UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS♦

JUDUL: **DUAL MODE MOBILE ROBOT: APPLICATION USING MATLAB GUI**

SESI PENGAJIAN: **2008/2009/I**

Saya **SAIFULLAH BIN AHMAD (860514-29-6145)**
(HURUF BESAR)

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

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
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:	
KG ALOR PIAH CHETOK, **17060 PASIR MAS,** **KELANTAN**	**REZA EZUAN BIN SAMIN** (Nama Penyelia)
Tarikh: **11 NOVEMBER 2008**	Tarikh: : **11 NOVEMBER 2008**

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 Engineering (Electronic)”

Signature : ....

Name of Supervisor : REZA EZUAN BIN SAMIN

Date : 11 NOVEMBER 2008

DUAL MODE MOBILE ROBOT: APPLICATION USING MATLAB GUI

SAIFULLAH BIN AHMAD

A thesis submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Electrical Engineering (Electronic)

Faculty of Electrical & Electronic Engineering
Universiti Malaysia Pahang

NOVEMBER 2008

“I declare that this thesis is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.”

Signature : ....
Name : SAIFULLAH BIN AHMAD
Date : 11 NOVEMBER 2008

To my beloved mother and father:
Zabiah Binti Saleh and Ahmad Bin Sulaiman
My siblings:
Zalina Ahmad, Zaleharani Ahmad, Zanariah Ahmad,
Sofiah Ahmad and my youngest brother, Hasan Ahmad
Last but not least a special people in my life, Asmahani Ahmad

ACKNOWLEDGEMENT

First of all, thanks to ALLAH s.w.t because I have completely finished my thesis. I want to appreciate to all people who had given their support to make ‘Dual Mode Mobile Robot: Application Using MATLAB GUI’ successful. I am greatly indebted to my supervisor, En Reza Ezuan Bin Samin for his advice and guide to finish this project.

Here, I also want to thank to all my beloved friends that always give a moral support and to complete this project. Suggestions and criticism from my friends have always been helpful in finding a solution to my problem. Thank you very much all.

Finally, I would like to express my thanks to those who were involved directly or indirectly in the completion of my thesis. Last but not least, to my family and also my love, for giving me a moral and inspiration throughout my project.

ABSTRAK

(Pergerakan robot 2 kaedah) ini mengandungi program komputer dan bahagian peralatan. Untuk bahagian program komputer, MATLAB GUI digunakan dan untuk bahagian peralatan pula cip PIC, LCD, alat pengesan jarak dan banyak lagi digunakan untuk menyiapkan projek ini. Projek ini terbahagi kepada dua bahagian iaitu dalam keadaan automatik dan kawalan secara biasa. Dalam keadaan automatik pula, alat pengesan jarak digunakan untuk mengawal robot dan robot itu akan berhenti dalam jarak yang tertentu. Jarak ini akan dipaparkan pada LCD. Kabel rs232 digunakan untuk menyambungkan peralatan dengan program MATLAB. Fokus utama projek ini adalah interaksi antara program MATLAB dengan peralatan dengan menggunakan kabel rs232. Pemaparan grafik (GUI) mengandungi peralatan dan komponen yang boleh melaksanakan pelbagai tugasan. Untuk melaksanakan tugasan ini, pengguna GUI tidak perlu membuat skrip atau suruhan. Tambahan lagi, pengguna tidak perlu mengetahui secara terperinci tentang tugasan. Cip PIC pula mengandungi sistem pemprosesan dan tempat untuk menyimpan data yang lengkap dalam satu pakej yang murah dan senang untuk digunakan.

ABSTRACT

Dual mode mobile robot has hardware and software part. For the software part, we use MATLAB GUI and for the hardware, it consists PIC 16F877A, sensor, LCD, relay and others. There are two mode to control the mobile using PIC 16F877A by interfacing with MATLAB GUI. The two modes are manual and autonomous mode. For the manual we control the mobile with GUI. For the autonomous mode, distance sensor is use to control the mobile and stop the mobile with certain distance that we have set up. The distance will display on LCD at hardware. Rs232 cable is use to connect hardware with software (PC). This project is focus on interfacing MATLAB software and hardware using rs232 cable. A graphical user interface (GUI) is a graphical display that contains devices, or components, that enable a user to perform interactive tasks. To perform these tasks, the user of the GUI does not have to create a script or type commands at the command line. Often, the user does not have to know the details of the task at hand. A PIC Microcontroller chip combines the function of microprocessor, ROM program memory, some RAM memory and input-output interface in one single package which is economical and easy to use.

TABLES OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Overview

This project is focus on designing the Graphical User Interface (GUI) through MATLAB to control the DC motor using PIC. The PIC is a programmable interface devices or controller between PC (MATLAB GUI) and the DC motor. The main contribution of this project is the interfacing of the MATLAB with PIC and Graphical User Interface (GUI).

The Peripheral Interface Controller (PIC) use in this project is as controller device between Personal Computer, analog distance sensor and the DC motor. The PIC is use because of wide availability and economical. Beside that PIC is a free development tools and can perform many function without needed extra circuitry. PIC also have analog to digital converter that will be use to connect with analog distance sensor.

The purpose using MATLAB in creating the GUI is because it already has Graphical User Interface Development Environment (GUIDE) that provides a set of tools for creating GUI. These tools simplify the process of lay out and programming GUIs. The GUI create in MATLAB with appropriate coding will control the DC motor via serial port that interface with the PIC.

The GUI create in MATLAB with appropriate coding will control the DC motor via serial port that interface with the PIC. There are many advantage by using the DC motor, among that the DC motor has no adverse effect on power quality and the speed is proportional to the magnetic flux.

This project is to control the mobile robot by using GUI in MATLAB and PIC controller. There are two modes to control the mobile robot. The first mode is control mobile robot manually. For the second mode (autonomous), to control the mobile robot we use analog distance sensor to detect the distance between the robot and wall. Then it will stop automatically in certain distance after detect the block or wall.

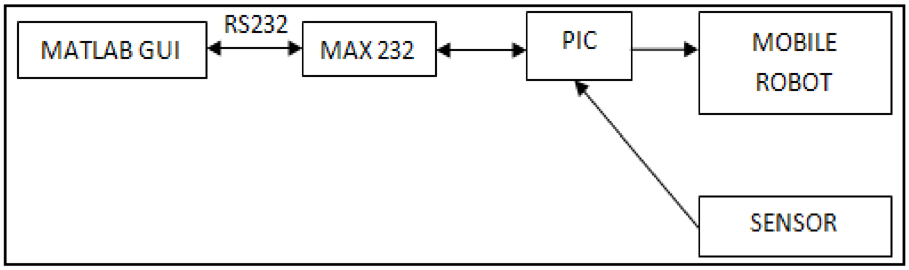


Figure 1.1: Block Diagram of Project

1.2 Objective

These projects have two main objectives. The objective of this project is to:

i. To control the mobile robot using GUI in MATLAB
ii. Able to interface the MATLAB GUI with hardware using PIC

The important part of this project is to interface the MATLAB GUI with the PIC. Then, important part of this project is to receive a signal from sensor that will transmit to MATLAB GUI and interface using PIC. After that, the programming will send the signal to control the mobile robot automatically.

1.3 Scope of project

The scopes of this project are laying out the GUI in MATLAB GUIDE and create programming for the GUI's. Secondly Prepare the PIC circuitry and serial 3 connections (DB9) circuit for interfacing part. For the third part is to build IR sensor circuit and interface with PIC. And the last part is creating program for PIC using PICBasic Pro Compiler to control the DC motor.

For this project, there are two scopes. The scope of project is dividing to software part and hardware part:

For the software part, we have:

i. MATLAB programming

ii. PIC programming
iii. PICBasic Pro Compiler

For the hardware part, we have:

i. 2 ways serial parallel port (transmit and received input or output)
ii. PIC 16F877A
iii. DC Motor and other components
iv. Distance sensor6V relays
v. LCD

1.4 Problem statement

The main objective in this project is to interface the MATLAB GUI with the PIC. It is a difficult part to develop the program for MATLAB and the PIC simultaneously to make the interfacing part. By using the PicBasic Pro Compiler software to develop programming to control DC motor, it can reduces the difficulty by comprises a list of statements that written in a programming language like assembler, C, or PBASIC. With this opportunity, the men in charge do not have to take long time to written and troubleshoot the program.

To interface MATLAB GUI with PIC controller we use RS232. For my project it will use bidirectional communication to transmit and receive the data. Previously, not much has develop application using PIC via RS232 bidirectional communication in MATLAB GUI due to its difficulties. The development of this project is for research purpose in MATLAB communication using rs232.

1.5 Thesis organization

This thesis consists of five chapters including of the first chapter. The contents of each chapter will explain details about this project.

Chapter 2 contains a detailed description each part of project. It will explain about the MATLAB and MATLAB GUI, PIC, sensor and DC motor. The project methodology is in Chapter 3. This will explain how the project is organized and the flow of the process in completing this project.

Chapter 4 will presents the expected result of simulation runs using MATLAB GUI interface with PIC. It also will show how the sensor will function to send an input signal to GUI and will transmit back to PIC to control the DC motor. In this chapter also will explain how the sensor works as automatic controller. Finally the conclusions for this project are presented in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

2.1 Graphical User Interface (GUI)

2.1.1 GUI definition

A graphical user interface (or GUI, often pronounced "gooey"), is a particular case of user interface for interacting with a computer which employs graphical images and widgets in addition to text to represent the information and actions available to the user. Usually the actions are performed through direct manipulation of the graphical elements [1].

The first graphical user interface was designed by Xerox Corporation's Palo Alto Research Center in the 1970s, but it was not until the 1980s and the emergence of the Apple Macintosh that graphical user interfaces became popular. One reason for their slow acceptance was the fact that they require considerable CPU power and a high-quality monitor, which until recently were prohibitively expensive [2].

A graphical user interface (GUI) is a pictorial interface to a program. 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 [3]. A true GUI includes standard formats for representing text and graphics [3]. The GUI should behave in an understandable and predictable manner, so that a user knows what to expect when he or she performs an action. For example, when a mouse click occurs on a pushbutton, the GUI should initiate the action described on the label of the button [4].

Many DOS programs include some features of GUIs, such as menus, but are not graphics based. Such interfaces are sometimes called graphical character-based user interfaces to distinguish them from true GUIs [4].

2.1.2 MATLAB GUI

A graphical user interface (GUI) is a graphical display that contains devices, or components, that enable a user to perform interactive tasks. To perform these tasks, the user of the GUI does not have to create a script or type commands at the command line. Often, the user does not have to know the details of the task at hand [5].

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 [5].

There are two basic tasks in process to implement a GUI. The two basic tasks in process of implementing a GUI are:

i. Laying out a GUI where MATLAB implement GUIs as figure windows containing various styles of uicontrol (User Interface) objects.
ii. Programming the GUI, where each object must be program to perform the intended action when activated by the user of GUI [5].

2.1.3 MATLAB GUIDE

GUIDE, the MATLAB graphical user interface development environment, provides a set of tools for creating graphical user interfaces (GUIs). These tools simplify the process of laying out and programming GUIs [6].

i. GUIDE is primarily a set of layout tools
ii. GUIDE also generates an M-file that contains code to handle the initialization and launching of the GUI

The M-file will provide a framework for the implementation of the callbacks, the functions that execute when users activate a component in the GUI [6].

2.1.4 GUI operation

The GUI is already associated with one or more user written routines known as callbacks. The execution of each callback is triggered by a particular user action such as, mouse click, pushbuttons, toggle buttons, lists, menus, text boxes, selection of a menu item, or the cursor passing over a component and so forth 7].

By clicking the button triggers the execution of a callback. A mouse click or a key press is an event, and the MATLAB program must respond to each event if the program is to perform its function. For example, if a user clicks on a button, that event must cause the MATLAB code that implements the function of the button to be executed. The code executed in response to an event is known as a call back [7].

This kind of programming is often referred to as event-driven programming. The event in the example is a button click. In event-driven programming, callback execution is asynchronous, controlled by events external to the software. In the case of MATLAB GUIs, these events usually take the form of user interactions with the GUI [7].

The writer of a callback has no control over the sequence of events that leads to its execution or, when the callback does execute, what other callbacks might be running simultaneously [7].

Callbacks:

i. Routine that executes whenever you activate the uicontrol object.
ii. Define this routine as a string that is a valid MATLAB expression or the name of an M-file.
iii. The expression executes in the MATLAB workspace.

2.2 PIC Microcontroller

PIC is a family of Harvard architecture microcontrollers made by Microchip Technology, derived from the PIC1650 originally developed by General Instrument's Microelectronics Division. PICs are popular with developers due to their low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming (and re-programming with flash memory) capability[8].

2.2.1 History of PIC

The original PIC was built to be used with GI's new 16-bit CPU, the CP1600. While generally a good CPU, the CP1600 had poor I/O performance, and the 8-bit PIC was developed in 1975 to improve performance of the overall system by offloading I/O tasks from the CPU [8].

The PIC used simple microcode stored in ROM to perform its tasks, and although the term wasn't used at the time, it is a RISC design that runs one instruction per cycle (4 oscillator cycles) [8].

In 1985 General Instruments spun off their microelectronics division, and the new ownership cancelled almost everything which by this time was mostly out-of-date. The PIC, however, was upgraded with EPROM to produce a programmable channel controller, and today a huge variety of PICs are available with various on-board peripherals (serial communication modules, UARTs, motor control kernels, etc.) and program memory from 512 words to 32k words and more[8].

2.2.2 The PIC16F877A Microcontroller

A PIC PIC16F877A Microcontroller chip combines the function of microprocessor, ROM program memory, some RAM memory and input-output interface in one single package which is economical and easy to use. The PIC devices generally feature is sleep mode (power saving), watchdog timer and various crystal or RC oscillator configuration, or an external clock [9].

Logicator system is designed to be used to program a range of 8, 18, 28 pin reprogrammable PIC microcontroller which provide a variety of input output, digital input and analogue input options to suit students project uses [9]. Reprogrammable "FLASH Memory" chips have been selected as the most economical for student use. If a student needs to amend to control system as the project is evaluated and developed, the chip can simply be taken out of the product and reprogrammed with an edited version of the floe sheet [9].

The features that available in PIC 16F877A:

i. General purpose i/o pins
ii. Internal clock oscillators
iii. 8/16 Bit Timers
iv. Internal EEPROM Memory
v. Synchronous/Asynchronous Serial Interface USART

vi. MSSP Peripheral for I²C and SPI Communications

vii. Capture/Compare and PWM modules

viii. Analog-to-digital converters

ix. USB, Ethernet, CAN interfacing support

x. External memory interface

xi. Integrated analog RF front ends (PIC16F639, and rfPIC)

xii. KEELOQ Rolling code encryption peripheral (encode/decode)

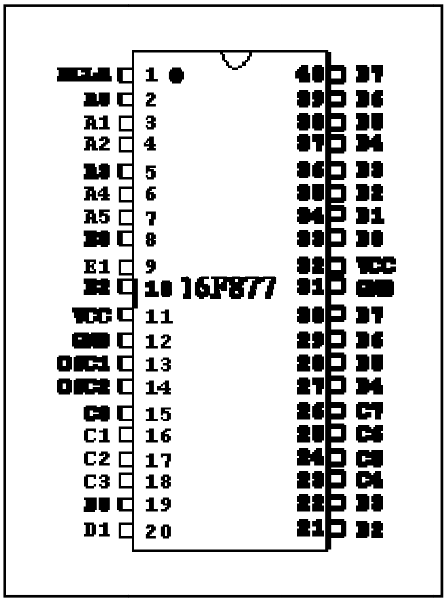


Figure 2.1: IC Configuration

2.2.3 PIC Basic Pro Compiler

The PicBasic Pro Compiler (or PBP) makes it even quicker and easier to program Microchip Technology's powerful PICmicro microcontrollers (MCUs). The

English-like BASIC language is much easier to read and write than the quirky Microchip assembly language [10].

The PicBasic Pro Compiler is “BASIC Stamp II like” and has most of the libraries and functions of both the BASIC Stamp I and II. Being a true compiler, programs execute much faster and may be longer than their Stamp equivalents. PBP is not quite as compatible with the BASIC Stamps as our original [10].

The PicBasic Pro Compiler produces code that may be programmed into a wide variety of PICmicro microcontrollers having from 8 to 84 pins and various on chip features including A/D converters, hardware timers and serial ports [10].

2.3 DC Motor

2.3.1 History of DC Motor

At the most basic level, electric motors exist to convert electrical energy into mechanical energy. This is done by way of two interacting magnetic fields one stationary, and another attached to a part that can move. A number of types of electric motors exist, but most BEAMbots use DC motors in some form or another. DC motors have the potential for very high torque capabilities (although this is generally a function of the physical size of the motor), are easy to miniaturize, and can be "throttled" via adjusting their supply voltage. DC motors are also not only the simplest, but the oldest electric motors [11].

The basic principles of electromagnetic induction were discovered in the early 1800's by Oersted, Gauss, and Faraday. By 1820, Hans Christian Oersted and Andre Marie Ampere had discovered that an electric current produces a magnetic field. The next 15 years saw a flurry of cross-Atlantic experimentation and innovation, leading finally to a simple DC rotary motor. A number of men were involved in the work, so proper credit for the first DC motor is really a function of just how broadly you choose to define the word "motor"[11].

2.3.2 DC Motor Operation

In any electric motor, operation is based on simple electromagnetism. A current-carrying conductor generates a magnetic field; when this is then placed in an external magnetic field, it will experience a force proportional to the current in the conductor, and to the strength of the external magnetic field. As you are well aware of from playing with magnets as a kid, opposite (North and South) polarities attract, while like polarities (North and North, South and South) repel. The internal configuration of a DC motor is designed to harness the magnetic interaction between a current-carrying conductor and an external magnetic field to generate rotational motion [12].

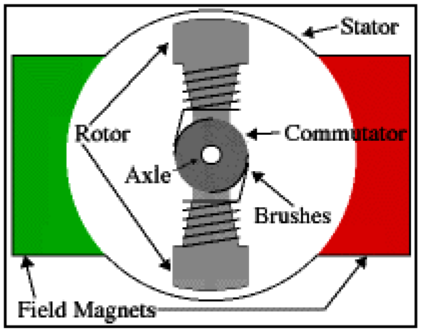


Figure 2.2: Part of DC Motor

Every DC motor has six basic parts axle, rotor (a.k.a., armature), stator, commutator, field magnet(s), and brushes. In most common DC motors (and all that BEAMers will see), the external magnetic field is produced by high-strength permanent magnets [12]. The stator is the stationary part of the motor this includes the motor casing, as well as two or more permanent magnet pole pieces. The rotor (together with the axle and attached commutator) rotates with respect to the stator. The rotor consists of windings (generally on a core), the windings being electrically connected to the commutator. The above diagram shows a common motor layout with the rotor inside the stator (field) magnets [13].

The geometry of the brushes, commutator contacts, and rotor windings are such that when power is applied, the polarities of the energized winding and the stator magnet(s) are misaligned, and the rotor will rotate until it is almost aligned with the stator's field magnets. As the rotor reaches alignment, the brushes move to the next commutator contacts, and energize the next winding. Given our example two-pole motor, the rotation reverses the direction of current through the rotor winding, leading to a "flip" of the rotor's magnetic field, driving it to continue rotating[13].

2.3.3 Advantage of DC Motor

The greatest advantage of DC motors may be speed control. Since speed is directly proportional to armature voltage and inversely proportional to the magnetic flux produced by the poles, adjusting the armature voltage and/or the field current will change the rotor speed [13].

2.4 Sensor

2.4.1 Definition of Sensor and the application

A sensor is a device which measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument. For example, a mercury thermometer converts the measured temperature into expansion and contraction of a liquid which can be read on a calibrated glass tube. A thermocouple converts temperature to an output voltage which can be read by a voltmeter. For accuracy, all sensors need to be calibrated against known standards [14].

Sensors are used in everyday objects such as touch-sensitive elevator buttons and lamps which dim or brighten by touching the base. There are also innumerable applications for sensors of which most people are never aware. Applications include automobiles, machines, aerospace, medicine, industry, and robotics [14].

A sensor's sensitivity indicates how much the sensor's output changes when the measured quantity changes. For instance, if the mercury in a thermometer moves 1cm when the temperature changes by 1°, the sensitivity is 1cm/1°. Sensors that measure very small changes must have very high sensitivities [14].

2.4.2 SHARP GP2Y0A21YK0F

GP2Y0A21YK0F is a distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector), IRED (infrared emitting diode) and signal processing circuit. The variety of the reflectivity of the object, the environmental temperature and the operating duration are not influenced easily to the distance detection because of adopting the triangulation method. This device outputs the voltage corresponding to the detection distance. So this sensor can also be used as a proximity sensor [14].

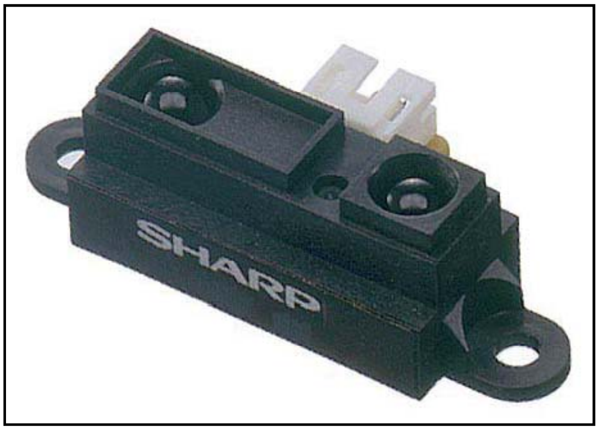


Figure 2.3: Analog distance sensor

The features that available in this sensor type:

i. Distance measuring range : 10 to 80 cm

ii. Analog output type

iii. Package size : 29.5×13×13.5 mm

iv. Consumption current : Typ. 30 mA

v. Supply voltage : 4.5 to 5.5 V

2.5 Relay

A relay is an electrically operated switch. Current flowing through the coil of the relay creates a magnetic field which attracts a lever and changes the switch contacts. The coil current can be on or off so relays have two switch positions and they are double throw (changeover) switches. Relays allow one circuit to switch a second circuit which can be completely separate from the first [15].

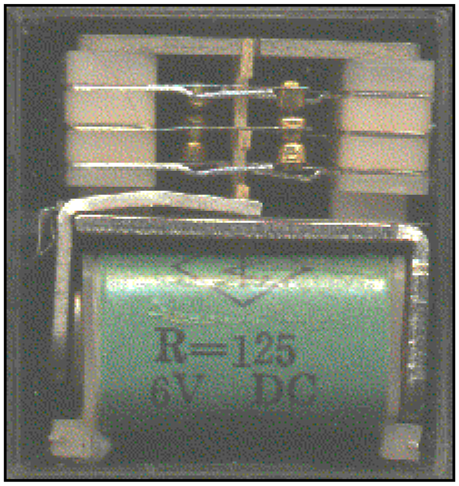


Figure 2.4: Coil and switch contact

Figure 2.5: 6V relays

In this project relay use to supply the voltage to DC motor. 6V relays need more than 6V voltage to energized the contact. The output voltage from PIC is use to energized the contact of relay [15].

2.6 LCD

A liquid crystal display (LCD) is an electro-optical amplitude modulator realized as 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 [16].

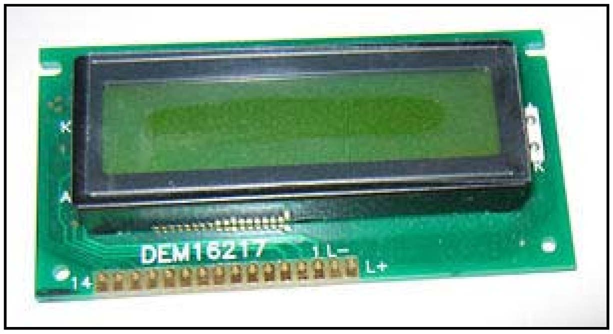


Figure 2.6: Liquid crystal display

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 [16].

2.7 MAX 233

The MAX220–MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where ±12V is not available. These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5µW. The MAX225, MAX233, MAX235, and MAX245/MAX246/MAX247 use no external components and are recommended for applications where printed circuit board space is critical [17].

2.8 Darlington Transistor C1815

Darlington transistor is a semiconductor device which combines two bipolar transistors in a single device so that the current amplified by the first is amplified further by the second. This configuration gives a high current gain (written β, h_{fe}, or h_{FE}) and can take less space than two separate transistors because the two transistors can use a shared collector. Integrated circuit packages are available, but it is still common also to use two separate transistors [18].

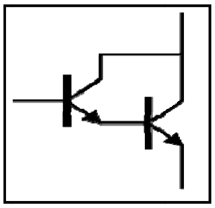


Figure 2.7: Circuit diagram of Darlington transistor

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter will present the whole methodology of this project and details of each part of hardware and software that use in this project. It will describe on how the project is organized and the flow of the steps in order to complete this project. The methodology is diverged in two parts, which is developing the hardware to interface with MATLAB. The other is developing the programming for MATLAB and the PIC to control DC motor. Then, another developing is the programming for MATLAB and PIC to receive the input from sensor and control the motor.

3.2 Software Development

There are three mains method in order to develop this project. Before the project is developing using MATLAB, it is needed to do the study on MATLAB GUIDE and the hardware (especially PIC). The flowchart in Figure 3.1 illustrated the sequence of steps for this project. The first method is developing GUI in MATLAB and programs every GUI component. Secondly is to develop PIC programming to control 5V DC and to receive data from sensor. And lastly is hardware design which is use to interface with MATLAB GUI.

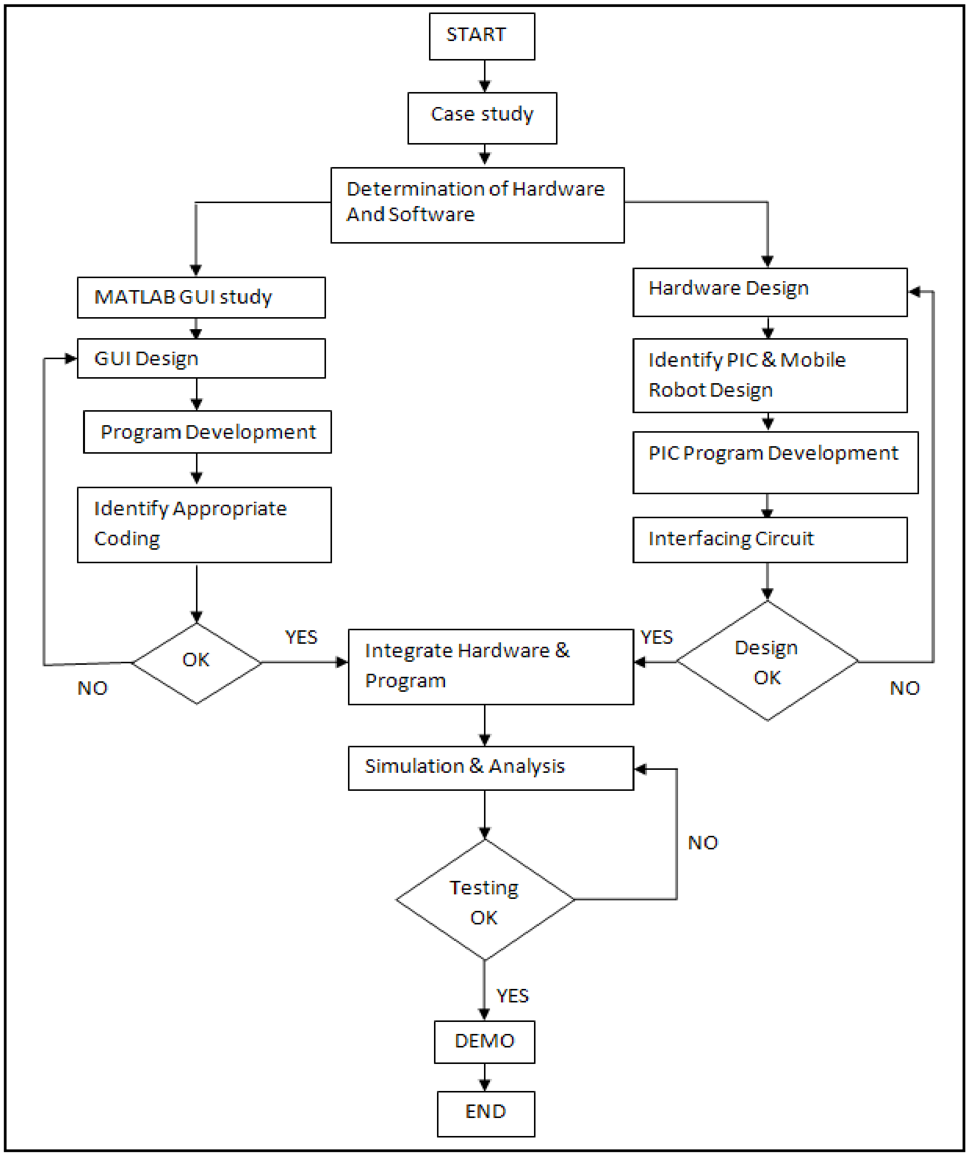


Figure 3.1: Flowchart of the whole project

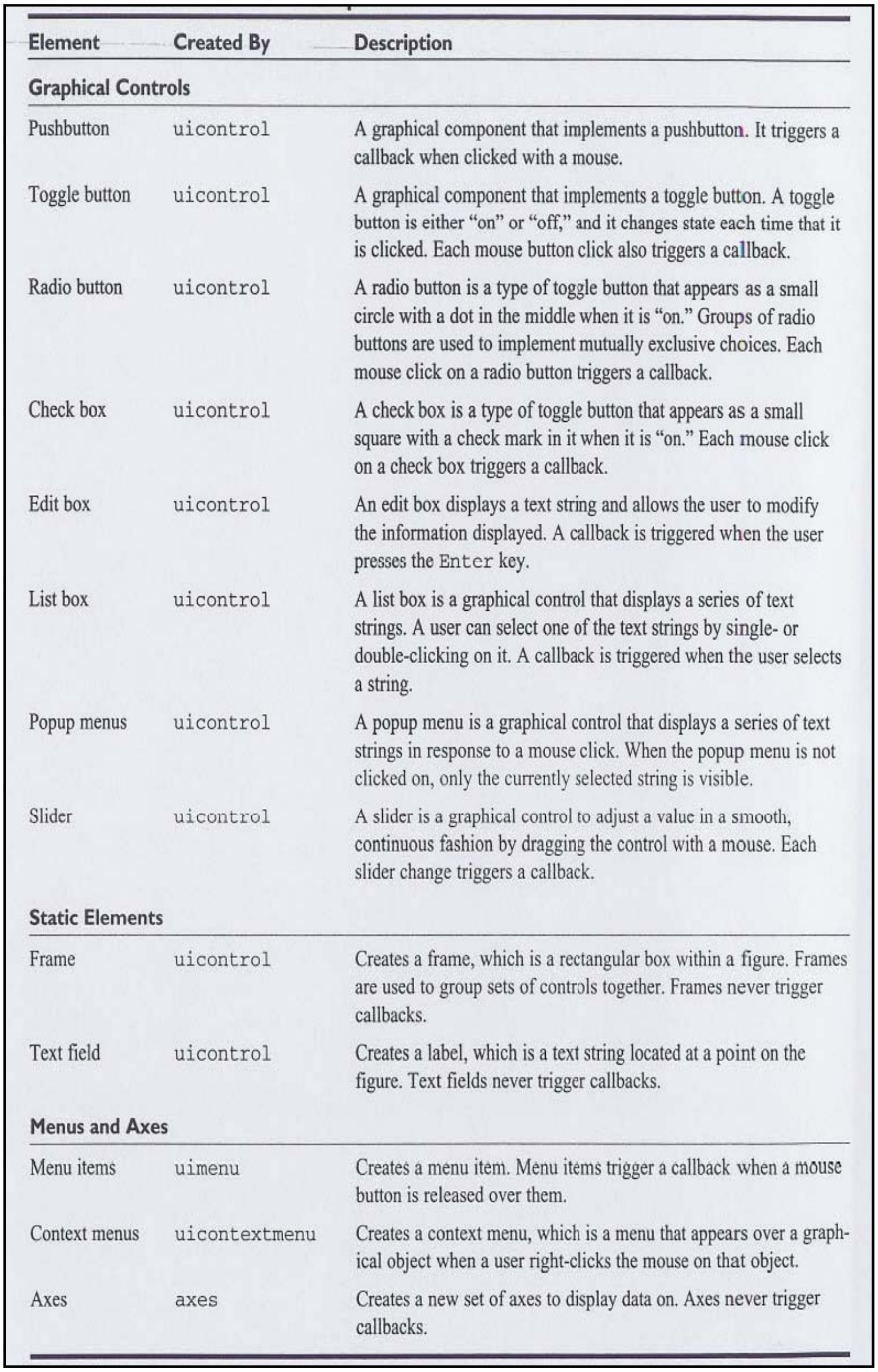
Figure 3.1 show the flow of the whole project. The flow of the project is divided into two main tasks that are hardware design and MATLAB GUI design. In hardware design part flow, the main target is to create appropriate programming for PIC to interface with personal computer via serial port to control DC motor and receive input from sensor. The second part, the prior task is to develop program in MATLAB to interface with PIC and the DC motor and the sensor. After that the both part is combine and do the analysis until achieve the needed objective. The main contribution of this project is to interface MATLAB GUI with the PIC using bidirectional communication.

3.2.1 Development MATLAB GUI Using MATLAB GUIDE

The MATLAB graphical user interface development environment provides a set of tools for creating graphical user interfaces (GUIs). These tools simplify the process of laying out and programming GUIs.

There are 5 steps in build the MATLAB GUI. First Use a MATLAB tool called guide (GUI Development Environment) to layout the components that show in Figure 3.2. This tool allows a programmer to layout the GUI, selecting and aligning the GUI components to be placed in it. The basic component of the MATLAB GUI is shown in Table 3.1.

Table 3.1: Basic MATLAB GUI Component



Element	Created By	Description
Graphical Controls		
Pushbutton	`uicontrol`	A graphical component that implements a pushbutton. It triggers a callback when clicked with a mouse.
Toggle button	`uicontrol`	A graphical component that implements a toggle button. A toggle button is either "on" or "off," and it changes state each time that it is clicked. Each mouse button click also triggers a callback.
Radio button	`uicontrol`	A radio button is a type of toggle button that appears as a small circle with a dot in the middle when it is "on." Groups of radio buttons are used to implement mutually exclusive choices. Each mouse click on a radio button triggers a callback.
Check box	`uicontrol`	A check box is a type of toggle button that appears as a small square with a check mark in it when it is "on." Each mouse click on a check box triggers a callback.
Edit box	`uicontrol`	An edit box displays a text string and allows the user to modify the information displayed. A callback is triggered when the user presses the `Enter` key.
List box	`uicontrol`	A list box is a graphical control that displays a series of text strings. A user can select one of the text strings by single- or double-clicking on it. A callback is triggered when the user selects a string.
Popup menus	`uicontrol`	A popup menu is a graphical control that displays a series of text strings in response to a mouse click. When the popup menu is not clicked on, only the currently selected string is visible.
Slider	`uicontrol`	A slider is a graphical control to adjust a value in a smooth, continuous fashion by dragging the control with a mouse. Each slider change triggers a callback.
Static Elements		
Frame	`uicontrol`	Creates a frame, which is a rectangular box within a figure. Frames are used to group sets of controls together. Frames never trigger callbacks.
Text field	`uicontrol`	Creates a label, which is a text string located at a point on the figure. Text fields never trigger callbacks.
Menus and Axes		
Menu items	`uimenu`	Creates a menu item. Menu items trigger a callback when a mouse button is released over them.
Context menus	`uicontextmenu`	Creates a context menu, which is a menu that appears over a graphical object when a user right-clicks the mouse on that object.
Axes	`axes`	Creates a new set of axes to display data on. Axes never trigger callbacks.

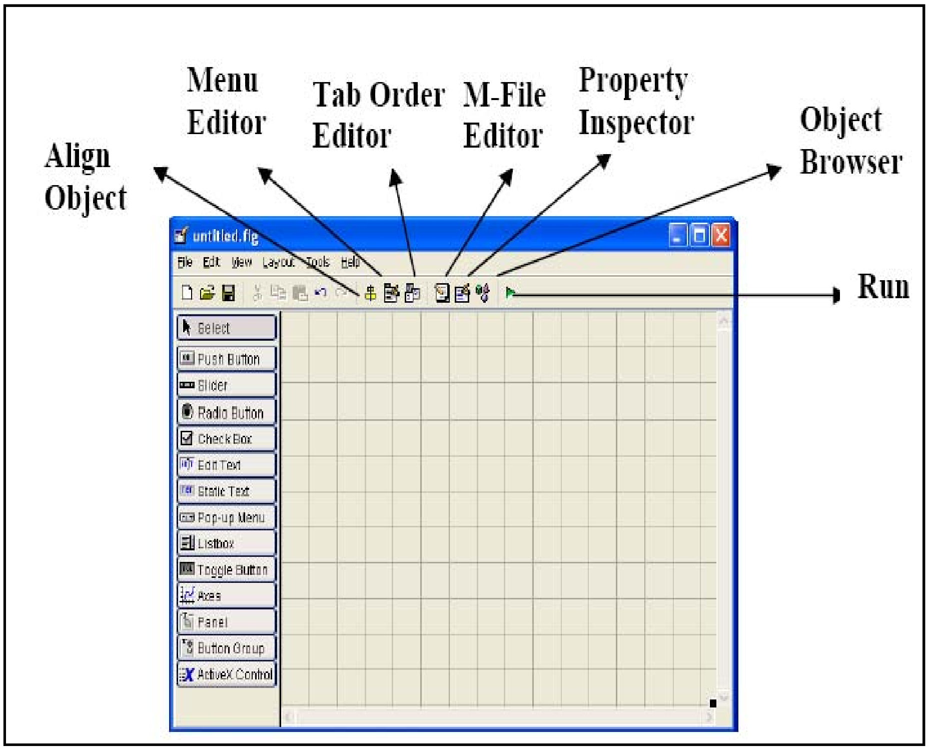
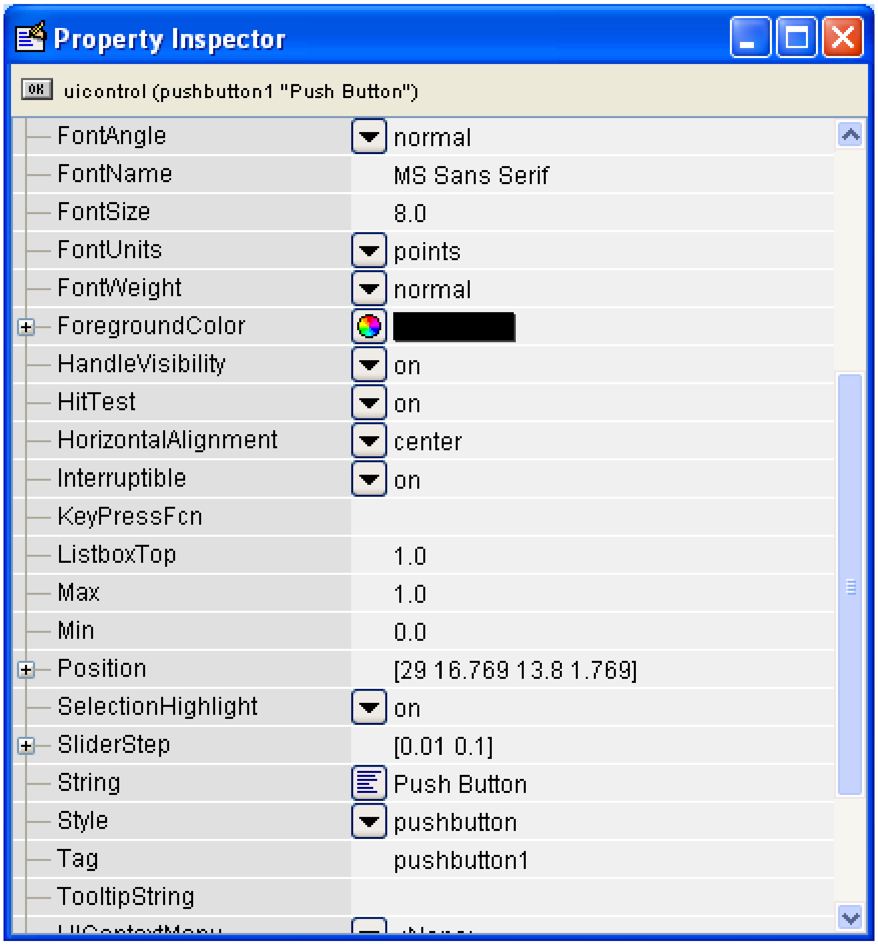


Figure 3.2**:** MATLAB GUIDE Layouts

Use a MATLAB tool called the Property Inspector (built into guide) to give each component a name (a "tag") and to set the characteristics of each component, such as its color, the text it displays, and so on. After that, we save the figure to the file. When the figure is saved, two files will be created on disk with the same name but different extents. The fig file contains the actual GUI that has been created, and the M-file contains the code to load the figure and skeleton call backs for each GUI element. These two files usually reside in the same directory. They correspond to the tasks of laying out and programming the GUI. When you lay out the GUI in the Layout Editor, your work is stored in the FIG-file. When you program the GUI, your work is stored in the corresponding M-file.



uicontrol (pushbutton1 "Push Button")

FontAngle	normal
FontName	MS Sans Serif
FontSize	8.0
FontUnits	points
FontWeight	normal
ForegroundColor	■
HandleVisibility	on
HitTest	on
HorizontalAlignment	center
Interruptible	on
KeyPressFcn	
ListboxTop	1.0
Max	1.0
Min	0.0
Position	[29 16.769 13.8 1.769]
SelectionHighlight	on
SliderStep	[0.01 0.1]
String	Push Button
Style	pushbutton
Tag	pushbutton1
TooltipString	
UIContextMenu	<None>

Figure 3.3: Property Inspector

After laying out the GUI component and set the property, the GUI will be look like in Figure 3.4 for example according to the user creativity.

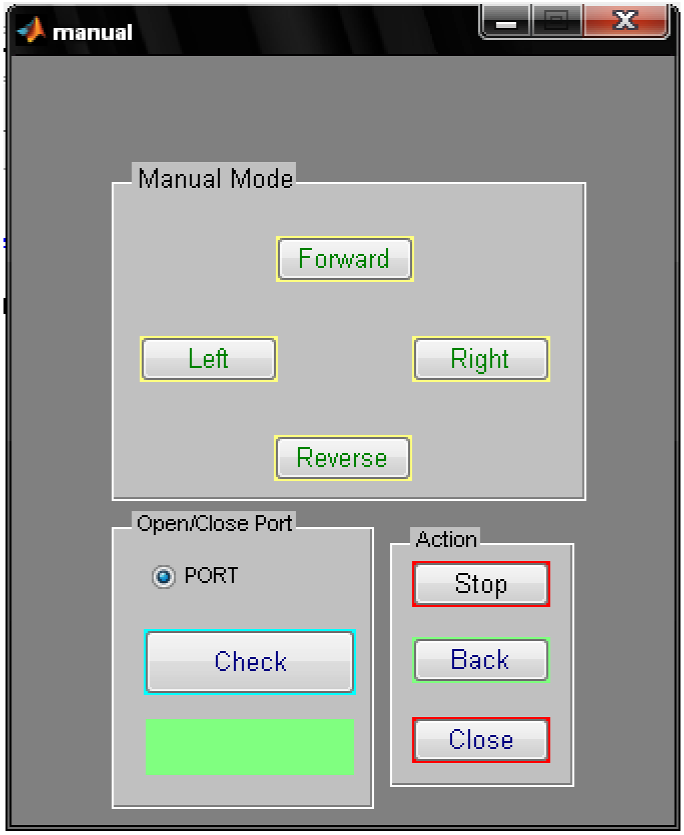
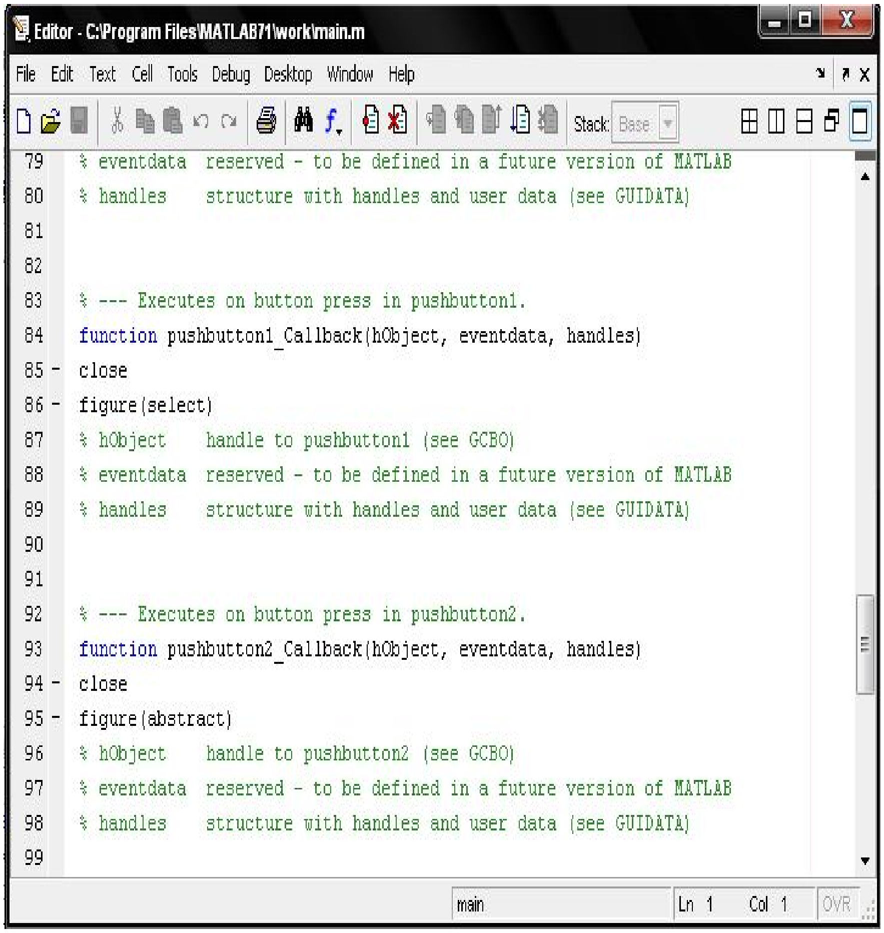


Figure 3.4: Example GUI

And finally write code to implement the behavior associated with each callback function in m-files show in Figure 3.5. A callback is a function that writes and associates with a specific GUI component or with the GUI figure. It controls GUI or component behavior by performing some action in response to an event for its component. This kind of programming is often called event-driven programming. This last step is the difficult one and has to make an extra reading on how to write the coding before the GUI component can perform some task that user desire.


Editor - C:\Program Files\MATLAB71\work\main.m
File Edit Text Cell Tools Debug Desktop Window Help
Stack: Base
main
Ln 1 Col 1
OVR

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
close
figure(select)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
close
figure(abstract)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

Figure 3.5 Example M-files for GUI

3.2.2 Build MATLAB Programming

After layed out the GUI, it need to program its behavior. The code is to write controls how the GUI responds to events such as button clicks, slider movement, menu item selection, or the creation and deletion of components. This programming takes the form of a set of functions, called callbacks, for each component and for the GUI figure itself.

A callback is a function that writes and associates with a specific GUI component or with the GUI figure. It controls GUI or component behavior by performing some action in response to an event for its component. This kind of programming is often called event-driven programming.

The GUI figure and each type of component have specific kinds of callbacks with which it can be associated. The callbacks that are available for each component are defined as properties of that component. Each kind of callback has a triggering mechanism or event that causes it to be called. The kind of callback is shown in Table 3.2.

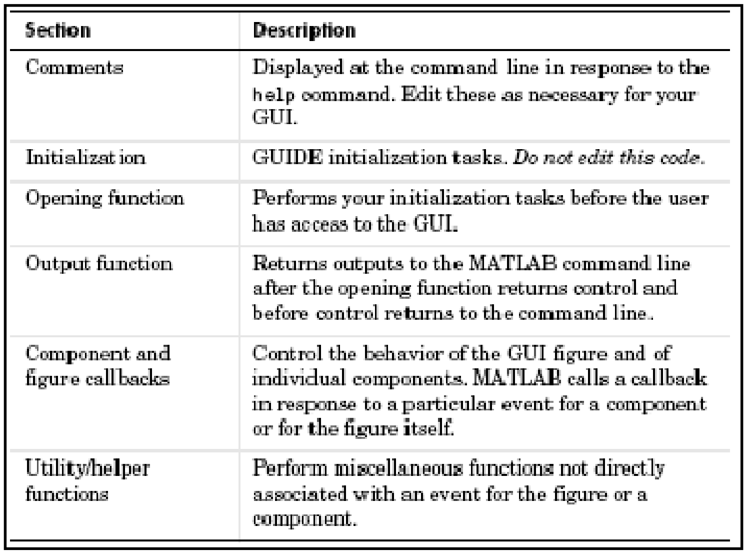
Table 3.2: Various kind of Callback

Callback Property	Triggering Event	Components
DeleteFcn	Component deletion. It can be used to perform cleanup operations just before the component or figure is destroyed.	Axes, figure, button group, context menu, menu, panel, user interface controls
KeyPressFcn	Executes when the user presses a keyboard key and the callback's component or figure has focus.	Figure, user interface controls
ResizeFcn	Executes when a user resizes a panel, button group, or figure whose figure. Resize property is set to On.	Button group, figure, panel
SelectionChangeFcn	Executes when a user selects a different radio button or toggle	Button group

	button in a button group component.	
WindowButtonDownFcn	Executes when you press a mouse button while the pointer is in the figure window.	Figure
WindowButtonMotionFcn	Executes when you move the pointer within the figure window.	Figure
WindowButtonUpFcn	Executes when you release a mouse button.	Figure
ButtonDownFcn	Executes when the user presses a mouse button while the pointer is on or within five pixels of a component or figure. If the component is a user interface control, its Enable property must be on.	Axes, figure, button group, panel, user interface controls
Callback	Component action. Executes, for example, when a user clicks a push button or selects a menu item.	Context menu, menu, user interface controls
CloseRequestFcn	Executes before the figure closes.	Figure
CreateFcn	Component creation. It can be use to initialize the component when it is created. It executes after the component or figure is created, but before it is displayed.	Axes, figure, button group, context menu, menu, panel, user interface controls

The GUI M-file that GUIDE generates is a function file. The name of the main function is the same as the name of the M-file. For example, if the name of the M-file is auto.m, then the name of the main function is auto. Each callback in the file is a sub function of the main function. When GUIDE generates an M-file, it automatically includes templates for the most commonly used callbacks for each component. The major sections of the GUI M-file are ordered as shown in Table 3.3.

Table 3.3: Major Sections of the GUI M-file



Section	Description
Comments	Displayed at the command line in response to the help command. Edit these as necessary for your GUI.
Initializat ion	GUIDE initialization tasks. *Do not edit this code.*
Opening function	Performs your initialization tasks before the user has access to the GUI.
Output function	Returns outputs to the MATLAB command line after the opening function returns control and before control returns to the command line.
Component and figure callbacks	Control the behavior of the GUI figure and of individual components. MATLAB calls a callback in response to a particular event for a component or for the figure itself.
Utility/helper functions	Perform miscellaneous functions not directly associated with an event for the figure or a component.

GUIDE automatically includes two callbacks, the opening function and the output function, in every GUI M-file it creates. The opening function programming is importance in initialize the communication port in MATLAB GUI before it can transmit data to the PIC. The data send from MATLAB GUI to PIC is in decimal form and PIC will control the DC motor with the preset programming according to the data received.

In opening and closing the communication port the command fclose (SerPIC) is use to disconnect a serial port object from the device. The baud rate from

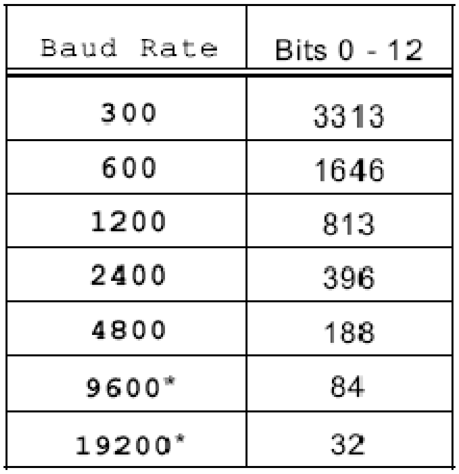
MATLAB GUI must be set same with the baud rate in PIC before it can transmit and receive the data. For example if baud rate in MATLAB GUI is 9600bps, so the baud rate in PIC also 9600bps.

3.2.3 Build PIC programming

The data from MATLAB GUI is send to PIC in decimal form, so the PIC is program to read or receive the data also in decimal form. The communication between MATLAB GUI and PIC is in standard asynchronous format where the device uses its own internal clock resulting in bytes that are transferred at arbitrary times. The baud rate is specifying according to MATLAB GUI. Some standard baud rates are listed in Table 3.4. For PIC programming, 9600bps is using which same with the MATLAB GUI.

The input data at PIC that transmit from MATLAB GUI is set to PORTC.0 before it run certain program to control the DC motor. Here is the example to program the stepper motor run in clockwise and anticlockwise direction. If MATLAB send data ‘001’, so the PIC will perform case 001 according the programming.

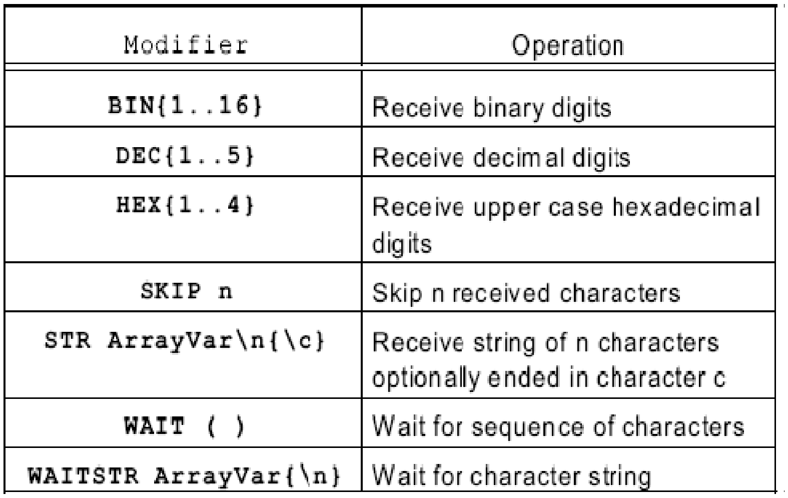
Table 3.4: List of Standard Baud Rate



Baud Rate	Bits 0 - 12
300	3313
600	1646
1200	813
2400	396
4800	188
9600*	84
19200*	32

To program the PIC, make sure the oscillator that defines in programming is same as use at hardware to avoid instability during transmit and receive data. The SERIN2 command in the program support many different data modifier which may be mixed and matches freely within single SERIN2 statement to provide various input formatting. The modifier support is shown in Table 3.5. The number 84 on "Serin2 SerI, 84, [dec3 B0]" command is refer to baud rate that equal to 9600bps according Table 3.4.

Table 3.5: Modifier Support by SERIN2 Command



Modifier	Operation
BIN{1..16}	Receive binary digits
DEC{1..5}	Receive decimal digits
HEX{1..4}	Receive upper case hexadecimal digits
SKIP n	Skip n received characters
STR ArrayVar\n{\c}	Receive string of n characters optionally ended in character c
WAIT ( )	Wait for sequence of characters
WAITSTR ArrayVar{\n}	Wait for character string

3.3 Hardware Development

For hardware design, first is to design the power supply module which is to supply 5V fixed to PIC and max232 IC. Power supply module is importance to PIC and max232 to prevent damage if users give the higher input supply to device. The schematic diagram for power supply module is like in Figure 3.6. Input to the power supply must greater than 7V to 7805 voltage regulator IC to achieve the 5V output supply to PIC and max233.

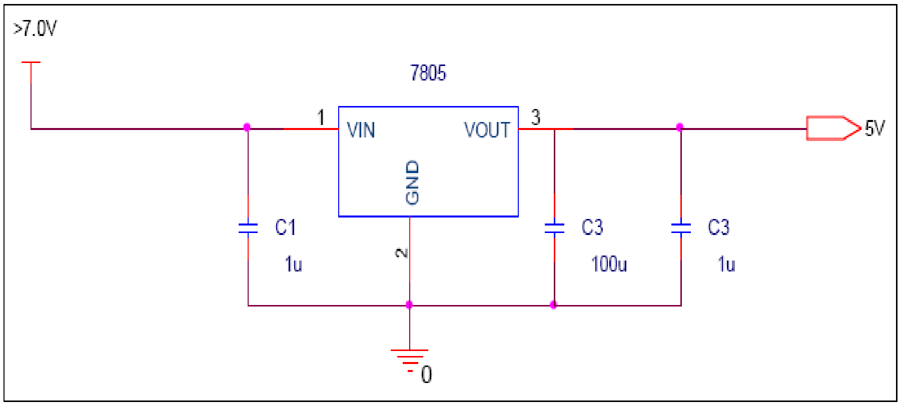


Figure 3.6: Power Supply Modules

Second is to design the connection from communication port (DB9 female connection) from computer to the device which is the pin assignment is shown in Table 3.6 below and the figure of RS 232 communication port shown on Figure 3.7. In fact, only three pins are required for serial port communications: one for receiving data, one for transmitting data, and one for the signal ground. The connection from computer to device is only on pin 2, 3 and pin 5. The circuit in Figure 3.8 shows the connection between RS232 with MAX232 and the PIC.

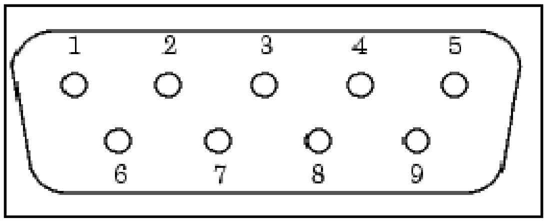
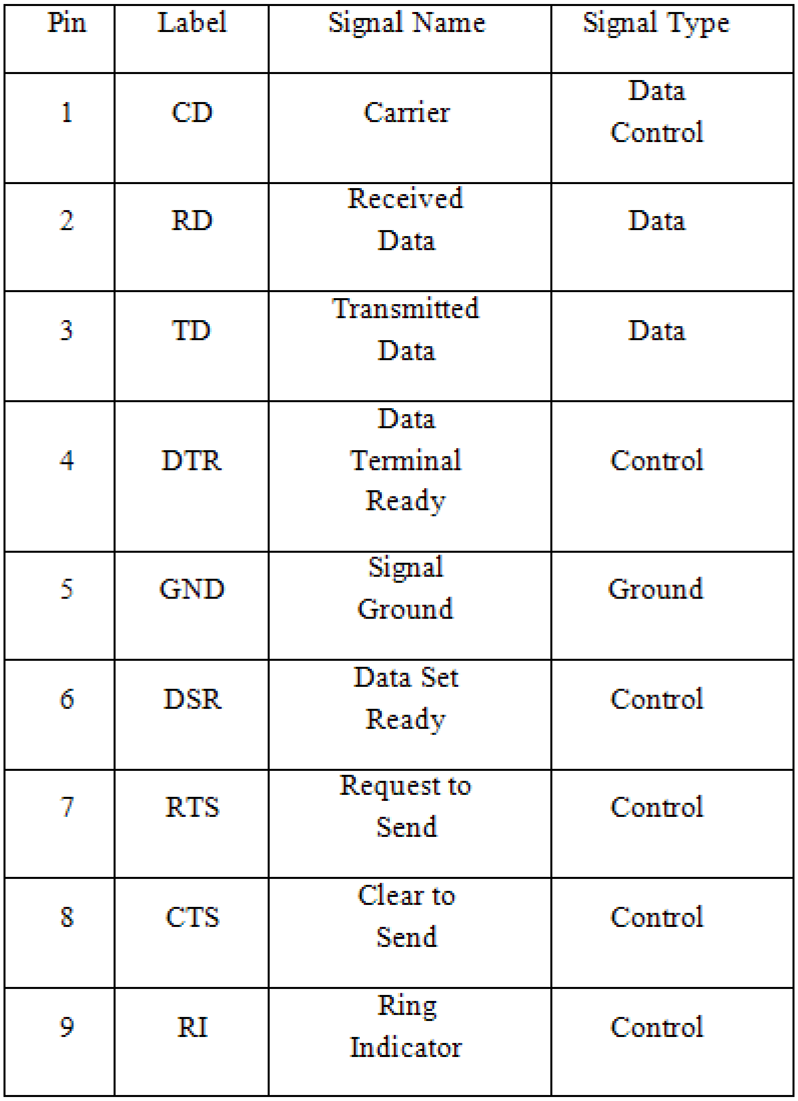


Figure 3.7: Pins and Signals Associated With the 9-pin Connector

Table 3.6: Serial Port Pin and Signal Assignments



Pin	Label	Signal Name	Signal Type
1	CD	Carrier	Data Control
2	RD	Received Data	Data
3	TD	Transmitted Data	Data
4	DTR	Data Terminal Ready	Control
5	GND	Signal Ground	Ground
6	DSR	Data Set Ready	Control
7	RTS	Request to Send	Control
8	CTS	Clear to Send	Control
9	RI	Ring Indicator	Control

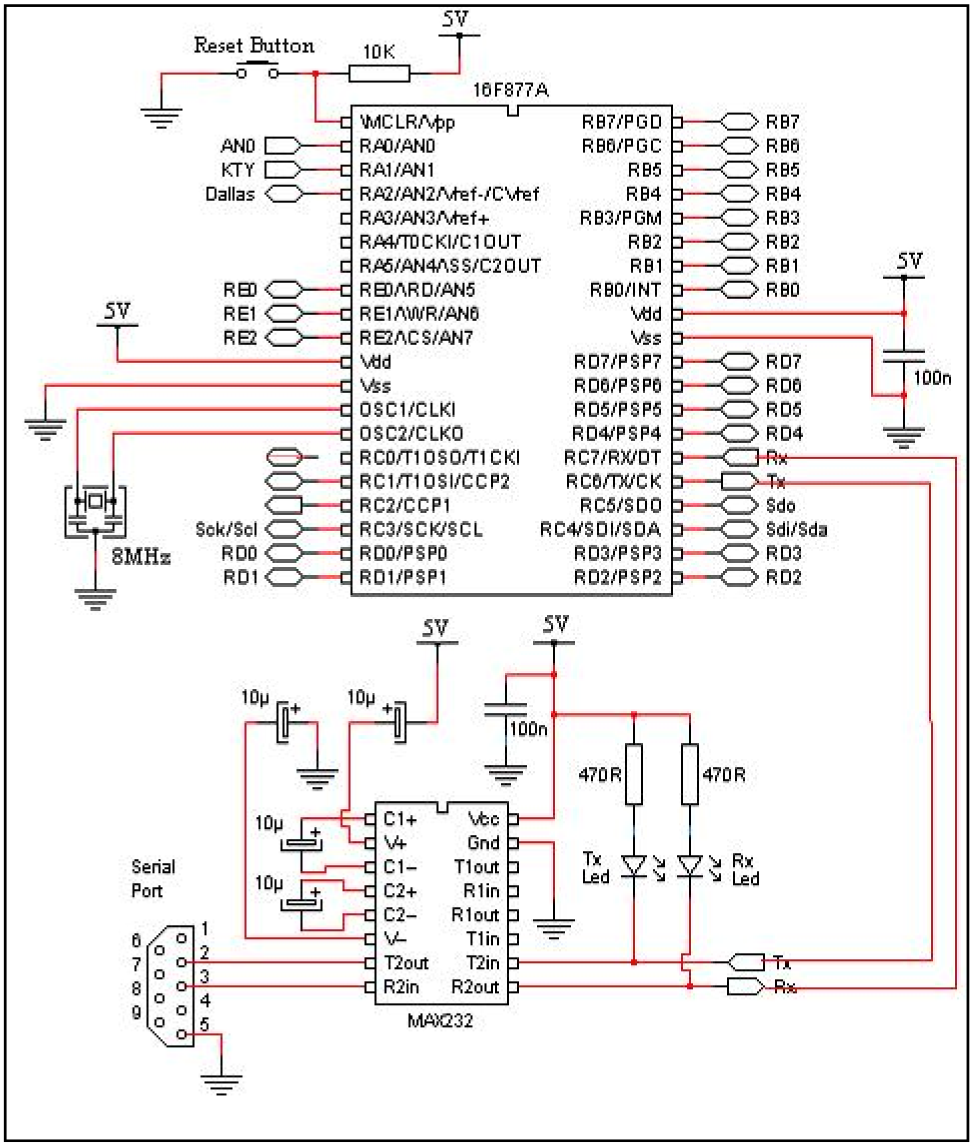


Figure 3.8: Serial Port Connection to PIC

In this project the output data from MAX233 is send directly to PIC at PORTC.7 and the sensor is connect to PIC at PORTA.0. This connection is depending on the PIC programming that has been developing before it can perform specific task according the data send from the MATLAB GUI. The oscillator use in the circuit diagram also same with the define one in the PIC programming to avoid instability.

The output on the PIC port is approximately 4.7 V low current which is cannot run the stepper motor or DC motor directly. So, to run the motor, switching approach is use by using additional source with high current supply. To done this method the Darlington transistor (C1815) is use like the circuit in Figure 3.9. To run the DC motor in forward or reverse direction it has to use relay because it cannot directly control via PIC. Relays use to give direct current to motor. In this project, the PORTB.0 to PORTB.3 will be use to control 5V DC motor like in Figure 3.9. The hardware installation for this project is shown in Figure 3.10.

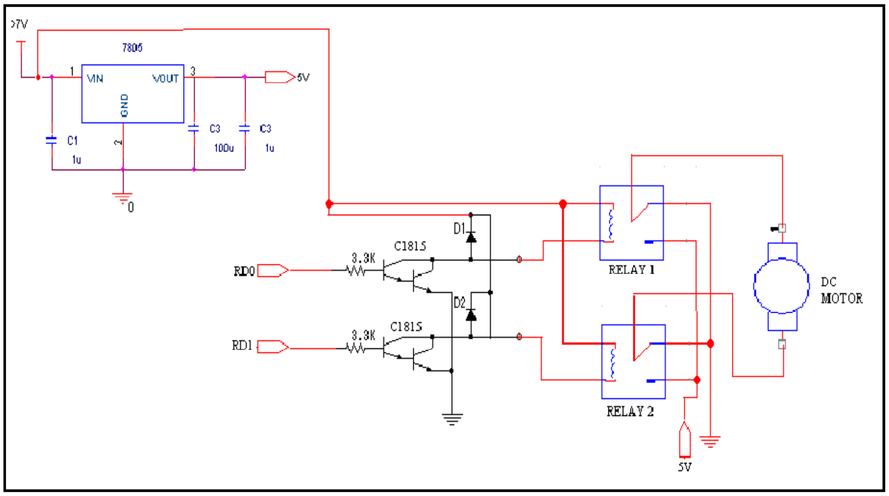


Figure 3.9: 5V DC Motor Connection

The input from sensor is connecting to PIC at PORTA.0. In this project, we use analog distance sensor that can sense a range between 8cm – 80cm. Analog distance sensor will give single output voltage. The input from sensor will convert from analog to digital form. We write the program that will convert analog input to digital. After the conversion, the digital output produce will display on LCD. The program to convert from analog to digital is in program 1. Before the character display at LCD, we must initialize LCD first. The program to initialize LCD is in program 2.

Program 1:

```
   ADCON1=4                 ' Set PortA 0, 1, 3 to A/D inputs
  Pauseus 50               ' Wait for channel to setup
  ADCON0.2 = 1              ' Start conversion
  Pauseus 50               ' Wait for conversion
  Return

' Subroutine to get pot x value
getx:
  ADCON0 = $41               ' Set A/D to Fosc/8, Channel 0, On
  Gosub getad
  x = ADRESH
  Return

mainloop:
  Gosub getx                          ' Get x value
  Lcdout $fe, 1, "Distance=", #x      ' Send to LCD

  if x>=100 then balik
  IF X<=10 THEN balik
  Goto mainloop                       ' Do it forever

balik:
  portb=%00000000
  Lcdout $fe, 1, "MOTOR STOP"       ' Send to LCD
  pause 1000
  goto start
```

Program 2:

```
Define LCD_DREG             PORTD          'initialize lcd
Define LCD_DBIT             4
Define LCD_RSREG PORTE
Define LCD_RSBIT   0
Define LCD_EREG             PORTE
Define LCD_EBIT             1

Low PORTE.2            ' LCD R/W line low (W)
Pause 500             ' Wait for LCD to start
```

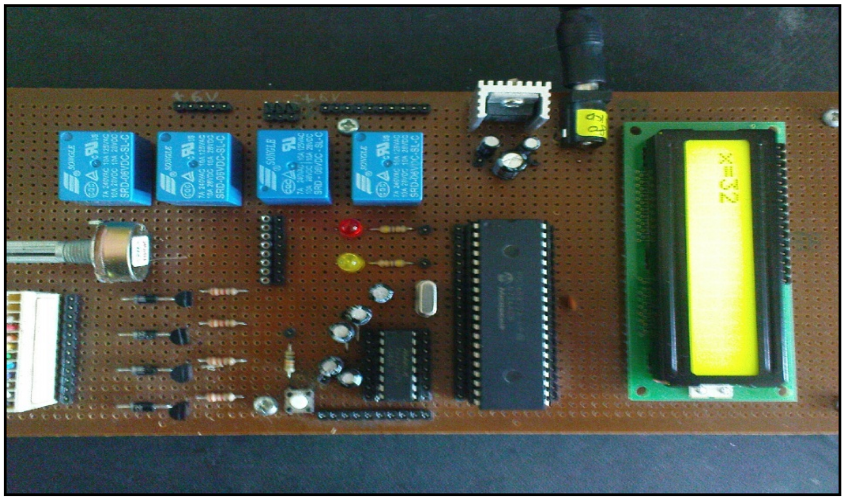


Figure 3.10: Hardware circuit (main)

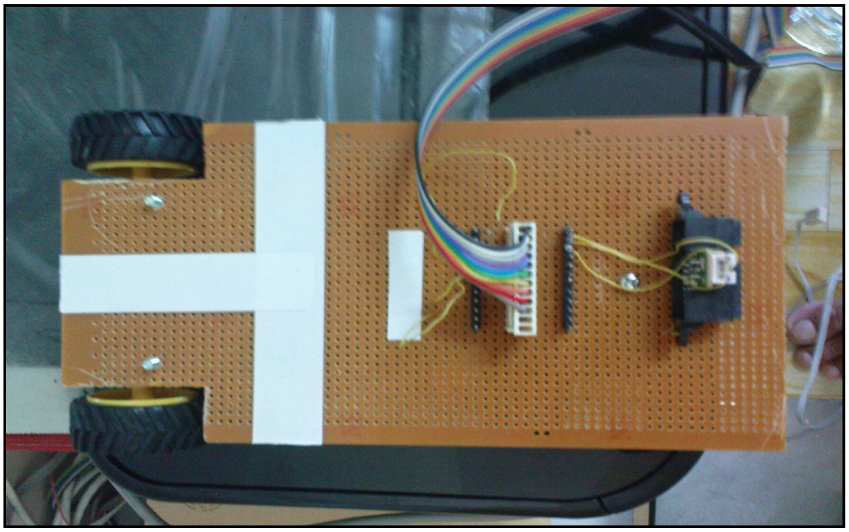


Figure 3.11: Mobile Robot

CHAPTER 4

RESULT AND DISCUSSION

4.1 Introduction

This chapter consists of the discussions on the results from the MATLAB GUI layout that has been developing using MATLAB Graphical User Interface Development Environment. The MATLAB GUI in this project can be divided to four parts. First part is main menu of the whole GUI. Second part is interfacing MATLAB GUI software.

4.2 Main Menu of GUI

The main menu of the GUI in this project contain of four pushbutton which link to motor control selection, general info about the project abstract, the credit and lastly is an exit button to close the whole program. The main menu of the GUI and info of the project is shown in Figure 4.1. For motor control pushbutton will explain detail in the next sub chapter. In credit part shown in Figure 4.2 contains the detail about the GUI developer and the supervisor. For the introduction to motor control will show in abstract. This abstract window is shown in Figure 4.3. For the exit button user will ask about the confirmation either to exit the GUI or not. The confirmation figure is shown in Figure 4.4.

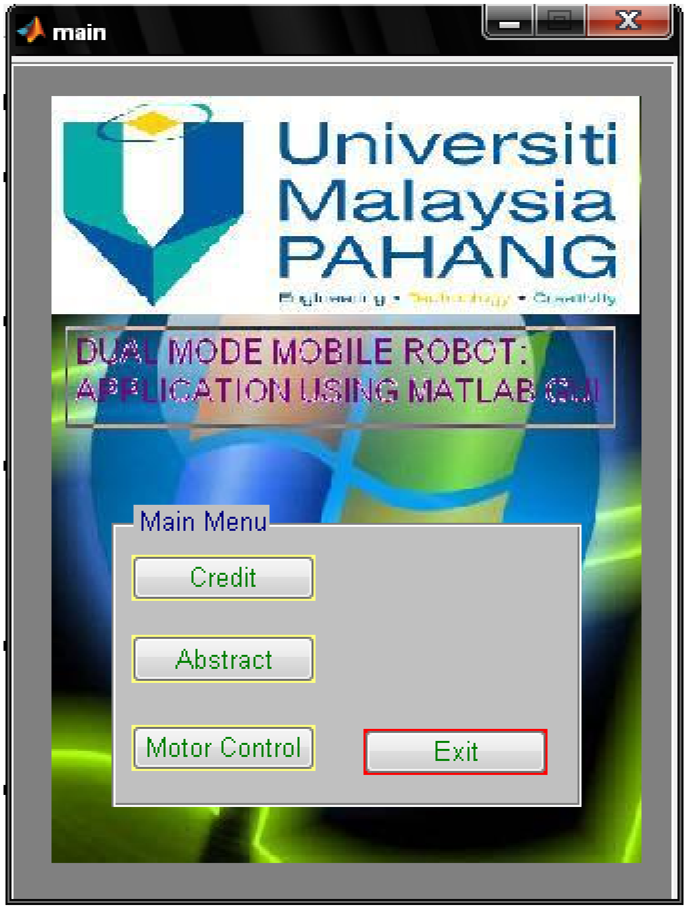
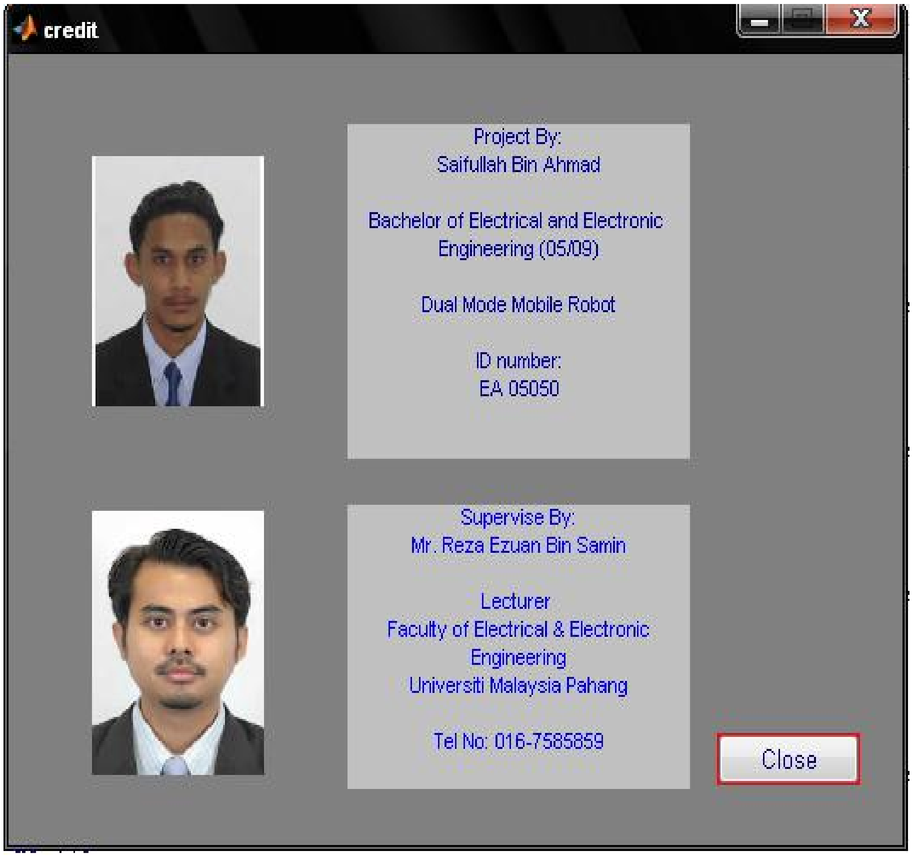


Figure 4.1: Main Menu

The main menu of GUI window consist four push buttons. The first push button is credit, abstract motor control and lastly exit button to close the main menu of GUI. When we push exit button it will close the window and pop-up window will display to do the action or not. The confirmation window is shown in Figure 4.5.



Project By:
Saifullah Bin Ahmad

Bachelor of Electrical and Electronic Engineering (05/09)

Dual Mode Mobile Robot

ID number:
EA 05050

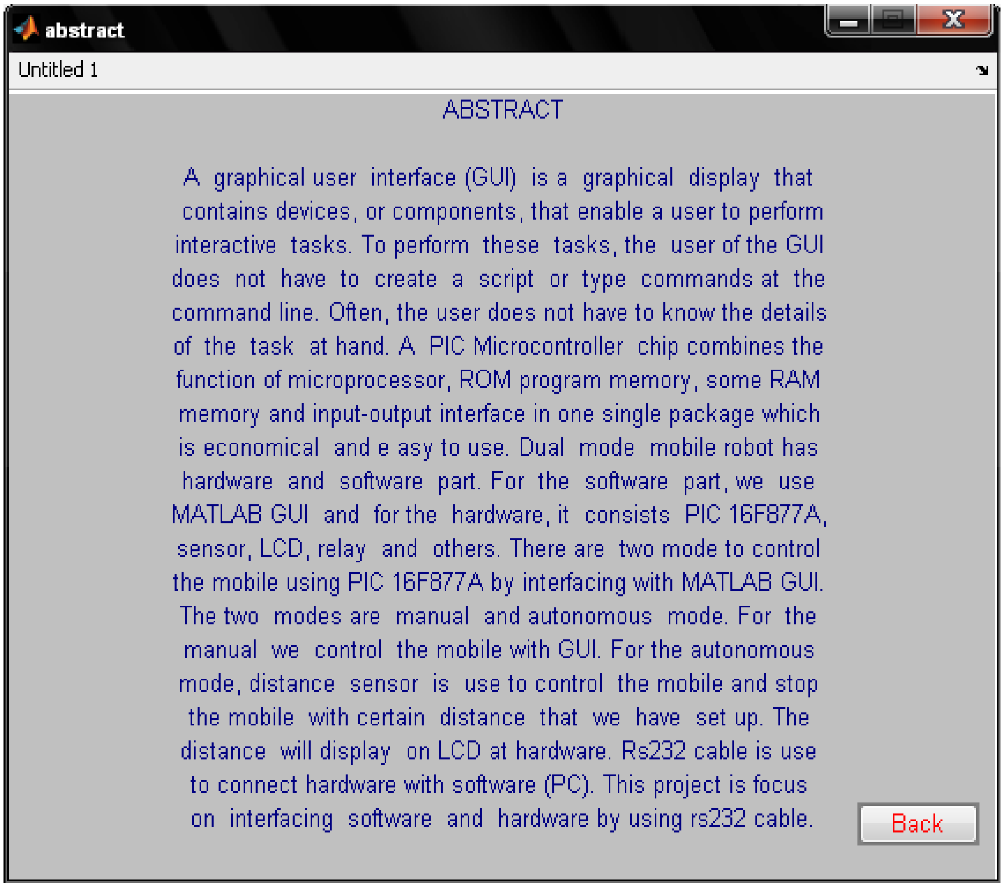
Supervise By:
Mr. Reza Ezuan Bin Samin

Lecturer
Faculty of Electrical & Electronic Engineering
Universiti Malaysia Pahang

Tel No: 016-7585859

Figure 4.2: Credit

The credit window will display when we push the credit button on the main menu. The credit will show GUI developer and the supervisor of the project. When we push the close button, it will display back main menu window.

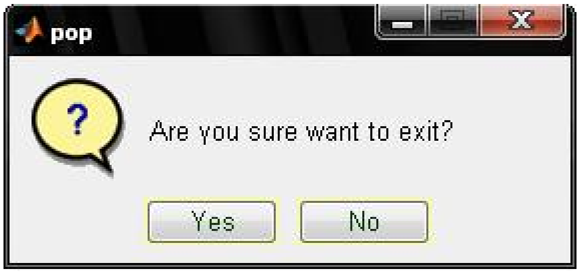


ABSTRACT

A graphical user interface (GUI) is a graphical display that contains devices, or components, that enable a user to perform interactive tasks. To perform these tasks, the user of the GUI does not have to create a script or type commands at the command line. Often, the user does not have to know the details of the task at hand. A PIC Microcontroller chip combines the function of microprocessor, ROM program memory, some RAM memory and input-output interface in one single package which is economical and e asy to use. Dual mode mobile robot has hardware and software part. For the software part, we use MATLAB GUI and for the hardware, it consists PIC 16F877A, sensor, LCD, relay and others. There are two mode to control the mobile using PIC 16F877A by interfacing with MATLAB GUI. The two modes are manual and autonomous mode. For the manual we control the mobile with GUI. For the autonomous mode, distance sensor is use to control the mobile and stop the mobile with certain distance that we have set up. The distance will display on LCD at hardware. Rs232 cable is use to connect hardware with software (PC). This project is focus on interfacing software and hardware by using rs232 cable.

Figure 4.3: Abstract

The abstract window will shown the description of the whole project. The back button is use to back to main menu.



Are you sure want to exit?

Figure 4.4: Pop-up Window

The pop-up window will show when we push exit button. When we push 'yes' button, it will exit the program. If we push 'no' button, it will back to main menu.

4.3 Interface MATLAB GUI Software

For motor control part, it divides into two parts where the first part is interfacing software and the second part is advance GUI development for future. The first part of the motor control GUI is the main objective of this project where to interface between MATLAB GUI with the device (motor) to control the motor. The figure of motor control menu is shown in Figure 4.5. The interface software is developing only for 5V DC motor. The rest is for advance development. In the menu motor control menu user can choose for manual mode or autonomous mode. For manual mode motor control is shown in Figure 4.6. For the autonomous mode motor control is shown in Figure 4.8.

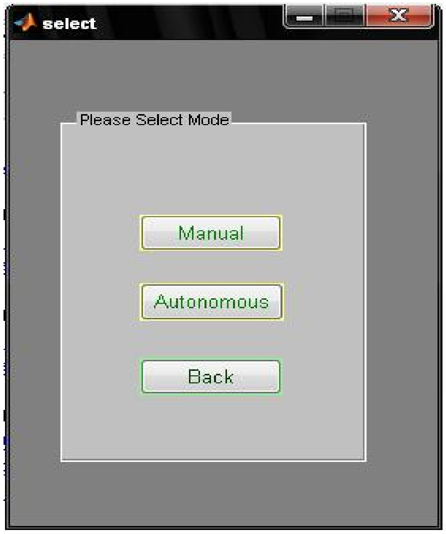


Figure 4.5: Mode Selection

Figure 4.5 shown will the menu selection of manual mode and autonomous mode. At this window, we can choose from two types of mode to control mobile robot. The button back is use for return to main menu.

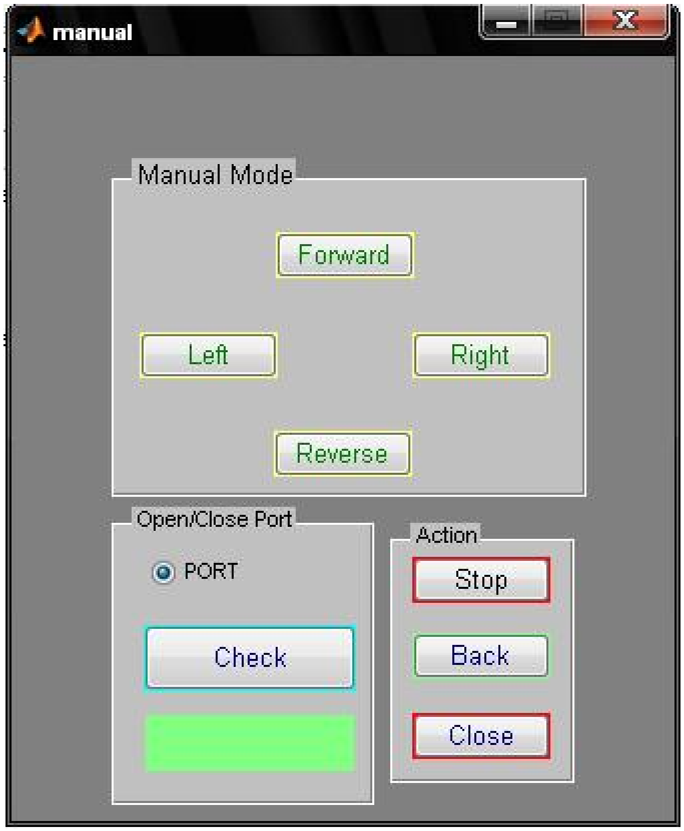


Figure 4.6: Manual Mode Motor Control

In manual mode window it has 3 main boxes. For manual mode box it consist four buttons that use to control the motor. Each button has assigned to control the mobile robot to move forward, reverse, left and right. In this window, we will be able to check the port in open or close condition. Finally it has the stop button and close button to exit the system.

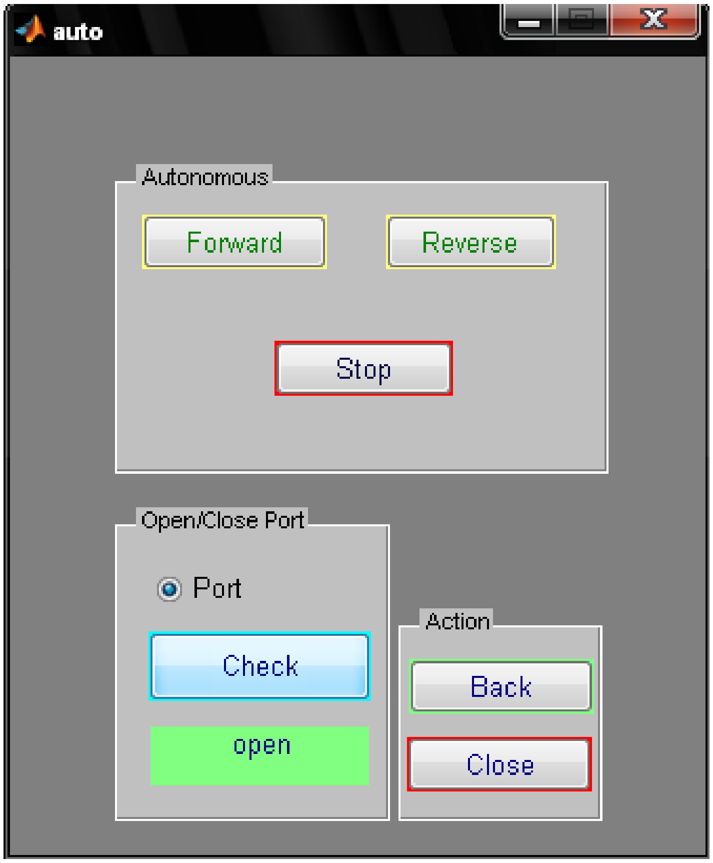


Figure 4.7: Autonomous Mode Motor Control

For autonomous mode, when we push start button the mobile robot will move forward. When the sensor detects the block, it will display the distance between robot and block. The distance will display at LCD at hardware board. Before we push start button, we must first check the port by clicking the button check. The communication port status is shown in Figure 4.8. If errors happen, user must restart the MATLAB and run the GUI back.

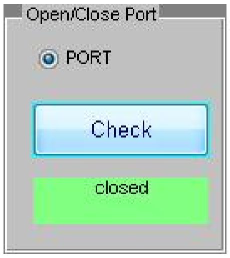


Figure 4.8: Communication Port Status

The communication port status is use to check the port connection opens or close. If the communication port status is close, we cannot use the entire button in the GUI window because the data from GUI programming are no sent to serial port. The push button is able to use when communication port status is open.

4.4 User information GUI

This part provides the user manual as guidance to use this MATLAB GUI software. The manual is important for the first time user to get the information on how to operate the GUI in right way. The user can get the information on how to setting the port when we click on radio button, because if this software use in different computer, the communication port configuration also differ. So the GUI software cannot control the motor or in other word the interface between MATLAB GUI and device is failed. The data is not sending to the PIC. The available port will display at MATLAB window, so we should setting back the programming.

4.5 Result

4.5.1 DC motor

DC motor is control by programming the PIC that will control the motor by clicking at push button on MATLAB GUI windows. The mobile robot is control using DC motor. All the movement of mobile robot is control trough GUI window.

4.5.2 Sensor (GP2Y0A21YK0F)

GP2Y0A21YK0F is a distance measuring sensor unit, composed of an integrated combination of PSD (position sensitive detector). IRED (infrared emitting diode) and signal processing circuit. The variety of the reflectivity of the object, the environmental temperature and the operating duration are not influenced easily to the distance detection because of adopting the triangulation method. This device outputs the voltage corresponding to the detection distance. So this sensor can also be used as a proximity sensor.

From the observation from distance sensor result, we have record some of characteristic of distance in such of to sense an object. Table 4.1 will show the accuracy of analog distance sensor to sense object.

Table 4.1: Observation from sensor

Paper type	Reflectance ratio
White paper	90%
Gray paper	18%

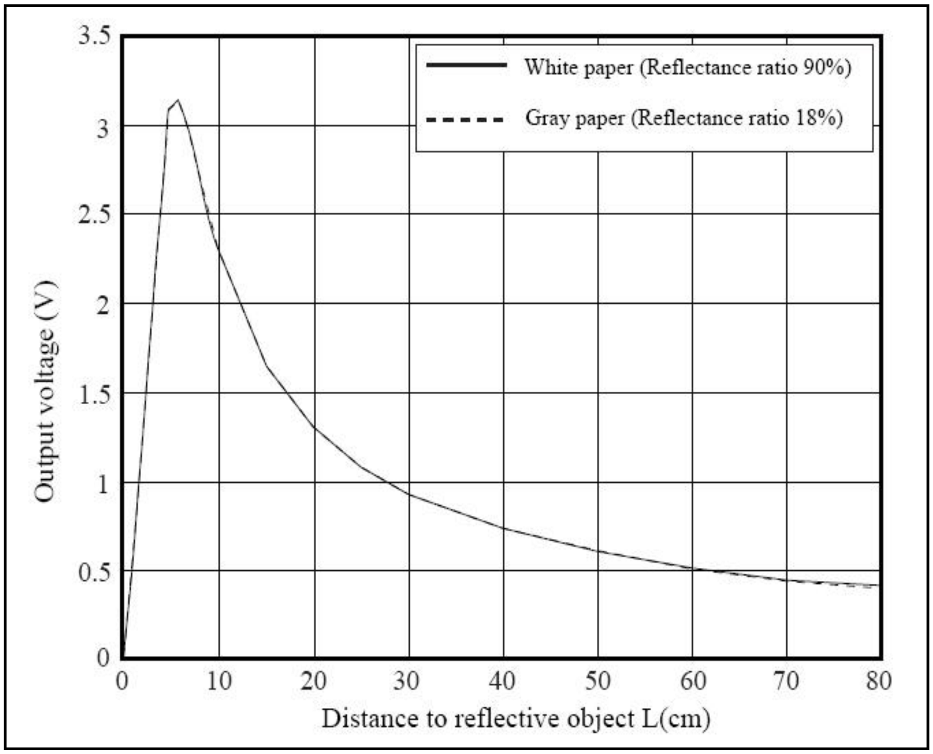


Figure 4.9: Distance versus Output Voltage of the Sensor

4.5.3 LCD Display

LCD module is use to display the distance measure from analog distance sensor. It also will display the mode and the status of motor.

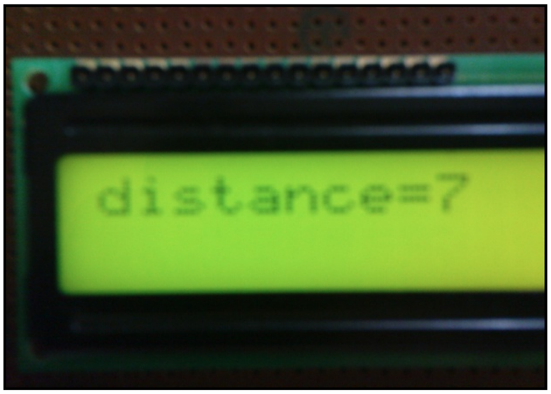


Figure 4.10: Mobile Robot Distance

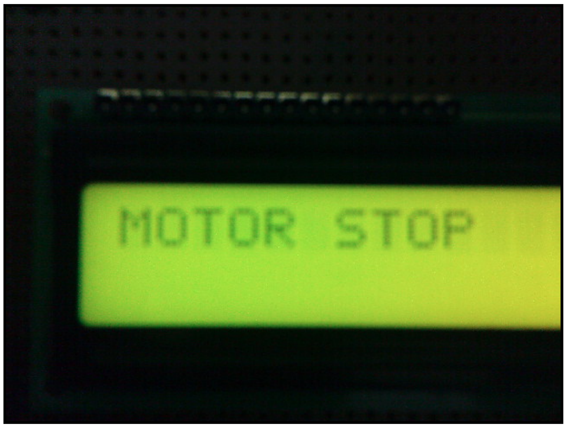


Figure 4.11: Motor Status

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

The GUI design and it implementation has fully develop to achieve the entire objectives. The mobile robot is able control using GUI via serial port communication. The development of the MATLAB GUI using MATLAB GUIDE and PIC16F877A programming using Microcode Studio was done after detail study and analysis. Through the development of this project it has conclude that the MATLAB GUI can control the motor and interface with the device using serial communication port with the proper hardware installation and a lot of knowledge to programmed the PIC.

The main objectives of this project are to control DC motor and interface the MATLAB GUI is fully achieved. The most important part in this project is interfacing MATLAB GUI with PIC 16F877A.

5.2 Future Recommendation

For the futures recommendation, to improve this project, others features in GUI is added and use as the main part of the project like slider to control the speed of motor. By using this slider, the accuracy of analog distance sensor to sense object is more precise and accurate. The input from sensor will send through serial port and display at GUI window.

For other recommendation of this project is to use two mobile robot for two mode we have. From GUI window, we can choose one of the two modes we have. To make more looked interested in this project; we use wireless systems to replace rs232 cable. By using wireless system, the mobile robot is able to control in wide range.

5.3 Costing and Commercialization

The cost of the project is divided into two parts. First part is for components that we get from laboratory and second part is for components we get from other sources. For components from laboratory, it will cost approximately RM 95. For components from other sources, it cost approximately RM 175. The total cost for hardware cost is approximately RM275. For software part, it will cost more on to get the license from MATLAB and usually the cost is high where the license must be renew by year.

This project can be used in industries for pick and place operation to replace SCARA robot that perfume same task. The operation system of this is project more effective because it has software part. This project cost more cheaply than SCARA robot and friendly users. This project approach of imparting advanced GUI capability to microcontrollers using MATLAB can be used to develop microcontroller-based low-cost control platforms. In addition, this approach can be used to impart GUI capability to any microcontroller that supports serial communication, such as the PIC series microcontrollers.

REFERENCES

1. UsersBrian R. Hunt, Ronald L. Lipsman, Jonathan Rosenberg, Kevin R. Coombes, John E. Osborn, Garrett J. Stuck. *A Guide to MATLAB: For Beginners and Experienced.* Published by Cambridge University Press. 2006.

2. MATLAB, high-performance numeric computation and visualization software: building a graphical user interface. *By MathWorks, Inc, Inc MathWorks.* Available at: http://www.mathworks.com

3. Chapman, Stephen J. *MATLAB Programming for Engineers*. Brooks Cole. 2001.

4. Scott T. Smith. *MATLAB Advanced GUI Development*: *Advanced GUI Development*. 2006.

5. Robert DeMoyer and E. Eugene Mitchell. *Use of the MATLAB Graphical User Interface Development Environment for Some Control System Applications*. 1999.

6. Marc E. Herniter. *Programming In MATLAB*. Northern Arizona.
University, Brooks/Cole. 2001.

7. Yan-Fang Li, Saul Harari, Hong Wong, and Vikram Kapila.
Matlab-Based Graphical User Interface Development for Basic Stamp 2. 2004.

8. Dan O'Sullivan; Tom Igoe. *Physical Computing: Sensing and Controlling the Physical World with Computers*. 2004.

9. Chuck Hellebuyck. *Programming PIC Microcontrollers with PICBASIC*.

10. PicBasic Pro Compiler. *MicroEngineering Labs, Inc*. 2004.
Available at: http://www.melabs.com

11. Gerald A. Moberg. *AC and DC Motor Control*. Published by Prentice Hall PTR. 1987.

12. Lab-Volt (Quebec) Ltd, Lab-Volt (Quebec) Ltd. Contributor Lab-Volt (Quebec) Ltd Staff. *DC Motor Drive*. Published by Lab-Volt. 2006.

13. Christopher Edwards and Sarah K. Spurgeon. *Sliding Mode Control: Theory and Applications*. Published by CRC Press. 1998.

14. David J. Perdue. *Unofficial LEGO NXT Inventor's Guide*. 2007.

15. Jonathan Chin. *Cisco Frame Relay Solutions Guide*. March 2004.

16. John Iovine. *PIC Robotics: A Beginner's Guide to Robotics Projects Using the PICmicro*. Published by McGraw-Hill Professional. 2004.

17. Jan Axelson. *Serial Port Complete: Programming and Circuits for RS-232 and RS-485 Links and Networks.* Published by lakeview research llc. 1998.

18. Mohamed A. El-Sharkawi. *Electric Energy: An Introduction*
Published by CRC Press. 2004.

APPENDIX A

PIC PROGRAMMING

```
INCLUDE      "bs2defs.bas"            'has some definition in it
DEFINE OSC 8                          'define the oscillator speed in MHz
Define LCD_DREG   PORTD               'initialize lcd
Define LCD_DBIT    4
Define LCD_RSREG PORTE
Define LCD_RSBIT  0
Define LCD_EREG   PORTE
Define LCD_EBIT    1

SerI   VAR PORTC.0           'define input port
x      var BYTE

TRISA = %00000001
TRISB = %00000000

Low PORTE.2                  ' LCD R/W line low (W)
Pause 500                    ' Wait for LCD to start

Start:

portc = %00000000            'clear port C
Serin2 SerI, 84, [dec3 B0]   'get three digit decimal number data from ‘MATLAB
                             GUI

SELECT CASE B0

CASE 001                     'forward
Lcdout $fe, 1, "MANUAL MODE"            'Send to LCD
PAUSE 1000
GOSUB forward
pause 100
goto start

CASE 002                                'reverse
Lcdout $fe, 1, "MANUAL MODE"            'Send to LCD
PAUSE 1000
GOSUB undur
pause 100
goto start

CASE 003                                'right
```

```
Lcdout $fe, 1, "MANUAL MODE"       'Send to LCD
PAUSE 1000
GOSUB right
pause 100
goto start

CASE 004                           'left
Lcdout $fe, 1, "MANUAL MODE"       'Send to LCD
PAUSE 1000
GOSUB left
pause 100
goto start

case 005
Lcdout $fe, 1, "MOTOR STOP"        'Send to LCD
portb=%00000000
pause 1000
goto start

case 006
pause 100
Lcdout $fe, 1, "AUTONOMOUS MODE"   'Send to LCD
portb=%00000101
PAUSE 1000
Lcdout $fe, 1, "MOTOR FORWARD"     'Send to LCD
PAUSE 1000
GOSUB mainloop

CASE 007
PAUSE 100
LCDOUT $fe, 1, "AUTONOMOUS MODE"         ' Send to LCD
portb=%00001010
PAUSE 1000
Lcdout $fe, 1, "MOTOR REVERSE"           ' Send to LCD
PAUSE 1000
GOSUB mainloop

end select

goto start

forward:
    portb=%00000101
    Lcdout $fe, 1, "MOTOR FORWARD"       ' Send to LCD
    pause 1000
```

```
        RETURN

undur:
        portb=%00001010
        Lcdout $fe, 1, "MOTOR REVERSE"     ' Send to LCD
        pause 1000
        RETURN

right:
        portb=%00001001
        Lcdout $fe, 1, "TURN RIGHT"        ' Send to LCD
        pause 1000
        portb=%00000000
        Lcdout $fe, 1, "MOTOR STOP"        ' Send to LCD
        pause 100
        RETURN

left:
        portb=%00000110
        Lcdout $fe, 1, "TURN LEFT"         ' Send to LCD
        pause 1000
        portb=%00000000
        Lcdout $fe, 1, "MOTOR STOP"        ' Send to LCD
        pause 100
        RETURN

' Subroutine to read a/d converter
getad:
ADCON1=4                    ' Set PortA 0, 1, 3 to A/D inputs
        Pauseus 50                  ' Wait for channel to setup
        ADCON0.2 = 1                ' Start conversion
        Pauseus 50                  ' Wait for conversion
        Return

' Subroutine to get pot x value
getx:
        ADCON0 = $41                       ' Set A/D to Fosc/8, Channel 0, On
        Gosub getad
        x = ADRESH
        Return

mainloop:
        Gosub getx                                 ' Get x value
        Lcdout $fe, 1, "Distance=", #x             ' Send to LCD
        Pause 500                  ' Do it about 5 times a second
```

```
          if x>=100 then balik
          IF X<=10 THEN balik
          Goto mainloop                                   ' Do it forever

balik:
          portb=%00000000
          Lcdout $fe, 1, "MOTOR STOP"              ' Send to LCD
          pause 1000
          goto start

end
```

APPENDIX B

AUTONOMOUS MODE

```
function varargout = auto(varargin)
% AUTO M-file for auto.fig
%      AUTO, by itself, creates a new AUTO or raises the existing
%      singleton*.
%
%      H = AUTO returns the handle to a new AUTO or the handle to
%      the existing singleton*.
%
%      AUTO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in AUTO.M with the given input arguments.
%
%      AUTO('Property','Value',...) creates a new AUTO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before auto_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to auto_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help auto

% Last Modified by GUIDE v2.5 14-Oct-2008 14:15:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @auto_OpeningFcn, ...
                   'gui_OutputFcn',  @auto_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
```

```
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before auto is made visible.
function auto_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to auto (see VARARGIN)

SerPIC=serial('COM3')   %define the port available
Check=SerPIC.status     %to check port status data
handles.status=Check    %store data
handles.op=SerPIC;  % store data
guidata(hObject, handles); %save data

% Choose default command line output for auto
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes auto wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = auto_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in check2.
function check2_Callback(hObject, eventdata, handles)
% hObject    handle to check2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of check2
```

```
if (get(hObject,'Value')==get(hObject,'Max'));
   SerPIC=handles.op  % retrieve data

set(SerPIC,'BaudRate',9600,'DataBits',8,'Parity','none','StopBits',1,'FlowControl','non
e');
   fopen(SerPIC)
   guidata(hObject,handles);     %save data    ;
else
   SerPIC=handles.op
   fclose(SerPIC)
   guidata(hObject,handles)

end
guidata(hObject,handles);

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

SerPIC=handles.op
fprintf(SerPIC,'%s','006');

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
close
figure(pop)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
close
figure(select)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
SerPIC=handles.op
fprintf(SerPIC,'%s','005');

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
SerPIC=handles.op
Check=handles.status
u=SerPIC.status

set(handles.text9,'String',u)

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

SerPIC=handles.op
fprintf(SerPIC,'%s','007');
```

APPENDIX C

MANUAL MODE

```
function varargout = manual(varargin)

% MANUAL M-file for manual.fig

%      MANUAL, by itself, creates a new MANUAL or raises the existing
%      singleton*.
%
%      H = MANUAL returns the handle to a new MANUAL or the handle to
%      the existing singleton*.
%
%      MANUAL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MANUAL.M with the given input arguments.
%      MANUAL('Property','Value',...) creates a new MANUAL or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before manual_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to manual_OpeningFcn via varargin.
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help manual
% Last Modified by GUIDE v2.5 11-Oct-2008 10:55:03
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @manual_OpeningFcn, ...
                   'gui_OutputFcn',  @manual_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before manual is made visible.
```

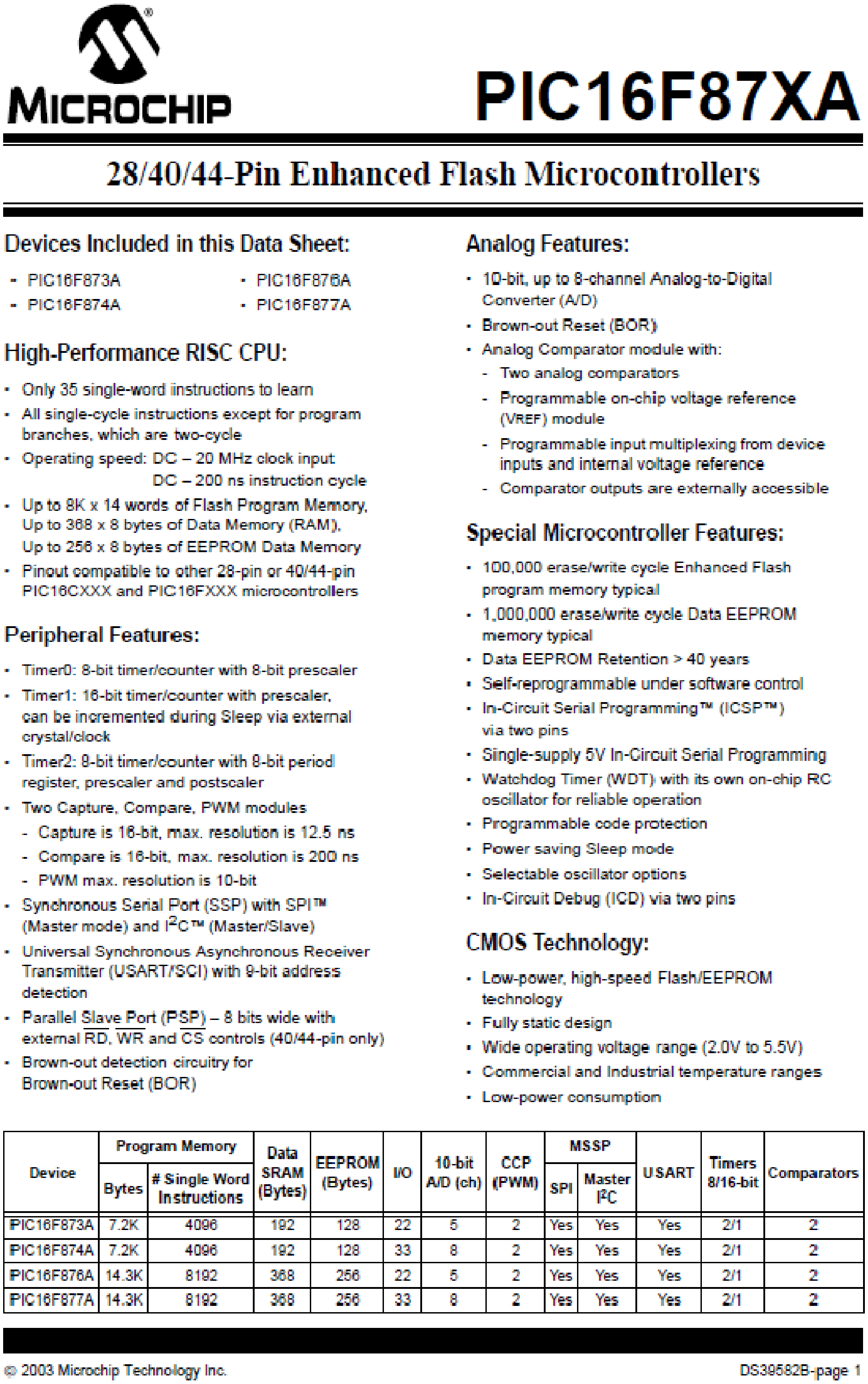
```
function manual_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to manual (see VARARGIN)

SerPIC=serial('COM3')   %define the port available
Check=SerPIC.status     %to check port status data
handles.status=Check    %store data
handles.op=SerPIC;  % store data
guidata(hObject, handles); %save data

% Choose default command line output for manual
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes manual wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = manual_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in check.
function check_Callback(hObject, eventdata, handles)
% hObject    handle to check (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of check

if (get(hObject,'Value')==get(hObject,'Max'));
   SerPIC=handles.op  % retrieve data

set(SerPIC,'BaudRate',9600,'DataBits',8,'Parity','none','StopBits',1,'FlowControl','non
e');
```

```
   fopen(SerPIC)
   guidata(hObject,handles);     %save data    ;
else
   SerPIC=handles.op
   fclose(SerPIC)
   guidata(hObject,handles)

end
guidata(hObject,handles);

% --- Executes on button press in fw.
function fw_Callback(hObject, eventdata, handles)
% hObject    handle to fw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

SerPIC=handles.op
fprintf(SerPIC,'%s','001');

% --- Executes on button press in rw.
function rw_Callback(hObject, eventdata, handles)
% hObject    handle to rw (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

SerPIC=handles.op
fprintf(SerPIC,'%s','002');

% --- Executes on button press in l.
function l_Callback(hObject, eventdata, handles)
% hObject    handle to l (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

SerPIC=handles.op
fprintf(SerPIC,'%s','004');

% --- Executes on button press in r.
function r_Callback(hObject, eventdata, handles)
% hObject    handle to r (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

SerPIC=handles.op
fprintf(SerPIC,'%s','003');
```

```
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
close
figure(select)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
close
figure(pop)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
SerPIC=handles.op
Check=handles.status
u=SerPIC.status

set(handles.text5,'String',u)

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

SerPIC=handles.op
fprintf(SerPIC,'%s','005');
```

APPENDICE D

PIC16F877A DATASHEET



MICROCHIP

PIC16F87XA

28/40/44-Pin Enhanced Flash Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

High-Performance RISC CPU:

- Only 35 single-word instructions to learn
- All single-cycle instructions except for program branches, which are two-cycle
- Operating speed: DC – 20 MHz clock input
  DC – 200 ns instruction cycle
- Up to 8K x 14 words of Flash Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM),
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to other 28-pin or 40/44-pin PIC16CXXX and PIC16FXXX microcontrollers

Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during Sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^{2}C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) – 8 bits wide with external $\overline{RD}$, $\overline{WR}$ and $\overline{CS}$ controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter (A/D)
- Brown-out Reset (BOR)
- Analog Comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (VREF) module
  - Programmable input multiplexing from device inputs and internal voltage reference
  - Comparator outputs are externally accessible

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Data EEPROM Retention > 40 years
- Self-reprogrammable under software control
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Single-supply 5V In-Circuit Serial Programming
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options
- In-Circuit Debug (ICD) via two pins

CMOS Technology:

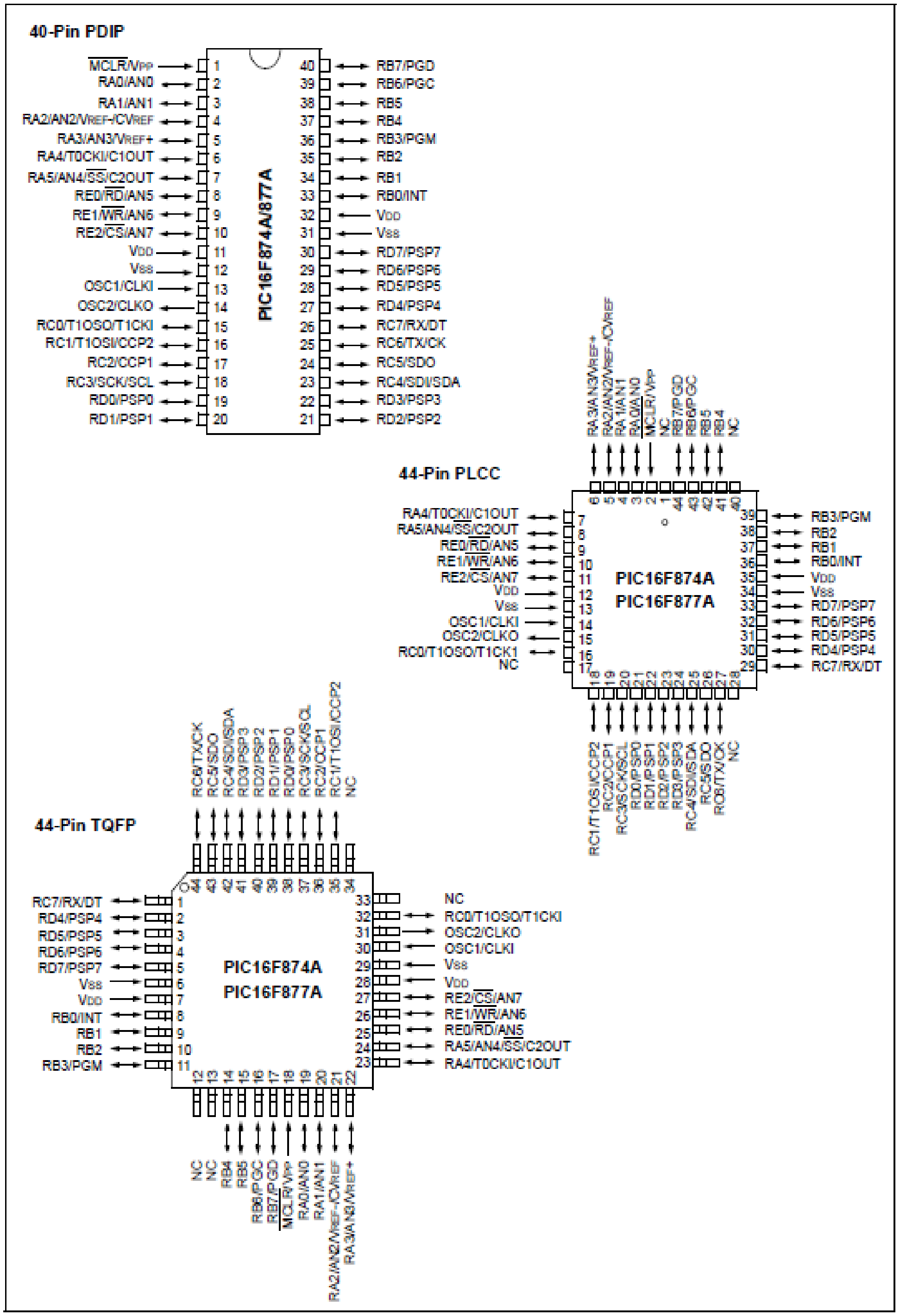
- Low-power, high-speed Flash/EEPROM technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Commercial and Industrial temperature ranges
- Low-power consumption

Device	Program Memory		Data SRAM (Bytes)	EEPROM (Bytes)	I/O	10-bit A/D (ch)	CCP (PWM)	MSSP		USART	Timers 8/16-bit	Comparators
	Bytes	# Single Word Instructions						SPI	Master I^{2}C			
PIC16F873A	7.2K	4096	192	128	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F874A	7.2K	4096	192	128	33	8	2	Yes	Yes	Yes	2/1	2
PIC16F876A	14.3K	8192	368	256	22	5	2	Yes	Yes	Yes	2/1	2
PIC16F877A	14.3K	8192	368	256	33	8	2	Yes	Yes	Yes	2/1	2

© 2003 Microchip Technology Inc.

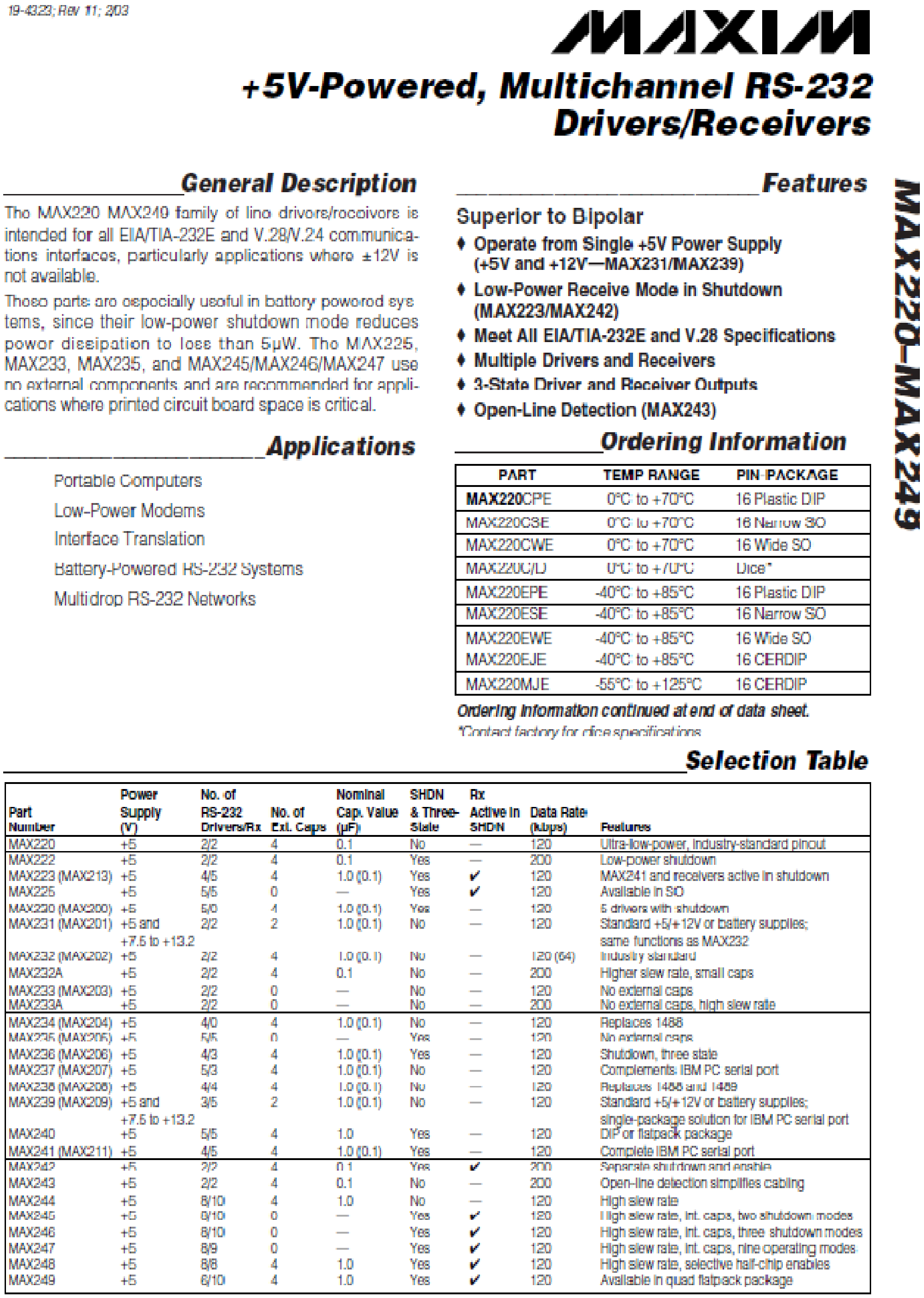
DS39582B-page 1

Pin Diagrams (Continued)



APPENDICE E

MAX 233 DATASHEET



19-4323; Rev 11; 2/03

MAXIM

+5V-Powered, Multichannel RS-232 Drivers/Receivers

MAX220–MAX249

General Description

The MAX220–MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where ±12V is not available.

These parts are especially useful in battery-powered systems, since their low-power shutdown mode reduces power dissipation to less than 5μW. The MAX225, MAX233, MAX235, and MAX245/MAX246/MAX247 use no external components and are recommended for applications where printed circuit board space is critical.

Applications

Portable Computers

Low-Power Modems

Interface Translation

Battery-Powered RS-232 Systems

Multidrop RS-232 Networks

Features

Superior to Bipolar

- **Operate from Single +5V Power Supply (+5V and +12V—MAX231/MAX239)**
- **Low-Power Receive Mode in Shutdown (MAX223/MAX242)**
- **Meet All EIA/TIA-232E and V.28 Specifications**
- **Multiple Drivers and Receivers**
- **3-State Driver and Receiver Outputs**
- **Open-Line Detection (MAX243)**

Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX220CPE	0°C to +70°C	16 Plastic DIP
MAX220CSE	0°C to +70°C	16 Narrow SO
MAX220CWE	0°C to +70°C	16 Wide SO
MAX220C/D	0°C to +70°C	Dice*
MAX220EPE	-40°C to +85°C	16 Plastic DIP
MAX220ESE	-40°C to +85°C	16 Narrow SO
MAX220EWE	-40°C to +85°C	16 Wide SO
MAX220EJE	-40°C to +85°C	16 CERDIP
MAX220MJE	-55°C to +125°C	16 CERDIP

Ordering Information continued at end of data sheet.

**Contact factory for dice specifications.*

Selection Table

Part Number	Power Supply (V)	No. of RS-232 Drivers/Rx	No. of Ext. Caps	Nominal Cap. Value (μF)	SHDN & Three-State	Rx Active in SHDN	Data Rate (kbps)	Features
MAX220	+5	2/2	4	0.1	No	—	120	Ultra-low-power, industry-standard pinout
MAX222	+5	2/2	4	0.1	Yes	—	200	Low-power shutdown
MAX223 (MAX213)	+5	4/5	4	1.0 (0.1)	Yes	✔	120	MAX241 and receivers active in shutdown
MAX225	+5	5/5	0	—	Yes	✔	120	Available in SO
MAX230 (MAX200)	+5	5/0	4	1.0 (0.1)	Yes	—	120	5 drivers with shutdown
MAX231 (MAX201)	+5 and +7.5 to +13.2	2/2	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; same functions as MAX232
MAX232 (MAX202)	+5	2/2	4	1.0 (0.1)	No	—	120 (64)	Industry standard
MAX232A	+5	2/2	4	0.1	No	—	200	Higher slew rate, small caps
MAX233 (MAX203)	+5	2/2	0	—	No	—	120	No external caps
MAX233A	+5	2/2	0	—	No	—	200	No external caps, high slew rate
MAX234 (MAX204)	+5	4/0	4	1.0 (0.1)	No	—	120	Replaces 1488
MAX235 (MAX205)	+5	5/5	0	—	Yes	—	120	No external caps
MAX236 (MAX206)	+5	4/3	4	1.0 (0.1)	Yes	—	120	Shutdown, three state
MAX237 (MAX207)	+5	5/3	4	1.0 (0.1)	No	—	120	Complements IBM PC serial port
MAX238 (MAX208)	+5	4/4	4	1.0 (0.1)	No	—	120	Replaces 1488 and 1489
MAX239 (MAX209)	+5 and +7.5 to +13.2	3/5	2	1.0 (0.1)	No	—	120	Standard +5/+12V or battery supplies; single-package solution for IBM PC serial port
MAX240	+5	5/5	4	1.0	Yes	—	120	DIP or flatpack package
MAX241 (MAX211)	+5	4/5	4	1.0 (0.1)	Yes	—	120	Complete IBM PC serial port
MAX242	+5	2/2	4	0.1	Yes	✔	200	Separate shutdown and enable
MAX243	+5	2/2	4	0.1	No	—	200	Open-line detection simplifies cabling
MAX244	+5	8/10	4	1.0	No	—	120	High slew rate
MAX245	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, two shutdown modes
MAX246	+5	8/10	0	—	Yes	✔	120	High slew rate, int. caps, three shutdown modes
MAX247	+5	8/9	0	—	Yes	✔	120	High slew rate, int. caps, nine operating modes
MAX248	+5	8/8	4	1.0	Yes	✔	120	High slew rate, selective half-chip enables
MAX249	+5	6/10	4	1.0	Yes	✔	120	Available in quad flatpack package

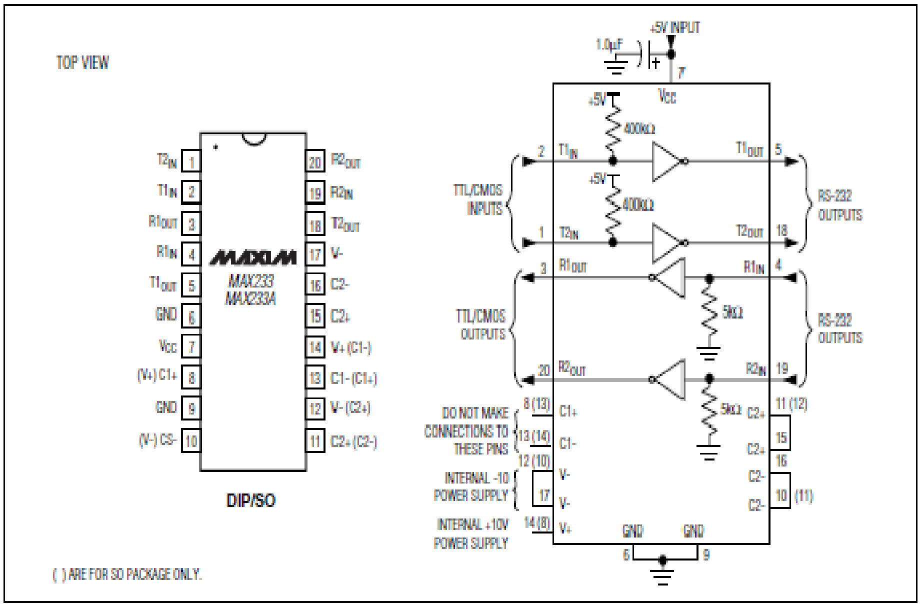


Figure 11. MAX233/MAX233A Pin Configuration and Typical Operating Circuit

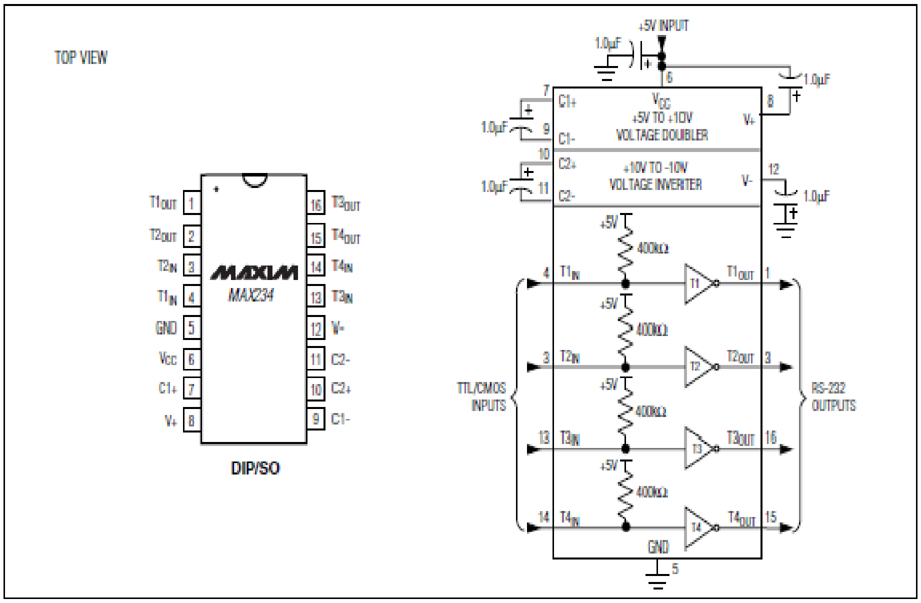


Figure 12. MAX234 Pin Configuration and Typical Operating Circuit