************
**********SEVENTH LAST************************


          TBA
          CLRB
          BRCLR     SWITCH,X  $01  LCRNT_6
          BRCLR     SWITCH,X  $02  LVOLT_6
          BRCLR     SWITCH,X  $04  LPOWR_6
          BRCLR     R_SWITCH  $01  LCRNT_6
          BRCLR     R_SWITCH  $02  LVOLT_6
          BRCLR     R_SWITCH  $04  LPOWR_6

LCRNT_6   LDAB $D004
          BRA  NEW_6
LVOLT_6   LDAB $D00C
          BRA  NEW_6
LPOWR_6   LDAB $D014
          BRA  NEW_6

NEW_6     BITB #$80
          BEQ  BIT6_7
          ADDA #1
BIT6_7    BITB #$40
          BEQ  BIT5_7
          ADDA #0
BIT5_7    BITB #$20
          BEQ  BIT4_7
          ADDA #0
BIT4_7    BITB #$10
          BEQ  BIT3_7
          ADDA #0
BIT3_7    BITB #$08
          BEQ  BIT2_7
          ADDA #0
```

```
BIT2_7    BITB #$04
          BEQ  BIT1_7
          ADDA #0
BIT1_7    BITB #$02
          BEQ  BIT0_7
          ADDA #0
BIT0_7    BITB #$01
          BEQ  CONVRT7
          ADDA #0

CONVRT7   CLRB
          CMPA #9
          BLS  ADJUST7
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST7
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST7
          ADDB #1
          SUBA #10
          CMPA #9
          BLO  ADJUST7
          ADDB #1
          SUBA #10
ADJUST7   ADDA    #$30
          STAA    $D060

*********************************************

          PULX
          RTS



DATA      FCC  "INTELLIGENT BOX"
          FCB  0
NAMA      FCC  "  BY M.KHAIRI  "
          FCB  0
DATA1     FCC  "CURRENT"
          FCB  0
DATA2     FCC  " NO OPERATION  "
          FCB  0
DATA3     FCC  "VOLTAGE"
          FCB  0
DATA4     FCC  "POWER"
          FCB  0
DATA5     FCC  "SELECT MODE NOW!"
```

```
          FCB  0
DATA6     FCC  "   WARNING  !! "
          FCB  0
DATA7     FCC  "  CURRENT HIGH "
          FCB  0
DATA8     FCC  "   WARNING  !! "
          FCB  0
DATA9     FCC  "  VOLTAGE HIGH "
          FCB  0
DATA10    FCC  "LIMITED CURRENT"
          FCB  0
DATA11    FCC  "LIMITED VOLTAGE"
          FCB  0
DATA12    FCC  "DEFAULT LIMITED"
          FCB  0
DATA13    FCC  " BUSY ! ! CHECK"
          FCB  0
DATA14    FCC  "EXTERNAL SWITCH"
          FCB  0
DATA15    FCC  "ALREADY RUNNING"
          FCB  0
DATA16    FCC  "ACCURACY CURRENT"
          FCB  0
DATA17    FCC  "TRANSFER TO PC"
          FCB  0
DATA18    FCC  "ACCURACY VOLTAGE"
          FCB  0
DATA19    FCC  "TRANSFER TO PC"
          FCB  0
DATA20    FCC  "HAND SHAKE ! ! "
          FCB  0
DATA21    FCC  "REQUEST FROM PC"
          FCB  0
OFF       FCC  " TURN OFF LOAD "
          FCB  0
ON        FCC  " TURN ON LOAD "
          FCB  0

          ORG  $FFFE
          FDB  $E000
          END
```