DEVELOPMENT OF PC BASED HOME ENERGY MANAGEMENT SYSTEM

MUHAMAD KHAIRI BIN CHE YUSOFF

UNIVERSITI MALAYSIA PAHANG
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.
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 exceed 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.

![Diagram of microprocessor and microcontroller](image)

**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.
2.1.1.1 Motorola MC68HC11

The most commonly used microcontroller nowadays are 8-bit and 16-bit microcontrollers. Other 32-bit embedded controllers 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 versions such as 8031, 8751, 8052, Microchip PIC such as P1C16F84 and BASIC STAMP and Motorola 68H005, 68HC08, 68HC11 and Zilog Z8.

The most commonly used microcontroller nowadays are 8-bit and 16-bit microcontrollers. Other 32-bit embedded controllers 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
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

<table>
<thead>
<tr>
<th>Part Number</th>
<th>EPROM</th>
<th>ROM</th>
<th>RAM</th>
<th>CONFIG</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC68HC11A8</td>
<td>—</td>
<td>—</td>
<td>512</td>
<td>256</td>
<td>50F Family Built Around This Device</td>
</tr>
<tr>
<td>MC68HC11A1</td>
<td>—</td>
<td>—</td>
<td>512</td>
<td>256</td>
<td>A8 with ROM Disabled</td>
</tr>
<tr>
<td>MC68HC11A0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>256</td>
<td>50D A8 with ROM and EEPROM Disabled</td>
</tr>
<tr>
<td>MC68HC811A8</td>
<td>—</td>
<td>—</td>
<td>8K+</td>
<td>256</td>
<td>50F EEPROM Emulator for A8</td>
</tr>
<tr>
<td>MC68HC11E9</td>
<td>—</td>
<td>12K</td>
<td>512</td>
<td>512</td>
<td>50F Four Input Capture/Bigger RAM 12K ROM</td>
</tr>
<tr>
<td>MC68HC11E1</td>
<td>—</td>
<td>—</td>
<td>512</td>
<td>512</td>
<td>50D E9 with ROM Disabled</td>
</tr>
<tr>
<td>MC68HC11E0</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>512</td>
<td>50C E9 with ROM and EEPROM Disabled</td>
</tr>
<tr>
<td>MC68HC611E2</td>
<td>—</td>
<td>—</td>
<td>2K^1</td>
<td>256</td>
<td>5FF No ROM Part for Expanded Systems</td>
</tr>
<tr>
<td>MC68HC711E9</td>
<td>12K</td>
<td>—</td>
<td>512</td>
<td>512</td>
<td>50F One-Time Programmable Version of E9</td>
</tr>
<tr>
<td>MC68HC11D3</td>
<td>4K</td>
<td>—</td>
<td>192</td>
<td>192</td>
<td>N/A Low-Cost 40-Pin Version</td>
</tr>
<tr>
<td>MC68HC711D9</td>
<td>4K</td>
<td>—</td>
<td>192</td>
<td>192</td>
<td>N/A One-Time Programmable Version of D3</td>
</tr>
<tr>
<td>MC68HC11F1</td>
<td>—</td>
<td>—</td>
<td>512</td>
<td>1K</td>
<td>5FF High-Performance Non-Multiplexed 68-B-Pin</td>
</tr>
<tr>
<td>MC68HC11K4</td>
<td>—</td>
<td>24K</td>
<td>640</td>
<td>768</td>
<td>5FF &gt; 1 Mbyte memory space, PWM, Cx, 84-Pin</td>
</tr>
<tr>
<td>MC68HC711K4</td>
<td>24K</td>
<td>—</td>
<td>640</td>
<td>768</td>
<td>5FF One-Time Programmable Version of K4</td>
</tr>
<tr>
<td>MC68HC11L6</td>
<td>—</td>
<td>16K</td>
<td>512</td>
<td>512</td>
<td>50F Like 'E9 with more ROM and more I/O, 64/68</td>
</tr>
<tr>
<td>MC68HC711L6</td>
<td>16K</td>
<td>—</td>
<td>512</td>
<td>512</td>
<td>50F One-Time Programmable Version of L4</td>
</tr>
</tbody>
</table>
2.1.1.2 Motorola 68HC11 Architecture

The MicroStamp11 module is built around the Motorola 68HC11 microcontroller 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](image)

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

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

*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 alphanumerical, 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.

![VCM162A LCD display](image)

**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
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.

![Voltage regulator](image)

**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:

```matlab
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.

```matlab
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:

```matlab
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 laying 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.

![GUIDE Layout Editor](image)

**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.